Online Dynamic Bin Packing

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

Mihai Burcea

October 2014

For Robert and my sister.

Ship: I have taught you about the classical Pandora and her box.
Panille: I know how this planet got its name.
Ship: Where would you hide when the serpents and shadows oozed out of the box?
Panille: Under the lid, of course.

–Kerro Panille, *Shiprecords*

by Frank Herbert, *The Jesus Incident*

But the Great Sea is terrible, Tuor son of Huor; and it hates the Noldor, for it works the Doom of the Valar. Worse things it holds than to sink into the abyss and so perish: loathing, and loneliness, and madness; terror of wind and tumult, and silence, and shadows where all hope is lost and all living shapes pass away. And many shores evil and strange it washes, and many islands of danger and fear infest it. I will not darken your heart, son of Middle-earth, with the tale of my labour seven years in the Great Sea from the North even into the South, but never to the West. For that is shut against us.

by J.R.R. Tolkien and Christopher Tolkien as editor, *Unfinished Tales*

Contents

Illustrations

List of Figures

List of Tables

List of Acronyms

1-D	One-Dimensional
2-D	Two-Dimensional
3-D	Three-Dimensional
AF	Any-Fit
APTAS	Asymptotic Polynomial Time Approximation Scheme
BF	Best-Fit
***d*-D**	*d*-Dimensional (equivalently, Multi-Dimensional)
FF	First-Fit
FFD	First-Fit Decreasing
FFDH	First-Fit Decreasing Height
FFM	First-Fit Modified
MFFD	Modified First-Fit Decreasing
NF	Next-Fit
NFDH	Next-Fit Decreasing Height
NP	Non-deterministic Polynomial
P	Polynomial
PF	Power Fraction
PTAS	Polynomial Time Approximation Scheme
RCP	Rectangle Packing
UF	Unit Fraction
WF	Worst-Fit

Preface

A significant part of this thesis is based on two peer-reviewed papers that have been published in international conferences. The two papers have been adapted for the purpose of this thesis and expanded to contain work that was omitted from the conference versions of the papers. The extended versions of the papers are also either under submission for journal publication or in preparation for submission. An additional chapter presents work that has not yet been published that is related to one of the above papers.

Specifically, Chapter 3 of the thesis is based on the paper entitled "An 8/3 Lower Bound for Online Dynamic Bin Packing", co-authored with Prudence W.H. Wong and Fencol C.C. Yung. The paper has been published in Proceedings of the 23rd International Symposium on Algorithms and Computation.

> We study the dynamic bin packing problem introduced by Coffman, Garey and Johnson. This problem is a generalization of the bin packing problem in which items may arrive and depart dynamically. The objective is to minimize the maximum number of bins used over all time. The main result is a lower bound of $8/3 \sim 2.666$ on the achievable competitive ratio, improving the best known 2.5 lower bound. The previous lower bounds were 2.388, 2.428, and 2.5. This moves a big step forward to close the gap between the lower bound and the upper bound, which currently stands at 2.788. The gap is reduced by about 60% from 0.288 to 0.122. The improvement stems from an adversarial sequence that forces an online algorithm $\mathcal{A}$ to open $2s$ bins with items having a total size of s only and this can be adapted appropriately regardless of the current load of other bins that have already been opened by $\mathcal{A}$. Comparing with the previous 2.5 lower bound, this basic step gives a better way to derive the complete adversary and a better use of items of slightly different sizes leading to a tighter lower bound. Furthermore, we show that the 2.5-lower bound can be obtained using this basic step in a much simpler way without case analysis.

Chapters 4 and 5 of the thesis are based on the paper entitled "Online Multi-dimensional Dynamic Bin Packing of Unit-Fraction Items", co-authored with Prudence W.H. Wong and Fencol C.C. Yung. The paper has been published in Proceedings of the 8th International Conference on Algorithms and Complexity.

We study the 2-D and 3-D dynamic bin packing problem, in which items arrive and depart at arbitrary times. The 1-D problem was first studied by Coffman, Garey, and Johnson motivated by the dynamic storage problem. Bar-Noy et al. have studied packing of unit fraction items (i.e., items with length $1/k$ for some integer $k \geq 1$), motivated by the window scheduling problem. In this paper, we extend the study of 2-D and 3-D dynamic bin packing problem to unit fractions items. The objective is to pack the items into unit-sized bins such that the maximum number of bins ever used over all time is minimized. We give a scheme that divides the items into classes and show that applying the First-Fit algorithm to each class is 6.7850- and 21.6108-competitive for 2-D and 3-D, respectively, unit fraction items. This is in contrast to the 7.4842 and 22.4842 competitive ratios for 2-D and 3-D, respectively, that would be obtained using only existing results for unit fraction items.

Finally Chapter 6 represents a continuation of the work in Chapters 4 and 5 that focuses on obtaining lower bounds for certain online algorithms for multi-dimensional dynamic bin packing. We consider several types of input and construct different adversaries suitable to each specific input.

Abstract

In this thesis we study online algorithms for dynamic bin packing. An online algorithm is presented with input throughout time and must make irrevocable decisions without knowledge of future input. The classical bin packing problem is a combinatorial optimization problem in which a set of items must be packed into a minimum number of uniform-sized bins without exceeding their capacities. The problem has been studied since the early 1970s and many variants continue to attract researchers' attention today. The dynamic version of the bin packing problem was introduced by Coffman, Garey and Johnson in 1983. The problem is a generalization of the bin packing problem in which items may arrive and depart dynamically. In this setting, an online algorithm for bin packing is presented with one item at a time, without knowledge of its departure time, nor arrival and departure times of future items, and must decide in which bin the item should be packed. Migration of items between bins is not allowed, however rearrangement of items within a bin is permitted. The objective of problem is to minimize the maximum number of bins used over all time. In multi-dimensional generalizations of the problem, multi-dimensional items must be packed without overlap in multi-dimensional bins of uniform size in each dimension. In this work, we study the setting where items are oriented and cannot be rotated.

We first consider online one-dimensional dynamic bin packing and present a lower bound of $8/3 \sim 2.666$ on the achievable competitive ratio of any deterministic online algorithm, improving the best known 2.5-lower bound. Since the introduction of the problem by Coffman, Garey and Johnson, the progress on the problem has focused on improving the original lower bound of 2.388 to 2.428, and to the best known 2.5-lower bound. Our improvement from 2.5 to $8/3 \sim 2.666$ makes a big step forward in closing the gap between the lower bound and the upper bound, which currently stands at 2.788.

Secondly we study the online two- and three-dimensional dynamic bin packing problem by designing and analyzing algorithms for special types of input. Bar-Noy et al. initiated the study of the one-dimensional unit fraction bin packing problem, a restricted version where all sizes of items are of the form $1/k$, for some integer $k > 0$. Another related problem is for power fraction items, where sizes are of the form $1/2^k$, for some integer $k \geq 0$. We initiate the study of online multi-dimensional dynamic bin packing of unit fraction items and power fraction items, where items have lengths unit fraction and power fraction in each dimension, respectively. While algorithms for general input are suitable for unit fraction and power fraction items, their worst-case performance guarantees are the same for special types of input. For unit fraction and power fraction items, we design and analyze online algorithms that achieve better worst-case performance guarantees compared to their classical counterparts. Our algorithms give careful consideration to unit and power fraction items, which allows us to reduce the competitive ratios for these types of inputs.

Lastly we focus on obtaining lower bounds on the performance of the family of Any-Fit algorithms (Any-Fit, Best-Fit, First-Fit, Worst-Fit) for online multi-dimensional dynamic bin packing. Any-Fit algorithms are classical online algorithms initially studied for the one-dimensional version of the bin packing problem. The common rule that the algorithms use is to never pack a new item to a new bin if the item can be packed in any of the existing bins. While the family of Any-Fit algorithms is always $O(1)$-competitive for one-dimensional dynamic bin packing, we show that this is no longer the case for multi-dimensional dynamic bin packing when using Best-Fit and Worst-Fit, even if the input consists of power fraction items or unit fraction items. For these restricted inputs, we prove that Best-Fit and Worst-Fit have unbounded competitive ratios, while for First-Fit we provide lower bounds that are higher than the lower bounds for any online algorithm. Furthermore, for general input we show that all classical Any-Fit algorithms are not competitive for online multi-dimensional dynamic bin packing.

Acknowledgements

My biggest thanks go to my supervisor, Prudence W.H. Wong, with whom I have had the privilege and pleasure of working from my second year as an undergraduate student to the very end of my research degree. Her guidance and support were invaluable to me throughout my summer project as an undergraduate student, following to my final year project, and leading to the research found here. I am honoured to have been her student, but regret I may not be able to repay all the time and energy she has invested in me. She has shown me the elegance of research, and for me this elegance represents saying something correctly and clearly. I have been allowed to contribute alongside her and our co-authors to a piece of knowledge, and for this I am forever grateful.

Of course, the work in this thesis and elsewhere would not have been possible without the help and enthusiasm of my other co-authors, Wing-Kai Hon, Hsiang-Hsuan Liu, David K.Y. Yau, and Fencol C.C. Yung. I am very thankful for the time they have invested in our work. Their dedication allowed us to bring our ideas to published pages.

A great aid in evaluating my progress throughout the years was provided by my advisors, Leszek A. Gąsieniec, Mingyu Guo, and Piotr Krysta. I am very thankful for their suggestions on improving and expanding the work in this thesis.

Lastly, I would like to thank my examiners, Stanley P.Y. Fung and Leszek A. Gąsieniec. I am very grateful for their thoroughness and advice in revising this thesis to its final version.

Chapter 1

Introduction

In this thesis we investigate the area of online algorithms in the context of the bin packing problem. An *online algorithm* must make irrevocable decisions without knowledge of future input. Our aim is the study of performance guarantees of online algorithms for optimization problems, specifically variants of the bin packing problem. The classical bin packing problem is a combinatorial optimization problem in which a set of items must be packed into a minimum number of uniform-sized bins without exceeding their capacities. In this setting, an online algorithm for bin packing is presented with one item at a time, without knowledge of future items, and must decide in which bin the item should be packed. Migration of items between bins is not allowed, however rearrangement of items within a bin is permitted. The objective of the problem is to minimize the total number of bins used. A classical approach to measure the performance of an online algorithm is *competitive analysis*, where the output of an online algorithm is compared to the output of an optimal offline algorithm that holds complete knowledge about the input. We first give an introduction to algorithms, including offline algorithms and the NP complexity class in Section 1.1.1, approximation algorithms in Section 1.1.2, and online algorithms and competitive analysis in Section 1.1.3. We further present the models for bin packing problems we consider in Section 1.2. The organization of the thesis and author's contributions are noted in Sections 1.3 and 1.4, respectively. Finally in Section 1.5 we note an additional contribution of the author that was not included as part of this thesis.

1.1 Algorithms

In this thesis we focus on deterministic algorithms for optimization problems. A deterministic algorithm is an algorithm which, presented with a particular input, will always produce the same output by passing through the same sequence of steps. Depending on the nature of the input, algorithms can be classified as offline and online. In the offline setting, all input is known in advance to the algorithm and can usually be preprocessed in such a way that it allows for good performance guarantees. On the other hand, in the online setting, input is received throughout time and preprocessing is not usually available. Our goal is to overcome these additional hurdles and design and analyze online deterministic algorithms with good performance guarantees. The class of problems for which we aim to prove performance guarantees are called optimization problems.

An optimization problem refers to a problem of either cost minimization or profit maximization. In our case, the bin packing problem is a cost minimization problem and we give definitions applicable to this type of optimization problem, while profit maximization problems can be defined similarly. An optimization problem $\mathcal{P}$ of cost minimization can be defined as follows [11]. $\mathcal{P}$ consists of a set $\mathcal{I}$ of inputs and a cost function C. Associated with every input I is a set of feasible outputs (or solutions) $F(I)$, and associated with each feasible solution O in $F(I)$ is a positive real, $C(I, O)$, representing the cost of the output O with respect to the input I. The aim of such a problem thus becomes to minimize $C(I, O)$ for any input $I \in \mathcal{I}$.

1.1.1 Offline Algorithms and the NP Complexity Class

Offline algorithms are algorithms that hold complete knowledge of the input for a problem and are required to output an answer or solution for a given problem. For many important optimization problems, including the bin packing problem, an optimal solution of minimum cost (or maximum profit, respectively) may not be determined in polynomial time, even if the input is received offline. These problems fall within the non-deterministic polynomial (NP) complexity class [45] and some of the hardest problems in NP are NP-complete and NP-hard problems. On the other hand, the polynomial (P) complexity class also falls within NP and its problems are polynomial-time solvable.

For both NP-complete and NP-hard problems, an optimal solution cannot be found in polynomial time, unless P=NP [29]. Given a solution for an optimization problem, the difference between NP-complete and NP-hard problems lies in whether we may be able to verify in polynomial time if the solution given is an optimal solution. While NP-complete problems are verifiable in polynomial time, NP-hard problems may or may not be verifiable in polynomial time. The bin packing problem, for example, is an NP-hard problem which is not verifiable in polynomial time, assuming P≠NP. This is due to the fact that an optimal solution is required beforehand in order to determine if the given solution is optimal. Up to now no polynomial-time algorithm for any NP-complete or NP-hard problem is known and researchers have focused on approximation algorithms that run in polynomial time.

1.1.2 Approximation Algorithms

In order to solve NP-complete or NP-hard optimization problems in polynomial time, a trade-off is typically made between solution cost and time complexity. Approximation algorithms are offline algorithms that are often used to obtain near-optimal solutions to such problems and can run in polynomial time. An approximation algorithm must produce a feasible solution with cost at most some factor away from the optimal cost. This factor is called an approximation ratio.

Let $\mathcal{A}(I)$ denote the cost of the solution for an approximation algorithm $\mathcal{A}$ on any input I, and $OPT(I)$ denote the cost of the optimal solution on I. For a minimization problem, we say that $\mathcal{A}$ has an asymptotic approximation ratio $c > 1$ (or alternatively is c-approximate) if for any input I, $\mathcal{A}(I) \leq c\,OPT(I) + b$, for a positive constant b. If the constant b is not a positive value, then we say that $\mathcal{A}$ has an *absolute* approximation ratio c (or alternatively is strictly c-approximate). The definition for a maximization problem is similar, except that we now require that $OPT(I) \leq c\,\mathcal{A}(I) + b$. These performance guarantees are similar for online algorithms, which are the main focus of this work, as we will see in the next subsection.

1.1.3 Online Algorithms and Competitive Analysis

Optimization problems in which the input is received one at a time and output must be produced throughout time are called online problems. In order to solve such problems, online algorithms are used. An online algorithm receives input throughout time and must make irrevocable decisions without knowledge of future input. The disadvantage of online algorithms becomes apparent compared to offline algorithms that have access to the whole input in advance. In the case of online algorithms, irrevocable decisions taken throughout their execution affect the cost of the solution produced. Nonetheless, the challenge of designing 'good' online algorithms has contributed to the theory of combinatorial optimization for almost half a century in the context of, e.g., bin packing, scheduling, and data structures (surveys [23], [64], and [1], respectively). Although many of the problems mentioned above are initially studied offline, their online counterparts reflect realistic models and online algorithms constitute an interesting class of algorithms by themselves. It is also worth mentioning that some online algorithms give good offline approximation results, thus their significance is not only limited to online problems.

Competitive analysis [11] was introduced by Sleator and Tarjan [65] as a measurement of the worst-case performance of an online algorithm $\mathcal{A}$ compared to an optimal offline algorithm OPT. As mentioned before, OPT knows the whole input in advance and is able to compute an optimal solution for a given problem. Even though the computational complexity of finding such an optimal solution may be very high and we may not be able to explicitly give such an optimal algorithm, a comparison of performance between $\mathcal{A}$ and OPT can still be made, as the minimum cost of OPT's solution is constrained by the problem model and input. Competitive analysis has also been compared to a game of two players [67], in which an online algorithm plays against an adversary that generates the input and acts as an offline algorithm, while it tries to minimize its cost relative to the online algorithm.

For a cost minimization problem, an online algorithm $\mathcal{A}$ is said to be c-competitive against an optimal offline algorithm OPT if there exists a constant b such that for any input I, $\mathcal{A}(I) \leq c\,OPT(I) + b$, where $\mathcal{A}(I)$ and $OPT(I)$ denote the cost of the solution of $\mathcal{A}$ and OPT, respectively, on the input I. In this case, we say that $\mathcal{A}$ is c-competitive or equivalently that $\mathcal{A}$ attains a competitive ratio c. When the constant

b is a positive value, c is called an asymptotic competitive ratio. Having a positive constant b reflects that for online problems we allow arbitrarily long input and thus the value of b becomes insignificant. On the other hand, when b is a constant with value at most zero, i.e., $\mathcal{A}(I) \leq c\,OPT(I)$, we say that $\mathcal{A}$ is a strictly c-competitive algorithm or equivalently that $\mathcal{A}$ attains an absolute competitive ratio c. Note that in the case of a cost minimization problem the competitive ratio is at least 1 and the smaller it is, the better an online algorithm performs against an optimal offline algorithm.

In this work we focus on the asymptotic performance of deterministic online algorithms for the bin packing problem. A different approach would be to use randomized online algorithms. Randomized algorithms for bin packing are algorithms that use coin flips to determine where to pack items. We recognize that randomization may be useful in improving upper bounds for optimization problems. However, for bin packing it is currently unclear whether randomized algorithms may help substantially. As noted in [23], Chandra [21] gave adversaries for any randomized online algorithm for some bin packing problems where the lower bounds were matching the lower bounds of deterministic online algorithms. Note that some lower bounds for deterministic online algorithms were further improved since [21]. The study of randomized online algorithms for bin packing would merit a topic of its own, however it is beyond the scope of this thesis. We will thus mainly study the bin packing problem from a deterministic point of view.

1.2 Bin Packing Problems

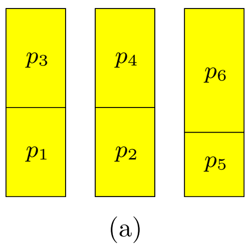
There are many variants of the classical bin packing problem that continue to attract researchers' attention (see the surveys [31, 27, 24, 23]). We introduce the classical bin packing problem in Section 1.2.1 before proceeding to online (static) bin packing in Section 1.2.2. In Section 1.2.3 we introduce the online dynamic bin packing problem, which is the model studied throughout this thesis. These sections give a brief overview of the problem models and some results for them, while Chapter 2 gives a detailed literature review. Most of the results obtained in this thesis are described in Section 1.2.3, with additional results in Section 1.2.4.

1.2.1 The Classical Bin Packing Problem

The bin packing problem is an NP-hard [45] classical combinatorial optimization problem that has been extensively studied throughout the years and different variants continue to attract researchers' attention (surveys [31, 27, 24, 23]).

Problem definition. In the offline setting [53], we are given a set of items, each with size a real number in $(0, 1]$ and an infinite supply of unit-size bins. The objective of the bin packing problem is to pack the set of items into a minimum number of bins such that the total size of the items in a bin does not exceed the bin capacity.

Motivation. The bin packing problem has many applications such as allocating memory in computers [20], assigning commercials to station breaks in television programming



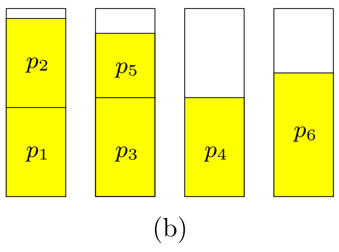


Figure 1.1: (a) The optimal solution with three bins and (b) the First-Fit online algorithm solution with four bins for item list L. The example shows items being packed along the height of a one-dimensional bin.

[13], and packing and cutting stock applications [2]. Multi-dimensional generalizations of the problem have applications in storing real objects. Furthermore, the dynamic setting is a more realistic model that takes into account departure of items at arbitrary times, and is suitable for dynamic storage allocation in computers and warehouse storage. It is worth mentioning that the online versions of the problems also reflect more realistic scenarios where the input is given one at a time, and decisions are irrevocable and will impact the overall cost of the solution.

1.2.2 Online Bin Packing

Problem definition. In the online setting [63, 69], items with sizes that are real numbers in $(0, 1]$ arrive at arbitrary times. The item size and arrival time are only known when the item arrives. Given an infinite supply of unit-size bins, the objective of the online bin packing problem is to pack items given throughout time into a minimum number of bins such that the total size of the items in a bin does not exceed the bin capacity. Migration of items between bins is not permitted[1].

Recall from Section 1.1.3 that the performance of an online algorithm is measured using competitive analysis [11]. Consider any online algorithm $\mathcal{A}$. Given an input I, let $\mathcal{A}(I)$ and $OPT(I)$ be the number of bins used at the end of the input by $\mathcal{A}$ and the optimal offline algorithm, respectively. Algorithm $\mathcal{A}$ is said to be c-competitive if there exists a constant b such that $\mathcal{A}(I) \le c\,OPT(I) + b$ for all I. In order to highlight the additional difficulty of the online model, let us consider the following example.

Example 1.1. *Suppose we have an input list L consisting of six items. Let $L = (p_1, p_2, p_3, p_4, p_5, p_6)$ and let s_i denote the size of p_i. For a small $\epsilon > 0$, suppose $s_1 = s_2 = 1/2-\epsilon, s_3 = s_4 = 1/2+\epsilon, s_5 = 1/3$, and $s_6 = 2/3$. In the offline setting, all items sizes are known in advance, while in the online setting item sizes are given one at a time in the order of their indices.*

For Example 1.1, one optimal solution is to pack items as shown in Figure 1.1(a), i.e., p_1 and p_3 are packed in one bin, p_2 and p_4 are packed in a second bin, and p_5 and p_6 are packed in a third bin. An online algorithm will also aim to minimize the "wasted"

[1]This reflects realistic scenarios where the overhead cost of migration may not be justified or where bins may be physically far away from each other.

space within a bin, even without knowledge of the input, in this case consisting of complementary size items. Consider the First-Fit (FF) algorithm. When a new item arrives, if there are occupied bins in which the item can be repacked, FF assigns the new item to the bin which has been occupied for the longest time. In other words, assuming bins are indexed in increasing order from left to right, FF aims to pack a new item in the lowest index bin in which it will fit. If no existing bin can accommodate the item, FF opens a new bin and packs the item there. Note that FF will not open a new bin unless no existing bin can accommodate the new item. For the item list L with items given in the order of their indices, FF requires four bins as shown in Figure 1.1(b).

Knowing the input in advance does provide better cost solutions. For instance, First-Fit Decreasing (FFD), an algorithm that sorts the items in non-increasing order of their sizes and then packs them according to the First-Fit rule, gives an optimal solution for Example 1.1. In general, FFD has an asymptotic approximation ratio of $11/9 \sim 1.223$ [52, 53], while the best lower bound for any algorithm for online bin packing currently stands at 1.54037 [4].

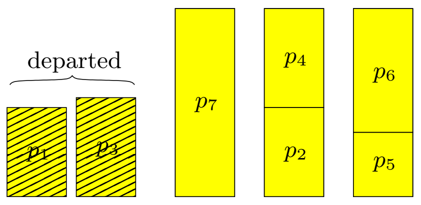
1.2.3 Online Dynamic Bin Packing

Most existing work focuses on "static" bin packing in the sense that items do not depart. In some potential applications like warehouse storage, a more realistic model takes into consideration dynamic arrival and departures of items. This natural generalization, known as dynamic bin packing, was introduced by Coffman, Garey and Johnson [26].

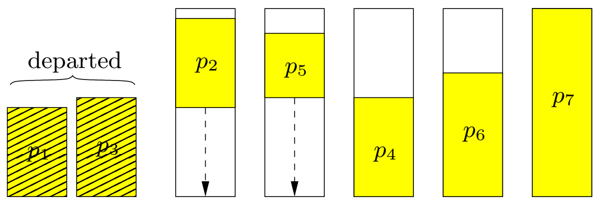
Problem definition. In online dynamic bin packing, items with sizes that are real numbers in $(0, 1]$ arrive over time, reside for some period of time, and may depart at arbitrary time. The size and arrival time of an item are only known when the item arrives and the departure time is only known when the item departs. We are also given an infinite supply of unit-size bins. Each item has to be assigned to a bin from the time it arrives until it departs such that the total size of the items in a bin does not exceed the bin capacity at any time. The objective is to minimize the maximum number of bins used over all time. Migration of items between bins is not allowed, while rearrangement of items within a bin is permitted[2].

While the generalization to include dynamic departure of items is indeed useful, it may further increase the ratio of the online solution cost to the optimal offline solution cost. Recall the item list L in Example 1.1. Suppose first all items in L are packed by the First-Fit (FF) online algorithm (defined in Section 1.2.2). After all items have been packed, we let p_1 and p_3 depart as shown in Figure 1.2(b). Finally, we release one new item p_7 with $s_7 = 1$. In the end, FF uses five bins as no existing bin can accommodate the new item. On the other hand, the cost of the optimal solution remains unchanged, as shown in Figure 1.2(a).

[2]Rearrangement of items within a bin is permitted as it does not affect the configuration of other bins. The related problem where position of items inside a bin are fixed is called "checkerboarding" [57] and has been studied in the context of dynamic storage allocation [62], where one is concerned with managing space within a bin.



(a) The first bin is emptied entirely as items p_1 and p_3 depart. The new item p_7 is packed in the first bin and the cost of the optimal solution remains unchanged.



(b) As items p_1 and p_3 depart the contents of existing bins may be shifted towards the bottom of the bin to accommodate new items. However, no existing bin can accommodate the new item p_7.

Figure 1.2: (a) The optimal solution with three bins and (b) the First-Fit online algorithm solution with five bins.

Existing results. The seminal paper [26] studied the First-Fit online algorithm. They showed that the competitive ratio of First-Fit lies between 2.75 and 2.897. However, the 2.75-lower bound assumes an optimal offline algorithm that allows migration of items between bins. They also showed that a modified First-Fit algorithm is 2.788-competitive. The modified First-Fit algorithm packs any item with size larger than 1/2 to a dedicated bin and all other items according to the First-Fit rule. They also gave a lower bound of 2.388 on the competitive ratio of *any* deterministic online algorithm. As usual, the 2.388-lower bound adversary assumes the optimal offline algorithm knows the input in advance, however it is not allowed to migrate items between bins, thus the result is "stronger" in this sense. This lower bound has later been improved to 2.428 [18] and then to the best known lower bound of 2.5 [19].

Our contribution. In Chapter 3 of this thesis, we improve the lower bound for any deterministic online algorithm to $8/3 \sim 2.666$, reducing the gap between the upper and lower bounds by about 60% from 0.288 to 0.122. We introduce a novel adversarial sequence that forces an online algorithm $\mathcal{A}$ to open $2s$ bins with items having a total size of s only and this can be adapted appropriately regardless of the current load of other bins that have already been opened by $\mathcal{A}$. Comparing with the previous 2.5 lower bound, this basic step gives a better way to derive the complete adversary and a better use of items of slightly different sizes leading to a tighter lower bound. Furthermore, we show that the 2.5-lower bound can be obtained using this basic step in a much simpler way.

Online Multi-dimensional Dynamic Bin Packing. The bin packing problem has been initially studied in one dimension, i.e, 1-D, and has been extended to multiple dimensions, i.e., d-D, where $d \geq 1$ (see survey [39]).

Problem definition. In online multi-dimensional dynamic bin packing (d-D packing), items with lengths in each dimension that are real numbers in $(0, 1]$ arrive over time, reside for some period of time, and may depart at arbitrary time. The lengths and arrival time of an item are only known when the item arrives and the departure time is only known when the item departs. We are also given an infinite supply of multi-dimensional bins of unit size in each dimension. Each item has to be assigned to a bin from the time it arrives until it departs such that items do not overlap and do not exceed the boundary of the bin. The items are oriented and cannot be rotated. The objective of the problem is to minimize the maximum number of bins used over all time. As in one-dimensional dynamic bin packing, migration of items to another bin is not allowed, yet rearrangement of items within a bin is allowed.

Unit fraction and power fraction items. In some real applications, item size is not represented by arbitrary real numbers in $(0, 1]$. Bar-Noy et al. [8] initiated the study of the one-dimensional *unit fraction bin packing problem*, a restricted version where all sizes of items are of the form $\frac{1}{k}$, for some integer $k > 0$. The problem was motivated by the window scheduling problem [7, 8]. Another related problem is for *power fraction* items, where sizes are of the form $\frac{1}{2^k}$, for some integer $k \geq 0$.

It is clear that items with lengths that are real numbers in $(0, 1]$, which we call *general size* items, form a superset of unit fraction and power fraction items, and algorithms for general size items obtain the same worst-case performance for unit fraction and power fraction items. A natural question arising is whether there are better algorithms for multi-dimensional dynamic bin packing of unit fraction and power fraction items. Currently, the best algorithm for online one-dimensional dynamic bin packing of unit and power fraction items is the First-Fit algorithm that attains a competitive ratio of 2.4842 [47], compared with 2.788 [26] for general size items.

Our contribution. We extend the answer to the above question in the affirmative and initiate the study of two- and three-dimensional dynamic bin packing of unit and power fraction items. In Chapter 4 we provide a new algorithm for online 2-D dynamic bin packing of unit fraction items and show that its worst-case performance is indeed better than its classical counterpart. Based on this result, we expand our analysis and provide new algorithms for 2-D power fraction items, and 3-D unit and power fraction items in Chapter 5.

1.2.4 Additional Lower Bounds

The last part of the thesis in Chapter 6 is dedicated to obtaining lower bounds on the performance of classical bin packing algorithms belonging to the Any-Fit family of algorithms. We give adversaries for First-Fit, Best-Fit, Worst-Fit, and Any-Fit for multi-dimensional dynamic bin packing. When a new item R arrives, if there are occupied bins in which R can be packed (allowing repacking for existing items), the algorithms assign R to one of these bins as follows: First-Fit assigns R to the bin which has been occupied for the longest time; Best-Fit assigns R to the heaviest loaded bin with ties broken arbitrarily; Worst-Fit assigns R to the lightest loaded bin with ties broken arbitrarily; and Any-Fit assigns R to any of the bins arbitrarily.

Our contribution. We first show that the First-Fit, Best-Fit, Worst-Fit, and Any-Fit algorithms without modification have unbounded competitive ratios for multi-dimensional dynamic bin packing of general size items. This is contrasted with the performance of First-Fit which is $O(1)$-competitive for one-dimensional dynamic bin packing of general size items [26]. Using an argument from [26], it can also be shown that Any-Fit (including Best-Fit and Worst-Fit) is also $O(1)$-competitive for one-dimensional dynamic bin packing of general size items. Additionally, we show that Best-Fit and Worst-Fit have unbounded competitive ratios for multi-dimensional dynamic bin packing even for power fraction items, while First-Fit is at least 5.45-competitive for 2-D unit fraction items and at least 6.45-competitive for 3-D unit fraction items. For First-Fit, the lower bound results for 2-D and 3-D power fraction items are slightly lower. These results are in contrast with the lower bounds of any algorithm of 3.70301 [37] and 4.85383 [37] for 2-D and 3-D unit fraction items, respectively.

1.3 Organization of the Thesis

The thesis is dedicated to the design and analysis of online algorithms for the dynamic bin packing problem. In particular, Chapter 2 gives a more detailed background of the bin packing problem, current results, and the online dynamic bin packing models that are considered in this thesis. Chapter 3 gives a new approach to improving the lower bound of any deterministic online algorithm for one-dimensional dynamic bin packing. The results are published in [71]. In Chapter 4 we consider two-dimensional online dynamic bin packing of unit fraction items and design and analyze the performance of an online algorithm for the model. Based on this result, we expand our analysis in Chapter 5 to two-dimensional power fraction items, and three-dimensional unit and power fraction items. The results in Chapters 4 and 5 are published in [15], however most of the analysis of Chapter 5 was omitted in [15]. In Chapter 6 we present several lower bounds for the family of Any-Fit algorithms for multi-dimensional dynamic bin packing of general size, unit fraction, and power fraction items. The work in Chapter 6 represents a continuation of the results in [15]. Finally in Chapter 7 we give concluding remarks, including some partial results on the lower bound for the algorithm in Chapter 4. We also propose future directions for the work.

1.4 Author's Contribution

The main construction of the adversary in Chapter 3 was completed by two co-authors, Prudence W.H. Wong and Fencol C.C. Yung. The author contributed to the proof for a simpler 2.5-lower bound (the adversary and the optimal offline packing configuration), as well as contributing to the optimal offline packing configuration for the new 2.666-lower bound. In Chapters 4 and 5, the work was completed by the author with significant contributions from the same co-authors. Finally, in Chapter 6, the work was primarily completed by the author, with guidance from one co-author, Prudence W.H. Wong.

The work in this thesis is in part based on the following two publications, that have been expanded and adapted for the purpose of this thesis:

- Prudence W.H. Wong, Fencol C.C. Yung, and Mihai Burcea. An 8/3 Lower Bound for Online Dynamic Bin Packing. *Proceedings of the 23rd International Symposium on Algorithms and Computation (ISAAC)*, 2012, pp. 44–53 ([71]).

  The journal version is currently under submission to Algorithmica; and

- Mihai Burcea, Prudence W.H. Wong, and Fencol C.C. Yung. Online Multi-dimensional Dynamic Bin Packing of Unit-Fraction Items. *Proceedings of the 8th International Conference on Algorithms and Complexity (CIAC)*, 2013, pp. 85–96 ([15]).

  The journal version is currently in preparation.

1.5 Additional Contribution

The author has also contributed to an offline scheduling problem arising in demand response management [17, 60] in Smart Grid [42]. However, the work resulted in part from a collaboration with another Ph.D. student and it has been decided not to include the results in this thesis. For completeness, we give the definition of the problem and results obtained.

Problem definition. We consider an offline scheduling problem where the input consists of a set of unit-sized jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$. The time is divided into integral timeslots $T = \{1, 2, 3, \ldots, \tau\}$ and each job $J_i \in \mathcal{J}$ is associated with a set of feasible timeslots $I_i \subseteq T$, in which it can be scheduled. In this model, each job J_i must be assigned to exactly one feasible timeslot from I_i. The *load* $\ell(t)$ of a timeslot t represents the total number of jobs assigned to the timeslot. We consider a general convex cost function f that measures the cost used in each timeslot t based on the load at t. The total cost used is the sum of cost over time. Over all timeslots this is $\sum_{t \in T} f(\ell(t))$. The objective is to find an assignment of all jobs in $\mathcal{J}$ to feasible timeslots such that the total cost is minimized.

Our contribution. We propose a polynomial time offline algorithm that gives an optimal solution. We show that the time complexity of the algorithm is $\mathcal{O}(n^2\tau)$, where n is the number of jobs and τ is the number of timeslots. We further show that if the feasible timeslots for each job to be served form a contiguous interval, we can improve the time complexity to $\mathcal{O}(n \log \tau + \min(n, \tau) n \log n)$.

The interested reader is referred to the following publication.

- Mihai Burcea, Wing-Kai Hon, Hsiang-Hsuan Liu, Prudence W.H. Wong, and David K.Y. Yau. Scheduling for Electricity Cost in Smart Grid. *Proceedings of the 7th International Conference on Combinatorial Optimization and Applications (COCOA)*, 2013, pp. 306–317 ([14]).

  The journal version is currently in preparation.

Chapter 2

Bin Packing Problems

In this chapter we give a more detailed history of the bin packing problem and some of its variants. We first survey results on the offline bin packing problem in Section 2.1 before proceeding to online bin packing in Section 2.2 and ultimately online dynamic bin packing in Section 2.3. We also present variations on the bin packing problem and other metrics to measure the performance of online algorithms in Section 2.4.

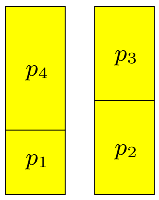
We also give a flavour of the results addressed in this thesis. Specifically, we give simple upper and lower bounds for the First-Fit algorithm in the context of online dynamic bin packing (Section 2.3). This serves to introduce the reader to the more complex analysis in later chapters, where several algorithms are analyzed.

2.1 The Classical Bin Packing Problem

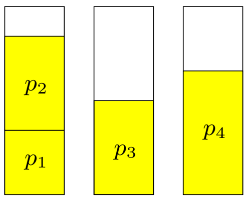
In the classical bin packing problem [53] we are given an infinite supply of uniform-sized bins and a list of items, each of which has size no larger than the common bin capacity. The goal of the optimization problem is to pack the items into a minimum number of bins, such that the sum of all items in a bin does not overflow the bin capacity. There are many applications of bin packing such as allocating memory in computers [20], assigning commercials to station breaks in television programming [13], and packing and cutting stock applications [2].

Due to the fact that the bin packing problem is NP-hard [45], research has been devoted to developing approximation algorithms for the offline version of the problem and online algorithms for its online counterpart. In fact, the research on bin packing has played an important role in the formation of complexity theory and in both the combinatorial and average-case analysis of approximation and online algorithms [27, 23]. Approximation algorithms (online algorithms, respectively) are sub-optimal offline algorithms (online algorithms, respectively) that guarantee worst-case performance within a factor of the optimal offline solution. For the bin packing problem, approximation and online algorithms that run in polynomial time are generally sought.

In Sections 1.1.2 and 1.1.3 we have defined the approximation ratio and the competitive ratio for approximation and online algorithms, respectively. For bin packing,



(a) The optimal solution requires two bins, by packing p_1 and p_4 in the first bin, and p_2 and p_3 in the second bin.



(b) The First-Fit algorithm requires three bins. p_1 and p_2 are packed in the first bin, p_3 is packed in the second bin, and p_4 is packed in the third bin.

Figure 2.1: (a) The optimal solution and (b) the First-Fit solution for item list L. The example shows items being packed along the height of a one-dimensional bin.

where input sequences may be arbitrarily long, we use the asymptotic approximation ratio and asymptotic competitive ratio as a metric for measuring the performance of approximation and online algorithms, respectively. We usually refer to the asymptotic approximation (competitive, respectively) ratio simply as the approximation (competitive, respectively) ratio when discussing worst-case performance of algorithms, unless both absolute and asymptotic ratios are relevant to the discussion.

Throughout the thesis we assume without loss of generality that each bin has a capacity of 1 and items have sizes that are real numbers in $(0, 1]$. To illustrate the classical bin packing problem, suppose we have an input list L consisting of four items. Let $L = (p_1, p_2, p_3, p_4)$ and let s_i denote the size of p_i. Suppose $s_1 = 1/3, s_2 = s_3 = 1/2$, and $s_4 = 2/3$. It is clear that an optimal solution requires at least $\lceil(s_1 + s_2 + s_3 + s_4)\rceil = 2$ bins; Figure 2.1(a) shows such an optimal solution. Nonetheless, it is not so straightforward how to produce an optimal solution. Consider the First-Fit (FF) algorithm. When a new item arrives, if there are occupied bins in which the item can be repacked, FF assigns the new item to the bin which has been occupied for the longest time. If no existing bin can accommodate the item, FF opens a new bin and packs the item there. For the item list L, FF requires three bins as shown in Figure 2.1(b). Note that FF greedily packs the items in order of their indices.

A better approximation algorithm would make use of the information of the input sequence, i.e., the item sizes. In order to obtain an improvement over First-Fit, algorithms that pack larger items first would be desirable. This is because we want to minimize the "wasted" space in a bin. A natural modification of First-Fit is the First-Fit Decreasing (FFD) algorithm that first sorts the items sizes in non-increasing order and then packs them according to the First-Fit rule. For the example list L given above FFD produces an optimal solution.

Xia and Tan [72] showed that, for any input, FF requires at most $1.7OPT + 0.7$ bins, where OPT denotes the minimum number of bins required by an optimal offline algorithm. Dósa and Sgall [35] improved the approximation ratio to exactly 1.7, thus the asymptotic approximation ratio and the absolute competitive ratio are indeed equal for FF. They also give a lower bound of 1.7 on the absolute approximation ratio of FF, thus

	1-D	2-D	3-D
Approximation ratio	1.184 [54]	1.525 [5]	2.8595 [16]
Admits APTAS?	Yes [43]	No [6]	No [6]
Admits PTAS?	No [3]	No [3]	No [3]

Table 2.1: Asymptotic approximation ratios, and APTAS and PTAS admissibility for offline bin packing.

the bound is shown to be tight. On the other hand, FFD originally analyzed by Johnson et al. [52, 53] has a much better asymptotic approximation ratio of $11/9 \sim 1.223$, while its absolute approximation was shown by Dósa et al. [34] to be $4/3 \sim 1.334$. The current best approximation algorithm was proposed by Johnson and Garey [54]. The algorithm is called Modified First-Fit Decreasing (MFFD) and is a modification based on FFD. The main difference from FFD is that MFFD attempts to packs items with sizes in $(1/6, 1/3]$ in bins containing items with size larger than $1/2$. The asymptotic approximation ratio is proven to be $71/60 \sim 1.184$.

There are of course many other offline algorithms for bin packing, including asymptotic polynomial-time approximation schemes [45] denoted by APTAS. An APTAS produces an approximation algorithm that uses $(1 + \epsilon)OPT + b$ bins, for a small $\epsilon > 0$, OPT being the number of bins used by an optimal offline algorithm, and b a positive constant, that has a running time that is polynomial in $1/\epsilon$. The bin packing problem is shown to admit an asymptotic PTAS in [43]. If the additive constant is not positive, there is no PTAS for the bin packing problem [3]. A recent survey [23] covers a wide range of results for classical bin packing algorithms.

There are multi-dimensional generalizations of the classical bin packing problem, where items have lengths in $(0, 1]$ in each dimension and must be packed in multi-dimensional unit-size bins. The best approximation ratio for two-dimensional bin packing is 1.525 [5], while for three-dimensional bin packing it is 2.8595 [16]. Bansal et al. [6] showed that even the two-dimensional problem does not admit an APTAS unless P=NP. Multi-dimensional generalizations of the classical bin packing problem are surveyed in [39]. Table 2.1 gives a summary of the best results for bin packing approximation algorithms. In the next sections we will also survey existing results for the online multi-dimensional model before discussing the online multi-dimensional dynamic setting.

2.2 Online Bin Packing

In the previous section, we have seen the disadvantage of the First-Fit (FF) algorithm against the First-Fit Decreasing (FFD) algorithm. FFD uses knowledge of the input and sorts the items according to their sizes. FF on the other hand does not use knowledge of the whole input and thus attains a worse approximation ratio. The challenge that online algorithms for the bin packing problem face is similar in this sense, as each item

		1-D	2-D	3-D
Upper bound	Static	1.58889 [63]	2.5545 [46]	4.3198 [46]
	Dynamic	2.788 [26]	7.788 [70]	22.788 [70]
Lower bound	Static	1.54037 [4]	1.907 [9]	2.111 [9]
	Dynamic	2.666 [*] [71]	3.70301 [37]	4.85383 [37]

Table 2.2: Asymptotic competitive ratios and lower bound results for online static bin packing and online dynamic bin packing of general size items. General size items have lengths that are real numbers in $(0, 1]$. [*] This result is also presented in Chapter 3.

is received one at a time, without knowledge of future input, and an online algorithm must make an irrevocable decision on where to pack the current item.

FF can be applied to the online (static) bin packing problem and its approximation ratio directly translates into a tight 1.7-competitive ratio based on the result of [35]. There are many [23] online algorithms for bin packing. The current best algorithm, Harmonic++ [63] has a competitive ratio of 1.58889. The algorithm works by defining types for the items based on their sizes and for each type the items are packed in dedicated bins which contain only items of one specific type. Harmonic++ defines 70 types of items, i.e., there are 70 disjoint subintervals of $(0, 1]$ which together fully cover $(0, 1]$. The lower bound for this algorithm, which belongs to the class of Harmonic algorithms [68], was already established to be $19/12 \sim 1.58333$ [61]. The lower bound for any algorithm for online bin packing currently stands at 1.54037 [4].

In multi-dimensional online (static) bin packing, the bins have capacity 1 in each dimension and items have lengths in $(0, 1]$ in each dimension. In this thesis, we study the model where items are oriented and cannot be rotated. For two-dimensional online bin packing, the current best upper bound of 2.5545 is due to Han et al. [46], and they achieve this result by using a Harmonic-type algorithm. The best lower bound known is 1.907 [9]. Han et al. [46] also mentioned that their result can be applied to three-dimensional online bin packing and this gives an algorithm with competitive ratio 4.3198, improving the previous upper bound of 4.8354 [38]. The lower bound for any online algorithm for three-dimensional bin packing is 2.111 [9]. A more detailed survey by Epstein and van Stee [39] covers offline and online results for multi-dimensional bin packing. Table 2.2 gives a summary of the best results for online static bin packing and also provides a comparison with the results for online dynamic bin packing. It is interesting to note that Harmonic-type approaches currently give the best algorithms for online static bin packing. Unfortunately, some of these approaches may not be suitable for online *dynamic* bin packing, as we will discuss in the next section.

2.3 Online Dynamic Bin Packing

Online dynamic bin packing is a natural generalization of the online bin packing problem. In potential applications such as warehouse storage and dynamic storage allocation in computers, the more realistic model of dynamic bin packing is required. This natural generalization was introduced by Coffman, Garey and Johnson [26]. In online dynamic bin packing, items arrive over time, reside for some period of time, and may depart at arbitrary time. Each item has to be assigned to a bin from the time it arrives until it departs. The objective is to minimize the maximum number of bins used over all time. Note that migration to another bin is not allowed, but rearrangement of items within a bin is permitted. In the online setting, the size and arrival time is only known when an item arrives and the departure time is only known when the item departs. For online dynamic bin packing, as usual, we measure the performance of an online algorithm using competitive analysis [11]. The subtle difference from online bin packing is that the maximum number of bins used by an online algorithm $\mathcal{A}$ and an optimal offline algorithm OPT are measured over *all time*. More formally, for any input I, we say that $\mathcal{A}$ is c-competitive if there exists a constant b such that $\mathcal{A}(I) \leq c\,OPT(I) + b$ for all I, where $\mathcal{A}(I)$ and $OPT(I)$ are the maximum number of bins used by $\mathcal{A}$ and OPT, respectively, over *all time*.

In the remainder of this section, we will discuss the one-dimensional dynamic bin packing problem, showing how to obtain simple bounds for the First-Fit algorithm, before proceeding to online multi-dimensional dynamic bin packing where we describe previous algorithms by Epstein and Levy [37]. These discussions serve to introduce the reader to the more complex analysis in later chapters, where several algorithms are analyzed. State of the art results for online dynamic bin packing are summarized in Table 2.2, where they are compared against results for online static bin packing. It is interesting to note that the lower bounds for any algorithm for online dynamic bin packing exceed the upper bound results for online static bin packing, making online dynamic bin packing "harder" than its static counterpart.

2.3.1 Online one-dimensional dynamic bin packing

Coffman, Garey, and Johnson [26] initiated the study of the online one-dimensional dynamic bin packing problem. Among their results, they analyze the worst-case performance of the First-Fit (FF) algorithm (see Section 2.1 for a description) and provide lower bounds for FF and any online algorithm. They showed that FF is 2.897-competitive and gave a lower bound for any online algorithm of 2.388. They also gave a modified version of First-Fit (FFM) that classifies items into two classes, items with size larger than 1/2 and smaller or equal to 1/2, and packs items of the first class in a dedicated bin and items of the second class according to the First-Fit rule. They show that FFM is 2.788-competitive.

To introduce the reader to the more complex analysis of bin packing algorithms in later chapters, we give a trivial method of proving a competitive ratio of 3 for FF, shown in [26]. We also show how a simple lower bound can be constructed for FF.

An upper bound of 3 for First-Fit ([26]). As noted in [26], to analyze the performance of FF we can restrict our attention to any input satisfying the following two conditions:

(i) FF uses the maximum number of bins when the last item is packed and not before.

(ii) No occupied bin ever becomes empty during the execution of FF on input sequences satisfying (i). 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. FF will work out the same final packing and the maximum number of bins used will remain unchanged.

Let $\mathrm{FF}(I)$ and $\mathrm{OPT}(I)$ denote the number of bins used by First-Fit and an optimal offline algorithm OPT, respectively, on any input I satisfying the above two conditions. Notice that by condition (ii) we can label the bins in increasing order and their labels do not change. The lowest index bin represents the longest occupied bin that FF uses and the highest index bin represents the latest occupied bin used by FF. We refer to bin B_i as the i-th bin opened by FF. The *load* of a bin B_i, denoted by ℓ_i, represents the sum of sizes of items that have been packed in B_i. Similarly, the total load $\mathcal{L}$ of all bins represents the total sum of loads of all bins. It is clear that an optimal solution requires at least $\lceil \mathcal{L} \rceil$ bins.

Coffman et al. give the following two observations that are sufficient to prove the upper bound of 3. First, any item that is packed in bin B_j with $j > 2\,\mathrm{OPT}(I)$ must have size greater than 1/2. Otherwise, if any item smaller than 1/2 is packed in B_j, the load of all bins before B_j must exceed 1/2, i.e., $\ell_i > 1/2$, for all $i < j$, and thus the total load of all bins used by FF exceeds $2\,\mathrm{OPT}(I) \times 1/2$. In this case, OPT would require more than $\mathrm{OPT}(I)$ bins to pack the load, which is a contradiction. The second observation is that there are at most $\mathrm{OPT}(I)$ items with size larger than 1/2, since more than that number requires more than $\mathrm{OPT}(I)$ bins. Then the upper bound of 3 follows.

Although the analysis presented above is simple, it contains important arguments when analyzing the performance of bin packing algorithms. We first want to ensure that when a new bin is opened, we can obtain a bound on the load of all other bins. Additionally, we want to deduce the sizes of items beyond a certain point in the packing configuration. Our approach in Chapters 4 and 5, albeit more complex, follows this line of arguments.

A lower bound of 2.25 for First-Fit. From an adversarial point of view, the lower bound construction aims to maximize the "wasted" space in a bin. For FF, we show how to obtain a simple 2.25 lower bound for online dynamic bin packing. Generally, lower bounds for dynamic bin packing consist of multiple stages and in each stage there may be item arrival and departure. Let k be an arbitrarily large integer and $\epsilon = 1/k$. We aim to show that FF uses $2.25k$ bins, while the optimal offline algorithm OPT uses $k+1$ bins. Then the competitive ratio of FF is at least 2.25. During the adversarial sequence, we keep the total load of items released and not departed to at most $k+1$.

In Stage 0, we release k/ϵ items of size ϵ and FF uses k bins. In Stage 1, we first depart a load of $1/2 - \epsilon$ of ϵ-sized items from each of the k bins opened in the previous stage. Afterwards, we have k bins, each packed with a load $1/2 + \epsilon$ of ϵ-sized items. The

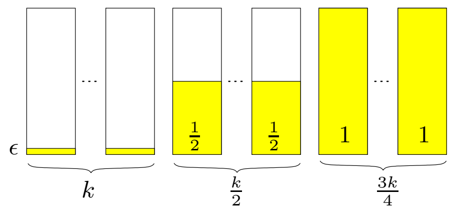


Figure 2.2: The final configuration of FF achieved by the adversary for a simple 2.25 lower bound.

total load in the packing at this point becomes $k \times (1/2 + \epsilon) = k/2 + 1$. We now release k items of size $1/2$, accounting for a load of $k/2$. No item can fit in any of the opened k bins, and FF packs them in $k/2$ new bins, two items in each bin. In Stage 2, we depart only one $1/2$-sized item from each of the $k/2$ bins opened in Stage 1. Additionally, we depart a load of $1/2$ of ϵ-sized items from each of the first k bins. We are left with k bins each packed with one ϵ-sized item and $k/2$ bins each packed with a $1/2$-sized item, with a total load of $k/4 + 1$. We can finally release $3k/4$ items of size 1. FF packs each item in a dedicated bin, thus the total number of bins used by FF is $9k/4 = 2.25k$. Figure 2.2 shows the final configuration of the packing of FF.

What remains to show is that there exists an optimal offline algorithm OPT that can use at any time no more than $k + 1$ bins. In Stage 0, all ϵ-sized items in the adversary that never depart are packed in one bin by OPT. The remaining ϵ-sized items are packed as follows. The items that will depart in Stage 1 are packed in $k/2 - 1$ bins and the ϵ-sized items that depart in Stage 2 are packed in $k/2$ bins. OPT uses k bins in Stage 0. At the beginning of Stage 1, $k/2 - 1$ bins are emptied. OPT packs the new $1/2$-sized items that never depart in $k/4$ of the emptied bins, and the remaining items in $k/4$ bins, opening only one additional new bin. In total OPT uses $k + 1$ bins. At the beginning of Stage 2, $k/4$ bins containing $1/2$-sized items are emptied and $k/2$ bins containing ϵ-sized items are emptied. The new $3k/4$ items of size 1 are packed in the empty bins and at the end OPT uses $k + 1$ bins. The lower bound of 2.25 then follows.

We have noted that in order to minimize the wasted space in a bin, an algorithm's aim is to pack larger items first. It is natural that adversaries used to construct lower bounds should present an algorithm with smaller items first and attempt to maximize the wasted space in a bin. Another technique of the adversary is to "block" items arriving at a later time, by leaving a bin packed with sufficient items, e.g., at the beginning of Stage 1 we have k bins each packed with a load of $1/2 + \epsilon$ and items of size $1/2$ later arrive. This prevents the new items from being packed in existing bins and forces FF to open additional bins. The construction of the lower bound becomes more complex as we attempt to reach the upper bound of an algorithm, especially if our aim is to prove a lower bound for *any* online algorithm (e.g., it also involves case analysis). It is also important to show that there exists an optimal offline algorithm OPT that can pack the input sequence into a desired number of bins, without allowing migration of items between bins at any time. OPT must also not encounter any "fitting" issues, e.g., for some small $\epsilon > 0$, three items of size $1/2 + \epsilon$ (with total load $1.5 + 3\epsilon$) may not be packed in less than three bins, as any two items packed in the same bin would overflow the bin capacity. Our $8/3 \sim 2.666$ lower bound for any online algorithm in Chapter 3 is based on these observations and additionally a novel way of constructing the adversary, by using items of slightly different sizes and items with complementary sizes. Furthermore, Chapter 6 also gives adversaries for several online algorithms for multi-dimensional dynamic bin packing.

A note on Harmonic-type algorithms. Recall that Harmonic-type algorithms are used for online non-dynamic (static) bin packing. The algorithms define types for the items based on their sizes and for each type the items are packed in dedicated bins which contain only items of one specific type. We can see that for online dynamic bin packing such algorithms may not be suitable if the number of types defined is large. We can construct an adversary that releases items of one type at a time, then departs most of them from the packing, without emptying any bin, and proceeds to release items of a different type. Previously opened bins will not pack items of a different type, thus the wasted space in existing bins can be maximized. Nonetheless, for online multi-dimensional dynamic bin packing we use in Chapters 4 and 5 a classification of items based on their lengths (three and four classes for two- and three-dimensional dynamic bin packing, respectively) in order to get constant competitive algorithms. In fact, without any classification we show that, amongst other algorithms, First-Fit is not competitive for multi-dimensional dynamic bin packing in Chapter 6. One of the aims in this thesis is also to strike a balance between the number of classes used and the performance of an algorithm. Indeed, the number of classes we define in Chapters 4 and 5 gives algorithms with good competitive ratios.

Before proceeding to online multi-dimensional dynamic bin packing, we briefly discuss special types of input for the bin packing problem, in particular unit fraction and power fraction items, for which we initiate the study in the context of online multi-dimensional dynamic bin packing in Chapters 4 and 5.

Unit fraction and power fraction items. Bar-Noy et al. [8] initiated the study of the *unit fraction bin packing problem*, a restricted version where all sizes of items are of the form $\frac{1}{k}$, for some integer $k > 0$. The problem was motivated by the windows scheduling problem [7, 8]. Another related problem is for *power fraction* items, where sizes are of the form $\frac{1}{2^k}$, for some integer $k \geq 0$. Bin packing with other restricted form of item sizes includes divisible item sizes [25] (where each possible item size can be divided by the next smaller item size) and discrete item sizes [22] (where possible item sizes are $\{1/k, 2/k, \cdots, j/k\}$ for some $1 \leq j \leq k$). For d-dimensional packing, for $d \geq 2$, items of restricted form have been considered, e.g., [51] considered strip packing ([67]) of items with one of the dimensions having discrete sizes and [59] considered bin packing of items where the lengths of each dimension are at most $1/m$, for some integer m. The study of these problems is motivated by applications in job scheduling.

Clearly items that have lengths real numbers in $(0, 1]$, which we call *general size* items, form a superset of unit fraction and power fraction items, and algorithms for general size items obtain the same worst-case performance for unit fraction and power fraction items. However, algorithms that exploit unit fraction input can obtain better worst-case performance for unit fraction items. For online dynamic bin packing, Chan et al. [18] obtained a competitive ratio of 2.4942 for First-Fit, which was recently improved by Han et al. to 2.4842 [47], while the lower bound for First-Fit was proven to be 2.45 [18]. The lower bound for any algorithm is 2.428 [18]. As FF is 2.4842-competitive for unit fraction items and the lower bound for any algorithm for online dynamic bin packing of general size items is 2.666 (Chapter 3 and [71]), online dynamic bin packing of unit fraction items is indeed "easier" than packing general size items. We also expect better competitive ratios for online multi-dimensional dynamic bin packing algorithms of unit fraction and power fraction items compared to packing general size items. This is in fact the case as discussed in Chapters 4 and 5. Table 2.3 shows a comparison between competitive ratios of general size, unit fraction, and power fraction items.

2.3.2 Online multi-dimensional dynamic bin packing

In multi-dimensional bin packing the bins have unit size in each dimension and items have lengths that are real numbers in $(0, 1]$ in each dimension. For the two-dimensional (2-D) problem the input consists of a set of rectangles that must be packed in unit squares. Similarly, for three-dimensional (3-D) bin packing, boxes must be packed in unit cubes. Recall that we study the setting where items are oriented and cannot be rotated. As for online dynamic bin packing, migration of items between bins is not allowed, while rearrangement within a bin is permitted.

Epstein and Levy [37] initiated the study of online multi-dimensional dynamic bin packing. For 2-D, they give a 8.5754-competitive algorithm and a lower bound for any online algorithm of 3.70301. For 3-D, they present a 35.346-competitive algorithm and a lower bound for any online algorithm of 4.85383. In the construction of the lower bounds only items with lengths in each dimension being unit fraction ($1/k$, for k integer) are used and the lower bounds also apply to online multi-dimensional dynamic bin packing of unit fraction items. They also address d-dimensional bin packing of boxes, for $d \geq 2$, and give an algorithm with a $2 \cdot 3.5^d$ competitive ratio and a lower bound of $d + 1$.

We will give a brief overview of the 2-D algorithms in [37] for packing rectangles and afterwards compare them to the best-in-class results by Wong and Yung [70]. The overview and comparison should also help the reader observe some of the common arguments for getting competitive algorithms for online multi-dimensional dynamic bin packing, arguments that are also used in bin packing of 2-D unit fraction items in Chapter 4, and 2-D power fraction items and 3-D unit and power fraction items in Chapter 5.

Online two-dimensional dynamic bin packing. First, let us consider the Next-Fit (NF) algorithm. After packing the first item, NF packs each successive item in the bin containing the last item that has been packed, if it fits in that bin; if it does not fit, NF closes that bin and packs the current item in an empty bin. On its own, NF is not suitable for dynamic bin packing, due to the fact that a bin cannot be reopened after the current item does not fit in the bin. Some items may depart from the closed bin, leaving the bin almost empty, and the bin cannot be reused for subsequent items. Nevertheless, a two-dimensional variant of the algorithm, the Next-Fit Decreasing Height (NFDH) algorithm is used originally for offline bin packing and is adapted to the online dynamic model in [37]. NFDH is a level (shelf) algorithm which uses the next-fit approach to pack the sorted (by non-increasing height) list of rectangles. A level algorithm creates levels by drawing horizontal lines of length 1, and placing rectangles between consecutive pairs of lines, including the bottom and top of the bin. The two lines which form a level are called the "floor" and "ceiling" of the level or the "bottom" and "top" of the level. The additional lines are typically created online starting from the bottom, where the "floor" of the first level is the bottom of the bin. The rectangles are packed, left-justified on the floor of a level until the next rectangle does not fit. This rectangle is used to define the height of a new level and the packing continues on this level. A new level is defined just above the previous level. If a new level of the given height cannot be opened, a new bin is opened and the rectangle defines the first level of that bin. As opposed to the algorithm First-Fit Decreasing Height (FFDH), earlier levels are not revisited. The levels are numbered from bottom to top, according to the order they are created.

NFDH can be used in the online dynamic setting when a new item p_i arrives in the following way. When p_i arrives, the algorithm attempts to pack it in the smallest index bin in which it is possible. p_i is merged into the sorted list of items already packed in the bin. Then NFDH uses the merged list of items to pack items according to its description. If the merged list can fit in a single bin, then the item is packed there. Otherwise, i.e., the merged list cannot fit in a single bin, we revert to the old configuration and NFDH attempts to pack p_i to the next bin. If none of the existing bins can fit the item, NFDH packs p_i to a new bin. Note that NFDH can be used in the online dynamic setting as it only uses information about items already arrived and not departed in the packing. Epstein and Levy use NFDH for one class of items that satisfies some constraints on the item lengths, while the other items are packed in a different way as we will see next.

For online two-dimensional dynamic bin packing, Epstein and Levy give an algorithm called Rectangle Packing (RCP) that classifies the items based on their width and height. Let w_i and h_i denote the width and height, respectively, of item p_i. Then p_i belongs to one of the following four disjoint classes, which together cover all possible input.

Class 1 if $w_i > 1/2$ and $h_i > 1/2$;

Class 2 if $w_i > 1/2$ and $h_i \leq 1/2$;

Class 3 if $w_i \leq 1/2$ and $h_i > 1/2$; or

Class 4 if $w_i \leq 1/2$ and $h_i \leq 1/2$.

RCP packs the items in each class separately, independent of other classes, by using sub-algorithms for each particular class. For each class of items, we can obtain a competitive ratio by comparing the sub-algorithm's performance to an optimal offline algorithm that also packs items in each class separately, independent of other classes. Then, the overall competitive ratio of RCP is no larger than the sum of the competitive ratios of the composed sub-algorithms for each class. Class 1 items are each packed in a separate bin as no two items can fit in the same bin. As Class 2 items have width greater than 1/2, no two items can fit along the width of a bin, and we can use the concept of *projection* of a higher dimension item to a lower dimension item. We project along the width of the items and the problem is reduced to 1-D, for which First-Fit (FF) is used. A similar argument is used for Class 3 items. Finally, Class 4 items are packed using the online dynamic version of NFDH described above.

It is easy to see that Class 1 items achieve a competitive ratio of 1, as no two items of the same class can fit in a single bin. By using a parametric version of FF, where items have size at most $1/k$, for some integer $k \geq 2$, Classes 2 and 3 both achieve a competitive ratio of 1.787682 for $k = 2$. The analysis for parametric FF with general k is given by Coffman et al. [26]. For NFDH, Epstein and Levy prove that if items have both width and height bounded by $1/k$ and a new item p_i does not fit in any of the existing bins, then the total occupied area of each existing bin (except possibly the last one) is at least $(1 - 1/k)^2$. For $k = 2$, this results in Class 4 items achieving a competitive ratio of 4. Overall, the competitive ratio of RCP is 8.5754.

The techniques of classifying items based on their lengths, repacking a list of items in a bin using a specific approach, parametric versions of algorithms, projection of a higher dimension to a lower dimension, and lower bounding the total area that items occupy in a bin are all used in designing and analyzing the performance of online algorithms for multi-dimensional dynamic bin packing. We also employ these techniques in Chapters 4 and 5. In these chapters we will in fact repack a list of items satisfying certain conditions using Steinberg's algorithm [66] for offline two-dimensional strip packing [39, 67].

Strip packing. In strip packing, a strip of finite width 1 and infinite height is given. The input consists of a list of rectangles (items) $R_1, R_2, \ldots, R_n$, and each item R_i has a given width w_i and height h_i. The goal is to pack the items into the strip such that the height to which the strip is filled is minimized. As in bin packing, the items cannot overlap and rotation of items is not permitted. Steinberg's algorithm [66] classifies items based on their lengths into seven classes and for each class they divide it into two smaller sub-classes and solve the problems recursively. Roughly speaking, the approach of the algorithm is to pack a list of items belonging to a particular class into a rectangle Q of width u and height v. Steinberg proves that if a list of rectangles is given such that their lengths and their total area satisfy certain conditions with regards to u and v, then the list may be packed in the rectangle Q, using some defined packing procedures for each class. Note that classes are defined based on the input, as the maximum item height can be arbitrarily large. Steinberg's algorithm is an approximation algorithm with

	1-D	2-D	3-D
General size	2.788 [26]	7.788 [70]	22.788 [70]
Unit fraction	2.4842 [47]	6.7850 [*] [15]	21.6108 [**] [15]
Power fraction	2.4842 [47]	6.2455 [**] [15]	20.0783 [**] [15]

Table 2.3: Competitive ratios for general size, unit fraction, and power fraction items. Results also presented in Chapters 4 and 5 are marked with "[*]" and "[**], respectively.

an absolute approximation ratio of 2. The current best result for offline strip packing with respect to the absolute approximation ratio is an algorithm based on Steinberg's algorithm that is $(5/3+\epsilon)$-approximate [48], for any $\epsilon > 0$.

In Chapters 4 and 5 we will use the following lemma implied by the results in [66] in order to be able to lower bound the total area that existing items occupy in a bin when no new item can repacked in that bin. Note that the lemma holds for online dynamic bin packing, as we only consider the list of items already packed in the bin but not departed and items may be repacked within a bin each time a new item arrives.

Lemma 4.2 ([66]). *Given a bin with width u and height v, if all items have width at most $\frac{u}{2}$ and height at most v, then any set of these items with total area at most $\frac{uv}{2}$ can fit into the same bin by using Steinberg's algorithm.*

State of the art. The current best algorithms for online multi-dimensional dynamic bin packing of general size items are due to Wong and Yung [70]. For 2-D, the algorithm uses a classification of items into three classes based on their width alone and achieves a competitive ratio of 7.788 compared to the previous result of 8.5754 [37]. For 3-D, they use four classes of items and achieve a competitive ratio of 22.788, compared with the previous approach of using eight classes that achieves a competitive ratio of 35.346 [37]. For d-dimensional packing they also improve the competitive ratio to 3^d from $2 \cdot 3.5^d$ [37].

Recall that general size items have lengths that are real numbers in $(0, 1]$, unit fraction items have lengths of the form $1/k$, for some integer $k > 0$, and power fraction items have lengths of the form $1/2^k$, for some integer $k \geq 0$. In Chapter 4 we also classify 2-D unit fraction items into three classes, while 3-D unit fraction items in Chapter 5 are classified into four classes. The same number of classes are used for 2-D and 3-D power fraction items in Chapter 5. Although the number of classes is the same as in [70], the analysis in Chapters 4 and 5 is different and requires careful consideration of unit and power fraction items, which allows us to reduce the competitive ratios for these types of inputs. Table 2.3 shows a comparison between competitive ratios of general size, unit fraction, and power fraction items.

2.4 Variations on Bin Packing

There are many variations of bin packing that continue to attract researchers' attention (e.g., see surveys [31, 27, 24, 23]). We will first give a brief overview of related models, although the analysis of such models is generally different from online dynamic bin packing. Afterwards, we discuss several other models for bin packing. While they are not directly related to our problem, these models show the broad scope of bin packing. We conclude this chapter with other types of analysis that may be introduced for online dynamic bin packing in future work.

Fully dynamic bin packing. Ivkovic and Lloyd [49, 50] studied the *fully dynamic bin packing problem* which is a variant of dynamic bin packing that allows existing items to migrate between bins each time an item arrives or departs. For the one-dimensional model they gave a 1.25-competitive online algorithm, showing that the variant is much "easier" than dynamic bin packing, for which the lower bound currently stands at 2.666 (Chapter 3 and [71]). Such an advantage of migrating items seems to negate in part the optimal offline algorithm's OPT full knowledge of the input, even while allowing migration of items for OPT.

Rotation of items. For online multi-dimensional static (non-dynamic) bin packing, where items are permanent, Epstein and van Stee [40] consider items that can be rotated and give the best results to date for the non-oriented problem. For two-dimensional bin packing, they give an algorithm with a competitive ratio of 2.25, while for three-dimensional bin packing, they give a 4.5-competitive algorithm. It is unclear whether rotation alone may make the problem "easier" as the optimal offline algorithm also has the advantage of rotating items. It is interesting to note that while the upper bound for the 2-D oriented problem (rotation is not allowed) is 2.5545 [46], thus greater than for the non-oriented problem, the upper bound for the 3-D oriented problem is 4.3198 [46], lower than for the non-oriented problem[1]. Rotation of items has not yet been introduced for online dynamic bin packing. In future research, it may indeed provide us with a better understanding of the difficulty of such problems compared to oriented problems.

Resource augmentation. Resource augmentation [55] has also been studied for bin packing [32, 41, 19]. In this variant, the online algorithm uses bins of uniform sizes that are larger than the unit size bins used by the optimal offline algorithm. As expected, in this model the competitive ratios improve. Chan et al. [19] first investigate online one-dimensional dynamic bin packing and show that bins of size 2 are both necessary and sufficient in order for an online algorithm to achieve 1-competitiveness. For online two-dimensional dynamic bin packing, Wong and Yung [70] show that using bins of width 3 and height 1 is sufficient to achieve 1-competitiveness, while for d-dimensional bin packing, for $d \geq 2$, they show that using bins of size $\{2\}^d$ is both necessary and sufficient to achieve 1-competitiveness. This setting is yet another direction which may yield new results, especially for two- and three-dimensional online dynamic bin packing.

Other variants. Many more variants of bin packing have been studied and have themselves provided more insight to other problems, while contributing to the field

[1]The best results for the oriented problem are obtained later than for the non-oriented problem.

of complexity theory. We briefly note some interesting variants including semi-online algorithms, bounded-space bin packing, and strip packing.

In the context of bin packing, semi-online algorithms are algorithms that can migrate items between bins, look ahead to later items before assigning the current one, or assume some pre-ordering of items. In the case of fully dynamic bin packing [49, 50], an algorithm uses all the above operations and can achieve a competitive ratio of 1.25. Algorithms that can use only one operation amongst those mentioned above are surveyed in [23]. Semi-online algorithms attempt to bridge the gap between optimal offline algorithms that hold complete knowledge of the input and online algorithms that receive items one at a time. In fact, advice complexity analysis [10, 33] aims to strike a balance between information given to online algorithms and their performance.

Bounded-space bin packing assumes that only a constant number of bins k may be open to receive items at any time. The Next-Fit algorithm, discussed in Section 2.3.2, is in fact a 1-bounded space algorithm. The problem is motivated by applications such as loading trucks, where only a certain number of vehicles are available at any time. The recent survey [23] covers different results for different values of k.

Strip packing [39, 67] is a two-dimensional packing problem in which we are given a strip of a finite width w but infinite height, and a set of boxes each of width at most w. The objective is to pack all the items into the strip to minimize the height used. Cutting-stock problems [2], where material of fixed width needs to be cut while minimizing the height of the strip used is an application of strip packing. Other applications of strip packing are in scheduling problems, e.g., [51].

Metrics for performance of online algorithms. For online algorithms, the performance is usually measured using competitive analysis [11]. Using competitive analysis, we provide a worst-case guarantee for the performance of an online algorithm. While this guarantees the algorithm's performance on any input, other metrics are nevertheless useful. For online static bin packing, other benchmarks such as average case analysis ([58]) have been studied [22, 28]. The difference from worst-case analysis is that average case analysis relies on input distribution and thus may not consider every possible input. Furthermore, other types of analysis such as advice complexity analysis ([10, 33]), random order analysis, and relative worst order analysis ([12]) have been introduced to online static bin packing in [73], [56], and [36], respectively. In the future, interesting results may arise by introducing such performance metrics for online *dynamic* bin packing.

Chapter 3

An 8/3 Lower Bound for Online Dynamic Bin Packing

3.1 Introduction

In this chapter, we study the online one-dimensional dynamic bin packing problem, introduced by Coffman, Garey and Johnson [26]. In online dynamic bin packing, items with sizes real values in $(0, 1]$ arrive over time, reside for some period of time, and may depart at arbitrary time. Each item has to be assigned to a unit-size bin from the time it arrives until it departs. The objective is to minimize the maximum number of bins used over all time. Note that migration to another bin is not allowed, but rearrangement of items within a bin is permitted. In the online setting, the size and arrival time are only known when an item arrives and the departure time is only known when the item departs. Recall that for online dynamic bin packing, we measure the performance of an online algorithm using competitive analysis [11]. For any input I, we say that $\mathcal{A}$ is c-competitive if there exists a constant b such that $\mathcal{A}(I) \leq c\,OPT(I) + b$ for all I, where $\mathcal{A}(I)$ and $OPT(I)$ are the maximum number of bins used by $\mathcal{A}$ and OPT, respectively, over all time.

We present a novel adversary for the one-dimensional dynamic bin packing problem that allows us to improve the lower bound from 2.5 [19] to $8/3 \sim 2.666$. Previously, the first lower bound of 2.388 [26] was improved to 2.428 [18], and subsequently to the best known 2.5-lower bound [19]. On the other hand, the best upper bound result currently stands at 2.788 [26]. The results have been published in Proceedings of the 23rd International Symposium on Algorithms and Computation (ISAAC), 2012 [71].

Previous results. The online dynamic bin packing problem was first studied by Coffman, Garey and Johnson [26]. They showed that the First-Fit (FF) algorithm (described in Section 2.1) has a competitive ratio lying between 2.75 and 2.897, and a modified First-Fit algorithm is 2.788-competitive. The modified First-Fit algorithm classifies items into two classes, large items with size larger than 1/2 and small items with size smaller or equal to 1/2, and then packs each large item in a dedicated bin while small items are packed according to the First-Fit rule. While the 2.75-lower bound also applies to the modified First-Fit algorithm, it is worth mentioning that in the construction of the adversary the optimal offline algorithm was allowed to migrate items between bins. They also give a "stronger" lower bound of 2.388 on the competitive ratio of any online deterministic algorithm and moreover the optimal offline algorithm is not permitted migration of items between bins. The result was improved to 2.428 [18] and subsequently to the best known lower bound of 2.5 [19].

Our contribution. We improve the lower bound on the competitive ratio of any online algorithm from 2.5 [19] to $8/3 \sim 2.666$, reducing the gap between the upper and lower bounds by about 60% from 0.288 to 0.122. The improvement of the lower bound to $8/3 \sim 2.666$ stems from an adversarial sequence that forces an online algorithm $\mathcal{A}$ to open $2s$ bins with items having a total size of s only and this can be adapted appropriately regardless of the load of other bins that have already been opened by $\mathcal{A}$. Comparing with the previous 2.5-lower bound, this basic step gives a better use of items of slightly different sizes leading to a tighter lower bound. Furthermore, we show in Section 3.3.4 that the 2.5-lower bound can be obtained using this basic step in a much simpler way. It is worth mentioning that we consider optimal packing without migration at any time.

The adversary. The adversarial sequence is composed of two operations, namely Op-Inc and Op-Comp. Roughly speaking, Op-Inc uses a load of at most s to make $\mathcal{A}$ open s bins, this is followed by some item departure such that each bin is left with only one item and the size is increasing across the bins. Op-Comp then releases items of complementary size such that for each item of size x, items of size $1 - x$ are released. The complementary size ensures that the optimal offline algorithm $\mathcal{O}$ is able to pack all these items using s bins while the sequence of arrival ensures that $\mathcal{A}$ has to pack these complementary items into separate bins.

Organization of the chapter. In Section 3.2, we introduce the notation used throughout the chapter. In Section 3.3, we detail the operations Op-Inc and Op-Comp that are used to force an online algorithm $\mathcal{A}$ to open new bins. Sections 3.4 and 3.5 are dedicated to proving the simple and complex cases of the 8/3 lower bound, respectively. In Section 3.6, we detail the packing configuration of the optimal offline algorithm. Finally, we give some concluding remarks in Section 3.7.

3.2 Preliminaries

In dynamic bin packing, items arrive and depart at arbitrary time. Each item comes with a size. We denote by *s-item* an item of size s. When an item arrives, it must be assigned to a unit-sized bin immediately without exceeding the bin capacity. At any time, the *load* of a bin is the total size of items currently assigned to that bin that have not yet departed. We denote by *ℓ-bin* a bin of load ℓ. Migration is not allowed, i.e., once an item is assigned to a bin, it cannot be moved to another bin. This also applies to the optimal offline algorithm. The objective is to minimize the maximum number of bins used over all time.

When we discuss how items are packed, we use the following notations:

- Item configuration ψ: y_{*z} describes a load y with $\frac{y}{z}$ items of size z, e.g., $\frac{1}{2}_{*\epsilon}$ means a load $\frac{1}{2}$ with $\frac{1}{2\epsilon}$ items of size ϵ. We skip the subscript when $y = z$.

- Bin configuration π: $(\psi_1, \psi_2, \cdots)$, e.g., $(\frac{1}{3}, \frac{1}{2}_{*\epsilon})$ means a bin has a load of $\frac{5}{6}$, with a $\frac{1}{3}$-item and an addition load $\frac{1}{2}$ with ϵ-items. In some cases, it is clearer to state the bin configuration in other ways, e.g., $(\frac{1}{2}, \frac{1}{2})$, instead of $1_{*\frac{1}{2}}$. Similarly, we will use $6 \times \frac{1}{6}$ instead of $1_{*\frac{1}{6}}$.

- Packing configuration ρ: $\{x_1{:}\pi_1, x_2{:}\pi_2, \cdots\}$ a packing where there are x_1 bins with bin configuration π_1, x_2 bins with π_2, and so on. E.g., $\{2k{:}1_{*\epsilon}, k{:}(\frac{1}{3}, \frac{1}{2}_{*\epsilon})\}$ means $2k$ bins are each packed with load 1 with ϵ-items and another k bins are each packed with a $\frac{1}{3}$-item and an addition load $\frac{1}{2}$ with ϵ-items.

- It is sometimes more convenient to describe a packing as $x{:}f(i)$, for $1 \leq i \leq x$, which means that there are x bins with different load, one bin with load $f(i)$ for each i. E.g., $k{:}\frac{1}{2}{-}i\delta$, for $1 \leq i \leq k$, means that there are k bins and one bin with load $\frac{1}{2}{-}i\delta$ for each i.

3.3 Op-Inc and Op-Comp

In this section, we discuss a process that the adversary uses to force an online algorithm $\mathcal{A}$ to open new bins. We first give the main idea and discuss its difference from previous approach in [19]. The adversary in [19] releases items of the same size in each stage and uses items of complementary sizes in two different stages (i.e., the two sizes sum to one, e.g., $\frac{1}{3}$-item and $\frac{2}{3}$-item). On the other hand, the adversary in this chapter releases items of slightly different size in each stage and continues to use items of complementary sizes in different stages. To realize the advantage of the latter approach, let us consider the following scenario.

Suppose we have s bins each with a $\frac{1}{3}$-item. If we want to force $\mathcal{A}$ to open s new bins using items of size $\frac{2}{3}$, we may need to release $2s$ such items because each existing bin can pack one item. The maximum load in total including the $\frac{1}{3}$-items would be $s(1 + \frac{2}{3})$. On the other hand, if the original s bins have items of size from $\frac{1}{3}$, $\frac{1}{3}{+}\delta$, $\frac{1}{3}{+}2\delta$, $\cdots$, $\frac{1}{3}{+}(s{-}1)\delta$, for some small $\delta > 0$, we can force $\mathcal{A}$ to open s new bins with a smaller maximum load at any time as follows. First release items of size $\frac{2}{3}{-}(s{-}1)\delta$ until A opens a new bin. At most $s+1$ such items are required. We then let all items of size $\frac{2}{3}{-}(s{-}1)\delta$ depart except the last one packed in the new bin. Next we release items of size $\frac{2}{3}{-}(s{-}2)\delta$. Note that these items can only be packed into the first $s{-}1$ existing bins. Therefore, at most s items are required to force $\mathcal{A}$ to open a new bin. We can repeat this process, for $1 \le i \le s$, with no more than $s-i+2$ items of size $\frac{2}{3}-(s-i)\delta$ to force $\mathcal{A}$ to open a new bin. Note that at any time, the total load of all items not departed is at most $s+\frac{2}{3}$, versus $s(1+\frac{2}{3})$ in the former case. It is better for an adversary to use less load to force $\mathcal{A}$ to open the same number of bins.

Our adversary is based on the above observations. Two operations are designed, namely, Op-Inc and Op-Comp, and the details are given in Sections 3.3.1 and 3.3.2, respectively. Op-Inc aims to force $\mathcal{A}$ to open some bins each with one item (of size $< \frac{1}{2}$) and the size of items is strictly increasing. Op-Comp then bases on the items in the bins opened by Op-Inc and releases items of complementary size. The complementary size is to ensure that an item released in Op-Inc can be packed with a corresponding item released in Op-Comp into the same bin by an optimal offline algorithm. The adversary to be described in Sections 3.4 and 3.5 works in stages. A stage of Op-Inc is associated with a corresponding stage of Op-Comp, but not necessarily consecutive, e.g., in one of the cases, Op-Inc is in Stage 1 and the corresponding Op-Comp is in Stage 4.

3.3.1 Operation Op-Inc

The aim of Op-Inc is to make $\mathcal{A}$ open at least s more bins, for some $s > 0$, such that each new bin contains one item with item size increasing over the s bins.

Pre-condition. Consider any value $0 < x < \frac{1}{2}$. Let h be the number of x-items that can be packed in existing bins.

Items to be involved. The items to be released have size in the range $[x, x+\epsilon]$, for some small ϵ, such that $x+\epsilon < \frac{1}{2}$. A total of $h + \lfloor \frac{1}{x+\epsilon} \rfloor \times s$ items are to be released.

Outcome. $\mathcal{A}$ opens at least s more new bins with increasing load in each new bin and the load of current bins remains unchanged.

The adversary. The adversary releases items of size x, $x+\frac{\epsilon}{s}$, $x+\frac{2\epsilon}{s}$, $\cdots$. Let $z_i = x+\frac{i\epsilon}{s}$. In each step i, the adversary releases z_i-items until $\mathcal{A}$ opens a new bin. We stop releasing items when $h+\lfloor \frac{1}{x+\epsilon} \rfloor \times s$ items have been released in total in a single step. By definitions of h, s, and x, $\mathcal{A}$ would have opened at least s new bins. We then let z_i-items depart except exactly one item of size z_i, for $0 \le i < s$, in the i-th new bin opened by $\mathcal{A}$.

Using Op-Inc. When we use Op-Inc later, we simply describe it as Op-Inc releasing at most $h + \lfloor \frac{s}{x} \rfloor$ items with the understanding that it works in phases and that items may depart at the end.

Remark. To simplify discussion, we denote the packing configuration of the new bins as $\{s{:}x + \frac{i\epsilon}{s}\}$ to mean that the s bins have different load with different value of i. Note that in some cases of the lower bound, items released in the very last step of Op-Inc do not depart the packing configuration unless explicitly mentioned.

3.3.2 Operation Op-Comp

Op-Comp is designed to work with Op-Inc and assumes that there are s existing bins each with load in the range $[x, y]$ where $x < y < \frac{1}{2}$. The outcome of Op-Comp is that $\mathcal{A}$

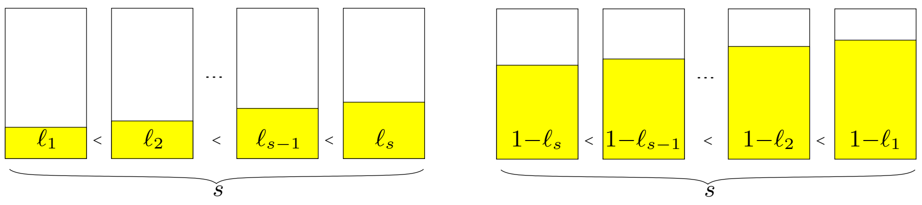


Figure 3.1: Op-Comp: Assuming $h = 0$. The s bins on the left are bins created by Op-Inc. The s new bins on the right are due to Op-Comp. Note that each existing item has a complementary new item such that the sum of sizes is 1.

opens s more bins. Figure 3.1 gives an illustration.

Pre-condition. Consider two values $x < y < \frac{1}{2}$. Suppose $\mathcal{A}$ uses s bins with load $x = \ell_1 < \ell_2 < \cdots < \ell_s = y$. Let $\ell = \sum_{1 \le i \le s} \ell_i$. Furthermore, suppose there are some additional bins with load smaller than x. Let h be the number of $(1-y)$-items that can be packed in other existing bins with load less than x.

Items to be involved. The items to be released have size in the range $[1 - y, 1 - x]$. Note that $1-x > 1-y > \frac{1}{2}$. In each step i, for $1 \le i \le s$, the number of $(1-\ell_{s+1-i})$-items released is at most $h + s + 2 - i$.

Outcome. $\mathcal{A}$ opens s more bins, each with an item $1 - \ell_{s+1-i}$, for $1 \le i \le s$, and the load of current bins remains unchanged.

The adversary. Starting from the largest load ℓ_s, we release items of size $1-\ell_s$ until $\mathcal{A}$ opens a new bin. At most $h + s + 1$ items are needed. Then we let all $(1-\ell_s)$-items depart except the one packed in the new bin. In general, in Step i, for $1 \le i \le s$, we release items of size $1-\ell_{s+1-i}$ until $\mathcal{A}$ opens a new bin. Note that such items can only be packed in the first $s + 1 - i$ bins and so at most $h + s + 2 - i$ items are required to force $\mathcal{A}$ to open another bin. We then let all $(1-\ell_{s+1-i})$-items depart except the one packed in the new bin.

Using Op-Comp. Similar to Op-Inc, when we use Op-Comp later, we describe it as Op-Comp with h and s and the understanding is that it works in phases and there are items released and departure in between. Note that the ℓ_i- and $(1 - \ell_i)$-items are complementary and the optimal offline algorithm would pack each pair of complementary items in the same bin.

3.3.3 Using Op-Inc and Op-Comp

In order to illustrate the use of Op-Inc and Op-Comp we will use the operations to force an online algorithm $\mathcal{A}$ to open $2s$ bins, for an arbitrarily large s using only a total load smaller than $s + \frac{2}{3} = s + O(1)$. The adversary is similar in construction to the example provided in the beginning of Section 3.3, except we now explicitly specify the parameters for Op-Inc and Op-Comp. Op-Inc will use items of slightly increasing sizes from $\frac{1}{3}$, while Op-Comp will use items of complementary sizes to the ones released by Op-Inc. Let s be an arbitrarily large integer and $\epsilon = \frac{1}{s}$. In the first part of our example, we aim to open s bins using Op-Inc.

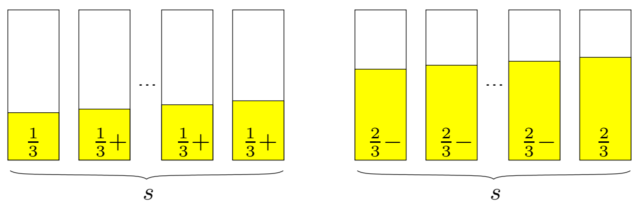
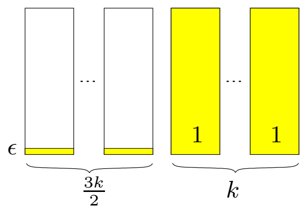


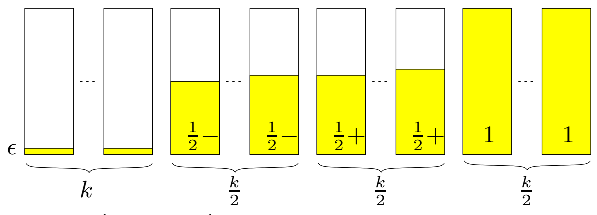
Figure 3.2: The s bins on the left are bins created by Op-Inc. The s bins on the right are due to Op-Comp. $\frac{1}{3}+$ and $\frac{2}{3}-$ denote items with increasing sizes larger than $\frac{1}{3}$ and smaller than $\frac{2}{3}$, respectively. Each item on the left has a complementary item on the right such that the sum of sizes is 1.

Op-Inc. We use Op-Inc to release items of size in $[\frac{1}{3}, \frac{1}{3}+\frac{(s-1)\epsilon}{s}]$, i.e., $x = \frac{1}{3}$. Note that there are no existing opened bins, thus no new items are packed in existing bins, i.e., $h = 0$. We have s steps, and in each step $0 \leq i < s$, Op-Inc releases identical items of size $z_i = \frac{1}{3}+\frac{i\epsilon}{s}$ until a new bin is opened by $\mathcal{A}$. Afterwards, we depart all z_i-items released except the one packed in the new bin by $\mathcal{A}$. In Step 0, one item of size $\frac{1}{3}$ is released and $\mathcal{A}$ packs it in a new bin. As there are no additional items of size z_0, except the one packed in the new bin, we proceed to Step 1 without departure of items. In Step 1, we release two items of size $\frac{1}{3}+\frac{1\epsilon}{s}$. As no more than two items of size $\frac{1}{3}+\frac{i\epsilon}{s}$, for $i \geq 1$, can be packed in the same bin, $\mathcal{A}$ opens at least one new bin. We now keep only the z_1-item that opened a new bin and depart the other z_1-item, regardless of where $\mathcal{A}$ packed it. In general, in Step i, we release $i+1$ items of size $z_i = \frac{1}{3}+\frac{i\epsilon}{s}$ and $\mathcal{A}$ opens a new bin in each step. Afterwards, we depart all z_i-items except the one that has been packed in the new bin. The maximum load used at any time for items released and not departed is achieved in Step $s-1$; this is $\frac{s}{3} + \sum_{j=0}^{s-1} \frac{j\epsilon}{s} + (s-1)(\frac{1}{3}+\frac{(s-1)\epsilon}{s}) < \frac{2s}{3}$. After departure of items the total load used becomes $\frac{s}{3} + \sum_{j=0}^{s-1} \frac{j\epsilon}{s} = \frac{s}{3} + O(1)$. We have obtained s bins, each packed with one item and the item size is increasing across the bins from $\frac{1}{3}$ up to $\frac{1}{3}+\frac{(s-1)\epsilon}{s}$. Figure 3.2 illustrates the bins obtained by Op-Inc.

Op-Comp. We can now use Op-Comp to release items of size in $[\frac{2}{3}-\frac{(s-1)\epsilon}{s}, \frac{2}{3}]$, i.e., $x = \frac{1}{3}$ and $y = \frac{1}{3}+\frac{(s-1)\epsilon}{s}$. Apart from bins opened by Op-Inc, there are no other existing bins in which $(1-y)$-items can be packed, thus $h = 0$. As for Op-Inc, we have s steps, and in each step $1 \leq i \leq s$, we release identical items of size $w_i = \frac{2}{3}-\frac{(s-i)\epsilon}{s}$ until $\mathcal{A}$ opens a new bin. Afterwards we depart all w_i-items released, except the one packed in the new bin. In Step 1, we release $s+1$ items of size $w_1 = \frac{2}{3}-\frac{(s-1)\epsilon}{s}$. $\mathcal{A}$ can pack at most s items in existing bins and opens at least one new bin. Afterwards, we depart all w_1 items released, except the one packed in the new bin. In Step 2, we release s items of size $w_2 = \frac{2}{3}-\frac{(s-2)\epsilon}{s}$. $\mathcal{A}$ can pack at most $s-1$ items in existing bins, as the s-th bin opened by Op-Inc is packed with an item of size $z_s = \frac{1}{3}+\frac{(s-1)\epsilon}{s}$, thus $w_2 + z_s > 1$. Note that no two items released by Op-Comp can be packed in the same bin. $\mathcal{A}$ opens a new bin in Step 2, and afterwards we depart all other w_2-items except the one packed in the new bin. Notice that in general, in Step i, we release $h+s+2-i$ items of size w_i and the



(a) Case 1



(b) Case 2. $\frac{1}{2}-$ and $\frac{1}{2}+$ denote items with increasing sizes smaller than $\frac{1}{2}$ and greater than $\frac{1}{2}$, respectively.

Figure 3.3: The final configuration of $\mathcal{A}$ achieved by the adversary for a simpler 2.5 lower bound in Cases 1 and 2.

number of items released decreases with each subsequent step. The items released are of increasing size and items of largest size $\frac{2}{3}$ are released in Step s. However, the maximum load used at any time for items released and not departed (including items released by Op-Inc) is achieved in Step 1; this is $\frac{s}{3} + \sum_{j=0}^{s-1} \frac{j\epsilon}{s} + (s+1)(\frac{2}{3} - \frac{(s-1)\epsilon}{s}) < s + \frac{2}{3} = s + O(1)$. At the end of Step s, we obtain s additional bins each packed with one item and the item size is increasing across the bins from $\frac{2}{3} - \frac{(s-1)\epsilon}{s}$ up to $\frac{2}{3}$. Each item in the final bin configuration that has been released by Op-Comp has a complementary item released by Op-Inc such that their sizes sum up to 1. This allows an optimal offline algorithm to pack the items together and use at most $s + O(1)$ bins in total. Figure 3.2 provides an illustration of the final bin configuration achieved by using Op-Inc and Op-Comp.

3.3.4 A 2.5 lower bound using Op-Inc and Op-Comp

In this section we demonstrate how we can benefit from using Op-Inc and Op-Comp by obtaining a 2.5 lower bound as in [19] for any deterministic online algorithm using the two operations in a much simpler way. The lower bound in [19] involves five cases, while using Op-Inc and Op-Comp only requires two cases and significantly fewer stages. We give an adversary for any online algorithm $\mathcal{A}$ such that at any time, the load of items released and not departed is at most $k + O(1)$, for some large even integer k. We prove that $\mathcal{A}$ uses $\frac{5k}{2}$ bins and give an optimal offline algorithm $\mathcal{O}$ that uses at most $k + O(1)$ bins at any time. Thus, the competitive ratio of $\mathcal{A}$ is at least 5/2.

Let $\epsilon = \frac{1}{k}$ and $\delta = \frac{\epsilon}{k+1}$. The adversary works in stages and in Stage 0 we release $\frac{k}{\epsilon}$ items of size ϵ. We denote by n_i the number of new bins opened by $\mathcal{A}$ in Stage i, thus $n_0 \geq k$ as $\mathcal{A}$ requires at least k bins. We consider two cases: $n_0 \geq \frac{3k}{2}$ and $\frac{3k}{2} > n_0 \geq k$. In Case 1, the adversary does not use Op-Inc and Op-Comp, while in Case 2 it uses them in pairs in Stages 1 and 2, respectively.

We first consider Case 1, where $n_0 \geq \frac{3k}{2}$. In Stage 1 we let items depart such that the configuration is $\{\frac{3k}{2}:\epsilon\}$. We can immediately release k items of size 1 and $\mathcal{A}$ needs an additional k bins, i.e., $n_1 = k$. The total number of bins used by $\mathcal{A}$ is thus $\frac{5k}{2}$. Figure 3.3(a) illustrates the final configuration of $\mathcal{A}$. On the other hand, $\mathcal{O}$ uses at most $k + 2$ bins, packing the ϵ-items that never depart in 2 bins and the remaining items in k bins.

$\boldsymbol{\mathcal{O}}$ **: # bins**	**1**	**1**	$\boldsymbol{\frac{k}{2}}$	$\boldsymbol{\frac{k}{2}}$	**total**
Stage 0	-	-	-	-	0
	$\mathbf{1}_{\boldsymbol{*\epsilon}}$	-	$\boldsymbol{(1-\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-\epsilon)_{*\epsilon}}$	k
Stage 1	$1_{*\epsilon}$	-	-	$\frac{1}{2}_{*\epsilon}$	$\frac{k}{4}+k\epsilon$
(Op-Inc)	$1_{*\epsilon}$	-	$\boldsymbol{\frac{1}{2}-i\delta, \frac{1}{2}-\delta}$	$\frac{1}{2}_{*\epsilon}$, $\boldsymbol{\frac{1}{2}-\delta}$	$k+k\epsilon-k\delta-\sum_{i=1}^{\frac{k}{2}} i\delta$
Stage 2	$1_{*\epsilon}$	-	$\frac{1}{2}-i\delta$	$(\frac{1}{2}-\epsilon)_{*\epsilon}$	$\frac{k}{2}+\frac{k\epsilon}{2}-\sum_{i=1}^{\frac{k}{2}} i\delta$
(Op-Comp)	$1_{*\epsilon}$	$\boldsymbol{\frac{1}{2}+\frac{k}{2}\delta}$	$\frac{1}{2}-i\delta$, $\boldsymbol{\frac{1}{2}+i\delta}$	$(\frac{1}{2}-\epsilon)_{*\epsilon}$, $\boldsymbol{\frac{1}{2}+\frac{k}{2}\delta}$	$k+\frac{k\epsilon}{2}+\frac{k^2\delta}{4}+(\frac{1}{2}+\frac{k}{2}\delta)$
Stage 3	$1_{*\epsilon}$	-	$\frac{1}{2}-i\delta, \frac{1}{2}+i\delta$	-	$\frac{k}{2}+k\epsilon$
(Final)	$1_{*\epsilon}$	-	$\frac{1}{2}-i\delta, \frac{1}{2}+i\delta$	$\mathbf{1}$	$k+k\epsilon$

Table 3.1: The optimal schedule for Case 2. For each stage, the first row is the configuration just before items arrival and the second row is the configuration when the maximum amount of load is released. The very last row is the final configuration. Bolded entries are new items arrived in the corresponding stage. The notation y_{*z} means packing a bin with a load y of z-items. The last column shows the total load of all bins.

We now consider Case 2, where $k \le n_0 < \frac{3k}{2}$. We first note that there are at least $\frac{k}{2}$ bins with load at least $\frac{1}{2}+\epsilon$ in each bin. Suppose by contradiction that there are less than $\frac{k}{2}$ of these bins. The $\frac{k}{2}-1$ highest load bins can have a load of at most 1 and the remaining bins (at most k of them) can have a load of at most $\frac{1}{2}$. Thus, the total load of the n_0 bins is at most $(\frac{k}{2}-1)\times 1+((\frac{3k}{2}-1)-\frac{k}{2}+1)\times\frac{1}{2} < k$, and this contradicts the fact that we release a load of k.

In Stage 1 we aim to force $\mathcal{A}$ to use at least $\frac{k}{2}$ new bins, i.e., $n_1 \ge \frac{k}{2}$. We let items depart until the configuration is $\{\frac{k}{2}{:}\epsilon, \frac{k}{2}{:}(\frac{1}{2}{+}\epsilon)_{*\epsilon}\}$ with k bins and a total load of $\frac{k}{4}+O(1)$. We use Op-Inc to release at most $\frac{3k}{2}$ items of size in $[\frac{1}{2}-\frac{k}{2}\delta, \frac{1}{2}-\delta]$, resulting in a total load of $k+O(1)$. For each existing ϵ-bin, at most one such new item can be packed because $\delta = \frac{\epsilon}{k+1}$, $(\frac{1}{2}-\frac{k}{2}\delta)+(\frac{1}{2}-\frac{k}{2}\delta)+\epsilon = 1-\frac{k}{k+1}\epsilon+\epsilon > 1$, and items released by Op-Inc are of increasing sizes. For a similar reason, no new item can be packed in the $((\frac{1}{2}{+}\epsilon)_{*\epsilon})$-bins. The parameters for Op-Inc are therefore $x = \frac{1}{2}-\frac{k}{2}\delta$, $h = \frac{k}{2}$ and $s = \frac{k}{2}$. According to Op-Inc, $\mathcal{A}$ opens at least $\frac{k}{2}$ new bins, each with one $(\frac{1}{2}-i\delta)$-item, for $1 \le i \le \frac{k}{2}$.

In Stage 2 we aim to force $\mathcal{A}$ to open at least another $\frac{k}{2}$ new bins, i.e., $n_2 \ge \frac{k}{2}$. The configuration of $\mathcal{A}$ at the end of Stage 1 and after departing one ϵ-item from each of the $((\frac{1}{2}{+}\epsilon)_{*\epsilon})$-bins is $\{\frac{k}{2}{:}\epsilon, \frac{k}{2}{:}\frac{1}{2}_{*\epsilon}, \frac{k}{2}{:}\frac{1}{2}{-}i\delta\}$, for $1 \le i \le \frac{k}{2}$, with $\frac{3k}{2}$ bins and a total load of $\frac{k}{2}+O(1)$. Note that in the last $\frac{k}{2}$ bins, the load increases by δ from $\frac{1}{2}-\frac{k}{2}\delta$ to $\frac{1}{2}-\delta$. We now use Op-Comp with $x = \frac{1}{2}-\frac{k}{2}\delta$, $y = \frac{1}{2}-\delta$, $h = \frac{k}{2}$, and $s = \frac{k}{2}$. I.e., Op-Comp releases items of size in the range $[\frac{1}{2}+\delta, \frac{1}{2}+\frac{k}{2}\delta]$ and at any time, at most $k+1$ items are released. Note that at most one item can be packed in the ϵ-bins and none in the $(\frac{1}{2}_{*\epsilon})$-bins, i.e., $h = \frac{k}{2}$ as said. According to Op-Comp, $\mathcal{A}$ needs to open $\frac{k}{2}$ new bins.

In Stage 3, we aim at $n_3 = \frac{k}{2}$. We let items depart until the configuration becomes $\{\frac{k}{2}{:}\epsilon, \frac{k}{2}{:}\epsilon, \frac{k}{2}{:}\frac{1}{2}{-}i\delta, \frac{k}{2}{:}\frac{1}{2}{+}i\delta\}$, for $1 \le i \le \frac{k}{2}$, with $2k$ bins and a total load of $\frac{k}{2}+O(1)$. We finally release $\frac{k}{2}$ items of size 1, and the total number of bins used by $\mathcal{A}$ becomes $\frac{5k}{2}$. Figure 3.3(b) illustrates the final configuration of $\mathcal{A}$.

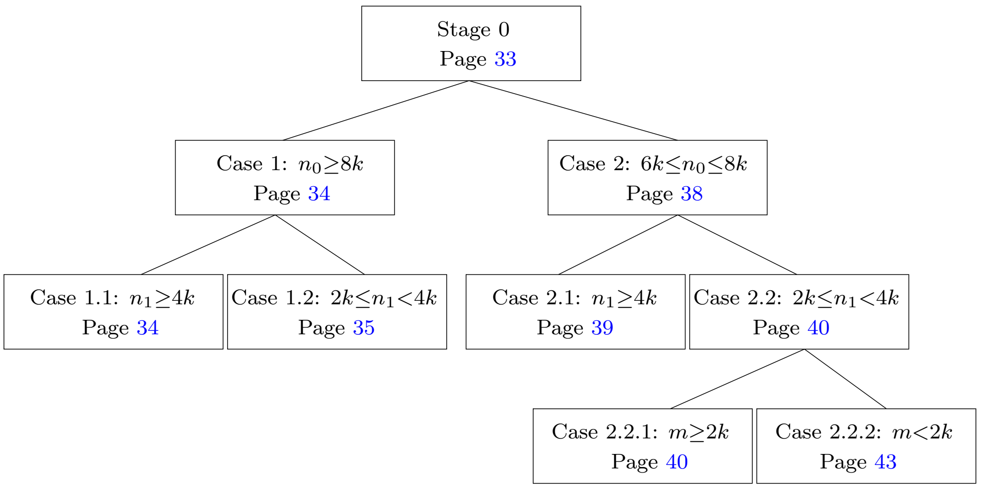


Figure 3.4: Different cases of the adversary.

On the other hand, the optimal offline algorithm $\mathcal{O}$ can use $k + 2$ bins to pack all items as shown in Table 3.1, which lists the packing configuration in each stage. Note that for stages involving Op-Inc and Op-Comp, we list the packing when the maximum amount of load is released in that stage (as each stage may involve item arrival and departure).

3.4 The 8/3 Lower Bound - The Simple Cases

We give an adversary such that at any time, the total load of items released and not departed is at most $6k + O(1)$, for some large integer k. We prove that any online algorithm $\mathcal{A}$ uses $16k$ bins, while the optimal offline algorithm $\mathcal{O}$ uses at most $6k + O(1)$ bins. Then, the competitive ratio of $\mathcal{A}$ is at least $\frac{8}{3}$. Figure 3.4 illustrates the different cases and gives an easy reference to the pages where the discussion resides.

The adversary works in stages and uses Op-Inc and Op-Comp in pairs. Case 1.1 does not use Op-Inc and Op-Comp, Cases 1.2 and 2.1 use them once while Cases 2.2.1 and 2.2.2 use them twice. Let $\epsilon = \frac{1}{6k}$, $\delta = \frac{\epsilon}{16k}$, and $\delta' = \frac{\delta}{16k}$. Notice that $\epsilon \gg \delta \gg \delta'$, for large k. To motivate our use of these three values, suppose that we have one ϵ-item and one $(\frac{1}{2} - \delta)$-item packed in the same bin. An additional $(\frac{1}{2} - \delta)$-item cannot be packed in the same bin, thus forcing an online algorithm $\mathcal{A}$ to open a new bin. On the other hand, one $(\frac{1}{2} - \delta)$-item can be packed in the same bin with any one of $(\frac{1}{2} + \delta)$-, $(\frac{1}{2} - \delta')$-, or $(\frac{1}{2} + \delta')$-item. These item sizes are useful for the optimal offline algorithm $\mathcal{O}$ that packs complementary items in the same bin.

In each stage we give the total size of items in the packing configurations of both $\mathcal{A}$ and $\mathcal{O}$. Recall that Op-Inc and Op-Comp use items of slightly different sizes in their respective stages. We omit the tedious calculations resulting as part of minor differences in sizes by giving a total size of items that involves the parameter symbol i used by Op-Inc and Op-Comp, even when i may take different values (see Remark

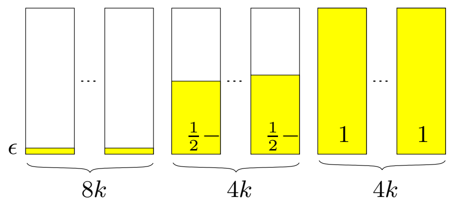


Figure 3.5: The final configuration of $\mathcal{A}$ achieved by the adversary in Case 1.1.

in Section 3.3.1). Note that this slightly imprecise notation serves to avoid diverting attention from the techniques used in our analysis. However, the exact total size of items may be calculated as in Table 3.1 by taking into account the parameters of Op-Inc and Op-Comp in the cases and stages that they are used.

The adversary. Let n_i be the number of new bins used by $\mathcal{A}$ in Stage i. In Stage 0, the adversary releases $\frac{6k}{\epsilon}$ items of size ϵ, with total load $6k$. It is clear that $\mathcal{A}$ needs at least $6k$ bins, i.e., $n_0 \geq 6k$. We distinguish between two cases: $n_0 \geq 8k$ and $8k > n_0 \geq 6k$. We distinguish cases by the ranges of the number of new bins opened and we usually let items depart to an exact value. We will then refer to n_i as this exact value. For example, in the case of $n_0 \geq 8k$, we let items depart until $8k$ bins are left and in the next stage we refer to $n_0 = 8k$ instead of $n_0 >= 8k$. In this section, we consider the case $n_0 \geq 8k$.

Case 1: $n_0 \geq 8k$.

Stage 1. In this stage, we aim at $n_1 \geq 2k$. We keep one ϵ-item in $8k$ bins and let all other items depart so that the configuration becomes $\{8k{:}\epsilon\}$. The total size of items departed is $6k - 8k\epsilon$ and the size of items remaining is $8k\epsilon$. We then release $12k$ items of size $\frac{1}{2}-\delta$, resulting in a total load of $6k + O(1)$. Note that $\frac{1}{2} > \frac{1}{2}-\delta > \frac{1}{2}-16k\delta = \frac{1}{2}-\epsilon$. At most one item can be packed in each existing bin and therefore $\mathcal{A}$ needs at least $2k$ new bins, i.e., $n_1 \geq 2k$. We further consider two sub-cases: $n_1 \geq 4k$ and $2k \leq n_1 < 4k$.

Case 1.1: $n_0 \geq 8k$ and $n_1 \geq 4k$.

In this case, we have $n_0 + n_1 \geq 12k$.

Stage 2. In this stage, we aim at $n_2 = 4k$. We keep one $(\frac{1}{2}-\delta)$-item in $4k$ bins and let all other $(\frac{1}{2}-\delta)$-items depart so that the configuration becomes $\{8k{:}\epsilon, 4k{:}\frac{1}{2}-\delta\}$, with $12k$ bins and a total load of $2k + O(1)$.

# bins	$8k$	$4k$
$\mathcal{A}$	ϵ	$\frac{1}{2}-\delta$

total size: $2k+8k\epsilon-4k\delta$
bins: $12k$

$\mathcal{O}$ **: # bins**	**2**	$\boldsymbol{2k}$	$\boldsymbol{4k}$	**total**
Stage 0	-	-	-	0
	$\mathbf{1}_{*\epsilon}$	$\boldsymbol{(1{-}2\epsilon)_{*\epsilon}}$	$\boldsymbol{(1{-}2\epsilon)_{*\epsilon}}$	$6k$
Stage 1	$\frac{8}{12}{}_{*\epsilon}$	-	-	$8k\epsilon$
	$\frac{8}{12}{}_{*\epsilon}$	$\boldsymbol{\frac{1}{2}{-}\delta, \frac{1}{2}{-}\delta}$	$\boldsymbol{\frac{1}{2}{-}\delta, \frac{1}{2}{-}\delta}$	$6k{+}8k\epsilon{-}12k\delta$
Stage 2	$\frac{8}{12}{}_{*\epsilon}$	$\frac{1}{2}{-}\delta, \frac{1}{2}{-}\delta$	-	$2k{+}8k\epsilon{-}4k\delta$
(Final)	$\frac{8}{12}{}_{*\epsilon}$	$\frac{1}{2}{-}\delta, \frac{1}{2}{-}\delta$	**1**	$6k{+}8k\epsilon{-}4k\delta$

Table 3.2: The optimal schedule for Case 1.1. For each stage, the first row is the configuration just before items arrival and the second row is the configuration when the maximum amount of load is released. The very last row is the final configuration. Bolded entries are new items arrived in the corresponding stage. The notation y_{*z} means packing a bin with a load y of z-items. The last column shows the total load of all bins.

We then release $4k$ items of size 1. Each of these items has to be packed in a new bin, i.e., $n_2 = 4k$. In total, $\mathcal{A}$ uses $8k + 4k + 4k = 16k$ bins. Figure 3.5 shows the final configuration of $\mathcal{A}$.

# bins	$\boldsymbol{8k}$	$\boldsymbol{4k}$	$\boldsymbol{4k}$
$\mathcal{A}$	ϵ	$\frac{1}{2}{-}\delta$	1

total size: $6k{+}8k\epsilon{-}4k\delta$
bins: $16k$

Lemma 3.1 summarizes the case and the packing of the optimal solution.

Lemma 3.1. *If $\mathcal{A}$ uses at least $8k$ bins in Stage 0 and $4k$ bins in Stage 1, then $\mathcal{A}$ uses $16k$ bins at the end while $\mathcal{O}$ uses $6k + 2$ bins.*

Proof. In order to complete the proof of the lemma, we list in Table 3.2 the packing configuration of the optimal offline algorithm $\mathcal{O}$, showing how items are packed in each stage. We show that $\mathcal{O}$ uses at most $6k + 2$ bins at any time. Each stage is composed of item departure and arrival. In Stage 0, the $8k$ ϵ-items released that never depart are packed in two bins, $4k$ ϵ-items in each bin, while the remaining ϵ-items that depart in Stage 1 are packed in the remaining space of the two bins and additionally in $6k$ bins. In Stage 1, $6k$ bins are emptied and $4k$ items of size $\frac{1}{2}{-}\delta$ released that never depart are packed in $2k$ bins, two items in each bin, with the remaining $8k$ items that depart in Stage 2 packed in $4k$ bins, two items in each bin. In the final stage, $4k$ 1-items released are packed in the last $4k$ bins that are emptied at the beginning of the stage. $\square$

Case 1.2: $\boldsymbol{n_0 \geq 8k}$ **and** $\boldsymbol{2k \leq n_1 < 4k}$**.**

In this case, we have $10k \leq n_0 + n_1 < 12k$. We first make an observation about the number of bins that contain one ϵ-item and one $(\frac{1}{2}{-}\delta)$-item. If this number is smaller than $4k$, then we have $n_1 > (12k - 4k)/2 = 4k$ because we have more than $12k - 4k$

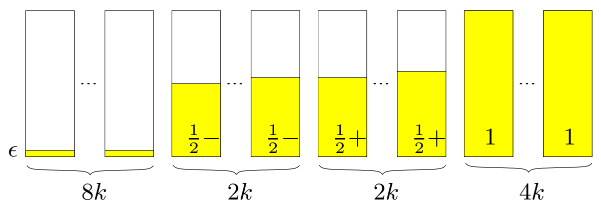


Figure 3.6: The final configuration of $\mathcal{A}$ achieved by the adversary in Case 1.2.

items of size $\frac{1}{2}-\delta$ to be packed in new bins and each bin can pack at most two of them, contradicting the definition of Case 1.2.

Observation 3.2. At the end of Stage 1 of Case 1.2, the number of bins containing one ϵ-item and one $(\frac{1}{2}-\delta)$-item is at least $4k$.

Stage 2. In this stage, we aim at $n_2 \geq 2k$. We keep $4k$ items of size $\frac{1}{2}-\delta$ in each of those bins that also have an ϵ-item and let other $(\frac{1}{2}-\delta)$-items depart. Then the configuration of $\mathcal{A}$ becomes $\{4k{:}\epsilon, 4k{:}(\epsilon, \frac{1}{2}-\delta)\}$, with $8k$ bins and a total load of $2k + O(1)$.

# bins	**$4k$**	**$4k$**
$\mathcal{A}$	ϵ	$\epsilon, \frac{1}{2}-\delta$

total size: $2k+8k\epsilon-4k\delta$
bins: $8k$

Recall that $\delta' = \frac{\delta}{16k}$. We use Op-Inc with $x = \frac{1}{2}-8k\delta'$, $h = 4k$, and $s = 2k$. For the first $4k$ bins, only one x-item can be packed; for the second $4k$ bins, no item can be packed since $16k\delta = \epsilon$, $16k\delta' = \delta$, and $(\epsilon + \frac{1}{2}-\delta) + \frac{1}{2}-8k\delta' > 1$. Therefore, we have $h = 4k$ as said. With $s = 2k$, Op-Inc releases $8k$ items of increasing size from $\frac{1}{2}-8k\delta'$ to at most $\frac{1}{2}-\delta'$. According to Op-Inc, $\mathcal{A}$ needs to open at least $2k$ bins, each with one $(\frac{1}{2}-(8k-i)\delta')$-item, for $0 \leq i < n_2$, and perhaps one other item. Note that $n_2 \geq 2k$ and at the end of Op-Inc we depart items such that $n_2 = 2k$.

Stage 3. In this stage, we aim at $n_3 = 2k$. The configuration at the end of Stage 2 is

$$\{4k{:}\epsilon, 4k{:}(\epsilon, \frac{1}{2}-\delta), 2k{:}\frac{1}{2}-(8k-i)\delta'\}\ \ , \ \text{for } 0 \leq i < 2k,$$

with $10k$ bins and a total load of $3k + O(1)$.

# bins	**$4k$**	**$4k$**	**$2k$**
$\mathcal{A}$	ϵ	$\epsilon, \frac{1}{2}-\delta$	$\frac{1}{2}-(8k-i)\delta'$

total size: $3k+8k\epsilon-4k\delta-2k(8k-i)\delta'$
bins: $10k$

Note that in the last $2k$ bins, the load increases by δ' from $\frac{1}{2}-8k\delta'$ to $\frac{1}{2}-6k\delta'$. We now use Op-Comp with $x = \frac{1}{2}-8k\delta'$, $y = \frac{1}{2}-6k\delta'$, $h = 4k$ and $s = 2k$, i.e., we release items of sizes in the range $\frac{1}{2}+6k\delta'$ to $\frac{1}{2}+8k\delta'$ and at any time at most $6k + 1$ items are released. Note that at most one new item can be packed in the first $4k$ bins, none in the next $4k$ bins, i.e., $h = 4k$. According to Op-Comp, $\mathcal{A}$ needs to open $2k$ bins and $n_3 = 2k$.

Stage 4. In this stage, we aim at $n_4 = 4k$. We let some items depart until the configuration is

$$\{4k{:}\epsilon, 4k{:}\epsilon, 2k{:}\frac{1}{2}-(8k-i)\delta', 2k{:}\frac{1}{2}+(8k-i)\delta'\}\ , \text{ for } 0 \le i < 2k,$$

with $12k$ bins and a total load of $2k + O(1)$.

# bins	**4*k***	**4*k***	**2*k***	**2*k***
$\mathcal{A}$	ϵ	ϵ	$\frac{1}{2}-(8k-i)\delta'$	$\frac{1}{2}+(8k-i)\delta'$

total size: $2k+8k\epsilon$
bins: $12k$

We further release $4k$ items of size 1, hence $n_4 = 4k$. In total, $\mathcal{A}$ uses $4k + 4k + 2k + 2k + 4k = 16k$ bins. Figure 3.6 shows the final configuration of $\mathcal{A}$.

# bins	**4*k***	**4*k***	**2*k***	**2*k***	**4*k***
$\mathcal{A}$	ϵ	ϵ	$\frac{1}{2}-(8k-i)\delta'$	$\frac{1}{2}+(8k-i)\delta'$	1

total size: $6k+8k\epsilon$
bins: $16k$

We observe that each item with size $\frac{1}{2}-(8k-i)\delta'$ has a corresponding item $\frac{1}{2}+(8k-i)\delta'$ such that the sum of sizes is 1. This allows the optimal offline algorithm to have a better packing. Lemma 3.3 summarizes the case and the packing of the optimal solution.

Lemma 3.3. *If $\mathcal{A}$ uses at least $8k$ bins in Stage 0 and $[4k, 8k)$ bins in Stage 1, then $\mathcal{A}$ uses $16k$ bins at the end while $\mathcal{O}$ uses $6k + 2$ bins.*

Proof. We list in Table 3.3 the packing configuration of the optimal offline algorithm $\mathcal{O}$, showing how items are packed in each stage. We show that $\mathcal{O}$ uses at most $6k + 2$ bins at any time. Each stage is composed of item departure and arrival. Note that in Stages 2 and 3 Op-Inc and Op-Comp are used, respectively, and we list the packing when maximum amount of load is released in that stage as each stage involves item arrival and departure. In Stage 0, the $8k$ items of size ϵ that never depart are packed by $\mathcal{O}$ in two bins, the first bin packed with $6k$ items and the second bin packed with $2k$ items. This allows the remaining space of the second bin to be reused for one item of size $\frac{1}{2}+(8k-i)\delta'$ in Stage 3. The remaining ϵ-items are packed in the remaining space of the second bin and additionally $6k$ bins. In Stage 1, $6k$ bins are emptied and $4k$ items of size $\frac{1}{2}-\delta$ released that depart only in the final stage are packed in $4k$ bins, one item in each bin. This allows some items released by Op-Inc and Op-Comp in Stages 2 and 3 to be temporarily packed in the $4k$ bins, as $(\frac{1}{2}-\delta) + (\frac{1}{2}+(8k-i)\delta') < 1$. The remaining $8k$ items released that depart in Stage 2 are packed in the remaining space of the $4k$ bins

$\mathcal{O}$: # bins	1	1	$4k$	$2k$	total
Stage 0	-	-	-	-	0
	$\mathbf{1_{*\epsilon}}$	$\mathbf{1_{*\epsilon}}$	$\mathbf{(1{-}2\epsilon)_{*\epsilon}}$	$\mathbf{(1{-}2\epsilon)_{*\epsilon}}$	$6k$
Stage 1	$1_{*\epsilon}$	$\frac{1}{3}_{*\epsilon}$	-	-	$8k\epsilon$
	$1_{*\epsilon}$	$\frac{1}{3}_{*\epsilon}$	$\boldsymbol{\frac{1}{2}{-}\delta, \frac{1}{2}{-}\delta}$	$\boldsymbol{\frac{1}{2}{-}\delta, \frac{1}{2}{-}\delta}$	$6k{+}8k\epsilon{-}12k\delta$
Stage 2	$1_{*\epsilon}$	$\frac{1}{3}_{*\epsilon}$	$\frac{1}{2}{-}\delta$	-	$2k{+}8k\epsilon{-}4k\delta$
(Op-Inc)	$1_{*\epsilon}$	$\frac{1}{3}_{*\epsilon}$	$\frac{1}{2}{-}\delta$, $\boldsymbol{\frac{1}{2}{-}(8k{-}i)\delta'}$	$\boldsymbol{\frac{1}{2}{-}(8k{-}i)\delta'}$, $\boldsymbol{\frac{1}{2}{-}(8k{-}i)\delta'}$	$6k{+}8k\epsilon{-}4k\delta$ $-8k(8k{-}i)\delta'$
Stage 3	$1_{*\epsilon}$	$\frac{1}{3}_{*\epsilon}$	$\frac{1}{2}{-}\delta$	$\frac{1}{2}{-}(8k{-}i)\delta'$	$3k{+}8k\epsilon{-}4k\delta$ $-2k(8k{-}i)\delta'$
(Op-Comp)	$1_{*\epsilon}$	$\frac{1}{3}_{*\epsilon}$, $\boldsymbol{\frac{1}{2}{+}(8k{-}i)\delta'}$	$\frac{1}{2}{-}\delta$, $\boldsymbol{\frac{1}{2}{+}(8k{-}i)\delta'}$	$\frac{1}{2}{-}(8k{-}i)\delta'$, $\boldsymbol{\frac{1}{2}{+}(8k{-}i)\delta'}$	$6k{+}8k\epsilon{-}4k\delta$ $+4k(8k{-}i)\delta'{+}(\frac{1}{2}{+}(8k{-}i)\delta')$
Stage 4	$1_{*\epsilon}$	$\frac{1}{3}_{*\epsilon}$	-	$\frac{1}{2}{-}(8k{-}i)\delta'$, $\frac{1}{2}{+}(8k{-}i)\delta'$	$2k{+}8k\epsilon$
(Final)	$1_{*\epsilon}$	$\frac{1}{3}_{*\epsilon}$	$\mathbf{1}$	$\frac{1}{2}{-}(8k{-}i)\delta'$, $\frac{1}{2}{+}(8k{-}i)\delta'$	$6k{+}8k\epsilon$

Table 3.3: The optimal schedule for Case 1.2. For each stage, the first row is the configuration just before items arrival and the second row is the configuration when the maximum amount of load is released. The very last row is the final configuration. Bolded entries are new items arrived in the corresponding stage. The notation y_{*z} means packing a bin with a load y of z-items. The last column shows the total load of all bins.

and an additional $2k$ bins. In the beginning of Stage 2, the $8k$ items depart. Afterwards Op-Inc is used to release $8k$ items of increasing sizes $\frac{1}{2}{-}(8k{-}i)\delta'$ and these are packed in the emptied space, with the condition that the $2k$ of the items that never depart are packed in the last $2k$ bins, one item in each bin. In Stage 3 we first depart $6k$ of the items released in the previous stage. Op-Comp is then used to release at most $6k+1$ items of increasing sizes $\frac{1}{2}{+}(8k{-}i)\delta'$. $2k$ of these items that never depart are packed in the last $2k$ bins, one item in each bin that has a complementary $(\frac{1}{2}{-}(8k{-}i)\delta')$-item. The remaining $4k+1$ items are packed in the remaining space. Finally in Stage 4 the first $4k$ bins are emptied and $4k$ 1-items released are packed there. □

3.5 The 8/3 Lower Bound - The Complex Cases

Case 2: $6k \le n_0 < 8k$.

This case is more complicated and involves three subcases. We make two observations about the load of the n_0 bins. If less than $4k$ bins have load at least $\frac{1}{2}+\epsilon$, then the total load of all bins is at most $(4k-1)+4k/2 = 6k-1$, contradicting the fact that total load of items released is $6k$. Similarly, if less than $5k$ bins have load at least $\frac{1}{4}+\epsilon$, then the total load of all bins is at most $(5k-1)+3k/4 < 6k$, leading to a contradiction.

Observation 3.4. At the end of Stage 0 of Case 2, (i) at least $4k$ bins have load at least $\frac{1}{2}+\epsilon$; (ii) at least $5k$ bins have load at least $\frac{1}{4}+\epsilon$.

Stage 1. We aim at $n_1 \ge 2k$. We let ϵ-items depart until the configuration of $\mathcal{A}$ becomes

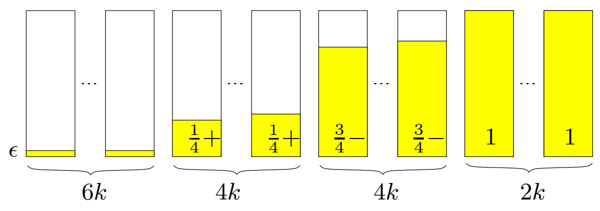


Figure 3.7: The final configuration of $\mathcal{A}$ achieved by the adversary in Case 2.1.

$$\{4k{:}(\frac{1}{2}{+}\epsilon)_{*\epsilon}, k{:}(\frac{1}{4}{+}\epsilon)_{*\epsilon}, k{:}\epsilon\}\ ,$$

with $6k$ bins and a total load of $9k/4 + O(1)$.

# bins	$4k$	k	k
$\mathcal{A}$	$(\frac{1}{2}{+}\epsilon)_{*\epsilon}$	$(\frac{1}{4}{+}\epsilon)_{*\epsilon}$	ϵ

total size: $\frac{9k}{4}{+}6k\epsilon$
bins: $6k$

We then use Op-Inc with $x = \frac{1}{4}{+}\delta$, $h = 9k$, and $s = 2k$. The first $4k$ bins can pack at most one x-item, the next k bins at most two, and the last k bins at most three, i.e., $h = 9k$. Any new bin can pack at most three items, implying that Op-Inc releases $15k = h+3s$ items of increasing sizes, from $\frac{1}{4}{+}\delta$ to at most $\frac{1}{4}{+}15k\delta$. According to Op-Inc, $\mathcal{A}$ opens at least $2k$ bins, i.e., $n_1 \geq 2k$, and Op-Inc does not depart items released in its last step. We consider two subcases: $n_1 \geq 4k$ and $2k \leq n_1 < 4k$.

Case 2.1: $6k \leq n_0 < 8k$ and $n_1 \geq 4k$.

In this case, we have $10k \leq n_0 + n_1$.

Stage 2. In this stage, we aim at $n_2 \geq 4k$. The configuration after departing some items released by Op-Inc and after departing a load of $\frac{1}{4}$ of ϵ-items from each of the $((\frac{1}{2}{+}\epsilon)_{*\epsilon})$-bins becomes

$$\{4k{:}(\frac{1}{4}{+}\epsilon)_{*\epsilon}, k{:}(\frac{1}{4}{+}\epsilon)_{*\epsilon}, k{:}\epsilon, 4k{:}\frac{1}{4}{+}i\delta\}\ ,\ \text{ for } 1 \leq i \leq 4k,$$

with $10k$ bins and a total load of $9k/4 + O(1)$.

# bins	$4k$	k	k	$4k$
$\mathcal{A}$	$(\frac{1}{4}{+}\epsilon)_{*\epsilon}$	$(\frac{1}{4}{+}\epsilon)_{*\epsilon}$	ϵ	$\frac{1}{4}{+}i\delta$

total size: $\frac{9k}{4}{+}6k\epsilon{+}4ki\delta$
bins: $10k$

Note that in the last $4k$ bins, the load increases by δ from $\frac{1}{4}{+}\delta$ to $\frac{1}{4}{+}4k\delta$. We now use Op-Comp with $x = \frac{1}{4}{+}\delta$, $y = \frac{1}{4}{+}4k\delta$, $h = k$, and $s = 4k$. I.e., Op-Comp releases items of sizes from $\frac{3}{4}{-}4k\delta$ to $\frac{3}{4}{-}\delta$ and at any time, at most $5k+1$ items are needed. None of these items can be packed in the first $5k$ bins, and only one can be packed in the next k bins, i.e., $h = k$ as said. According to Op-Comp, $\mathcal{A}$ requires $4k$ new bins.

Stage 3. In this stage, we aim at $n_3 = 2k$. We let items depart until the configuration becomes

$$\{4k{:}\epsilon, k{:}\epsilon, k{:}\epsilon, 4k{:}\frac{1}{4}{+}i\delta, 4k{:}\frac{3}{4}{-}i\delta\}\ \ ,\ \text{for } 1 \le i \le 4k,$$

with $14k$ bins and a total load of $4k + O(1)$.

# bins	$4k$	k	k	$4k$	$4k$
$\mathcal{A}$	ϵ	ϵ	ϵ	$\frac{1}{4}{+}i\delta$	$\frac{3}{4}{-}i\delta$

total size: $4k{+}6k\epsilon$
bins: $14k$

We further release $2k$ items of size 1. $\mathcal{A}$ needs to open $2k$ new bins, so $n_3 = 2k$. In total, $\mathcal{A}$ uses $6k + 4k + 4k + 2k = 16k$ bins. Figure 3.7 shows the final configuration of $\mathcal{A}$.

# bins	$4k$	k	k	$4k$	$4k$	$2k$
$\mathcal{A}$	ϵ	ϵ	ϵ	$\frac{1}{4}{+}i\delta$	$\frac{3}{4}{-}i\delta$	1

total size: $6k{+}6k\epsilon$
bins: $16k$

We note that each item with size $\frac{1}{4}{+}i\delta$ has a corresponding item $\frac{3}{4}{-}i\delta$ such that the sum of sizes is 1. This allows the optimal offline algorithm to have a better packing. The details are given in Section 3.6.

Lemma 3.5. *If $\mathcal{A}$ uses $[6k, 8k)$ bins in Stage 0 and at least $4k$ bins in Stage 1, then $\mathcal{A}$ uses $16k$ bins at the end while $\mathcal{O}$ uses $6k+4$ bins.*

Case 2.2: $6k \le n_0 < 8k$ and $2k \le n_1 < 4k$.

In this case, the Op-Inc in Stage 1 is paired with an Op-Comp in Stage 4 (not consecutively), and in between, there is another pair of Op-Inc and Op-Comp in Stages 2 and 3, respectively. Let m be the number of bins among the n_1 new bins that have been packed with at least two items. We further distinguish two subcases: $m \ge 2k$ and $m < 2k$.

Case 2.2.1: $6k \le n_0 < 8k$, $2k \le n_1 < 4k$ and $m \ge 2k$.

In this case, we have $8k \le n_0 + n_1 < 10k$ and $m \ge 2k$. We make an observation about the bins containing some ϵ-items. In particular, we claim that there are at least k bins that are packed with

- either one ϵ-item and at least two $(\frac{1}{4}{+}i\delta)$-items,

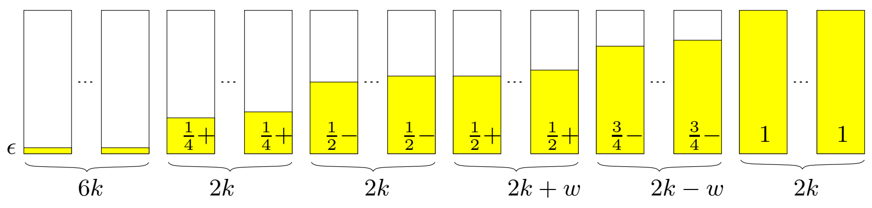


Figure 3.8: The final configuration of $\mathcal{A}$ achieved by the adversary in Case 2.2.1.

- or one $(\frac{1}{4}+i\delta)$-item plus at least a load of $\frac{1}{4}+\epsilon$.

We note that in Stage 1, $15k$ items are released and at most three items can be packed in any of the $n_1 < 4k$ new bins, i.e., at most $12k$ items. So, at least $3k$ of them have to been packed in the first $6k$ bins. Let a and b be the number of bins in the first $5k$ bins (with load at least $\frac{1}{4}+\epsilon$) that are packed with at least one $(\frac{1}{4}+i\delta)$-item and at least two $(\frac{1}{4}+i\delta)$-items, respectively; z_1, z_2, z_3 be the number of bins in the next k bins (with one ϵ-item) that are packed one, two, and three $(\frac{1}{4}+i\delta)$-items, respectively. Note that $z_1+z_2+z_3 \le k$. Since $3k$ items have to be packed in these bins, we have $a+2b+z_1+2z_2+3z_3 \ge 3k$, hence $a+2b+z_2+2z_3 \ge 2k$. The last inequality implies that $a+b+z_2+z_3 \ge k$ and the claim holds.

Observation 3.6. At the end of Stage 1 of Case 2.2.1, at least k bins are packed with either one ϵ-item and at least two $(\frac{1}{4}+i\delta)$-items, or one $(\frac{1}{4}+i\delta)$-item plus at least a load of $\frac{1}{4}+\epsilon$.

Stage 2. We aim at $n_2 \ge 2k$. Let $z = z_2+z_3$. We let items depart until the configuration becomes

$$\{3k{:}(\frac{1}{2}+\epsilon)_{*\epsilon}, k-z{:}((\frac{1}{4}+\epsilon)_{*\epsilon}, \frac{1}{4}+i\delta), z{:}(\epsilon, \frac{1}{4}+i\delta, \frac{1}{4}+i\delta), 2k{:}\epsilon, 2k{:}(\frac{1}{4}+i\delta, \frac{1}{4}+i\delta)\} \ ,$$

with $8k$ bins and a total load of $3k+O(1)$.

# bins	$3k$	$k-z$	z	$2k$	$2k$
$\mathcal{A}$	$(\frac{1}{2}+\epsilon)_{*\epsilon}$	$(\frac{1}{4}+\epsilon)_{*\epsilon}, \frac{1}{4}+i\delta$	$\epsilon, \frac{1}{4}+i\delta, \frac{1}{4}+i\delta$	ϵ	$\frac{1}{4}+i\delta, \frac{1}{4}+i\delta$

total size: $3k+6k\epsilon+5ki\delta+zi\delta$
bins: $8k$

Recall that $\delta' = \frac{\delta}{16k}$. We use Op-Inc with $x = \frac{1}{2}-6k\delta'$, $h = 2k$, and $s = 2k$. The x-items can only be packed in the $2k$ bins with load ϵ, at most one item in one bin, i.e., $h = 2k$. Any new bin can pack at most two, implying that Op-Inc releases $6k = h+2s$ items of increasing sizes, from $\frac{1}{2}-6k\delta'$ to at most $\frac{1}{2}-\delta'$. According to Op-Inc, $\mathcal{A}$ has to open at least $2k$ new bins, i.e., $n_2 \ge 2k$. In the last step of Op-Inc we depart items such that $n_2 = 2k$.

Stage 3. In this stage, we aim at $n_3 \ge 2k$. We use Op-Comp which corresponds to Op-Inc in Stage 2. We let items depart until the configuration becomes

$$\{3k{:}(\frac{1}{2}{+}\epsilon)_{*\epsilon}, k{-}z{:}((\frac{1}{4}{+}\epsilon)_{*\epsilon}, \frac{1}{4}{+}i\delta), z{:}(\epsilon, \frac{1}{4}{+}i\delta, \frac{1}{4}{+}i\delta), 2k{:}\epsilon,$$
$$2k{:}(\frac{1}{4}{+}i\delta, \frac{1}{4}{+}i\delta), 2k{:}\frac{1}{2}{-}i\delta'\} \ ,$$

with $10k$ bins and a total load of $4k + O(1)$.

# bins	$3k$	$k{-}z$	z	$2k$	$2k$	$2k$
$\mathcal{A}$	$(\frac{1}{2}{+}\epsilon)_{*\epsilon}$	$(\frac{1}{4}{+}\epsilon)_{*\epsilon}, \frac{1}{4}{+}i\delta$	$\epsilon, \frac{1}{4}{+}i\delta, \frac{1}{4}{+}i\delta$	ϵ	$\frac{1}{4}{+}i\delta, \frac{1}{4}{+}i\delta$	$\frac{1}{2}{-}i\delta'$

total size: $4k{+}6k\epsilon{+}5ki\delta{+}zi\delta{-}2ki\delta'$
bins: $10k$

We then use Op-Comp with $x = \frac{1}{2}{-}6k\delta'$, $y = \frac{1}{2}{-}4k\delta'$, $h = 2k$, and $s = 2k$. I.e., we release items of increasing size from $\frac{1}{2}{+}4k\delta'$ to $\frac{1}{2}{+}6k\delta'$, and at any time, at most $4k + 1$ items are needed. The $2k$ bins of load ϵ can pack one such item. Suppose there are w, out of $2k$, ϵ-bins that are not packed with a $(\frac{1}{2}{+}i\delta')$-item. According to Op-Comp, $\mathcal{A}$ has to open $2k{+}w$ new bins.

Stage 4. In this stage, we aim at $n_4 \geq 2k{-}w$. We use Op-Comp which corresponds to Op-Inc in Stage 1. We let items depart until the configuration is

$$\{3k{:}(\frac{1}{4}{+}\epsilon)_{*\epsilon}, k{-}z{:}(\frac{1}{4}{+}\epsilon)_{*\epsilon}, z{:}(\epsilon, \frac{1}{4}{+}i\delta), 2k{-}w{:}(\epsilon, \frac{1}{2}{+}i\delta'), w{:}\epsilon, 2k{:}\frac{1}{4}{+}i\delta,$$
$$2k{:}\frac{1}{2}{-}i\delta', 2k{+}w{:}\frac{1}{2}{+}i\delta'\} \ ,$$

with $12k + w$ bins and a total load of $9k/2 + O(1)$.

# bins	$3k$	$k{-}z$	z	$2k{-}w$	w	$2k$	$2k$	$2k{+}w$
$\mathcal{A}$	$(\frac{1}{4}{+}\epsilon)_{*\epsilon}$	$(\frac{1}{4}{+}\epsilon)_{*\epsilon}$	$\epsilon, \frac{1}{4}{+}i\delta$	$\epsilon, \frac{1}{2}{+}i\delta'$	ϵ	$\frac{1}{4}{+}i\delta$	$\frac{1}{2}{-}i\delta'$	$\frac{1}{2}{+}i\delta'$

total size: $\frac{9k}{2}{+}6k\epsilon{+}2ki\delta{+}zi\delta{+}2ki\delta'$
bins: $12k{+}w$

We then use Op-Comp with $x = \frac{1}{4}{+}\delta$, $y = \frac{1}{4}{+}2k\delta$, $h = w$, and $s = 2k - w$. I.e., we release items of sizes from $\frac{3}{4}{-}2k\delta$ to $\frac{3}{4}{-}\delta$ and at any time, at most $2k + 1$ items are needed. Only w ϵ-bins can pack such item, i.e., $h = w$ as said. According to Op-Comp, $\mathcal{A}$ has to open $2k{-}w$ new bins.

Stage 5. In this final stage, we aim at $n_5 = 2k$. We let items depart until the configuration is

$$\{3k{:}\epsilon, k{-}z{:}\epsilon, z{:}\epsilon, 2k{-}w{:}\epsilon, w{:}\epsilon, 2k{:}\frac{1}{4}{+}i\delta, 2k{:}\frac{1}{2}{-}i\delta', 2k{+}w{:}\frac{1}{2}{+}i\delta', 2k{-}w{:}\frac{3}{4}{-}i\delta\} \ ,$$

with $14k$ bins and a total load of $4k - \frac{w}{4} + O(1)$.

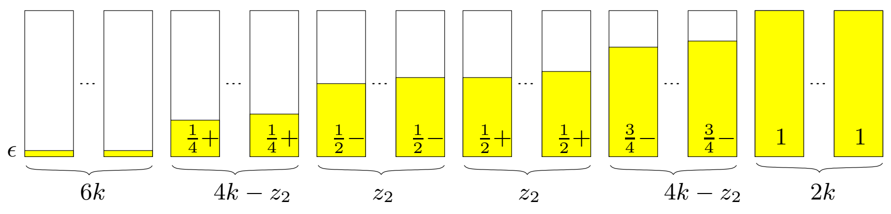


Figure 3.9: The final configuration of $\mathcal{A}$ achieved by the adversary in Case 2.2.2.

# bins	$\boldsymbol{3k}$	$\boldsymbol{k-z}$	$\boldsymbol{z}$	$\boldsymbol{2k-w}$	$\boldsymbol{w}$	$\boldsymbol{2k}$	$\boldsymbol{2k}$	$\boldsymbol{2k+w}$	$\boldsymbol{2k-w}$
$\mathcal{A}$	ϵ	ϵ	ϵ	ϵ	ϵ	$\frac{1}{4}+i\delta$	$\frac{1}{2}-i\delta'$	$\frac{1}{2}+i\delta'$	$\frac{3}{4}-i\delta$
total size: $4k+6k\epsilon+wi\delta+wi\delta'-\frac{w}{4}$									
# bins: $14k$									

Finally, we release $2k$ items of size 1 and $\mathcal{A}$ has to open $2k$ new bins. In total, $\mathcal{A}$ uses $3k+(k-z)+z+(2k-w)+w+2k+2k+(2k+w)+(2k-w)+2k = 16k$ bins. Figure 3.8 shows the final configuration of $\mathcal{A}$. The packing of $\mathcal{O}$ is given in Section 3.6.

# bins	$\boldsymbol{3k}$	$\boldsymbol{k-z}$	$\boldsymbol{z}$	$\boldsymbol{2k-w}$	$\boldsymbol{w}$	$\boldsymbol{2k}$	$\boldsymbol{2k}$	$\boldsymbol{2k+w}$	$\boldsymbol{2k-w}$	$\boldsymbol{2k}$
$\mathcal{A}$	ϵ	ϵ	ϵ	ϵ	ϵ	$\frac{1}{4}+i\delta$	$\frac{1}{2}-i\delta'$	$\frac{1}{2}+i\delta'$	$\frac{3}{4}-i\delta$	1
total size: $6k+6k\epsilon+wi\delta+wi\delta'-\frac{w}{4}$										
# bins: $16k$										

Lemma 3.7. *If $\mathcal{A}$ uses $[6k, 8k)$ bins in Stage 0, $[2k, 4k)$ bins in Stage 1, and $m \geq 2k$, then $\mathcal{A}$ uses $16k$ bins at the end while $\mathcal{O}$ uses $6k+3$ bins.*

Case 2.2.2: $\boldsymbol{6k \leq n_0 < 8k}$, $\boldsymbol{2k \leq n_1 < 4k}$ and $\boldsymbol{m < 2k}$.

We recall that in Stage 1, $15k$ items of size $\frac{1}{4}+i\delta$ are released and $\mathcal{A}$ uses $[2k, 4k)$ new bins for these items.

Observation 3.8. (i) At most $8k$ items of size $\frac{1}{4}+i\delta$ can be packed to the n_1 new bins. (ii) At least k of the following bins have load more than $\frac{1}{2}$: $\{k{:}(\frac{1}{4}+\epsilon)_{*\epsilon}, k{:}\epsilon\}$. (iii) At least $2k$ of the $\{4k{:}(\frac{1}{2}+\epsilon)_{*\epsilon}\}$ bins are packed with at least one $(\frac{1}{4}+i\delta)$-item.

Proof. (i) By the definition of m, we note m new bins can pack at most three items and $n_1 - m$ can pack at most one item. Therefore, the total number of items that can be packed is at most $n_1+2m < 4k+2m < 8k$ because $m < 2k$. (ii) By (i), at least $7k$ items have to be packed in existing bins. Each of the first $4k$ bins can pack at most one item, meaning that at least $3k$ items have to be packed in $\{k{:}(\frac{1}{4}+\epsilon)_{*\epsilon}, k{:}\epsilon\}$. Suppose there are a of the $((\frac{1}{4}+\epsilon)_{*\epsilon})$-bins and b of the ϵ-bins that have load at least $\frac{1}{2}$. Therefore, we require that $2a+3b+(k-b) \geq 3k$ or equivalently $a+b \geq k$. (iii) We note that at most $2k$ items can be packed in $((\frac{1}{4}+\epsilon)_{*\epsilon})$-bins and at most $3k$ items to ϵ-bins. Therefore, at least $7k-5k = 2k$ items needed to be packed in the first $4k$ bins, each packing at most one item, implying that there are at least $2k$ such bins. □

Let z_1 and z_2 be the number of new bins that are packed one and at least two, respectively, $(\frac{1}{4}{+}i\delta)$-items. The following observation gives a bound on z.

Observation 3.9. (i) At most $9k$ items of size $\frac{1}{4}{+}i\delta$ can be packed in existing bins. (ii) $z_2 \geq k$. (iii) $z_1 \geq 3(2k - z_2)$.

Proof. (i) We observe that at most $9k$ items can be packed in existing bins because at most $4k$ items can be packed in the first $4k$ bins, $2k$ items in the next k bins and $3k$ items in the next k bins. (ii) At least $7k$ out of $n_0{+}n_1$ bins have load at least $\frac{1}{2}$. Otherwise, the total load that can be packed is less than $7k \times \frac{3}{4} + (10k - 7k) \times \frac{1}{4} = 6k$, contradicting that there is more than $6k$ of load released. (iii) Therefore, at least $15k - 9k = 6k$ items have to be packed in the new bins. So we require that $z_1 + 3z_2 \geq 6k$ and $z_1 \geq 3(2k - z_2)$. ☐

Stage 2. In this stage, we target $n_2 \geq z_2$. Recall that due to Observation 3.8(ii) at least k of the following bins have load more than $\frac{1}{2}$: $\{k{:}(\frac{1}{4}{+}\epsilon)_{*\epsilon}, k{:}\epsilon\}$. Let $k - q$, for $q \leq k$, be the number of bins among the $((\frac{1}{4}{+}\epsilon)_{*\epsilon})$-bins that have load more than $\frac{1}{2}$. Then there are at least q bins among the remaining ϵ-bins that have load more than $\frac{1}{2}$. We let items depart until the configuration becomes

- $2k{:}(\frac{1}{2}{+}\epsilon)_{*\epsilon}$,
- $2k{:}((\frac{1}{4}{+}\epsilon)_{*\epsilon}, \frac{1}{4}{+}i\delta)$, this is possible because of Observation 3.8(iii),
- $q{:}(\epsilon, \frac{1}{4}{+}i\delta, \frac{1}{4}{+}i\delta)$,
- $k{-}q{:}((\frac{1}{4}{+}\epsilon)_{*\epsilon}, \frac{1}{4}{+}i\delta)$, this is possible because of Observation 3.8(ii),
- $k{:}\epsilon$,
- $z_2{:}(\frac{1}{4}{+}i\delta, \frac{1}{4}{+}i\delta)$, this is possible because of Observation 3.9(ii),
- $2(2k{-}z_2){:}(\frac{1}{4}{+}i\delta)$, this is possible because of Observation 3.9(iii),

with $10k - z_2$ bins and a total load of $7k/2 + O(1)$.

# bins	$\mathbf{2k}$	$\mathbf{2k}$	$\mathbf{q}$	$\mathbf{k{-}q}$	$\mathbf{k}$	$\mathbf{z_2}$	$\mathbf{2(2k{-}z_2)}$
$\mathcal{A}$	$(\frac{1}{2}{+}\epsilon)_{*\epsilon}$	$(\frac{1}{4}{+}\epsilon)_{*\epsilon}$, $\frac{1}{4}{+}i\delta$	$\epsilon, \frac{1}{4}{+}i\delta$, $\frac{1}{4}{+}i\delta$	$(\frac{1}{4}{+}\epsilon)_{*\epsilon}$, $\frac{1}{4}{+}i\delta$	ϵ	$\frac{1}{4}{+}i\delta$, $\frac{1}{4}{+}i\delta$	$\frac{1}{4}{+}i\delta$

total size: $\frac{7k}{2}{+}6k\epsilon{+}7ki\delta{+}xi\delta$
bins: $10k{-}z_2$

We then use Op-Inc with $x = \frac{1}{2}{-}5k\delta'$, $h = 5k{-}2z_2$ and $s = z_2$. The x-items can only be packed in k of ϵ-bins and $2(2k - z_2)$ of $(\frac{1}{4}{+}i\delta)$-bins, i.e., $h = k + 2(2k - z_2) = 5k - 2z_2$ as said. Any new bin can pack at most two, implying that Op-Inc releases $5k = h + 2s$ items of increasing sizes from $\frac{1}{2}{-}5k\delta'$ to $\frac{1}{2}{-}\delta'$. According to Op-Inc, $\mathcal{A}$ has to open at least z_2 bins, i.e., $n_2 \geq z_2$. Op-Inc departs items released in the last step such that $n_2 = z_2$.

Stage 3. In this stage, we target $n_3 \geq z_2$. We let items depart until the configuration becomes

$$\{2k{:}(\frac{1}{2}{+}\epsilon)_{*\epsilon}, 2k{:}((\frac{1}{4}{+}\epsilon)_{*\epsilon}, \frac{1}{4}{+}i\delta), q{:}(\epsilon, \frac{1}{4}{+}i\delta, \frac{1}{4}{+}i\delta), k{-}q{:}((\frac{1}{4}{+}\epsilon)_{*\epsilon}, \frac{1}{4}{+}i\delta), k{:}\epsilon,$$

$$z_2{:}(\frac{1}{4}{+}i\delta, \frac{1}{4}{+}i\delta), 2(2k{-}z_2){:}\frac{1}{4}{+}i\delta, z_2{:}\frac{1}{2}{-}i\delta'\} \ ,$$

with $10k$ bins and a total load of $7k/2 + z_2/2 + O(1)$.

# bins	$\boldsymbol{2k}$	$\boldsymbol{2k}$	$\boldsymbol{q}$	$\boldsymbol{k{-}q}$	$\boldsymbol{k}$	$\boldsymbol{z_2}$	$\boldsymbol{2(2k{-}z_2)}$	$\boldsymbol{z_2}$
$\mathcal{A}$	$(\frac{1}{2}{+}\epsilon)_{*\epsilon}$	$(\frac{1}{4}{+}\epsilon)_{*\epsilon}$, $\frac{1}{4}{+}i\delta$	$\epsilon, \frac{1}{4}{+}i\delta$, $\frac{1}{4}{+}i\delta$	$(\frac{1}{4}{+}\epsilon)_{*\epsilon}$, $\frac{1}{4}{+}i\delta$	ϵ	$\frac{1}{4}{+}i\delta$, $\frac{1}{4}{+}i\delta$	$\frac{1}{4}{+}i\delta$	$\frac{1}{2}{-}i\delta'$

total size: $\frac{7k}{2}{+}6k\epsilon{+}7ki\delta{+}xi\delta{+}\frac{z_2}{2}{-}z_2i\delta'$
bins: $10k$

We use Op-Comp with $s = z_2$ to release items of increasing size from $\frac{1}{2}{+}\delta'$. These items can only be packed in ϵ-bins (k of them) and $(\frac{1}{4}{+}i\delta)$-bins ($2(2k - z_2)$ of them). At any time, at most $(5k - z_2) + 1$ items are needed. According to Op-Comp, $\mathcal{A}$ has to open z_2 bins, i.e., $n_3 \geq z_2$.

Stage 4. We target $n_4 \geq (4k - z_2)$. We let items depart until the configuration becomes

$$\{4k{-}q{:}(\frac{1}{4}{+}\epsilon)_{*\epsilon}, k{+}q{:}(\epsilon, \frac{1}{4}{+}i\delta), k{:}\epsilon, 4k{-}z_2{:}\frac{1}{4}{+}i\delta, z_2{:}\frac{1}{2}{-}i\delta', z_2{:}\frac{1}{2}{+}i\delta'\} \ ,$$

with $10k + z_2$ bins and a total load of $9k/4 + 3z_2/4 + O(1)$.

# bins	$\boldsymbol{4k{-}q}$	$\boldsymbol{k{+}q}$	$\boldsymbol{k}$	$\boldsymbol{4k{-}z_2}$	$\boldsymbol{z_2}$	$\boldsymbol{z_2}$
$\mathcal{A}$	$(\frac{1}{4}{+}\epsilon)_{*\epsilon}$	$\epsilon, \frac{1}{4}{+}i\delta$	ϵ	$\frac{1}{4}{+}i\delta$	$\frac{1}{2}{-}i\delta'$	$\frac{1}{2}{+}i\delta'$

total size: $\frac{9k}{4}{+}6k\epsilon{+}5ki\delta{+}xi\delta{+}\frac{3z_2}{4}{-}z_2i\delta$
bins: $10k{+}z_2$

We then use Op-Comp with $s = 4k - z_2$ and items of increasing size $\frac{3}{4}{-}i\delta$. Using similar ideas as before, $\mathcal{A}$ has to open $(4k - z_2)$ new bins.

Stage 5. In this stage, we target $n_5 = 2k$. We let items depart until the configuration becomes

$$\{4k{-}q{:}\epsilon, k{+}q{:}\epsilon, k{:}\epsilon, 4k{-}z_2{:}\frac{1}{4}{+}i\delta, z_2{:}\frac{1}{2}{-}i\delta', z_2{:}\frac{1}{2}{+}i\delta', 4k{-}z_2{:}\frac{3}{4}{-}i\delta, \}\ ,$$

with $14k$ bins and a total load of $4k + O(1)$.

# bins	$4k{-}q$	$k{+}q$	k	$4k{-}z_2$	z_2	z_2	$4k{-}z_2$
$\mathcal{A}$	ϵ	ϵ	ϵ	$\frac{1}{4}{+}i\delta$	$\frac{1}{2}{-}i\delta'$	$\frac{1}{2}{+}i\delta'$	$\frac{3}{4}{-}i\delta$
total size: $4k{+}6k\epsilon$							
# bins: $14k$							

We finally release $2k$ items of size 1 and $\mathcal{A}$ has to open $2k$ new bins. In total $\mathcal{A}$ uses $6k + 8k + 2k = 16k$ bins. Figure 3.9 shows the final configuration of $\mathcal{A}$. The packing of $\mathcal{O}$ is given in Section 3.6.

# bins	$4k{-}q$	$k{+}q$	k	$4k{-}z_2$	z_2	z_2	$4k{-}z_2$	$2k$
$\mathcal{A}$	ϵ	ϵ	ϵ	$\frac{1}{4}{+}i\delta$	$\frac{1}{2}{-}i\delta'$	$\frac{1}{2}{+}i\delta'$	$\frac{3}{4}{-}i\delta$	1
total size: $6k{+}6k\epsilon$								
# bins: $16k$								

Lemma 3.10. *If $\mathcal{A}$ uses $[6k, 8k)$ bins in Stage 0, $[2k, 4k)$ bins in Stage 1, and $m < 2k$, then $\mathcal{A}$ uses $16k$ bins at the end while $\mathcal{O}$ uses $6k + 5$ bins.*

Theorem 3.11. *No online algorithm can be better than 8/3-competitive.*

3.6 The optimal offline packing

In this section we discuss how the optimal offline algorithm $\mathcal{O}$ packs items in different stages for the complex cases in the previous section. We list in Tables 3.4–3.6 the packing configuration in each stage. Note that for stages involving Op-Inc and Op-Comp, we list the packing when the maximum amount of load is released in that stage (as each stage may involve item arrival and departure). In each table, for each stage the first row is the configuration just before items arrival and the second row is the configuration when the maximum amount of load is released. The very last row is the final configuration. Bolded entries are new items arrived in the corresponding stage. The notation y_{*z} means packing a bin with a load y of z-items. The last column shows the total load of all bins.

Case 2.1: We note that each item with size $\frac{1}{4}{+}i\delta$ in Stage 1 has a corresponding item $\frac{3}{4}{-}i\delta$ in Stage 2 such that the sum of sizes is 1. Table 3.4 lists the optimal offline schedule.

Case 2.2.1: We note that each item with size $\frac{1}{4}{+}i\delta$ in Stage 1 has a corresponding item $\frac{3}{4}{-}i\delta$ in Stage 4 and each item with size $\frac{1}{2}{-}i\delta'$ in Stage 2 has a corresponding item $\frac{1}{2}{+}i\delta'$ in Stage 3 such that the sum of sizes is 1. Table 3.5 lists the optimal offline schedule.

$\boldsymbol{\mathcal{O}}$ **: # bins**	**3**	**1**	$\boldsymbol{4k}$	$\boldsymbol{k}$	$\boldsymbol{k}$	**total**
Stage 0	-	-	-	-	-	0
	$\mathbf{1}_{*\epsilon}$	$\mathbf{1}_{*\epsilon}$	$\boldsymbol{(1{-}4\epsilon)_{*\epsilon}}$	$\boldsymbol{(1{-}5\epsilon)_{*\epsilon}}$	$\boldsymbol{(1{-}3\epsilon)_{*\epsilon}}$	$6k$
Stage 1	$1_{*\epsilon}$	$1_{*\epsilon}$	$(\frac{1}{4}{-}3\epsilon)_{*\epsilon}$	$(\frac{1}{4}{-}3\epsilon)_{*\epsilon}$	$(1{-}3\epsilon)_{*\epsilon}$	$\frac{9k}{4}{+}6k\epsilon$
(Op-Inc)	$1_{*\epsilon}$	$1_{*\epsilon}$	$(\frac{1}{4}{-}3\epsilon)_{*\epsilon}$, $\boldsymbol{3{\times}(\frac{1}{4}{+}i\delta)}$	$(\frac{1}{4}{-}3\epsilon)_{*\epsilon}$, $\boldsymbol{3{\times}(\frac{1}{4}{+}i\delta)}$	$(1{-}3\epsilon)_{*\epsilon}$	$6k{+}6k\epsilon{+}15ki\delta$
Stage 2	$\frac{2}{3}_{*\epsilon}$	-	$\frac{1}{4}{+}i\delta$	$(\frac{1}{4}{-}3\epsilon)_{*\epsilon}$	$(1{-}3\epsilon)_{*\epsilon}$	$\frac{9k}{4}{+}6k\epsilon{+}4ki\delta$
(Op-Comp)	$\frac{2}{3}_{*\epsilon}$	$\boldsymbol{\frac{3}{4}{-}i\delta}$	$\frac{1}{4}{+}i\delta$,$\boldsymbol{\frac{3}{4}{-}i\delta}$	$(\frac{1}{4}{-}3\epsilon)_{*\epsilon}$,$\boldsymbol{\frac{3}{4}{-}i\delta}$	$(1{-}3\epsilon)_{*\epsilon}$	$6k{+}6k\epsilon{-}ki\delta{+}(\frac{3}{4}{-}i\delta)$
Stage 3	$\frac{1}{3}_{*\epsilon}$	-	$\frac{1}{4}{+}i\delta$, $\frac{3}{4}{-}i\delta$	-	-	$4k{+}6k\epsilon$
(Final)	$\frac{1}{3}_{*\epsilon}$	-	$\frac{1}{4}{+}i\delta$, $\frac{3}{4}{-}i\delta$	**1**	**1**	$6k{+}6k\epsilon$

Table 3.4: The optimal schedule for Case 2.1.

Case 2.2.2: We note that each item with size $\frac{1}{4}{+}i\delta$ in Stage 1 has a corresponding item $\frac{3}{4}{-}i\delta$ in Stage 4 and each item with size $\frac{1}{2}{-}i\delta'$ in Stage 2 has a corresponding item $\frac{1}{2}{+}i\delta'$ in Stage 3 such that the sum of sizes is 1. Table 3.6 lists the optimal offline schedule.

3.7 Conclusion

We have derived an $8/3 \sim 2.666$ lower bound on the competitive ratio for dynamic bin packing, improving the best known 2.5 lower bound [19]. We designed two operations that release items of slightly increasing sizes and items with complementary sizes. These operations make a more systematic approach to release items: the type of item sizes used in a later case is a superset of those used in an earlier case. This is in contrast to the previous 2.5 lower bound in [19] in which rather different sizes are used in different cases. Furthermore, in each case, we use one or two pairs of Op-Inc and Op-Comp, which makes the structure clearer and the proof easier to understand. We also show that the new operations defined lead to a much easier proof for a 2.5 lower bound. An obvious open problem is to close the gap between the upper and lower bounds.

$\boldsymbol{\mathcal{O}}$ **: # bins**	$\boldsymbol{2}$	$\boldsymbol{1}$	$\boldsymbol{z}$	$\boldsymbol{2k-w-z}$	$\boldsymbol{w}$	$\boldsymbol{k-w}$	$\boldsymbol{k}$	$\boldsymbol{w}$	$\boldsymbol{2k}$	**total**
Stage 0	-	-	-	-	-	-	-	-	-	0
	$\boldsymbol{1_{*\epsilon}}$	$\boldsymbol{1_{*\epsilon}}$	$\boldsymbol{(1-3\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-3\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-3\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-3\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-3\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-3\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-3\epsilon)_{*\epsilon}}$	$6k$
Stage 1	$1_{*\epsilon}$	$1_{*\epsilon}$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$	$(\frac{3}{4}-2\epsilon)_{*}\epsilon$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$	$\frac{9k}{4}+6k\epsilon$
(Op-Inc-1)	$1_{*\epsilon}$	$1_{*\epsilon}$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{4}+i\delta, \frac{1}{4}+i\delta}$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{4}+i\delta, \frac{1}{4}+i\delta}$	$(\frac{3}{4}-2\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{4}+i\delta}$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{4}+i\delta, \frac{1}{4}+i\delta}$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\boldsymbol{3\times(\frac{1}{4}+i\delta)}$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\boldsymbol{3\times(\frac{1}{4}+i\delta)}$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\boldsymbol{3\times(\frac{1}{4}+i\delta)}$	$6k+6k\epsilon+15ki\delta$
Stage 2	$1_{*\epsilon}$	$\frac{1}{3}_{*\epsilon}$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$	$(\frac{3}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta, \frac{1}{4}+i\delta$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $3\times(\frac{1}{4}+i\delta)$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	-	$3k+6k\epsilon+5ki\delta$ $+zi\delta$
(Op-Inc-2)	$1_{*\epsilon}$	$\frac{1}{3}_{*\epsilon}$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$, $\boldsymbol{\frac{1}{2}-i\delta'}$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{2}-i\delta'}$	$(\frac{3}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta, \frac{1}{4}+i\delta$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $3\times(\frac{1}{4}+i\delta)$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$, $\boldsymbol{\frac{1}{2}-i\delta'}$	$\boldsymbol{\frac{1}{2}-i\delta', \frac{1}{2}-i\delta'}$	$6k+6k\epsilon+5ki\delta$ $+zi\delta-6ki\delta'$
Stage 3	$1_{*\epsilon}$	$\frac{1}{3}_{*\epsilon}$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$	$(\frac{3}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta, \frac{1}{4}+i\delta$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $3\times(\frac{1}{4}+i\delta)$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$\frac{1}{2}-i\delta'$	$4k+6k\epsilon+5ki\delta$ $+zi\delta-2ki\delta'$
(Op-Comp-2)	$1_{*\epsilon}$	$\frac{1}{3}_{*\epsilon}$, $\boldsymbol{\frac{1}{2}+i\delta'}$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$, $\boldsymbol{\frac{1}{2}+i\delta'}$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{2}+i\delta'}$	$(\frac{3}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta, \frac{1}{4}+i\delta$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $3\times(\frac{1}{4}+i\delta)$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$, $\boldsymbol{\frac{1}{2}+i\delta'}$	$\frac{1}{2}-i\delta'$, $\boldsymbol{\frac{1}{2}+i\delta'}$	$6k+6k\epsilon+5ki\delta$ $+zi\delta+2ki\delta'+(\frac{1}{4}+i\delta)$
Stage 4	$\frac{5}{6}_{*\epsilon}$	-	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$, $\frac{1}{2}+i\delta'$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$, $\frac{1}{2}+i\delta'$	$\frac{1}{4}_{*}\epsilon$	$\frac{1}{4}+i\delta$	$\frac{1}{4}+i\delta$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$, $\frac{1}{2}+i\delta'$	$\frac{1}{2}-i\delta', \frac{1}{2}+i\delta'$	$\frac{9k}{2}+6k\epsilon+2ki\delta$ $+zi\delta+2ki\delta'$
(Op-Comp-1)	$\frac{5}{6}_{*\epsilon}$	$\boldsymbol{\frac{3}{4}-i\delta}$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$, $\frac{1}{2}+i\delta'$	$(\frac{1}{2}-2\epsilon)_{*}\epsilon$, $\frac{1}{2}+i\delta'$	$\frac{1}{4}_{*}\epsilon$, $\boldsymbol{\frac{3}{4}-i\delta}$	$\frac{1}{4}+i\delta$, $\boldsymbol{\frac{3}{4}-i\delta}$	$\frac{1}{4}+i\delta$, $\boldsymbol{\frac{3}{4}-i\delta}$	$(\frac{1}{4}-2\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$, $\frac{1}{2}+i\delta'$	$\frac{1}{2}-i\delta', \frac{1}{2}+i\delta'$	$6k+6k\epsilon+zi\delta$ $+2ki\delta'+(\frac{3}{4}-i\delta)$
Stage 5	$\frac{1}{2}_{*\epsilon}$	-	-	-	-	$\frac{1}{4}+i\delta, \frac{3}{4}-i\delta$	$\frac{1}{4}+i\delta, \frac{3}{4}-i\delta$	$\frac{1}{4}+i\delta, \frac{1}{2}+i\delta'$	$\frac{1}{2}-i\delta', \frac{1}{2}+i\delta'$	$4k+6k\epsilon+wi\delta$ $+wi\delta'-\frac{w}{4}$
(Final)	$\frac{1}{2}_{*\epsilon}$	-	$\boldsymbol{1}$	$\boldsymbol{1}$	$\boldsymbol{1}$	$\frac{1}{4}+i\delta, \frac{3}{4}-i\delta$	$\frac{1}{4}+i\delta, \frac{3}{4}-i\delta$	$\frac{1}{4}+i\delta, \frac{1}{2}+i\delta'$	$\frac{1}{2}-i\delta', \frac{1}{2}+i\delta'$	$6k+6k\epsilon+wi\delta$ $+wi\delta'-\frac{w}{4}$

Table 3.5: The optimal schedule for Case 2.2.1.

$\boldsymbol{\mathcal{O}}$ **: # bins**	$\mathbf{4}$	$\mathbf{1}$	$\boldsymbol{z_2}$	$\boldsymbol{4k-2z_2}$	$\boldsymbol{z_2}$	$\boldsymbol{q}$	$\boldsymbol{k-q}$	$\boldsymbol{q}$	$\boldsymbol{z_2-k-q}$	$\boldsymbol{2k-z_2}$	**total**
Stage 0	-	-	-	-	-	-	-	-	-	-	0
	$\boldsymbol{1_{*\epsilon}}$	-	$\boldsymbol{(1-4\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-4\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-4\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-4\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-4\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-4\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-4\epsilon)_{*\epsilon}}$	$\boldsymbol{(1-4\epsilon)_{*\epsilon}}$	$6k$
Stage 1	$1_{*\epsilon}$	-	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$	$(\frac{3}{4}-3\epsilon)_{*}\epsilon$	$(\frac{3}{4}-3\epsilon)_{*}\epsilon$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$	$\frac{9k}{4}+6k\epsilon$
(Op-Inc-1)	$1_{*\epsilon}$	-	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\boldsymbol{3\times(\frac{1}{4}+i\delta)}$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\boldsymbol{3\times(\frac{1}{4}+i\delta)}$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\boldsymbol{3\times(\frac{1}{4}+i\delta)}$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{4}+i\delta, \frac{1}{4}+i\delta}$	$(\frac{3}{4}-3\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{4}+i\delta}$	$(\frac{3}{4}-3\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{4}+i\delta}$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{4}+i\delta, \frac{1}{4}+i\delta}$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{4}+i\delta, \frac{1}{4}+i\delta}$	$6k+6k\epsilon+15ki\delta$
Stage 2	$1_{*\epsilon}$	-	$(\frac{1}{4}-6\epsilon)_{*}\epsilon$, $3\times(\frac{1}{4}+i\delta)$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	-	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta, \frac{1}{4}+i\delta$	$(\frac{3}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$\frac{7k}{2}+6k\epsilon+7ki\delta$ $+xi\delta$
(Op-Inc-2)	$1_{*\epsilon}$	-	$(\frac{1}{4}-6\epsilon)_{*}\epsilon$, $3\times(\frac{1}{4}+i\delta)$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$,$\boldsymbol{\frac{1}{2}-i\delta'}$	$\boldsymbol{\frac{1}{2}-i\delta', \frac{1}{2}-i\delta'}$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta, \frac{1}{4}+i\delta$	$(\frac{3}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{2}-i\delta'}$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{2}-i\delta'}$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$, $\boldsymbol{\frac{1}{2}-i\delta'}$	$6k+6k\epsilon+7ki\delta$ $+xi\delta-5ki\delta'$
Stage 3	$1_{*\epsilon}$	-	$(\frac{1}{4}-6\epsilon)_{*}\epsilon$, $3\times(\frac{1}{4}+i\delta)$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$\frac{1}{2}-i\delta'$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta, \frac{1}{4}+i\delta$	$(\frac{3}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$\frac{7k}{2}+6k\epsilon+7ki\delta$ $+xi\delta+\frac{z_2}{2}-z_2i\delta'$
(Op-Comp-2)	$1_{*\epsilon}$	$\boldsymbol{\frac{1}{2}+i\delta'}$	$(\frac{1}{4}-6\epsilon)_{*}\epsilon$, $3\times(\frac{1}{4}+i\delta)$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$, $\boldsymbol{\frac{1}{2}+i\delta'}$	$\frac{1}{2}-i\delta'$, $\boldsymbol{\frac{1}{2}+i\delta'}$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta, \frac{1}{4}+i\delta$	$(\frac{3}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{2}+i\delta'}$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{1}{2}+i\delta'}$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$, $\boldsymbol{\frac{1}{2}+i\delta'}$	$6k+6k\epsilon+7ki\delta$ $+xi\delta-2z_2i\delta'$ $+5ki\delta'+(\frac{1}{2}+i\delta')$
Stage 4	$\frac{1}{2}_{*\epsilon}$	-	$\frac{1}{4}+i\delta$	$\frac{1}{4}+i\delta$	$\frac{1}{2}-i\delta', \frac{1}{2}+i\delta'$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta, \frac{1}{4}+i\delta$	$(\frac{3}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$	$\frac{9k}{4}+6k\epsilon+5ki\delta$ $+xi\delta+\frac{3z_2}{4}-z_2i\delta$
(Op-Comp-1)	$\frac{1}{2}_{*\epsilon}$	$\boldsymbol{\frac{3}{4}-i\delta}$	$\frac{1}{4}+i\delta$, $\boldsymbol{\frac{3}{4}-i\delta}$	$\frac{1}{4}+i\delta$, $\boldsymbol{\frac{3}{4}-i\delta}$	$\frac{1}{2}-i\delta', \frac{1}{2}+i\delta'$	$(\frac{1}{2}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta, \frac{1}{4}+i\delta$	$(\frac{3}{4}-3\epsilon)_{*}\epsilon$, $\frac{1}{4}+i\delta$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{3}{4}-i\delta}$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{3}{4}-i\delta}$	$(\frac{1}{4}-3\epsilon)_{*}\epsilon$, $\boldsymbol{\frac{3}{4}-i\delta}$	$6k+6k\epsilon+xi\delta$ $+(\frac{3}{4}-i\delta)$
Stage 5	$\frac{1}{4}_{*\epsilon}$	-	$\frac{1}{4}+i\delta$, $\frac{3}{4}-i\delta$	$\frac{1}{4}+i\delta$, $\frac{3}{4}-i\delta$	$\frac{1}{2}-i\delta', \frac{1}{2}+i\delta'$	-	-	-	-	-	$4k+6k\epsilon$
(Final)	$\frac{1}{4}_{*\epsilon}$	-	$\frac{1}{4}+i\delta$, $\frac{3}{4}-i\delta$	$\frac{1}{4}+i\delta$, $\frac{3}{4}-i\delta$	$\frac{1}{2}-i\delta', \frac{1}{2}+i\delta'$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$6k+6k\epsilon$

Table 3.6: The optimal schedule for Case 2.2.2.

Chapter 4

Online Two-dimensional Dynamic Bin Packing of Unit Fraction Items

4.1 Introduction

In this chapter, we design and analyze algorithms for online two-dimensional dynamic bin packing of unit fraction items. Recall that general size items have lengths that are real numbers in $(0, 1]$, unit fraction items have lengths of the form $1/k$, for some integer $k > 0$, and power fraction items have lengths of the form $1/2^k$, for some integer $k \geq 0$. In multi-dimensional bin packing, items with lengths unit fraction or power fraction in each dimension must be packed into multi-dimensional bins with lengths unit-size in each dimension. In dynamic bin packing, items arrive over time, reside for some period of time, and may depart at arbitrary time. Each item has to be assigned to a bin from the time it arrives until it departs. The objective is to minimize the maximum number of bins used over all time. Note that migration to another bin and rotation of items are not allowed, but rearrangement of items within a bin is permitted. In the online setting, the size and arrival time is only known when an item arrives and the departure time is only known when the item departs. As in Chapter 3, we measure the performance of an online algorithm using competitive analysis [11] (detailed in Section 1.1.3 and recalled in Chapter 3).

We provide a new algorithm for online two-dimensional dynamic bin packing of unit fraction items and show that its worst-case performance is better than its classical counterpart for general size items. The results have been published in Proceedings of the 8th International Conference on Algorithms and Complexity (CIAC), 2013 [15]. In Chapter 5 we continue our analysis by designing and analyzing algorithms for two-dimensional dynamic bin packing of power fraction items, and three-dimensional dynamic bin packing

	1-D	2-D	3-D
General size	2.788 [26]	7.788 [70]	22.788 [70]
Unit fraction	2.4842 [47]	6.7850 [*]	21.6108 [**]
Power fraction	2.4842 [47]	6.2455 [**]	20.0783 [**]

Table 4.1: Competitive ratios for general size, unit fraction, and power fraction items. The results obtained in this chapter is marked with "[*]". The results obtained in Chapter 5 are marked with "[**]".

of unit and power fraction items. While these results were described in [15], their analysis was omitted. In this chapter we also introduce power fraction items and notation used in Chapter 5.

Previous results. The dynamic bin packing problem was first studied in 1-D for general size items by Coffman, Garey and Johnson [26], showing that the First-Fit (FF) algorithm has a competitive ratio lying between 2.75 and 2.897, and a modified First-Fit algorithm is 2.788-competitive. They gave a formula of the competitive ratio of FF when the item size is at most $\frac{1}{k}$. When $k = 2$ and 3, the ratios are 1.7877 and 1.459, respectively. They also gave a lower bound of 2.388 for any deterministic online algorithm, which was improved to 2.428 [18], 2.5 [19], and then to 2.666 (detailed in Chapter 3 and published in [71]). For unit fraction items, Chan et al. [18] obtained a competitive ratio of 2.4942, which was recently improved by Han et al. to 2.4842 [47], while the lower bound was proven to be 2.428 [18]. Multi-dimensional dynamic bin packing of general size items was initially studied by Epstein and Levy [37], who showed that the competitive ratios are 8.5754, 35.346 and $2 \cdot 3.5^d$ for 2-D, 3-D and d-D respectively. The ratios are then improved to 7.788, 22.788, and 3^d correspondingly by Wong and Yung [70]. For 2-D and 3-D general size items, the lower bounds are 3.70301 and 4.85383 [37], respectively. In the construction of the adversaries in [37], only items with lengths unit fraction are used, thus the lower bounds also apply to 2-D and 3-D unit fraction items.

Our contribution. In this chapter, we extend the study of the 2-D online dynamic bin packing problem to unit fraction items. We observe in Section 4.2 that using the 1-D results on unit fraction items [47], the competitive ratio of 7.788 for 2-D [70] naturally becomes 7.4842, while the competitive ratio of 22.788 for 3-D [70] becomes 22.4842. An immediate question arising is whether we can have an even smaller competitive ratio. We answer the questions affirmatively as follows (see Table 4.1 for a summary). In this chapter, for 2-D, we obtain a competitive ratio of 6.7850 for unit fraction items, while in Chapter 5 for 2-D power fraction items we obtain a competitive ratio of 6.2455, while for 3-D we obtain competitive ratios of 21.6108 and 20.0783 for unit and power fraction items, respectively.

We adopt the typical approach of dividing items into classes and analyzing each class individually. We propose several natural classes and define different packing schemes based on the classes[1]. In particular, we show that two schemes lead to better results.

[1]The proposed classes are not necessarily disjoint while a packing scheme is a collection of disjoint classes that cover all types of items.

We show that one scheme is better than the other for unit fraction items, and vice versa for power fraction items (to be detailed in Chapter 5). Our approach gives a systematic way to explore different combinations of classes. One observation we have made is that dividing 2-D items into three classes gives comparable results but dividing into four classes would lead to much higher competitive ratios.

Organization of the chapter. In Section 4.2 we introduce the preliminaries required for discussion. In Section 4.3 we analyze the components of two different classification schemes for unit fraction items, while in Section 4.4 we propose the two classification algorithms for unit fraction items.

4.2 Preliminaries

Notations and definitions. We consider the online dynamic bin packing problem, in which 2-D and 3-D items must be packed into 2-D and 3-D unit-sized bins, respectively, without overflowing. The items arrive over time, reside for some period of time, and may depart at arbitrary times. Each item must be packed into a bin from its arrival to its departure. Migration to another bin is not allowed and the items are oriented and cannot be rotated. Yet, repacking of items within the same bin is permitted. The *load* refers to the total area or volume of a set of 2-D or 3-D items, respectively. The objective of the problem is to minimize the total number of bins used over all time. For both 2-D and 3-D, we consider two types of input: unit fraction and power fraction items.

A *general size item* is an item such that the length in each dimension is in $(0, 1]$. A *unit fraction item* is an item with lengths of the form $\frac{1}{k}$, where $k \geq 1$ is an integer, A *power fraction item* has lengths of the form $\frac{1}{2^k}$, where $k \geq 0$ is an integer.

A packing is said to be *feasible* if all items do not overlap and the packing in each bin does not exceed the boundary of the bin; otherwise, the packing is said to *overflow* and is *infeasible*.

Some of the algorithms discussed in this chapter repack existing items (and possibly include a new item) in a bin to check if the new item can be packed into this bin. If the repacking is infeasible, it is understood that we would restore the packing to the original configuration.

For 2-D items, we use the notation $\mathrm{T}(w, h)$ to refer to the type of items with width w and height h. We use '$*$' to mean that the length can take any value at most 1, e.g., $\mathrm{T}(*, *)$ refers to all items. The parameters w (and h) may take an expression $\leq x$ meaning that the width is at most x. For example, $\mathrm{T}(\frac{1}{2}, \leq\frac{1}{2})$ refers to the items with width $\frac{1}{2}$ and height at most $\frac{1}{2}$. In the following discussion, we divide the items into seven disjoint types as showed in Table 4.2.

The bin assignment algorithm that we use for all types of 2-D and 3-D unit and power fraction items is the First-Fit (FF) algorithm. When a new item arrives, if there are occupied bins in which the item can be repacked, FF assigns the new item to the bin which has been occupied for the longest time.

$\mathrm{T}(1,1)$	$\mathrm{T}(1,\frac{1}{2})$	$\mathrm{T}(\frac{1}{2},1)$	$\mathrm{T}(\frac{1}{2},\frac{1}{2})$	$\mathrm{T}(1,\leq\frac{1}{3})$	$\mathrm{T}(\frac{1}{2},\leq\frac{1}{3})$	$\mathrm{T}(\leq\frac{1}{3},\leq 1)$

Table 4.2: Types of unit fraction items considered.

Scheme in [70]		
Classes	Types of items	Competitive ratios
Class A	$\mathrm{T}(\leq\frac{1}{3},\leq 1)$	3 [70]
Class B	$\mathrm{T}(\frac{1}{2},1), \mathrm{T}(\frac{1}{2},\frac{1}{2}), \mathrm{T}(\frac{1}{2},\leq\frac{1}{3})$	2 [70]
Class C	$\mathrm{T}(1,1), \mathrm{T}(1,\frac{1}{2}), \mathrm{T}(1,\leq\frac{1}{3})$	2.4842 [47]
Overall	All items	7.4842

Table 4.3: The 2-D results of [70] for unit-fraction items.

Remark on existing result on unit fraction items. Using this notation, the algorithm in [70] effectively classifies unit fraction items into the classes as shown in Table 4.3. Items in the same class are packed separately, independent of other classes. The overall competitive ratio is the sum of the competitive ratios of all classes. By the result in [47], the competitive ratio for Class C reduces from 2.788 [26] to 2.4842 [47] and the overall competitive ratio immediately reduces from 7.778 to 7.4842. The algorithm in [70] for 3-D items also uses classification of items based on their lengths. The 2-D algorithm is used as part of the 3-D algorithm for one class of 3-D items that can be projected along the third dimension such that the problem is reduced to 2-D. This results in the particular class of 3-D items achieving the same competitive ratio as the overall competitive ratio for 2-D items. Substituting the reduced 7.4842 competitive ratio for 2-D unit fraction items as part of the classification for 3-D items results in a reduced overall upper bound of 22.4842 for 3-D unit fraction items compared with 22.788 for 3-D general size items.

Corollary 4.1. *The* 2*-D and* 3*-D packing algorithms in* [70] *are* 7.4842*-competitive for* 2*-D unit fraction items and* 22.4842*-competitive for* 3*-D unit fraction items, respectively.*

Repacking. To determine if an item can be packed into an existing bin, we will need some repacking. Here we make some simple observations about the load of items if repacking is not feasible. We first note the following lemma which is implied by Theorem 1.1 in [66].

Lemma 4.2 ([66])**.** *Given a bin with width u and height v, if all items have width at most $\frac{u}{2}$ and height at most v, then any set of these items with total area at most $\frac{uv}{2}$ can fit into the same bin by using Steinberg's algorithm.*

The implication of Lemma 4.2 is that if packing a new item of width $w \leq \frac{u}{2}$ and height h into a bin results in infeasible packing, then the total load of the existing items is at least $\frac{uv}{2} - wh$.

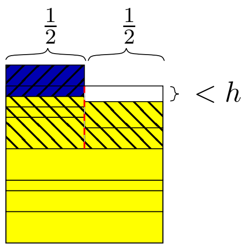


Figure 4.1: Infeasible repacking of existing items of types $\mathrm{T}(1, \leq\frac{1}{3})$ and $\mathrm{T}(\frac{1}{2}, \leq\frac{1}{3})$ and a new item of type $\mathrm{T}(1, *)$. The empty space has width $\frac{1}{2}$ and height less than h.

$\beta\langle x, y\rangle$	$y = 3$	4	5	6
$x = 3$	1	1	1	1
4	$\frac{3}{4}$	$\frac{5}{6}$ $= \frac{1}{3}+\frac{1}{4}+\frac{1}{4}$	$\frac{5}{6}$	$\frac{11}{12}$ $= \frac{2}{3}+\frac{1}{4}$
5	$\frac{7}{10}$ $= \frac{1}{4}+\frac{1}{4}+\frac{1}{5}$	$\frac{47}{60}$ $= \frac{1}{3}+\frac{1}{4}+\frac{1}{5}$	$\frac{5}{6}$	$\frac{17}{20}$ $= \frac{1}{4}+\frac{3}{5}$
6	$\frac{7}{10}$	$\frac{23}{30}$ $= \frac{1}{6}+\frac{3}{5}$	$\frac{49}{60}$ $= \frac{1}{4}+\frac{1}{6}+\frac{2}{5}$	$\frac{17}{20}$

Table 4.4: Values of $\beta\langle x, y\rangle$ for $3 \leq x \leq 6$ and $3 \leq y \leq 6$.

Lemma 4.3. *Consider packing of two types of items* $\mathrm{T}(\frac{1}{2}, \leq h)$ *and* $\mathrm{T}(1, *)$*, for some* $0 < h < 1$*. If we have an item of type* $\mathrm{T}(1, h')$ *that cannot be packed to an existing bin, then the current load of the bin is at least* $1 - \frac{h}{2} - h'$*.*

Proof. We first pack all items with width 1, including the new type $\mathrm{T}(1, h')$ item, one by one on top of the previous one. For the remaining space, we divide it into two equal halves each with width $\frac{1}{2}$. We then try to pack the $\mathrm{T}(\frac{1}{2}, \leq h)$ items into one compartment until it overflows, and then continue packing into the other compartment. The space left in the second compartment has a height less than h, otherwise, the overflow item can be packed there (see Figure 4.1). As a result, the total load of items is at least $1 - \frac{h}{2}$. Since the new item has a load of h', the total load of existing items is at least $1 - \frac{h}{2} - h'$ as claimed. □

In the case of 1-D packing, Chan et al. [18] have defined the following notion. Let x and y be positive integers. Suppose that a 1-D bin is already packed with some items whose sizes are chosen from the set $\{1, \frac{1}{2}, \ldots, \frac{1}{x}\}$. They defined the notion of the minimum load of such a bin that an additional item of size $\frac{1}{y}$ cannot fit into the bin. We modify this notion such that the set in concern becomes $\{\frac{1}{3}, \frac{1}{4}, \ldots, \frac{1}{x}\}$. We define $\beta\langle x, y\rangle$ to be the minimum load of this bin containing items with length at least $\frac{1}{x}$ and at most $\frac{1}{3}$ such that an item of size $\frac{1}{y}$ cannot be packed into this bin. Precisely,

$$\beta\langle x, y\rangle = \min_{3\leq j\leq x \text{ and } n_j\geq 0} \{\frac{n_3}{3} + \frac{n_4}{4} + \ldots + \frac{n_x}{x} \mid \frac{n_3}{3} + \frac{n_4}{4} + \ldots + \frac{n_x}{x} > 1 - \frac{1}{y}\}.$$

Table 4.4 shows the values of this function for $3 \leq x \leq 6$ and $3 \leq y \leq 6$.

Classes	Types of items	Competitive ratio
Class 1	$\mathrm{T}(\leq\frac{1}{3}, \leq 1)$	2.8258
Class 2	$\mathrm{T}(1, \leq\frac{1}{3})$, $\mathrm{T}(\frac{1}{2}, \leq\frac{1}{3})$	1.7804
Class 3	$\mathrm{T}(1,1)$, $\mathrm{T}(1,\frac{1}{2})$, $\mathrm{T}(\frac{1}{2},1)$, $\mathrm{T}(\frac{1}{2},\frac{1}{2})$	2.25
Class 4	$\mathrm{T}(1,\frac{1}{2})$, $\mathrm{T}(1,\leq\frac{1}{3})$, $\mathrm{T}(\frac{1}{2},\frac{1}{2})$, $\mathrm{T}(\frac{1}{2},\leq\frac{1}{3})$	2.4593
Class 5	$\mathrm{T}(1,1)$, $\mathrm{T}(\frac{1}{2},1)$	1.5

Table 4.5: Classifications of 2-D unit fraction items and their competitive ratios.

4.3 Classification of 2-D unit fraction items

Following the idea in [70], we also divide the type of items into classes. In Table 4.5, we list the different classes we considered in this chapter. We propose two packing schemes, each of which makes use of a subset of the classes that are disjoint. The competitive ratio of a packing scheme is the sum of the competitive ratio we can achieve for each of the classes in the scheme. In this section, we focus on individual classes and in the next section, we discuss the two packing schemes. For each class, we use FF (First-Fit) to determine which bin to assign an item. For each bin, we check if the new item can be packed together with the existing items in the bin; this is done by some repacking procedures and the repacking is different for different classes. Using a similar idea as in [23], one can show that the maximum number of bins used by FF is achieved when the items released are in order of non-decreasing area size. More specifically, suppose $\mathrm{T}(w_1, h_1)$ and $\mathrm{T}(w_2, h_2)$ are two types of items given such that $w_1 \times h_1 < w_2 \times h_2$. We denote by $x_1 \geq 0$ the last bin that FF ever packs a $\mathrm{T}(w_1, h_1)$-item, and by $x_1 + x_2$, where x_2 may be a negative integer, the last bin that FF ever packs a $\mathrm{T}(w_2, h_2)$-item. Then the maximum number of bins used by FF is achieved if and only if $x_1 + x_2 \geq x_1$, i.e., x_2 is non-negative [23]. Our analysis in this and the next chapter assumes such sequences.

We start with simpler cases of Classes 5 and 3. We then continue with Class 2 while the analysis of Class 1 applies the result of Class 2 and Class 4 follows the approach of Class 2.

4.3.1 Class 5: $\mathrm{T}(1,1), \mathrm{T}(\frac{1}{2},1)$

We first consider a simple case where the items are either $\mathrm{T}(1,1)$ or $\mathrm{T}(\frac{1}{2},1)$. We show that FF is 1.5-competitive.

Lemma 4.4. *FF is* 1.5*-competitive for unit fraction items of types* $\mathrm{T}(1,1)$ *and* $\mathrm{T}(\frac{1}{2},1)$.

Proof. Suppose the maximum load at any time is n. Then OPT uses at least n bins. Let x_1 be the last bin that FF ever packs an $\mathrm{T}(\frac{1}{2},1)$-item and $x_1 + x_2$ for $\mathrm{T}(1,1)$. When FF packs such a $\mathrm{T}(\frac{1}{2},1)$-item in bin-$x_1$, all the $x_1 - 1$ bins before that must have a load of 1. Therefore, we have $(x_1 - 1) + \frac{1}{2} \leq n$. When FF packs a $\mathrm{T}(1,1)$-item into bin-$(x_1 + x_2)$, the first x_1 bins, which may have $\mathrm{T}(\frac{1}{2},1)$-item, have a load of at least $\frac{1}{2}$ and the next x_2 bins have a load of at least 1 (the latter holds because of the definition of x_1). Therefore, we have $\frac{x_1}{2} + x_2 \le n$. The maximum value of $x_1 + x_2$ is obtained by setting $x_1 = n$ and $x_2 = \frac{n}{2}$, such that $x_1 + x_2 = 1.5n \le 1.5OPT$. $\square$

4.3.2 Class 3: $\mathbf{T(1,1), T(1,\frac{1}{2}), T(\frac{1}{2},1), T(\frac{1}{2},\frac{1}{2})}$

We now consider Class 3 for which both the width and height are at least $\frac{1}{2}$. We show that FF is 2.25-competitive.

Lemma 4.5. *FF is* 2.25*-competitive for unit fraction items of types* $\mathrm{T}(1,1)$*,* $\mathrm{T}(1,\frac{1}{2})$*,* $\mathrm{T}(\frac{1}{2},1)$*,* $\mathrm{T}(\frac{1}{2},\frac{1}{2})$*.*

Proof. Suppose the maximum load at any time is n. Then OPT uses at least n bins. Let x_1 be the last bin that FF ever packs a $\mathrm{T}(\frac{1}{2},\frac{1}{2})$-item, x_1+x_2 for $\mathrm{T}(1,\frac{1}{2})$ and $\mathrm{T}(\frac{1}{2},1)$, and $x_1 + x_2 + x_3$ for $\mathrm{T}(1,1)$. When FF packs a $\mathrm{T}(\frac{1}{2},\frac{1}{2})$-item to bin-$x_1$, all the $x_1 - 1$ before that must have a load of 1. Therefore, $(x_1 - 1) + \frac{1}{4} \le n$. When FF packs a $\mathrm{T}(1,\frac{1}{2})$ or $\mathrm{T}(\frac{1}{2},1)$-item to bin-$(x_1 + x_2)$, all the bins before that must have a load of $\frac{1}{2}$. Hence, $\frac{x_1+x_2}{2} \le n$. When FF packs a $\mathrm{T}(1,1)$-item to bin-$(x_1 + x_2 + x_3)$, the first x_1 bins must have a load of at least $\frac{1}{4}$, the next x_2 bins must have a load of at least $\frac{1}{2}$, and the last $x_3 - 1$ bins must have a load of 1. Therefore, $\frac{x_1}{4} + \frac{x_2}{2} + (x_3 - 1) + 1 \le n$. The maximum value of $x_1 + x_2 + x_3$ is obtained by setting $x_1 = x_2 = n$ and $x_3 = \frac{n}{4}$. Then, $x_1 + x_2 + x_3 = 2.25n \le 2.25OPT$. $\square$

4.3.3 Class 2: $\mathbf{T(1,\le\frac{1}{3}), T(\frac{1}{2},\le\frac{1}{3})}$

In this section, we consider items whose width is at least $\frac{1}{2}$ and height is at most $\frac{1}{3}$. For this class, the repack when a new item arrives is done according to the description in the proof of Lemma 4.3. We are going to show that FF is 1.7804-competitive for Class 2.

Suppose the maximum load at any time is n. Let x_1 be the last bin that FF ever packs a $\mathrm{T}(\frac{1}{2},\le\frac{1}{3})$-item. Using the analysis in [26] for 1-D items with size at most $\frac{1}{3}$, one can show that $x_1 \le 1.4590n$.

Lemma 4.6 ([26])**.** *Suppose we are packing unit-fraction items of type* $\mathrm{T}(1,\le\frac{1}{3})$*,* $\mathrm{T}(\frac{1}{2},\le\frac{1}{3})$ *and the maximum load over time is* n*. We have* $x_1 \le 1.4590n$*, where* x_1 *is the last bin that FF ever packs a* $\mathrm{T}(\frac{1}{2},\le\frac{1}{3})$*-item.*

Lemma 4.6 implies that FF only packs items of $\mathrm{T}(1,\le\frac{1}{3})$ in bin-y for $y > 1.459n$. The following lemma further asserts that the height of these items is at least $\frac{1}{6}$.

Lemma 4.7. *Suppose we are packing unit-fraction items of type* $\mathrm{T}(1,\le\frac{1}{3})$*,* $\mathrm{T}(\frac{1}{2},\le\frac{1}{3})$ *and the maximum load over time is* n*. Any item that is packed by FF to bin-*y*, for* $y > 1.459n$*, must be of type* $\mathrm{T}(1,h)$*, where* $\frac{1}{6} \le h \le \frac{1}{3}$*.*

Proof. Suppose on the contrary that FF packs a $\mathrm{T}(1,\le\frac{1}{7})$-item in bin-$y$ for $y > 1.459n$. This means that packing the item in any of the first $1.459n$ bins results in an infeasible packing. By Lemma 4.3, with $h = \frac{1}{3}$ and $h' = \frac{1}{7}$, the load of each of the first $1.459n$

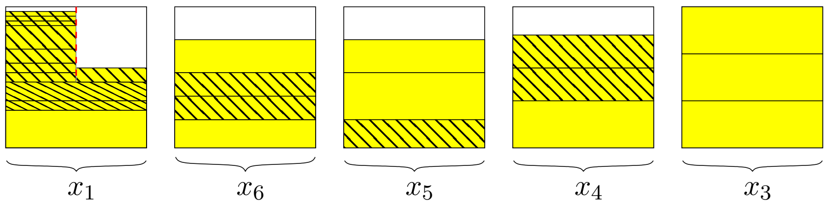


Figure 4.2: Illustration of the proof of Lemma 4.8. In each set of bins, the shaded items are the item types that do not appear in subsequent bins. For example, items of type $T(\frac{1}{2}, \leq \frac{1}{3})$ in the first x_1 bins do not appear in the subsequent bins.

bins is at least $1 - \frac{1}{6} - \frac{1}{7} = 0.69$. Then the total is at least $1.459n \times 0.69 > 1.0067n$, contradicting that the maximum load at any time is n. Therefore, the lemma follows. □

We are now ready to analyze the competitive ratio of FF for this class (Figure 4.2 gives an illustration).

Lemma 4.8. *FF is* 1.7804*-competitive for unit fraction items of types* $T(1, \leq\frac{1}{3})$ *and* $T(\frac{1}{2}, \leq\frac{1}{3})$.

Proof. Let $(x_1+x_6), (x_1+x_6+x_5), (x_1+x_6+x_5+x_4)$, and $(x_1+x_6+x_5+x_4+x_3)$ be the last bin that FF ever packs a $T(1, \frac{1}{6})$-, $T(1, \frac{1}{5})$-, $T(1, \frac{1}{4})$-, and $T(1, \frac{1}{3})$- item, respectively. When FF packs a $T(1, \frac{1}{6})$-item to bin-$(x_1 + x_6)$, the load of the first x_1 bins is at least $1 - \frac{1}{6} - \frac{1}{6} = \frac{2}{3}$, by Lemma 4.3. By Lemma 4.7, only type $T(1, k)$-item, for $\frac{1}{6} \leq k \leq \frac{1}{3}$, could be packed in the x_6 bins. These items all have width 1 and thus can be considered as 1-D case. Therefore, when we cannot pack a $T(1, \frac{1}{6})$-item, the current load must be at least $\beta\langle 6, 6\rangle$. Then we have $x_1(\frac{2}{3}) + x_6\,\beta\langle 6, 6\rangle \leq n$. Similarly, we have

1. $x_1(\frac{2}{3}) + x_6\,\beta\langle 6, 6\rangle \leq n$,
2. $x_1(1 - \frac{1}{6} - \frac{1}{5}) + x_6\,\beta\langle 6, 5\rangle + x_5\,\beta\langle 5, 5\rangle \leq n$,
3. $x_1(1 - \frac{1}{6} - \frac{1}{4}) + x_6\,\beta\langle 6, 4\rangle + x_5\,\beta\langle 5, 4\rangle + x_4\,\beta\langle 4, 4\rangle \leq n$,
4. $x_1(1 - \frac{1}{6} - \frac{1}{3}) + x_6\,\beta\langle 6, 3\rangle + x_5\,\beta\langle 5, 3\rangle + x_4\,\beta\langle 4, 3\rangle + x_3\,\beta\langle 3, 3\rangle \leq n$.

We note that for each inequality, the coefficients are increasing, e.g., for (1), we have $\frac{2}{3} \leq \beta\langle 6, 6\rangle = \frac{17}{20}$, by Table 4.4. Therefore, the maximum value of $x_1+x_6+x_5+x_4+x_3$ is obtained by setting the maximum possible value of x_6 satisfying (1), and then the maximum possible value of x_5 satisfying (2), and so on. Using Table 4.4, we compute the corresponding values as $1.4590n$, $0.0322n$, $0.0597n$, $0.0931n$ and $0.1365n$, respectively. As a result, $x_1 + x_6 + x_5 + x_4 + x_3 \leq 1.7804n \leq 1.7804OPT$. □

4.3.4 Class 1: $T(\leq\frac{1}{3}, \leq 1)$

Items of type $T(\leq\frac{1}{3}, \leq 1)$ are further divided into three subtypes: $T(\leq\frac{1}{3}, \leq\frac{1}{3})$, $T(\leq\frac{1}{3}, \frac{1}{2})$, $T(\leq\frac{1}{3}, 1)$. We first describe how to repack these items.

1. When the new item is $T(\leq\frac{1}{3},\leq\frac{1}{3})$, we use Steinberg's algorithm [66] to repack the new and existing items. Note that the item width satisfies the criteria of Lemma 4.2.

2. When the new item is $T(\leq\frac{1}{3},\frac{1}{2})$ or $T(\leq\frac{1}{3},1)$ and the bin contains $T(\leq\frac{1}{3},\leq\frac{1}{3})$-item, we divide the bin into two compartments, one with width $\frac{1}{3}$ and the other $\frac{2}{3}$ and both with height 1. We reserve the small compartment for the new item and try to repack the existing items in the large compartment using Steinberg's algorithm. This idea originates from [70].

3. When the new item is $T(\leq\frac{1}{3},\frac{1}{2})$ or $T(\leq\frac{1}{3},1)$ and the bin does not contain $T(\leq\frac{1}{3},\leq\frac{1}{3})$-item, we use the repacking method as in Lemma 4.3 but with the width becoming the height and vice versa. Note that this implies that Lemma 4.8 applies for these items.

We now analyze the performance of FF. Suppose the maximum load at any time is n. Let x_1 be the last bin that FF ever packs a $T(\leq\frac{1}{3},\leq\frac{1}{3})$-item, and x_1+x_2 be the last bin that FF ever packs a $T(\leq\frac{1}{3},\frac{1}{2})$ or $T(\leq\frac{1}{3},1)$ item.

Consider x_1. When a $T(\leq\frac{1}{3},\leq\frac{1}{3})$-item cannot be packed using Steinberg's algorithm, the total bin load including the new items is at least $\frac{1}{2}$, by Lemma 4.2. The load of the new item is at most $\frac{1}{9}$, hence, the total load of existing items is at least $\frac{1}{2}-\frac{1}{9}=\frac{7}{18}$. Therefore, we have $(x_1-1)\times\frac{7}{18}\leq n$, implying that $x_1\leq 2.5715n$, for large n. As for the other two subtypes, when FF packs such items into bin-(x_1+x_2), the first x_1 bins have load at least $\frac{1}{3}$ because overflow of the repacking means that existing items cannot be packed in the large compartment; by Lemma 4.2, the total load is at least $\frac{1}{2}\cdot\frac{2}{3}=\frac{1}{3}$. In the next x_2 bins, there could only be items of the two subtypes. Recall how we repack these items in (3) above and that Lemma 4.8 applies. Note that we can have at most a load of $n-\frac{x_1}{3}$ in the x_2 bins and by the proof of Lemma 4.8, this means $x_2\leq 1.7804(n-\frac{x_1}{3})$. For similar reason as before x_1+x_2 takes the maximum when $x_1=2.5715n$ and $x_2=0.2544n$. Therefore, we have the following lemma.

Lemma 4.9. *FF is 2.8258-competitive for unit fraction items of types* $T(\leq\frac{1}{3},\leq 1)$.

4.3.5 Class 4: $\mathbf{T(1,\frac{1}{2}), T(1,\leq\frac{1}{3}), T(\frac{1}{2},\frac{1}{2}), T(\frac{1}{2},\leq\frac{1}{3})}$

The analysis of Class 4 follows a similar framework as in Class 2. Similar to Class 2, the repack when a new item arrives is done according to the description in the proof of Lemma 4.3. We are going to show that FF is 2.4593-competitive for Class 4.

Suppose the maximum load at any time is n. Let x_1 be the last bin that FF ever packs a $T(\frac{1}{2},\leq\frac{1}{2})$-item. Using the analysis in [26] for 1-D items with size at most $\frac{1}{2}$, one can show that $x_1\leq 1.7877n$.

Lemma 4.10 ([26])**.** *Suppose we are packing unit-fraction items of type* $T(1,\leq\frac{1}{2})$, $T(\frac{1}{2},\leq\frac{1}{2})$ *and the maximum load over time is n. We have $x_1\leq 1.7877n$, where x_1 is the last bin that FF ever packs a* $T(\frac{1}{2},\leq\frac{1}{2})$*-item.*

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

Table 4.6: Values of $\gamma\langle x,y\rangle$ for $2\le x\le 6$ and $2\le y\le 6$.

Lemma 4.10 implies that FF only packs items of $\mathrm{T}(1,\le\frac{1}{2})$ in bin-y for $y>1.7877n$. The following lemma further asserts that the height of these items is at least $\frac{1}{5}$.

Lemma 4.11. *Suppose we are packing unit-fraction items of type* $\mathrm{T}(1,\le\frac{1}{2})$, $\mathrm{T}(\frac{1}{2},\le\frac{1}{2})$ *and the maximum load over time is n. Any item that is packed by FF to bin-y, for $y>1.7877n$, must be of type* $\mathrm{T}(1,h)$, *where* $\frac{1}{5}\le h\le\frac{1}{2}$.

Proof. Suppose on the contrary that FF packs a $\mathrm{T}(1,\le\frac{1}{6})$-item in bin-$y$ for $y>1.7877n$. This means that packing the item in any of the first $1.7877n$ bins results in an infeasible packing. By Lemma 4.3, with $h=\frac{1}{2}$ and $h'=\frac{1}{6}$, the load of each of the first $1.7877n$ bins is at least $1-\frac{1}{4}-\frac{1}{6}=0.5833$. Then the total is at least $1.7877n\times 0.5833>1.042n$, contradicting that the maximum load at any time is n. Therefore, the lemma follows. □

To further analyze the performance of FF, we consider the function $\gamma\langle x,y\rangle$, which defines the minimum load of this bin containing items with length at least $\frac{1}{x}$ and at most $\frac{1}{2}$ such that an item of size $\frac{1}{y}$ cannot be packed into this bin. See Table 4.6 for the values. We now state the result.

Lemma 4.12. *FF is 2.4593-competitive for unit fraction items of types* $\mathrm{T}(1,\frac{1}{2})$, $\mathrm{T}(1,\le\frac{1}{3})$, $\mathrm{T}(\frac{1}{2},\frac{1}{2})$, $\mathrm{T}(\frac{1}{2},\le\frac{1}{3})$.

Proof. Let (x_1+x_5), $(x_1+x_5+x_4)$, $(x_1+x_5+x_4+x_3)$, and $(x_1+x_5+x_4+x_3+x_2)$ be the last bin that FF ever packs a $\mathrm{T}(1,\frac{1}{5})$-, $\mathrm{T}(1,\frac{1}{4})$-, $\mathrm{T}(1,\frac{1}{3})$-, and $\mathrm{T}(1,\frac{1}{2})$- item, respectively. When FF packs a $\mathrm{T}(1,\frac{1}{5})$-item to bin-$(x_1+x_5)$, the load of the first x_1 bins is at least $1-\frac{1}{4}-\frac{1}{5}=\frac{11}{20}$, by Lemma 4.3. By Lemma 4.11, only type $\mathrm{T}(1,k)$-item, for $\frac{1}{5}\le k\le\frac{1}{2}$, could be packed in the x_5 bins. These items all have width 1 and thus can be considered as 1-D case. Therefore, when we cannot pack a $\mathrm{T}(1,\frac{1}{5})$-item, the current load must be at least $\gamma\langle 5,5\rangle$. Then we have $x_1(\frac{11}{20})+x_5\,\gamma\langle 5,5\rangle\le n$. Similarly, we have

1. $x_1(\frac{11}{20})+x_5\,\gamma\langle 5,5\rangle\le n$,
2. $x_1(1-\frac{1}{4}-\frac{1}{4})+x_5\,\gamma\langle 5,4\rangle+x_4\,\gamma\langle 4,4\rangle\le n$,

2DDynamicPackUFS1		
Classes	Types of items	Competitive ratios
Class 1	T($\leq\frac{1}{3}$, $\leq$1)	2.8258
Class 4	T(1, $\frac{1}{2}$), T(1, $\leq\frac{1}{3}$), T($\frac{1}{2}$, $\frac{1}{2}$), T($\frac{1}{2}$, $\leq\frac{1}{3}$)	2.4593
Class 5	T(1, 1), T($\frac{1}{2}$, 1)	1.5
Overall	All of the above	6.7850

Table 4.7: Competitive ratios for 2-D unit fraction items.

2DDynamicPackUFS2		
Classes	Types of items	Competitive ratios
Class 1	T($\leq\frac{1}{3}$, $\leq$1)	2.8258
Class 2	T(1, $\leq\frac{1}{3}$), T($\frac{1}{2}$, $\leq\frac{1}{3}$)	1.7804
Class 3	T(1, 1), T(1, $\frac{1}{2}$), T($\frac{1}{2}$, 1), T($\frac{1}{2}$, $\frac{1}{2}$)	2.25
Overall	All of the above	6.8561

Table 4.8: Competitive ratios for 2-D unit fraction items.

3. $x_1(1-\frac{1}{4}-\frac{1}{3}) + x_5\,\gamma\,\langle 5,3\rangle + x_4\,\gamma\,\langle 4,3\rangle + x_3\,\gamma\,\langle 3,3\rangle \leq n$,

4. $x_1(1-\frac{1}{4}-\frac{1}{2}) + x_5\,\gamma\,\langle 5,2\rangle + x_4\,\gamma\,\langle 4,2\rangle + x_3\,\gamma\,\langle 3,2\rangle + x_2\,\gamma\,\langle 2,2\rangle \leq n$.

We note that for each inequality, the coefficients are increasing, e.g., for (1), we have $\frac{11}{20} \leq \gamma\,\langle 5,5\rangle = \frac{5}{6}$, by Table 4.4. Therefore, the maximum value of $x_1+x_5+x_4+x_3+x_2$ is obtained by setting the maximum possible value of x_5 satisfying (1), and then the maximum possible value of x_4 satisfying (2), and so on. Using Table 4.4, we compute the corresponding values as $1.7877n$, $0.0202n$, $0.1085n$, $0.1917n$ and $0.3514n$, respectively. As a result, $x_1+x_6+x_5+x_4+x_3 \leq 2.4593n \leq 2.4593OPT$. □

4.4 Packing of 2-D unit fraction items

Our algorithm, named as 2DDynamicPackUF, classifies items into classes and then pack items in each class independent of other classes. In each class, FF is used to pack the items as described in Section 4.3. In this section, we present two schemes and show their competitive ratios.

Table 4.7 shows the classification and associated competitive ratios for 2DDynamicPackUFS1. This scheme contains Classes 1, 4, and 5, which cover all items.

Theorem 4.13. 2DDynamicPackUFS1 *is* 6.7850*-competitive for* 2*-D unit fraction items.*

Scheme 2DDynamicPackUFS2 has a higher competitive ratio than Scheme 2DDynamicPackUFS1, nevertheless, Scheme 2DDynamicPackUFS2 has a smaller competitive ratio for power fraction items to be discussed in the next chapter. Table 4.8 shows the classification and associated competitive ratios for 2DDynamicPackUFS2. This scheme contains Classes 1, 2, and 3, which cover all items.

Lemma 4.14. 2DDynamicPackUFS2 *is* 6.8561*-competitive for* 2*-D unit fraction items.*

Based on the results obtained in this chapter, we are ready to extend our analysis to two-dimensional power fraction items and three-dimensional unit and power fraction items in Chapter 5. We will also give concluding remarks on our study in Chapter 5.

Chapter 5

Online Multi-dimensional Dynamic Bin Packing of Unit and Power Fraction Items

5.1 Introduction

In this chapter we first study online two-dimensional dynamic bin packing of power fraction items. We use a similar approach for proving the competitive ratio of 2-D power fraction items in Section 5.2 as that for 2-D unit fraction items in Chapter 4. Secondly, we extend our study to 3-D unit and power fraction items in Section 5.3. The 3-D results are based in part on the analysis of 2-D unit and power fraction items. While this chapter contains results that were described in the paper published in Proceedings of the 8th International Conference on Algorithms and Complexity (CIAC), 2013 [15], their analysis was omitted from [15] and we present it here for the first time. In some cases, the analysis for 2-D power fraction items is very similar to 2-D unit fraction items. In order to avoid detailing such a similar analysis, we give a description on how this can be performed as in Chapter 4.

Recall that a general size item is an item such that the length in each dimension is in $(0, 1]$, a unit fraction item is an item with lengths of the form $\frac{1}{k}$, where $k \geq 1$ is an integer, and a power fraction item has lengths of the form $\frac{1}{2^k}$, where $k \geq 0$ is an integer. The notation used for 2-D power fraction items is identical to the one used for 2-D unit fraction items in Chapter 4 (Section 4.2), while for 3-D unit and power fraction items we introduce the notation used in Section 5.3. Once again, the bin assignment algorithm that we use for all types of 2-D power fraction items and 3-D unit and power fraction items is the First-Fit (FF) algorithm. When a new item arrives, if there are occupied bins in which the item can be repacked, FF assigns the new item to the bin which has been occupied for the longest time. As in Chapter 4, we adopt the typical approach of dividing items into classes and analyzing each class individually. We propose the classes and define different packing schemes based on the classes.

	1-D	2-D	3-D
General size	2.788 [26]	7.788 [70]	22.788 [70]
Unit fraction	2.4842 [47]	6.7850 [**]	21.6108 [*]
Power fraction	2.4842 [47]	6.2455 [*]	20.0783 [*]

Table 5.1: Competitive ratios for general size, unit fraction, and power fraction items. The results obtained in this chapter are marked with "[*]". The result obtained in Chapter 4 is marked with "[**]".

$T(1,1)$	$T(1,\frac{1}{2})$	$T(\frac{1}{2},1)$	$T(\frac{1}{2},\frac{1}{2})$	$T(1,\leq\frac{1}{4})$	$T(\frac{1}{2},\leq\frac{1}{4})$	$T(\leq\frac{1}{4},\leq 1)$

Table 5.2: Types of power fraction items considered.

Our contribution. In this chapter, we design and analyze algorithms for online two-dimensional dynamic bin packing of power fraction items, and online three-dimensional dynamic bin packing of unit and power fraction items. For 2-D power fraction items we obtain a competitive ratio of 6.2455, while for 3-D we obtain competitive ratios of 21.6108 and 20.0783 for unit and power fraction items, respectively. Table 5.1 compares these results to general size items and the 2-D unit fraction item result obtained in Chapter 4.

Organization of the chapter. In Section 5.2 we study our new algorithm for 2-D power fraction items, while in Section 5.3 we design and analyze our new algorithms for 3-D unit and power fraction items. Finally in Section 5.4 we give concluding remarks on our study of unit and power fraction items in Chapters 4 and 5.

5.2 2-D power fraction items

We consider several types of 2-D power fraction items, given in Table 5.2. The difference from unit fraction items lies in the boundary values of $\frac{1}{3}$ being replaced with $\frac{1}{4}$ for power fraction items. We will first consider in Section 5.2.1 the classification scheme of 2-D general size items in [70] and show that using some existing results and a new analysis for a class of items leads to a competitive ratio of 7.1509. The purpose of this is to show the new analysis which will be reused in the proof for 3-D power fraction items in Section 5.3. Afterwards, we present a new classification scheme and analysis for 2-D power fraction items in Section 5.2.2 for which the competitive ratio is further reduced to 6.2455.

5.2.1 Existing scheme

The algorithm in [70] for general size items effectively classifies power fraction items into the classes as shown in Table 5.3. The overall competitive ratio reduces from 7.788 to 7.1509 by using a new analysis for Class A items and existing results for Classes B and C. We now show how to improve the competitive ratio for Class A items of type

Scheme in [70]		
Classes	Types of items	Competitive ratios
Class A	T($\leq\frac{1}{4}$, $\leq$1)	2.667
Class B	T($\frac{1}{2}$, 1), T($\frac{1}{2}$, $\frac{1}{2}$), T($\frac{1}{2}$, $\leq\frac{1}{4}$)	2 [70]
Class C	T(1, 1), T(1, $\frac{1}{2}$), T(1, $\leq\frac{1}{4}$)	2.4842 [47]
Overall	All items	7.1509

Table 5.3: The 2-D results of [70] in the context of power-fraction items.

2DDynamicPackPF		
Class	Types of items	Competitive Ratio
Class 1	T($\leq\frac{1}{4}$, $\leq$1)	2.4995 [*]
Class 2	T(1, $\leq\frac{1}{4}$), T($\frac{1}{2}$, $\leq\frac{1}{4}$)	1.496025 [*]
Class 3	T(1, 1), T(1, $\frac{1}{2}$), T($\frac{1}{2}$, 1), T($\frac{1}{2}$, $\frac{1}{2}$)	2.25
Overall	All items	6.2455

Table 5.4: Competitive ratios for 2-D power fraction items. Marked with [*] are the competitive ratios that are reduced as compared to unit fraction items.

T($\leq\frac{1}{4}$, $\leq$1). This result will also be used in the analysis of 3-D power fraction items in Section 5.3.

2DRepackNarrowPF. We will make use of the procedure 2DRepackNarrowPF that packs 2-D power fraction items with width $w \leq \frac{1}{4}$ and height $h \leq 1$ into a 2-D unit size bin in the following way: we divide a bin into two partitions, one with width $\frac{1}{4}$ and height 1, and the other with width $\frac{3}{4}$ and height 1. When a new item arrives, we place it into the first partition and try to repack existing items to the second partition using Steinberg's algorithm [66]. If the existing items cannot be repacked to the second partition, it is understood that we revert to the old configuration. Lemma 4.2 states that if a bin is given with width u and height v, and the width of all items is smaller than $\frac{u}{2}$, any set of these with total area at most $\frac{uv}{2}$ can fit into the same bin by using Steinberg's algorithm. In our case, all the items have width $w \leq \frac{1}{4}$, and the width of the bin that they need to be fitted into is $\frac{3}{4}$. The condition that the items must have width $w \leq \frac{u}{2}$, i.e., $\frac{1}{4} \leq \frac{3}{8}$, is satisfied. Additionally, we can express the following: if the items cannot fit into the second partition, the area must exceed $\frac{uv}{2}$, i.e., $\frac{3}{8}$. Using 2DRepackNarrowPF for Class A items results in a competitive ratio of $\frac{8}{3} = 2.(6)$.

For Class B we reuse the existing competitive ratios from [70] and for Class C the problem is reduced to 1-D and we reuse the competitive ratios from [47], noting that power fraction items are a subset of unit fraction items.

Corollary 5.1. *The* 2*-D packing algorithm in* [70] *is* 7.1509*-competitive for power fraction items.*

$\beta_{\mathrm{p}} \langle x, y \rangle$	$y = 4$	8
$x = 4$	1	1
8	$\frac{7}{8}$	1

Table 5.5: Values of $\beta_{\mathrm{p}} \langle x, y \rangle$ for $x \in \{4, 8\}$ and $y \in \{4, 8\}$.

5.2.2 Classification of 2-D power fraction items

For 2-D power fraction items we present a new classification scheme in Table 5.4 based on 2DDynamicPackUFS2 for unit fraction items (Table 4.8) where the competitive ratio reduces to 6.2455. We reduce the competitive ratio for Classes 1 and 2 of unit fraction items and we reuse the competitive ratio of 2.25 obtained for Class 3 (Section 4.3.2), as the unit fraction items coincide with the power fraction items. A possible scheme based on 2DDynamicPackUFS1 (Table 4.7) for unit fraction items would lead to a higher competitive ratio of 6.3669 for power fraction items and we do not present it.

Repacking procedures. We note that repacking procedures for power fraction items are similar to those for unit fraction items described in Sections 4.3.4, 4.3.3, and 4.3.2, for Classes 1, 2, and 3, respectively. Even though the boundary value of $\frac{1}{3}$ changes to $\frac{1}{4}$ (for items $\mathrm{T}(1, \leq\frac{1}{4})$, $\mathrm{T}(\frac{1}{2}, \leq\frac{1}{4})$, and $\mathrm{T}(\leq\frac{1}{4}, \leq 1)$, including its three subtypes), the procedures remain the same. The bin assignment algorithm description remains the same, but we rename it 2DDynamicPackPF.

We are now ready to proceed with the analysis of Classes 2 and 1.

Class 2: $\mathbf{T(1, \leq\frac{1}{4})}$, $\mathbf{T(\frac{1}{2}, \leq\frac{1}{4})}$

We now consider items whose width is at least $\frac{1}{2}$ and height is at most $\frac{1}{4}$. For this class, the repacking of a newly arrived item is done according to the description in the proof of Lemma 4.3. We can show that FF is 1.496025-competitive for Class 2 in a similar manner to the proof for Class 2 unit fraction items (Section 4.3.3).

Suppose the maximum load at any time is n. If x_1 is the last bin that FF ever packs a $\mathrm{T}(\frac{1}{2}, \leq\frac{1}{4})$-item, using the analysis in [26] for 1-D items with size at most $\frac{1}{4}$, one can show that $x_1 \leq 1.3192n$ (Lemma 5.2 (1)). Lemma 5.2 (1) implies that FF only packs items of $\mathrm{T}(1, \leq\frac{1}{4})$ in bin-y for $y > 1.3192n$. Therefore, any item that is packed by FF to bin-y, for $y > 1.3192n$, must be of type $\mathrm{T}(1, h)$, where $\frac{1}{8} \leq h \leq \frac{1}{4}$ (Lemma 5.2 (2)).

We now further modify the $\beta \langle x, y \rangle$ function such that the new function $\beta_{\mathrm{p}} \langle x, y \rangle$ denotes the minimum load of a bin containing items of lengths at least $\frac{1}{x}$ and at most $\frac{1}{4}$ such that an item of length $\frac{1}{y}$ will not be able to pack in this bin. Table 5.5 shows the values of the function for $x \in \{4, 8\}$ and $y \in \{4, 8\}$. Using Table 5.5, it can be proved that FF is 1.496025-competitive for power fraction items of types $\mathrm{T}(1, \leq\frac{1}{4})$ and $\mathrm{T}(\frac{1}{2}, \leq\frac{1}{4})$ (Lemma 5.2 (3)). The arguments are similar to those for Class 2 of 2-D unit fraction items in Section 4.3.3.

Lemma 5.2. (1) *Suppose we are packing power-fraction items of type* $\mathrm{T}(1, \le\frac{1}{4})$, $\mathrm{T}(\frac{1}{2}, \le\frac{1}{4})$ *and the maximum load over time is* n*. We have* $x_1 \le 1.3192n$*, where* x_1 *is the last bin that FF ever packs a* $\mathrm{T}(\frac{1}{2}, \le\frac{1}{4})$*-item. The proof is similar to Lemma 4.6.*

(2) *Suppose we are packing power-fraction items of type* $\mathrm{T}(1, \le\frac{1}{4})$, $\mathrm{T}(\frac{1}{2}, \le\frac{1}{4})$ *and the maximum load over time is* n*. Any item that is packed by FF to bin-*y*, for* $y > 1.3192n$*, must be of type* $\mathrm{T}(1, h)$*, where* $\frac{1}{8} \le h \le \frac{1}{4}$*. The proof is similar to Lemma 4.7.*

(3) *FF is* 1.496025*-competitive for power fraction items of types* $\mathrm{T}(1, \le\frac{1}{4})$ *and* $\mathrm{T}(\frac{1}{2}, \le\frac{1}{4})$*. The proof is similar to Lemma 4.8.*

Class 1: $\mathbf{T}(\le\frac{1}{4}, \le 1)$

As with unit fraction items, we further divide the power fraction items of type $\mathrm{T}(\le\frac{1}{4}, \le 1)$ into three subtypes $\mathrm{T}(\le\frac{1}{4}, \le\frac{1}{4})$, $\mathrm{T}(\le\frac{1}{4}, \frac{1}{2})$, $\mathrm{T}(\le\frac{1}{4}, 1)$. The repacking of these items is similar to Class 1 unit fraction items (Section 4.3.4):

1. When the new item is $\mathrm{T}(\le\frac{1}{4}, \le\frac{1}{4})$, we use Steinberg's algorithm [66] to repack the new and existing items.
2. When the new item is $\mathrm{T}(\le\frac{1}{4}, \frac{1}{2})$ or $\mathrm{T}(\le\frac{1}{4}, 1)$ and the bin contains $\mathrm{T}(\le\frac{1}{4}, \le\frac{1}{4})$-item, we divide the bin into two compartments, one with width $\frac{1}{4}$ and the other $\frac{3}{4}$ and both with height 1. We reserve the small compartment for the new item and try to repack the existing items in the large compartment using Steinberg's algorithm. This idea originates from [70].
3. When the new item is $\mathrm{T}(\le\frac{1}{4}, \frac{1}{2})$ or $\mathrm{T}(\le\frac{1}{4}, 1)$ and the bin does not contain $\mathrm{T}(\le\frac{1}{4}, \le\frac{1}{4})$-item, we use the repacking method as in Lemma 4.3 but with the width becoming the height and vice versa. Note that this implies that Lemma 4.8 applies for these items.

In a similar way to Section 4.3.4 (Lemma 4.9) we can show that FF is 2.4995-competitive for power fraction items of types $\mathrm{T}(\le\frac{1}{4}, \le 1)$.

Lemma 5.3. *FF is* 2.4995*-competitive for power fraction items of types* $\mathrm{T}(\le\frac{1}{4}, \le 1)$*.*

Finally, for Class 3 items we reuse the existing result from Section 4.3.2 and combined with Lemmas 5.2 (3) and 5.3, the theorem follows.

Theorem 5.4. 2DDynamicPackPF *is* 6.2455*-competitive for power fraction items.*

5.3 3-D unit and power fraction items

For 3-D unit fraction and power fraction items we propose two classification schemes as shown in Tables 5.6 and 5.7, respectively. We first introduce the notation necessary for

3DDynamicPackUF [70]		
Classes	Types of items	Competitive ratios
Class 1	T($>\frac{1}{2}, *, *$)	6.7850 [*]
Class 2	T($\leq\frac{1}{2}, >\frac{1}{2}, *$)	4.8258 [**]
Class 3	T($\leq\frac{1}{2}, (\frac{1}{3}, \frac{1}{2}], *$)	4 [70]
Class 4	T($\leq\frac{1}{2}, \leq\frac{1}{3}, *$)	6 [70]
Overall	All items	21.6108

Table 5.6: Competitive ratios for 3-D unit fraction items. [*] This result uses Theorem 4.13. [**] This result uses Lemma 4.9.

3DDynamicPackPF		
Classes	Types of items	Competitive ratios
Class 1	T($>\frac{1}{2}, *, *$)	6.2455 [*]
Class 2	T($\leq\frac{1}{2}, >\frac{1}{2}, *$)	4.4995 [**]
Class 3	T($\leq\frac{1}{2}, (\frac{1}{4}, \frac{1}{2}], *$)	4 [70]
Class 4	T($\leq\frac{1}{2}, \leq\frac{1}{4}, *$)	5.334 [***]
Overall	All items	20.0783

Table 5.7: Competitive ratios for 3-D power fraction items. [*] This result uses Theorem 5.4. [**] This result uses the competitive ratio of Class 1 from Lemma 5.3. [***] This result uses the procedure 2DRepackNarrowPF from Section 5.2.1.

discussion and afterwards we apply our new results for 2-D unit and power fraction items and some results for 2-D general size from [70] in order to prove the 3-D competitive ratios.

Notations. For 3-D items, we use the notation T(x_1, x_2, x_3) to refer to the type of items with lengths x_1, x_2, x_3 in the three dimensions, in a similar way to 2-D items. We use '$*$' to mean that the length can take any value at most 1, e.g., T($*, *, *$) refers to all items. A parameter may take an expression $> x$, meaning that the length is larger than x. In addition to 2-D items, a parameter may take an interval, $(x, y]$ meaning that the length is in the interval $(x, y]$. For example T($\leq\frac{1}{2}, (\frac{1}{3}, \frac{1}{2}], *$) refers to items with x_1 in $(0, \frac{1}{2}]$, x_2 in $(\frac{1}{3}, \frac{1}{2}]$, and x_3 in $(0, 1]$. Lastly, we denote the length of an item R along dimension x_i by $x_i(R)$.

3-D unit fraction items. The bin assignment algorithm, 3DDynamicPackUF, classifies the items as they arrive into the four classes from Table 5.6. Items in each class are assigned to bins independently of other classes. 3DDynamicPackUF then packs a new item into the first bin of the corresponding class in which it will repack along with existing items or, if the item is not packable in any bin, opens a new bin of the corresponding class and places the new item there.

3-D power fraction items. For power fraction items we slightly modify the classification for 3-D items, such that boundary values of $\frac{1}{3}$ are replaced by $\frac{1}{4}$. Table 5.7 details this classification. The bin assignment algorithm for power fraction items, 3DDynamicPackPF, is identical to 3DDynamicPackUF.

We will now prove the competitive ratio of 21.6108 for 3DDynamicPackUF based on our new results for 2-D unit fraction items and existing results for 2-D and 3-D general size items by Wong and Yung [70].

Theorem 5.5. *Algorithm* 3DDynamicPackUF *is* 21.6108*-competitive for unit fraction items.*

Proof. The proof is similar to the one provided in [70] for 3DDynamicPack. For Class 1 items, as only one item can fit along dimension x_1, we project along this dimension and the problem is reduced to 2-D; we pack the items using our 2-D algorithm for unit fraction items, 2DDynamicPackUFS1. The algorithm has a competitive ratio of 6.7850. Class 2 items are further divided into two subclasses: T($\leq \frac{1}{3}, > \frac{1}{2}, *$) and T($(\frac{1}{3}, \frac{1}{2}], > \frac{1}{2}, *$). Projecting the items along dimension x_2, the first subclass corresponds to 2-D unit fraction items of type T($\leq\frac{1}{3}, \leq 1$) and for the second subclass we reuse the procedure 2DRepackMedium from [70]. The first and second subclasses attain competitive ratios of 2.8258 and 2, respectively. For Classes 3 and 4, we reuse the competitive ratios of 4 and 6, respectively, from [70]. Denoting by n_1, n_2, n_3, and n_4 the number of bins used by Classes 1, 2, 3, and 4, respectively, and by n the total number of bins used by all classes, we have: $n \leq n_1 + n_2 + n_3 + n_4 \leq 21.6108 OPT$, where OPT is the total number of bins used by the optimal offline algorithm. □

Lastly, we will prove the competitive ratio of 20.0783 for 3DDynamicPackPF based on our new results for 2-D power fraction items, the 2DRepackNarrowPF procedure from Section 5.2.1, and existing results for 2-D and 3-D general size items by Wong and Yung [70].

Theorem 5.6. *Algorithm* 3DDynamicPackPF *is* 20.0783*-competitive for power fraction items.*

Proof. Similar to unit fraction items, Class 1 power fraction items will repack items using the 2-D power fraction algorithm, 2DDynamicPackPF, by projecting along dimension x_1. 2DDynamicPackPF attains a competitive ratio of 6.2455. Class 2 items are further divided into two subclasses: T($\leq \frac{1}{4}, > \frac{1}{2}, *$) and T($(\frac{1}{4}, \frac{1}{2}], > \frac{1}{2}, *$). We note that by projecting along dimension x_2, the first subclass corresponds to power fraction items of type T($\leq\frac{1}{4}, \leq 1$) and for the second subclass we again reuse the procedure 2DRepackMedium from [70]. The first and second subclasses attain competitive ratios of 2.4995 and 2, respectively. For Class 3, we reuse the competitive ratio of 4 from [70].

For Class 4 items, we first sort the items in non-increasing order of their lengths in dimension x_1. Items are then packed into layers constructed along dimension x_1 in a next-fit-decreasing-height manner (detailed in Section 2.3.2). The first layer always has length $\frac{1}{2}$ along dimension x_1. We project the items along dimension x_1, and pack items into this layer along dimensions x_2 and x_3 by using 2DRepackNarrowPF from

Section 5.2.1. I.e., we create two partitions with lengths $\frac{1}{4}$ and $\frac{3}{4}$ along dimension x_2, and length 1 along dimension x_3, corresponding to widths and heights of 2-D items, respectively, and repack items using 2DRepackNarrowPF. If some item Q cannot pack into this layer, we create a new layer with length $x_1(Q)$ along dimension x_1 and pack Q into the first partition of this layer. Then we use 2DRepackNarrowPF and try to repack Q and subsequent items to the second partition. We repeat this process until all items are packed or an item overflows dimension x_1 of the bin, in which case we revert to the previous feasible configuration. Note that once an item cannot fit into a layer, the layer is closed and closed layers are not revisited; however, each time a new item arrives the layers are reconstructed.

As for unit fraction items, we denote by n_1, n_2, n_3, and n_4 the number of bins used by Classes 1, 2, 3, and 4, respectively, and by n the total number of bins used by all classes. When we open the n_4-th bin for a new Class-4 item, by Lemma 4.2, in the repacking of each bin, by projecting items along dimension x_1, the total area of all Class-4 items in the second partition in each layer and the first partition in the next layer is more than $\frac{3}{8}$. Furthermore, the length of all items along dimension x_1 in each layer is at least the length of the next layer. Since existing items cannot be repacked into the same bin by using 2DRepackNarrowPF, the total length of all layers along dimension x_1 is more than 1 and that of all but the first layer is more than $\frac{1}{2}$. Therefore, the total volume of existing items in each bin is more than $\frac{3}{16}$ and the total volume of all existing items is more than $(n_4 - 1)\frac{3}{16}$. Concluding, we have $n \leq n_1 + n_2 + n_3 + n_4 \leq 20.0783OPT$, where OPT is the total number of bins used by the optimal offline algorithm. □

5.4 Conclusion

In Chapters 4 and 5 we have extended the study of 2-D and 3-D dynamic bin packing problem to unit and power fraction items. We have improved the competitive ratios that would be obtained using only existing results for unit fraction items from 7.4842 to 6.7850 for 2-D, and from 22.4842 to 21.6108 for 3-D. For power fraction items, the competitive ratios are further reduced to 6.2455 and 20.0783 for 2-D and 3-D, respectively. Our approach is to divide items into classes and analyzing each class individually. We have proposed several classes and defined different packing schemes based on the classes. This approach gives a systematic way to explore different combinations of classes.

An open problem is to further improve the competitive ratios for various types of items. The gap between the upper and lower bounds could also be reduced by improving the lower bounds. Another problem is to consider higher dimension bin packing. For d-dimensional static and dynamic bin packing, for $d \geq 2$, the competitive ratio grows exponentially with d. Yet there is no matching lower bound that also grows exponentially with d. It is believed that this is the case [30] and any such lower bound would be of great interest.

Another direction is to consider the packing of unit fraction and power fraction hypercubes, where all sides of an item are the same length. We note in Lemma 5.7 that the competitive ratio for the packing of 2-D unit fraction square items would reduce to 3.9654 compared to the competitive ratio of 2-D general size square items of 4.2154 [37]. For 3-D unit fraction cubes, this would reduce to 5.24537 compared to 5.37037 for 3-D general size cubes [37].

Lemma 5.7. *There exists a* 3.9654-*competitive algorithm for* 2-*D unit fraction squares and a* 5.24537-*competitive algorithm for* 3-*D unit fraction cubes.*

Proof. We denote 2-D unit fraction square items using the regular notation for 2-D unit fraction items, i.e., they are of type $\mathrm{T}(x, y)$, where $x = y$. A similar notation is used for 3-D unit fraction cubes of type $\mathrm{T}(x, y, z)$, where $x = y = z$. For 2-D unit fraction squares, we divide the items into two classes, class of item types $\mathrm{T}(\leq \frac{1}{3}, \leq \frac{1}{3})$ and class of items types $\mathrm{T}(\geq \frac{1}{2}, \geq \frac{1}{2})$, and the two classes cover all types of items. By the analysis in [37], the first class of items achieves a competitive ratio of 2.2154 using the Next-Fit Decreasing Height (NFDH) algorithm (detailed in Section 2.3.2). Using the First-Fit (FF) algorithm for the second class of items, we can achieve a competitive ratio of 1.75. The proof is similar to the analysis of Class 5-items ($\mathrm{T}(1, 1)$- and $\mathrm{T}(\frac{1}{2}, 1)$-items) detailed in Section 4.3.1. Overall the competitive ratio obtained is 3.9654.

For 3-D unit fraction cubes, we have three classes of items: $\mathrm{T}(\leq\frac{1}{4}, \leq\frac{1}{4}, \leq\frac{1}{4})$, $\mathrm{T}(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, and $\mathrm{T}(\geq \frac{1}{2}, \geq \frac{1}{2}, \geq \frac{1}{2})$. The first two classes are shown to achieve competitive ratios of 2.37037 and 1 in [37], using the NFDH algorithm and using dedicated bins, respectively. For the third class of items using FF would result in a competitive ratio of 1.875 through a similar analysis as in Section 4.3.1. Overall the competitive ratio obtained is 5.24537. ◻

By using similar arguments as the previous lemma, the competitive ratios for 2-D and 3-D power fraction hypercubes are further reduced to 3.52778 and 4.24537, respectively.

Corollary 5.8. *There exists a* 3.52778-*competitive algorithm for* 2-*D power fraction squares and a* 4.24537-*competitive algorithm for* 3-*D power fraction cubes.*

Chapter 6

Lower Bounds for Online Multi-dimensional Dynamic Bin Packing

6.1 Introduction

In this chapter we give lower bounds of certain online algorithms for multi-dimensional dynamic bin packing. We separately consider lower bounds for three types of items, general size items, unit fraction items and power fraction items. While in Chapter 3 we gave a lower bound for *any* online algorithm, in this chapter we focus on the family of Any-Fit algorithms, specifically First-Fit, Best-Fit, Worst-Fit, and Any-Fit. We first introduce the algorithms for which we will construct adversaries for online multi-dimensional dynamic bin packing. When a new item R arrives, if there are occupied bins in which R can be packed (allowing repacking for existing items), the algorithms assign R to one of these bins as follows:

First-Fit (FF) assigns R to the bin which has been occupied for the longest time.

Best-Fit (BF) assigns R to the heaviest loaded bin. Ties are broken arbitrarily.

Worst-Fit (WF) assigns R to the lightest loaded bin. Ties are broken arbitrarily.

Any-Fit (AF) assigns R to any of the bins arbitrarily.

Our contribution. We first consider general size items and show that the family of Any-Fit algorithms (First-Fit, Best-Fit, Worst-Fit, and Any-Fit) without modification is not competitive for online multi-dimensional dynamic bin packing. Note that it is sufficient to show that the algorithms are not competitive for online two-dimensional dynamic bin packing. This result is in contrast to the $O(1)$-competitiveness of First-Fit without modification for online 1-D dynamic bin packing [26]. In fact, it can also be shown that Any-Fit, including Best-Fit and Worst-Fit, is $O(1)$-competitive for online

		1-D bounds		2-D bounds		3-D bounds	
Algorithm	Input	Upper	Lower	Upper	Lower	Upper	Lower
First-Fit	General	2.897 [26]	2.75 [26]	∞	∞	∞	∞
	UF	2.4842 [47]	2.45 [18]	?	**5.45**	?	**6.45**
	PF	2.4842 [47]	?	?	**5.4375**	?	**6.4375**
Best-Fit	General	3 [18, 26]	3 [18]	∞	∞	∞	∞
	UF	3 [18]	3 [18]	∞	∞	∞	∞
	PF	3 [18]	3 [18]	∞	∞	∞	∞
Worst-Fit	General	3 [18, 26]	3 [18]	∞	∞	∞	∞
	UF	3 [18]	3 [18]	∞	∞	∞	∞
	PF	3 [18]	3 [18]	∞	∞	∞	∞
Any-Fit	General	3 [18, 26]	2.666 [*]	∞	∞	∞	∞
	UF	3 [18]	2.428 [18]	?	3.70301 [37]	?	4.85383 [37]
	PF	3 [18]	?	?	?	?	?
$\mathcal{A}$	General	-	2.666 [*]	-	3.70301 [37]	-	4.85383 [37]
	UF	-	2.428 [18]	-	3.70301 [37]	-	4.85383 [37]
	PF	-	?	-	?	-	?

Table 6.1: Bounds for the family of Any-Fit algorithms without modification. Bolded entries are results obtained in this chapter. For input, General, UF, and PF refer to general size, unit fraction, and power fraction items, respectively. $\mathcal{A}$ refers to any online algorithm. "∞" denotes unbounded competitive ratios and "?" denotes unknown bounds. [*] This result is detailed in Chapter 3 and published in [71].

1-D dynamic bin packing of general size items (Section 6.3). Secondly we consider unit fraction and power fraction items. We show that Best-Fit and Worst-Fit without modification are not competitive for online multi-dimensional dynamic bin packing of unit and power fraction items, while the lower bound for First-Fit is at least 5.4375 for 2-D power fraction items and at least 5.45 for 2-D unit fraction items. These results again show that modification of the algorithms is required in order to achieve competitiveness. These results are in contrast to the $O(1)$-competitiveness for online 1-D dynamic bin packing of unit fraction items of Best-Fit and Worst-Fit [18] and the general lower bound of any algorithm of 3.70301 [37] for 2-D unit fraction items. Lower bound results for Any-Fit are yet to be investigated in the context of unit and power fraction items, although it is expected that these will not exceed the corresponding lower bounds for First-Fit. In Table 6.1 we give a summary of existing and new results for the family of Any-Fit algorithms for online multi-dimensional dynamic bin packing.

Organization of the chapter. In Section 6.2 we give preliminaries required for discussion. In Section 6.3 we consider the lower bound result for general size items, while in Section 6.4 we consider lower bounds for unit and power fraction items. We give concluding remarks in Section 6.5.

6.2 Preliminaries

Recall that a general size item is an item such that the length in each dimension is in $(0, 1]$; a unit fraction item is an item with lengths of the form $\frac{1}{k}$, where $k \geq 1$ is an integer; and a power fraction item has lengths of the form $\frac{1}{2^k}$, where $k \geq 0$ is an integer.

Algorithms without modification. Algorithms without modification are algorithms that belong to the family of Any-Fit that do not classify items based on their lengths and then pack items of different classes in different bins independently of each other. We also refer to these algorithms as algorithms without classification of items.

Notations. When we discuss how items are packed, we use the following notations:

- Item configuration ψ: $\mathrm{T}(w, h)$ describes an item with width w and height h, with an area $w \times h$, e.g., $\mathrm{T}(\frac{1}{2}, \frac{1}{2})$ means an item with width $\frac{1}{2}$ and height $\frac{1}{2}$, with an area $\frac{1}{4}$.
- Bin configuration π: $(\psi_1, \psi_2, \cdots)$, e.g., $(\mathrm{T}(\frac{1}{2}, \frac{1}{2}), \mathrm{T}(1, \frac{1}{2}))$ means a bin has a load of $\frac{3}{4}$, with a $\mathrm{T}(\frac{1}{2}, \frac{1}{2})$-item and a $\mathrm{T}(1, \frac{1}{2})$-item. Multiple items of the same type in a bin are represented as $m{\times}\mathrm{T}(w, h)$, for some integer $m \geq 1$, e.g., $4{\times}\mathrm{T}(\frac{1}{2}, \frac{1}{2})$.
- Packing configuration ρ: $\{x_1{:}\pi_1, x_2{:}\pi_2, \cdots\}$ is a packing where there are x_1 bins with bin configuration π_1, x_2 bins with π_2, and so on. E.g., $\{2n{:}\mathrm{T}(\frac{1}{2}, 1), n{:}\mathrm{T}(1, 1)\}$ means $2n$ bins are each packed with load $\frac{1}{2}$ with one $\mathrm{T}(\frac{1}{2}, 1)$-item and another n bins are each packed with a load 1 with one $\mathrm{T}(1, 1)$-item.

6.3 Lower bounds for general size items

Only one adversary is required for 2-D general size items in order to prove that the family of Any-Fit algorithms without modification is not competitive for online multi-dimensional dynamic bin packing. The idea behind the adversary is to use items of slightly different sizes that are "tall" and "thin", and items that are "short" and "wide", in each stage. Furthermore, in each stage the choice of where the family of Any-Fit algorithms can pack new items is limited to only a new bin due to two-dimensional packing constraints.

Theorem 6.1. *The competitive ratio of the family of Any-Fit algorithms (First-Fit, Best-Fit, Worst-Fit, and Any-Fit) without classification of items for the online dynamic bin packing problem of* 2*-D general size items is unbounded.*

Proof. Let z be an arbitrarily large integer and n be the largest integer such that $\sum_{0 \leq i \leq n-1} \frac{1}{z+i} \leq 1$. The adversary works in n stages. In Stage i, the adversary releases one $\mathrm{T}(1, \frac{1}{z+i})$-item and n items of type $\mathrm{T}(\frac{1}{z+i}, 1{-}\frac{1}{z+i})$. The adversary then lets items released in this stage depart until one $\mathrm{T}(1, \frac{1}{z+i})$-item and one $\mathrm{T}(\frac{1}{z+i}, 1{-}\frac{1}{z+i})$-item remain. We observe that by the definition of z and n, the items released in each stage can be packed into one bin. Furthermore, for $i' > i$, a $\mathrm{T}(\frac{1}{z+i'}, 1{-}\frac{1}{z+i'})$-item cannot be packed in the same bin with a $\mathrm{T}(1, \frac{1}{z+i})$-item.

All algorithms in the family of any-fit algorithms pack in the same way. Let $\mathcal{A}$ be such an algorithm. The items in each stage will be packed by $\mathcal{A}$ in the same new bin and, after the departure of items, only one $\mathrm{T}(1, \frac{1}{z+i})$-item and one $\mathrm{T}(\frac{1}{z+i}, 1-\frac{1}{z+i})$-item are left in this new bin. By the above observation, no items in later stages can be packed into this bin and $\mathcal{A}$ is forced to use one new bin in each stage. In total $\mathcal{A}$ uses n bins. On the other hand, the optimal algorithm, $\mathcal{O}$, can reserve two bins for the items that do not depart at all, one for $\mathrm{T}(1, \frac{1}{z+i})$-items and one for $\mathrm{T}(\frac{1}{z+i}, 1-\frac{1}{z+i})$-items. A third bin is needed for the other items, which can be reused in the next stage after these items depart. The competitive ratio of $\mathcal{A}$ is $n/3$, which can be made arbitrarily large by having an arbitrarily large z. $\square$

Online one-dimensional dynamic bin packing. While the family of Any-Fit algorithms without classification is not competitive for online multi-dimensional dynamic bin packing of general size items, it is $O(1)$-competitive for one-dimensional dynamic bin packing as follows. First-Fit is 2.897-competitive for online dynamic bin packing [26]. A lower bound of 2.75 for First-Fit is also shown in [26], however the lower bound construction makes use of an optimal offline algorithm that is allowed to migrate items between bins at any time. Any-Fit, including Best-Fit and Worst-Fit, can be shown to be 3-competitive by using the same arguments as in Section 2.3.1 for the simple upper bound of 3 for First-Fit. The arguments originate from [26]. The lower bounds for Best-Fit and Worst-Fit are 3 [18], while the lower bound for any algorithm, including Any-Fit, is 2.666 (detailed in Chapter 3 and published in [71]).

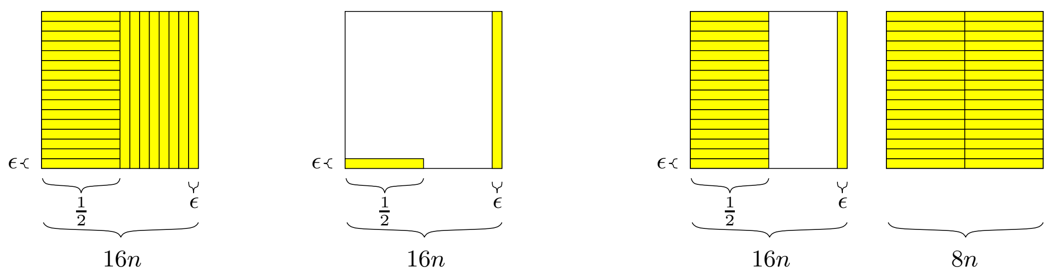
6.4 Lower bounds for unit and power fraction items

We show that when no classification of items based on their lengths is used, the lower bound of FF is at least 5.4375 for 2-D power fraction items and 5.45 for 2-D unit fraction items (Section 6.4.1). In contrast to FF, the competitive ratios of BF and WF are unbounded even for power fraction items (Sections 6.4.2 and 6.4.3, respectively). When classification of items is used, items in different classes are packed independently of each other and competitive ratios may improve. The lower bound for any algorithm that may also use classification of items is 3.70301 [37] for 2-D unit fraction items.

The adversaries work in stages and each stage may consist of item arrival and departure. In each stage, an online algorithm may open new bins. We denote the number of new bins opened in Stage i as s_i.

6.4.1 First-Fit

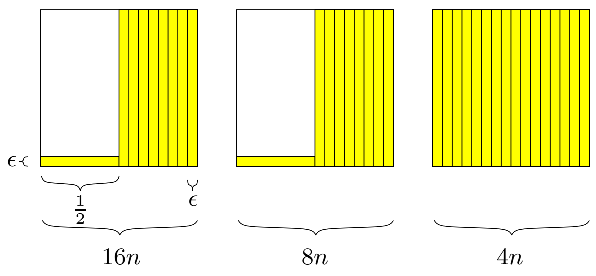
In this section, we give an adversary for the First-Fit (FF) algorithm when the input consists of power fraction items and no classification of items is used. Recall that power fraction items have lengths of the form $\frac{1}{2^k}$, for some integer $k \geq 0$. The adversary consists of multiple stages and at any time the load of items released and not departed is at most $16n + O(1)$, for some large integer n that is a power of 2. We prove that FF uses $87n$ bins, while the optimal offline algorithm $\mathcal{O}$ uses at most $16n+O(1)$ bins. Then,



(a) Stage 0.1 bins after the arrival of items.

(b) Stage 0.1 bins after the departure of items.

(c) Stage 0.2 bins after the first arrival of items.



(d) Stage 0.2 bins after the second arrival of items.

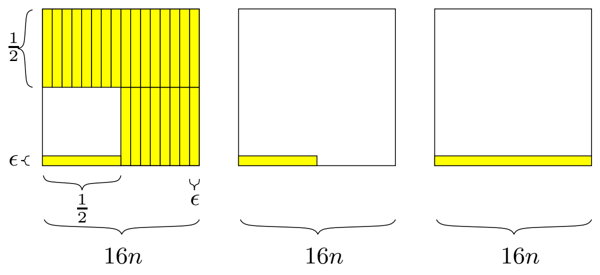
Figure 6.1: Select bin configurations for First-Fit in Stage 0.

the competitive ratio of FF is at least 5.4375 (Lemma 6.2). Note that power fraction items are a subset of unit fraction items and the lower bound also applies to the latter input. For unit fraction items, the lower bound of FF for one-dimensional dynamic bin packing of unit fraction items is 2.45 [18]. Using this result, we improve slightly the lower bound of FF without classification of two-dimensional unit fraction items to 5.45 (Corollary 6.3). On the other hand, the lower bound for unit fraction items for any algorithm is 3.70301 [37].

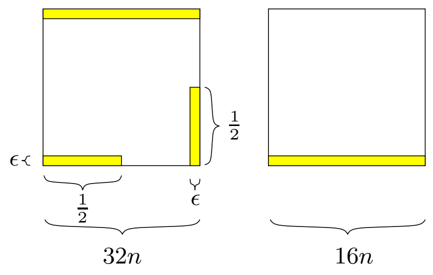
The adversary. Let n be an arbitrarily large integer that is a power of 2 and $\epsilon = \frac{1}{16n}$.

Stage 0. In this initial stage consisting of item arrival and departure, we aim to force FF to use $32n$ bins, i.e., $s_0 = 32n$, using a total load of $16n + O(1)$. This stage consists of two parts and the adversary forces FF to open $16n$ new bins in each part.

Stage 0.1. The first sequence of item arrival consists of alternating between releasing some items of types $\mathrm{T}(\frac{1}{2}, \epsilon)$ and $\mathrm{T}(\epsilon, 1)$. Notice that releasing $16n$ items of type $\mathrm{T}(\frac{1}{2}, \epsilon)$ and $8n$ items of type $\mathrm{T}(\epsilon, 1)$ results in FF packing them in one bin with a load of 1. We first aim to open $16n$ bins and this can be achieved by releasing $16n$ items of type $\mathrm{T}(\frac{1}{2}, \epsilon)$ and $8n$ items of type $\mathrm{T}(\epsilon, 1)$ and repeating the process for $16n$ bins. In total, $\frac{16n}{\epsilon}$ items of type $\mathrm{T}(\frac{1}{2}, \epsilon)$ and $\frac{8n}{\epsilon}$ items of type $\mathrm{T}(\epsilon, 1)$ are needed, resulting in a total load of $16n$. Figure 6.1(a) shows how items are packed by FF. We now depart items such that in each of $16n$ bins there remains one $\mathrm{T}(\frac{1}{2}, \epsilon)$-item and one $\mathrm{T}(\epsilon, 1)$-item. The configuration becomes $\{16n{:}(\mathrm{T}(\frac{1}{2}, \epsilon), \mathrm{T}(\epsilon, 1))\}$, with $16n$ bins and a total load of 1.5. Figure 6.1(b) shows an illustration.



(a) Stage 2 bins after the first arrival of items.



(b) Bins at the end of Stage 2.

Figure 6.2: Select bin configurations for First-Fit in Stage 2.

Stage 0.2. We now aim to open an additional $16n$ bins by alternating releasing $\mathrm{T}(\frac{1}{2},\epsilon)$-items and $\mathrm{T}(\epsilon,1)$-items (with item departure in-between). We release $\frac{32n}{\epsilon}$ items of type $\mathrm{T}(\frac{1}{2},\epsilon)$, resulting in a total load of $16n+1.5$. Note that only $16n-1$ new items of type $\mathrm{T}(\frac{1}{2},\epsilon)$ can be packed in each of the first $16n$ bins due to them containing one $\mathrm{T}(\epsilon,1)$-item and one $\mathrm{T}(\frac{1}{2},\epsilon)$-item. Thus, FF packs $\frac{16n-1}{\epsilon}$ items in the first $16n$ bins and opens $8n+1$ new bins for the remaining items (see Figure 6.1(c) for an illustration). Among these items released, we keep one $\mathrm{T}(\frac{1}{2},\epsilon)$-item in each of the last $8n$ bins and let all other items depart such that the configuration becomes $\{16n{:}(\mathrm{T}(\frac{1}{2},\epsilon),\mathrm{T}(\epsilon,1)), 8n{:}\mathrm{T}(\frac{1}{2},\epsilon)\}$, with $24n$ bins and a total load of 1.75. We now release $\frac{16n}{\epsilon}$ items of type $\mathrm{T}(\epsilon,1)$. By similar observations as before, FF packs $\frac{8n-1}{\epsilon}$ items in the first $16n$ bins, $\frac{4n}{\epsilon}$ items in the next $8n$ bins, and opens $4n+1$ new bins for the remaining items (see Figure 6.1(d) for an illustration). We now let items depart such that the configuration becomes

$$\{16n{:}(\mathrm{T}(\tfrac{1}{2},\epsilon),\mathrm{T}(\epsilon,1)), 8n{:}(\mathrm{T}(\tfrac{1}{2},\epsilon),\mathrm{T}(\epsilon,1)), 4n{:}\mathrm{T}(\epsilon,1)\}\ ,$$

with $28n$ bins and a total load of 2.5. Let s'_{0_t} be the number of existing bins in Stage 0 at some time t. We observe that alternating between releasing $\mathrm{T}(\frac{1}{2},\epsilon)$- and $\mathrm{T}(\epsilon,1)$-items (with departure of items in-between) forces FF to open more than $16n - s'_{0_t}/2$ new bins, as each existing bin can pack a load slightly less than $1/2$ of $\mathrm{T}(\frac{1}{2},\epsilon)$- and $\mathrm{T}(\epsilon,1)$-items. By repeating this process we obtain a total of $8n+4n+2n+n+n/2+\cdots$ new bins, which converges to $16n$. Overall, in Stage 0, we obtain $32n$ bins, each packed with one $\mathrm{T}(\frac{1}{2},\epsilon)$-item and one $\mathrm{T}(\epsilon,1)$-item, thus $s_0 = 32n$.

Stage 1. In this stage, we aim at $s_1 = 16n$. At the end of Stage 0, the packing configuration is $\{32n{:}(\mathrm{T}(\frac{1}{2},\epsilon),\mathrm{T}(\epsilon,1))\}$, with $32n$ bins and a total load of 3. We then release $\frac{16n}{\epsilon}$ items of type $\mathrm{T}(1,\epsilon)$ and none of the existing bins can pack any new items due to the existence of $\mathrm{T}(\epsilon,1)$-items. FF opens $16n$ new bins, i.e., $s_1 = 16n$ as said.

Stage 2. In this stage, we aim to modify the configuration of the first $32n$ bins such that $\mathrm{T}(\epsilon,1)$-items depart, and each bin will be packed with one of each $\mathrm{T}(\frac{1}{2},\epsilon)$-, $\mathrm{T}(\epsilon,\frac{1}{2})$-, and $\mathrm{T}(1,\epsilon)$-item at the end of the stage. The last $16n$ bins will each be packed with one $\mathrm{T}(1,\epsilon)$-item. This configuration aims to force FF to open additional new bins in subsequent stages. No new bins are opened during this process. This stage follows the

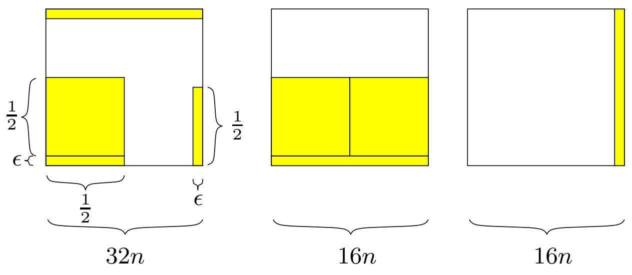


Figure 6.3: The bin configuration of First-Fit at the end of Stage 4.

approach of Stage 0, by alternating release of $T(\epsilon, \frac{1}{2})$-items and $T(1, \epsilon)$-items with item departure in-between.

We begin by departing items such that the configuration becomes $\{32n{:}T(\frac{1}{2}, \epsilon), 16n{:}T(1, \epsilon)\}$, with $48n$ bins and a total load of 2. We now release $\frac{24n}{\epsilon}$ items of type $T(\epsilon, \frac{1}{2})$ and FF packs them in the first $16n$ bins, as each bin can pack $24n$ items (see Figure 6.2(a) for an illustration). We now keep one $T(\epsilon, \frac{1}{2})$-item in each of the first $16n$ bins and let all other $T(\epsilon, \frac{1}{2})$-items depart. Similar to Stage 0, we release $\frac{16n}{\epsilon}$ items of type $T(1, \epsilon)$. In the first $16n$ bins, FF packs $\frac{8n}{\epsilon}$ items, and in the next $\lceil \frac{8n}{1-1/16n} \rceil > 8n$ bins FF packs the remaining $\frac{8n}{\epsilon}$ items. We now let items depart such that the configuration becomes

$$\{16n{:}(T(\tfrac{1}{2}, \epsilon), T(\epsilon, \tfrac{1}{2}), T(1, \epsilon)), 8n{:}(T(\tfrac{1}{2}, \epsilon), T(1, \epsilon)), 8n{:}T(\tfrac{1}{2}, \epsilon), 16n{:}T(1, \epsilon)\} \ ,$$

with $48n$ bins and a total load of 4. We can further alternate between releasing $\frac{32n}{\epsilon}$ items of type $T(\epsilon, \frac{1}{2})$ and releasing $\frac{16n}{\epsilon}$ items of type $T(1, \epsilon)$, while keeping one item of each type in each bin, and departing all other items released. We can repeat this process for $32n$ bins as in Stage 0. (Note that one $T(\frac{1}{2}, \epsilon)$-item is already packed in each of the $32n$ bins and the total number of bins that are each packed with one $T(\frac{1}{2}, \epsilon)$-, $T(\epsilon, \frac{1}{2})$-, and $T(1, \epsilon)$-item converges faster to $32n$ compared to Stage 0.) Figure 6.2(b) shows the bin configuration of FF at the end of the stage.

Stage 3. In this stage, we aim at $s_3 = 16n$. At the end of Stage 2, the packing configuration is $\{32n{:}(T(\frac{1}{2}, \epsilon), T(\epsilon, \frac{1}{2}), T(1, \epsilon)), 16n{:}T(1, \epsilon)\}$, with $48n$ bins and a total load of 5. We now release $\frac{16n}{\epsilon}$ items of type $T(\epsilon, 1)$ and FF opens $16n$ new bins, as $T(\epsilon, 1)$-items cannot be packed in any existing bins due to the existence of $T(1, \epsilon)$-items, i.e., $s_3 = 16n$ as said.

Stage 4. Similar to Stage 2, we aim to modify the configuration of the first $48n$ bins in order to force FF to open additional new bins in Stage 5. We keep one $T(\epsilon, 1)$-item in each of the last $16n$ bins and let all other $T(\epsilon, 1)$-items depart such that the configuration becomes

$$\{32n{:}(T(\tfrac{1}{2}, \epsilon), T(\epsilon, \tfrac{1}{2}), T(1, \epsilon)), 16n{:}T(1, \epsilon), 16n{:}T(\epsilon, 1)\} \ ,$$

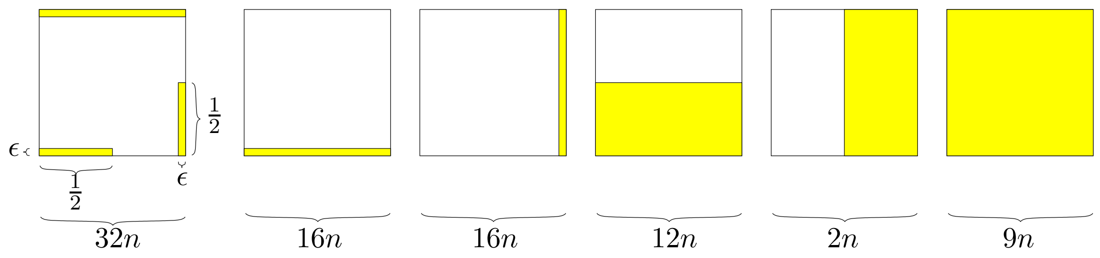


Figure 6.4: The final configuration achieved by First-Fit.

with $64n$ bins and a total load of 6. We then release $64n$ items of type $\mathrm{T}(\frac{1}{2}, \frac{1}{2})$. In the first $32n$ bins FF packs $32n$ items, one item in each bin, and in the next $16n$ bins FF packs $32n$ items, two items in each bin. The last $16n$ bins do not pack any new item and no new bins are opened. Figure 6.3 shows the bin configuration of FF at the end of the stage.

Stage 5. In this stage, we aim at $s_5 = 12n$. We keep one $\mathrm{T}(\frac{1}{2}, \frac{1}{2})$-item in each of the second $16n$ bins and let all other $\mathrm{T}(\frac{1}{2}, \frac{1}{2})$-items depart such that the configuration becomes

$$\{32n{:}(\mathrm{T}(\tfrac{1}{2}, \epsilon), \mathrm{T}(\epsilon, \tfrac{1}{2}), \mathrm{T}(1, \epsilon)), 16n{:}(\mathrm{T}(1, \epsilon), \mathrm{T}(\tfrac{1}{2}, \tfrac{1}{2})), 16n{:}\mathrm{T}(\epsilon, 1)\} \ ,$$

with $64n$ bins and a total load of $4n + 6$. We then release $24n$ items of type $\mathrm{T}(1, \frac{1}{2})$. No new items can be packed in existing bins and FF opens $12n$ new bins by packing two items in each new bin, i.e., $s_5 = 12n$ as said.

Stage 6. We aim at $s_6 = 2n$. We keep one $\mathrm{T}(1, \frac{1}{2})$-item in each of the last $12n$ bins and let all other $\mathrm{T}(1, \frac{1}{2})$-items depart, and depart all $\mathrm{T}(\frac{1}{2}, \frac{1}{2})$-items such that the configuration becomes

$$\{32n{:}(\mathrm{T}(\tfrac{1}{2}, \epsilon), \mathrm{T}(\epsilon, \tfrac{1}{2}), \mathrm{T}(1, \epsilon)), 16n{:}\mathrm{T}(1, \epsilon), 16n{:}\mathrm{T}(\epsilon, 1), 12n{:}\mathrm{T}(1, \tfrac{1}{2})\} \ ,$$

with $76n$ bins and a total load of $6n + 6$. We now release $20n$ items of type $\mathrm{T}(\frac{1}{2}, 1)$. FF packs $16n$ items in the existing $16n$ bins each packed with one $\mathrm{T}(\epsilon, 1)$-item, one item in each bin, and FF opens $2n$ new bins for the remaining items, by packing two items in each new bin, thus $s_6 = 2n$ as said.

Stage 7. In this final stage, we aim at $s_7 = 9n$. We keep one $\mathrm{T}(\frac{1}{2}, 1)$-item in each of the last $2n$ bins and let all other $\mathrm{T}(\frac{1}{2}, 1)$-items depart such that the configuration becomes

$$\{32n{:}(\mathrm{T}(\tfrac{1}{2}, \epsilon), \mathrm{T}(\epsilon, \tfrac{1}{2}), \mathrm{T}(1, \epsilon)), 16n{:}\mathrm{T}(1, \epsilon), 16n{:}\mathrm{T}(\epsilon, 1), 12n{:}\mathrm{T}(1, \tfrac{1}{2}), 2n{:}\mathrm{T}(\tfrac{1}{2}, 1)\} \ ,$$

with $78n$ bins and a total load of $7n+6$. We finally release $9n$ items of type $\mathrm{T}(1, 1)$ and FF opens $9n$ new bins, i.e., $s_7 = 9n$. In total FF uses $32n + 16n + 16n + 12n + 2n + 9n = 87n$ bins. Figure 6.4 shows an illustration of the final configuration of FF.

The optimal offline packing. The optimal offline algorithm $\mathcal{O}$ can use $16n + 6$ bins to pack all items as shown in Table 6.2. In each stage, we list the packing configuration

$\mathcal{O}$: # bins	2	2	1	1	n	$3n$	$4n$	$2n$	$6n$	total
Stage 0 Stage 0.1–A	$\boldsymbol{8n\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{8n\times\mathrm{T}(\frac{1}{2},\epsilon)}$	-	-	$\boldsymbol{(16n-2)\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{(16n-2)\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{(16n-2)\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{(32n-2)\times\mathrm{T}(\frac{1}{2},\epsilon)}$	$\boldsymbol{(32n-2)\times\mathrm{T}(\frac{1}{2},\epsilon)}$	$16n$
Stage 0.1–D	$8n\times\mathrm{T}(\epsilon,1)$	$8n\times\mathrm{T}(\frac{1}{2},\epsilon)$	-	-	-	-	-	-	-	$24n\epsilon$
Stage 0.2–A	$8n\times\mathrm{T}(\epsilon,1)$, $\boldsymbol{8n\times\mathrm{T}(\epsilon,1)}$	$8n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $\boldsymbol{8n\times\mathrm{T}(\frac{1}{2},\epsilon)}$	-	-	$\boldsymbol{(16n-1)\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(\epsilon,1)}$	$16n{+}32n\epsilon$
Stage 0.2–D	$8n\times\mathrm{T}(\epsilon,1)$, $8n\times\mathrm{T}(\epsilon,1)$	$8n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $8n\times\mathrm{T}(\frac{1}{2},\epsilon)$	-	-	-	-	-	-	-	$48n\epsilon$
Stage 1–A	$8n\times\mathrm{T}(\epsilon,1)$, $8n\times\mathrm{T}(\epsilon,1)$	$8n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $8n\times\mathrm{T}(\frac{1}{2},\epsilon)$	$\boldsymbol{16n\times\mathrm{T}(1,\epsilon)}$	-	$\boldsymbol{(16n-1)\times\mathrm{T}(1,\epsilon)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(1,\epsilon)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(1,\epsilon)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(1,\epsilon)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(1,\epsilon)}$	$16n{+}48n\epsilon$
Stage 2–D	-	$8n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $8n\times\mathrm{T}(\frac{1}{2},\epsilon)$	$16n\times\mathrm{T}(1,\epsilon)$	-	-	-	-	-	-	$32n\epsilon$
Stage 2–A	$\boldsymbol{16n\times\mathrm{T}(1,\epsilon)}$	$16n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $\boldsymbol{16n\times\mathrm{T}(\epsilon,\frac{1}{2})}$	$16n\times\mathrm{T}(1,\epsilon)$	-	$\boldsymbol{(16n-1)\times\mathrm{T}(1,\epsilon)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(1,\epsilon)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(1,\epsilon)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(1,\epsilon)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(1,\epsilon)}$	$16n{+}64n\epsilon$
Stage 2–D	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $16n\times\mathrm{T}(\epsilon,\frac{1}{2})$	$16n\times\mathrm{T}(1,\epsilon)$	-	-	-	-	-	-	$80n\epsilon$
Stage 3–A	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $16n\times\mathrm{T}(\epsilon,\frac{1}{2})$	$16n\times\mathrm{T}(1,\epsilon)$	$\boldsymbol{16n\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(\epsilon,1)}$	$\boldsymbol{(16n-1)\times\mathrm{T}(\epsilon,1)}$	$16n{+}80n\epsilon$
Stage 4–D	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $16n\times\mathrm{T}(\epsilon,\frac{1}{2})$	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\epsilon,1)$	-	-	-	-	-	$96n\epsilon$
Stage 4–A	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $16n\times\mathrm{T}(\epsilon,\frac{1}{2})$	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\epsilon,1)$	$\boldsymbol{4\times\mathrm{T}(\frac{1}{2},\frac{1}{2})}$	$\boldsymbol{4\times\mathrm{T}(\frac{1}{2},\frac{1}{2})}$	$\boldsymbol{4\times\mathrm{T}(\frac{1}{2},\frac{1}{2})}$	$\boldsymbol{4\times\mathrm{T}(\frac{1}{2},\frac{1}{2})}$	$\boldsymbol{4\times\mathrm{T}(\frac{1}{2},\frac{1}{2})}$	$16n{+}96n\epsilon$
Stage 5–D	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $16n\times\mathrm{T}(\epsilon,\frac{1}{2})$	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\epsilon,1)$	$4\times\mathrm{T}(\frac{1}{2},\frac{1}{2})$	$4\times\mathrm{T}(\frac{1}{2},\frac{1}{2})$	-	-	-	$4n{+}96n\epsilon$
Stage 5–A	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $16n\times\mathrm{T}(\epsilon,\frac{1}{2})$	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\epsilon,1)$	$4\times\mathrm{T}(\frac{1}{2},\frac{1}{2})$	$4\times\mathrm{T}(\frac{1}{2},\frac{1}{2})$	$\boldsymbol{2\times\mathrm{T}(1,\frac{1}{2})}$	$\boldsymbol{2\times\mathrm{T}(1,\frac{1}{2})}$	$\boldsymbol{2\times\mathrm{T}(1,\frac{1}{2})}$	$16n{+}96n\epsilon$
Stage 6–D	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $16n\times\mathrm{T}(\epsilon,\frac{1}{2})$	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\epsilon,1)$	-	-	$2\times\mathrm{T}(1,\frac{1}{2})$	$2\times\mathrm{T}(1,\frac{1}{2})$	-	$6n{+}96n\epsilon$
Stage 6–A	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $16n\times\mathrm{T}(\epsilon,\frac{1}{2})$	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\epsilon,1)$	$\boldsymbol{2\times\mathrm{T}(\frac{1}{2},1)}$	$\boldsymbol{2\times\mathrm{T}(\frac{1}{2},1)}$	$2\times\mathrm{T}(1,\frac{1}{2})$	$2\times\mathrm{T}(1,\frac{1}{2})$	$\boldsymbol{2\times\mathrm{T}(\frac{1}{2},1)}$	$16n{+}96n\epsilon$
Stage 7–D	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $16n\times\mathrm{T}(\epsilon,\frac{1}{2})$	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\epsilon,1)$	$2\times\mathrm{T}(\frac{1}{2},1)$	-	$2\times\mathrm{T}(1,\frac{1}{2})$	$2\times\mathrm{T}(1,\frac{1}{2})$	-	$7n{+}96n\epsilon$
Stage 7–A	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\frac{1}{2},\epsilon)$, $16n\times\mathrm{T}(\epsilon,\frac{1}{2})$	$16n\times\mathrm{T}(1,\epsilon)$	$16n\times\mathrm{T}(\epsilon,1)$	$2\times\mathrm{T}(\frac{1}{2},1)$	$\boldsymbol{\mathrm{T}(1,1)}$	$2\times\mathrm{T}(1,\frac{1}{2})$	$2\times\mathrm{T}(1,\frac{1}{2})$	$\boldsymbol{\mathrm{T}(1,1)}$	$16n{+}96n\epsilon$

Table 6.2: The optimal schedule used by the adversary for First-Fit. "A" and "D" denote arrival and departure of items, respectively.

of $\mathcal{O}$ when the maximum number of bins is used by FF and the maximum load of items is released. Bolded entries are new items arrived in the corresponding stage. Where applicable, we also list the packing configuration after departure of items. Some items that never depart are packed in a constant number of bins. The last column shows the total load of all bins.

Theorem 6.2. *The First-Fit algorithm without classification is not c-competitive for any $c < 5.4375$ for two-dimensional power fraction items.*

Unit fraction items lower bound. The lower bound of FF for one-dimensional dynamic bin packing of unit fraction items was proven to be 2.45 in [18]. We can modify our adversary for power fraction items given above such that the lower bound of FF without classification of two-dimensional unit fraction items is 5.45 in the following way. We first complete Stages 0–2 of our adversary without modification. Notice that at the end of Stage 2, the packing configuration is $\{32n{:}(\mathrm{T}(\frac{1}{2},\epsilon),\mathrm{T}(\epsilon,\frac{1}{2}),\mathrm{T}(1,\epsilon)), 16n{:}\mathrm{T}(1,\epsilon)\}$, with $48n$ bins and a total load of $5 = O(1)$. We can now use the adversary for 1-D items by fixing the height of items released to be 1 and the width corresponds to the length of 1-D items. Items released are of type $\mathrm{T}(x,1)$, for variable width x, and no items released can be packed in existing bins. The 1-D adversary can also be modified in such a way that the optimal offline algorithm uses $16n$ bins, while FF uses $16n \times 2.45$ bins. Using this approach the lower bound can be improved to $48n/16n + 39.2n/16n = 5.45$.

Corollary 6.3. *The First-Fit algorithm without classification is not c-competitive for any* $c < 5.45$ *for two-dimensional unit fraction items.*

3-D unit and power fraction items. The lower bounds for First-Fit for 3-D unit fraction items and 3-D power fraction items can be obtained as 6.45 and 6.4375, respectively, as follows. Recall that $\mathrm{T}(x_1, x_2, x_3)$ refers to the type of items with lengths x_1, x_2, x_3 in the three dimensions, in a similar way to 2-D items. Before Stage 0 of the adversaries for 2-D unit and power fraction items, we release a load of $16n$ of items of type $\mathrm{T}(1, 1, \epsilon)$ and FF uses $16n$ bins. Afterwards, we depart most of the items, such that in each of the $16n$ bins we are only left with one $\mathrm{T}(1, 1, \epsilon)$-item. The adversaries for 2-D unit and power fraction items may now be used by releasing items of type $\mathrm{T}(w, h, 1)$, where w and h correspond to the widths and heights of 2-D items used in the 2-D adversaries. Due to 3-D packing restrictions, none of the items released from Stage 0 onwards can be packed in the first $16n$ bins. The optimal offline algorithm will only require one additional bin in order to pack $\mathrm{T}(1, 1, \epsilon)$-items that never depart.

Corollary 6.4. *The First-Fit algorithm without classification is not c-competitive for any* $c < 6.45$ *for three-dimensional unit fraction items and is not c-competitive for any* $c < 6.4375$ *for three-dimensional power fraction items.*

6.4.2 Best-Fit

We give an adversary for Best-Fit (BF) that consists of n stages, for an arbitrarily large integer n that is a power of 2, and aims to force BF to open n bins using at most a total load of $2 = O(1)$ only. The adversary is constructed taking into account the fact that BF packs a new item to the heaviest loaded bin, breaking ties arbitrarily. Roughly speaking, in each stage the adversary will force BF to open a new bin using a small load of items, regardless of how many existing bins are already opened in the packing configuration. This is made possible by 2-D packing constraints. The adversary then modifies the configuration of the existing bins (without completely emptying the bins at any time) such that they will be packed with the same item types as the newly opened bin. This will allow the adversary to open a new bin in each subsequent stage.

Theorem 6.5. *The competitive ratio of the Best-Fit algorithm without classification for two-dimensional power fraction items is unbounded.*

Proof. Let n be an arbitrarily large integer that is a power of 2 and $\epsilon = \frac{1}{n}$. The adversary aims to force BF to open one bin in each stage, such that BF uses $i + 1$ bins at the end of Stage i, for $0 \leq i < n$. Notice that in Stage i, $i + 1$ corresponds to the index of the opened bin in that stage. In a slight abuse of notation, we identify the bins by their indices, with the understanding that bin i refers to the opened bin in Stage $i - 1$. We

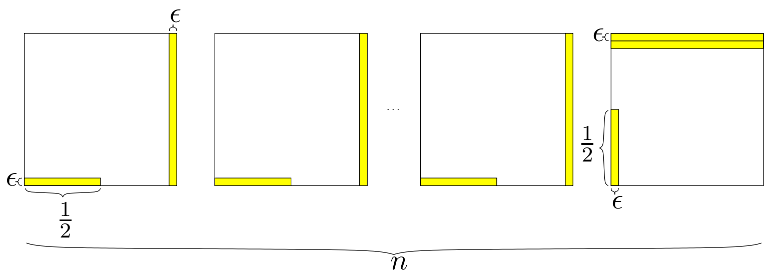


Figure 6.5: The final configuration achieved by Best-Fit for n bins.

denote by ℓ_i the load of bin i. Note that the indices of bins do not change as no bin is emptied entirely during the sequence.

The adversary will use a limited number of different items in order to open a new bin in each stage. In consecutive stages, the existing bins will alternate between the packing configuration of $\{(\mathrm{T}(\epsilon, \frac{1}{2}), \mathrm{T}(1, \epsilon))\}$ and $\{(\mathrm{T}(\frac{1}{2}, \epsilon), \mathrm{T}(\epsilon, 1))\}$. To unify the discussion, we let $((w_1, h_1), (w_2, h_2)) \in \{((\epsilon, \frac{1}{2}), (1, \epsilon)), ((\frac{1}{2}, \epsilon), (\epsilon, 1))\}$. Notice that (a) a $\mathrm{T}(w_2, h_2)$-item cannot be packed in the same bin with a $\mathrm{T}(h_2, w_2)$-item due to 2-D packing constraints, e.g., if $\mathrm{T}(w_2, h_2) = \mathrm{T}(1, \epsilon)$ then $\mathrm{T}(h_2, w_2) = \mathrm{T}(\epsilon, 1)$ and $w_2 + h_2 > 1$; and (b) the load of each pair of $\mathrm{T}(w_1, h_1)$-item and $\mathrm{T}(w_2, h_2)$-item is $\frac{3\epsilon}{2}$. By (a) the adversary can force BF to open a new bin using one type of item if all other existing bins contain at least one item of different type. By (b) the adversary can ensure that even if two bins each contain different pairs of items, their load is the same, and this helps with modifying the configuration of existing bins. I.e., when modifying the configuration of a bin we ensure that it has highest load, while all other bins have equal lighter load, regardless of their configuration. The adversary sequence ensures that at the beginning of each stage, each existing bin will be packed with one pair of items defined above. Suppose that in the beginning of Stage i each bin is packed with one $\mathrm{T}(w_1, h_1)$-item and one $\mathrm{T}(w_2, h_2)$-item belonging to the set of items defined above. Then, the adversary releases $\mathrm{T}(h_2, w_2)$-items and $\mathrm{T}(h_1, w_1)$-items during the stage, such that a new bin is opened. Afterwards the adversary aims to modify the configuration of all bins such that at the end of the stage each bin will contain one $\mathrm{T}(h_2, w_2)$-item and $\mathrm{T}(h_1, w_1)$-item. In the next stage, the process is repeated by first releasing $\mathrm{T}(w_1, h_1)$-items and $\mathrm{T}(w_2, h_2)$-items.

In the beginning of Stage 0, there are no open bins and we set $((w_1, h_1), (w_2, h_2)) = ((\epsilon, \frac{1}{2}), (1, \epsilon))$. We start in Stage 0 and release one $\mathrm{T}(w_1, h_1)$-item and one $\mathrm{T}(w_2, h_2)$-item. BF packs them in one bin. For any later Stage i, with $i > 0$, we have the following steps (repeating $n - 1$ times) consisting of sequences of arrival and departure of items.

Step 1. We first release two $\mathrm{T}(h_2, w_2)$-items and BF packs both items to a new bin with index $i+1$ as no existing bin can pack the items. We further release one $\mathrm{T}(h_1, w_1)$-item and BF packs the item in bin $i+1$ as $\ell_{i+1} = 2 \times (\epsilon \times 1) > \ell_i = \ell_{i-1} = \cdots = \ell_1 = \frac{3\epsilon}{2}$. If $i + 1 = n$, we have obtained n bins and stop, otherwise we continue as follows.

Step 2. In this step, we aim to modify the configuration of all $i + 1$ bins without closing any bin such that, at the end of the process, each bin will be packed with one

$\mathrm{T}(h_1, w_1)$-item and one $\mathrm{T}(h_2, w_2)$-item.

(i) We first depart one $\mathrm{T}(h_2, w_2)$-item from bin $i+1$.

For each of the first $1 \leq j \leq i$ bins we repeat the following sequence of item departure and arrival in Steps 2(ii)–2(vii).

(ii) Depart one $\mathrm{T}(w_1, h_1)$-item from bin j.

(iii) If $\mathrm{T}(w_1, h_1) = \mathrm{T}(\epsilon, \frac{1}{2})$, release one $\mathrm{T}(1, \frac{1}{2})$-item, otherwise release one $\mathrm{T}(\frac{1}{2}, 1)$-item. BF packs the item in bin j as no other bin can accommodate the item. This is because all bins with index smaller than j and bin $i+1$ are packed with one $\mathrm{T}(h_1, w_1)$-item and one $\mathrm{T}(h_2, w_2)$-item and all remaining bins with index larger than j, except for bin $i+1$, are packed with one $\mathrm{T}(w_1, h_1)$-item and one $\mathrm{T}(w_2, h_2)$-item.

(iv) Release four $\mathrm{T}(h_1, w_1)$-items. All existing bins can accommodate the new items, however BF packs them to bin j as it has highest load, i.e., $\ell_j = (1\times\epsilon)+(1\times\frac{1}{2}) > \ell_k = \frac{3\epsilon}{2}$, for any bin $k \neq j$.

(v) Depart the $\mathrm{T}(1, \frac{1}{2})$-item or $\mathrm{T}(\frac{1}{2}, 1)$-item packed in bin j and additionally depart the $\mathrm{T}(w_2, h_2)$-item from bin j; note that ℓ_j becomes $4 \times (\frac{1}{2}\times\epsilon) = 2\epsilon$.

(vi) Release one $\mathrm{T}(h_2, w_2)$-item; BF packs it in bin j for a similar reason as before.

(vii) Finally depart three $\mathrm{T}(h_1, w_1)$-items from bin j, and the bin remains packed with one $\mathrm{T}(h_1, w_1)$-item and one $\mathrm{T}(h_2, w_2)$-item, for a total load of $\frac{3\epsilon}{2}$.

At the end we obtain $i+1$ bins, each packed with one $\mathrm{T}(h_1, w_1)$-item and one $\mathrm{T}(h_2, w_2)$-item.

Step 3. Set $((w_1, h_1), (w_2, h_2)) = ((h_1, w_1), (h_2, w_2))$ and go back to Step 1, where we proceed to the next stage.

Figure 6.5 shows an illustration of the final configuration of BF.

The optimal offline packing. On the other hand, the optimal offline algorithm $\mathcal{O}$ uses four bins. The first bin packs at most n items of type $\mathrm{T}(\epsilon, \frac{1}{2})$ and n items of type $\mathrm{T}(\frac{1}{2}, \epsilon)$, the second bin packs at most n items of type $\mathrm{T}(\epsilon, 1)$, and the third bin packs at most n items of type $\mathrm{T}(1, \epsilon)$. The last bin packs one item of type $\mathrm{T}(\frac{1}{2}, 1)$ and three additional items of type $\mathrm{T}(\epsilon, \frac{1}{2})$, or one item of type $\mathrm{T}(1, \frac{1}{2})$ and three additional items of type $\mathrm{T}(\frac{1}{2}, \epsilon)$, at any time. Thus, the competitive ratio of BF is $n/4$ and for an arbitrarily large n the theorem follows. $\square$

6.4.3 Worst-Fit

In this section, we consider the lower bound of the Worst-Fit (WF) algorithm for two-dimensional power fraction items when no classification of items is used. The items used by the adversary for WF are a subset of the items used for the adversary of Best-Fit (BF) in Section 6.4.2. For completeness, we give the structure of the adversary in Theorem 6.6. Recall that WF assigns a new item R to the lightest loaded bin, breaking ties arbitrarily. Thus, in contrast to BF, in order to individually modify the configuration

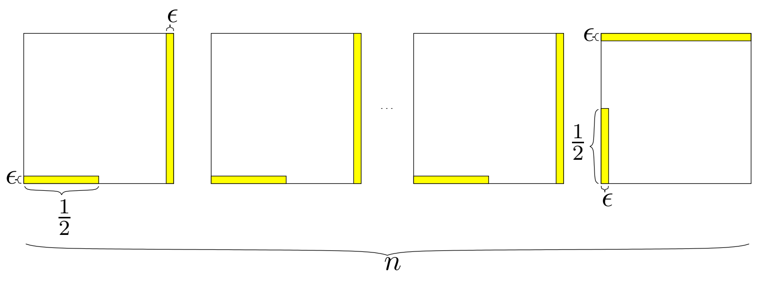


Figure 6.6: The final configuration achieved by Worst-Fit for n bins.

of an existing bin we require that it has lightest load. Similar to BF, the adversary consists of n stages and aims to force WF to open n bins using at most a total load of $3/2 = O(1)$ only.

Theorem 6.6. *The competitive ratio of the Worst-Fit algorithm without classification for two-dimensional power fraction items is unbounded.*

Proof. Let n be an arbitrarily large integer that is a power of 2 and $\epsilon = \frac{1}{n}$. The adversary aims to force WF to open one bin in each stage, such that WF uses $i+1$ bins at the end of Stage i, for $0 \le i < n$. Notice that in Stage i, $i+1$ corresponds to the index of the opened bin in that stage. In a slight abuse of notation, we identify the bins by their indices, with the understanding that bin i refers to the opened bin in Stage $i-1$. We denote by ℓ_i the load of bin i. Note that the indices of bins do not change as no bin is emptied entirely during the sequence.

The adversary will use a limited number of different items in order to open a new bin in each stage. In consecutive stages, the existing bins will alternate between the packing configuration of $\{(\mathrm{T}(\epsilon, \frac{1}{2}), \mathrm{T}(1, \epsilon))\}$ and $\{(\mathrm{T}(\frac{1}{2}, \epsilon), \mathrm{T}(\epsilon, 1))\}$. To unify the discussion, we let $((w_1, h_1), (w_2, h_2)) \in \{((\epsilon, \frac{1}{2}), (1, \epsilon)), ((\frac{1}{2}, \epsilon), (\epsilon, 1))\}$. The adversary makes use of the fact that (a) a $\mathrm{T}(w_2, h_2)$-item cannot be packed in the same bin with a $\mathrm{T}(h_2, w_2)$-item due to 2-D packing constraints and (b) each pair of items has equal load of $\frac{3\epsilon}{2}$. By (a) the adversary can force WF to open a new bin using one type of item if all other existing bins contain at least one item of different type. By (b) the adversary can ensure that even if two bins each contain different pairs of items, their load is the same. This helps with modifying the configuration of an existing bin, as we ensure that it has lightest load, while all other bins have equal higher load, regardless of their configuration. The adversary sequence ensures that at the beginning of each stage, each existing bin will be packed with one pair of items defined above. If in Stage i each bin is packed with one $\mathrm{T}(w_1, h_1)$-item and one $\mathrm{T}(w_2, h_2)$-item then the adversary releases $\mathrm{T}(h_2, w_2)$-items and $\mathrm{T}(h_1, w_1)$-items during the stage, such that a new bin is opened. Afterwards the adversary aims to modify the configuration of all bins such that at the end of the stage each bin will contain one $\mathrm{T}(h_2, w_2)$-item and $\mathrm{T}(h_1, w_1)$-item. In the next stage, the process is repeated by first releasing $\mathrm{T}(w_1, h_1)$-items and $\mathrm{T}(w_2, h_2)$-items.

In the beginning of Stage 0, there are no open bins and we set $((w_1, h_1), (w_2, h_2)) = ((\epsilon, \frac{1}{2}), (1, \epsilon))$. We start in Stage 0 and release one $\mathrm{T}(w_1, h_1)$-item and one $\mathrm{T}(w_2, h_2)$-item. WF packs them in one bin. For any later Stage i, with $i > 0$, we have the following steps (repeating $n - 1$ times) consisting of sequences of arrival and departure of items.

Step 1. Release one $\mathrm{T}(h_2, w_2)$-item and WF opens a new bin with index $i + 1$ as no existing bins can pack the item. We further release one $\mathrm{T}(h_1, w_1)$-item and WF packs the item in bin $i + 1$ as $\ell_{i+1} = \epsilon \times 1 < \ell_i = \ell_{i-1} = \cdots = \ell_1 = \frac{3\epsilon}{2}$. If $i + 1 = n$, we have obtained n bins and stop, otherwise we continue as follows.

Step 2. In this step, we aim to modify the configuration of the first i bins without closing any bin such that, at the end of the process, each bin will be packed with one $\mathrm{T}(h_1, w_1)$-item and one $\mathrm{T}(h_2, w_2)$-item.

(i) Depart all $\mathrm{T}(w_2, h_2)$-items from the first i bins.

(ii) Release i items of type $\mathrm{T}(h_1, w_1)$ and WF packs one item in each of the first i bins as they have equal lightest load $\frac{\epsilon}{2}$.

(iii) Depart all $\mathrm{T}(w_1, h_1)$-items from the first i bins.

(iv) Release i items of type $\mathrm{T}(h_2, w_2)$. As before, WF packs one item in each of the first i bins.

At the end we have obtained $i + 1$ bins, each packed with one $\mathrm{T}(h_1, w_1)$-item and one $\mathrm{T}(h_2, w_2)$-item.

Step 3. Set $((w_1, h_1), (w_2, h_2)) = ((h_1, w_1), (h_2, w_2))$ and go back to Step 1, where we proceed to the next stage.

Figure 6.6 shows an illustration of the final configuration of WF.

The optimal offline packing. The optimal offline algorithm $\mathcal{O}$ uses three bins, one bin for at most n items of type $\mathrm{T}(\epsilon, \frac{1}{2})$ and n items of type $\mathrm{T}(\frac{1}{2}, \epsilon)$, one bin for at most n items of type $\mathrm{T}(\epsilon, 1)$, and one bin for at most n items of type $\mathrm{T}(1, \epsilon)$. Thus, the competitive ratio of WF is $n/3$ and for an arbitrarily large n the theorem follows. □

6.5 Conclusion

We have shown that the family of Any-Fit algorithms that does not use classification of items is not competitive for online multi-dimensional dynamic bin packing of general size items. In contrast, First-Fit without classification has a competitive ratio of 2.897 [26] for online one-dimensional dynamic bin packing of general size items. It can also be shown that Any-Fit (including Best-Fit and Worst-Fit) is 3-competitive for 1-D general size items using an argument from [26] for First-Fit. Furthermore, we show that Best-Fit and Worst-Fit without classification are also not competitive for online multi-dimensional dynamic bin packing of power fraction items. Lastly, First-Fit without classification of items has lower bounds of at least 5.4375 for 2-D power fraction items, at least 6.4375 for 3-D power fraction items, at least 5.45 for 2-D unit fraction items, and at least 6.45 for 3-D unit fraction items. All these results are in contrast to the general lower bounds of any algorithm of 3.70301 [37] for 2-D unit fraction items and 4.85383 [37] for 3-D unit fraction items. One of the directions for future work would be to obtain a better lower bound for any algorithm, as the upper bounds for 2-D and 3-D unit fraction items currently stand at 6.7850 and 21.6108, respectively.

Chapter 7

Conclusion

In this work we have studied the online multi-dimensional dynamic bin packing problem. In the online model of the bin packing problem items arrive one at a time and irrevocable decisions must be made on where to pack items such that the total number of bins used is minimized. In the dynamic setting items may also depart at unknown arbitrary time and the goal becomes to minimize the total number of bins used over all time.

For online one-dimensional dynamic bin packing, we have improved the best known lower bound of 2.5 [19] to $8/3 \sim 2.666$ in Chapter 3. The previous lower bounds were 2.388 [26], 2.428 [18], and 2.5 [19]. We designed two operations called Op-Inc and Op-Comp that release items of slightly increasing sizes and items with complementary sizes. These operations make a more systematic approach to release items: the type of item sizes used in a later case is a superset of those used in an earlier case. This is in contrast to the previous 2.5 lower bound in [19] in which rather different sizes are used in different cases. Furthermore, in each case, we use one or two pairs of Op-Inc and Op-Comp, which makes the structure clearer and the proof easier to understand. We also show that the new operations defined lead to a much easier proof for a 2.5 lower bound. We believe the design of this adversary may be useful in other optimization problems.

The upper bound result for the one-dimensional dynamic bin packing problem, based on the analysis of a modified version of First-Fit, currently stands at 2.788 [26]. There has been no progress on improving the upper bound for more than 30 years at this time. The current best approach for online static bin packing is using Harmonic-type algorithms that classify items into a large number of classes and then pack items in each class independently of each other. This approach does not seem to be suitable for online dynamic bin packing as discussed in Section 2.3.1. It is also not yet clear whether the lower bound can be improved beyond 2.666. As we have seen throughout this work, First-Fit appears to give the best competitive ratios for online dynamic bin packing. A different analysis of First-Fit may be worth pursuing towards a better upper bound, perhaps using weighting function techniques ([44, 53]), through which the absolute approximation ratio and asymptotic approximation ratio of First-Fit were proven to be equal and exactly 1.7 [35] for static bin packing.

We have initiated the study of two- and three-dimensional dynamic bin packing of unit fraction and power fraction items, providing algorithms with better worst-case performance compared to algorithms for general size items in Chapters 4 and 5. Specifically, we have improved the competitive ratios that would be obtained using only existing results for unit fraction items from 7.4842 to 6.7850 for 2-D, and from 22.4842 to 21.6108 for 3-D. For power fraction items, the competitive ratios are further reduced to 6.2455 and 20.0783 for 2-D and 3-D, respectively. Our approach is to divide items into classes and analyze each class individually. We have proposed several classes and defined different packing schemes based on the classes. This approach gives a systematic way to explore different combinations of classes and future work may easily build on our results.

The technique of dividing items into classes for online multi-dimensional dynamic bin packing seems to be in certain cases necessary. As we have shown in Chapter 6, the family of Any-Fit algorithms that does not use classification of items is not competitive for online multi-dimensional dynamic bin packing of general size items. Furthermore, we have also shown that Best-Fit and Worst-Fit without classification are also not competitive for online multi-dimensional dynamic bin packing of power fraction items. First-Fit without classification of items has a lower bound of at least 5.4375 for 2-D power fraction items and at least 5.45 for 2-D unit fraction items. For 3-D power and unit fraction items, First-Fit is at least 6.4375- and 6.45-competitive, respectively.

For 2-D and 3-D unit fraction items, the lower bounds for any algorithm are 3.70301 and 4.85383 [37], respectively. There is still quite a gap between these and the upper bounds for unit fraction items (6.7850 for 2-D and 21.6108 for 3-D), especially for 3-D, and future work may focus on improving the lower bound. A different direction is improvement specifically on the lower bounds of any algorithm for general size items, as the current best results are obtained using only 2-D and 3-D unit fraction items in the construction of the adversaries [37]. The upper bounds for 2-D and 3-D general size items currently stand at 7.788 and 22.788 [70], respectively.

For the classes given for 2-D unit fraction items in Chapter 4, we have obtained some preliminary results on a lower bound for any online algorithm that uses the given classification of items. These types of lower bounds would allow to determine how good a classification and packing configuration are, an approach not yet explored in online dynamic bin packing literature. For completeness we will give the details of our preliminary lower bounds for two classes of 2-D unit fraction items in Observations 7.1 and 7.2 (the notations used are described in Section 6.2).

The following observation shows a lower bound of 1.5 for any online algorithm for Class 5 two-dimensional unit fraction items. As detailed in Section 4.3.1, First-Fit obtains a competitive ratio of 1.5 for Class 5 items, thus the bound is tight for this input.

Observation 7.1. For any online dynamic bin packing algorithm $\mathcal{A}$ with input of types $\mathrm{T}(1,1)$ and $\mathrm{T}(\frac{1}{2},1)$, $\mathcal{A}$ is not c-competitive for any $c < 1.5$.

Proof. We give an adversary for any online algorithm $\mathcal{A}$ for items of types $\mathrm{T}(1,1)$ and $\mathrm{T}(\frac{1}{2},1)$, i.e., Class 5 items, such that at any time the load of items released and not departed is at most n, for some large even integer n. We consider two stages of a sequence of arrivals and departures of items of types $\mathrm{T}(1,1)$ and $\mathrm{T}(\frac{1}{2},1)$. We prove that $\mathcal{A}$ uses $3n/2$ bins, while the optimal offline algorithm $\mathcal{O}$ uses at most n bins at any time. Then, the competitive ratio of $\mathcal{A}$ is at least 1.5, which matches the upper bound achieved by First-Fit (FF) for Class 5 items.

Stage 0. We release $2n$ items of type $\mathrm{T}(\frac{1}{2},1)$, with total load n. $\mathcal{A}$ needs at least n bins by packing at most two $\mathrm{T}(\frac{1}{2},1)$-items in each bin, thus $s_0 \geq n$.

Stage 1. In this stage, we aim at $s_1 = n/2$. We keep one $\mathrm{T}(\frac{1}{2},1)$-item in each of n bins and let all other items depart such that the configuration becomes $\{n{:}\mathrm{T}(\frac{1}{2},1)\}$, with n bins and a total load of $n/2$. We then release $n/2$ items of type $\mathrm{T}(1,1)$. The first n bins cannot pack any $\mathrm{T}(1,1)$-item and $\mathcal{A}$ opens $n/2$ new bins, thus $s_1 = n/2$. In total $\mathcal{A}$ uses $3n/2$ bins.

The optimal offline packing. In Stage 0, n items of type $\mathrm{T}(\frac{1}{2},1)$ that never depart are packed in $n/2$ bins and the remaining n items in an additional $n/2$ bins. In Stage 1, the second $n/2$ are emptied and $n/2$ items of type $\mathrm{T}(1,1)$ released are packed in these bins. Thus $\mathcal{O}$ uses at most n bins at any time and the lemma follows. □

This final observation shows a lower bound of 1.9375 for any online algorithm for Class 3 two-dimensional unit fraction items. The bound is not tight, as First-Fit obtains a competitive ratio of 2.25 (Section 4.3.2).

Observation 7.2. For any online dynamic bin packing algorithm $\mathcal{A}$ with input of types $\mathrm{T}(1,1)$, $\mathrm{T}(1,\frac{1}{2})$, $\mathrm{T}(\frac{1}{2},1)$, and $\mathrm{T}(\frac{1}{2},\frac{1}{2})$, $\mathcal{A}$ is not c-competitive for any $c < 1.9375$.

Proof. We give an adversary for any online algorithm $\mathcal{A}$ for items of types $\mathrm{T}(1,1)$, $\mathrm{T}(1,\frac{1}{2})$, $\mathrm{T}(\frac{1}{2},1)$, and $\mathrm{T}(\frac{1}{2},\frac{1}{2})$, i.e., Class 3 items, such that at any time the load of items released and not departed is at most $16n$, for some large integer n. We prove that $\mathcal{A}$ uses $31n$ bins, while the optimal offline algorithm $\mathcal{O}$ uses at most $16n$ bins at any time. Then, the competitive ratio of $\mathcal{A}$ is at least 1.9375.

Stage 0. We release $64n$ items of type $\mathrm{T}(\frac{1}{2},\frac{1}{2})$, with total load $16n$, and $\mathcal{A}$ needs at least $16n$ bins, i.e., $s_0 \geq 16n$. $\mathcal{A}$ can pack at most four $\mathrm{T}(\frac{1}{2},\frac{1}{2})$-items in each bin, thus $\mathcal{A}$ needs at least $64n/4 = 16n$ bins as said.

Stage 1. In this stage, we aim at $s_1 \geq 4n$. We keep one $\mathrm{T}(\frac{1}{2},\frac{1}{2})$-item in each of $16n$ bins and let other items depart such that the configuration is $\{16n{:}\mathrm{T}(\frac{1}{2},\frac{1}{2})\}$, with $16n$ bins a total load of $4n$. We then release $24n$ items of type $\mathrm{T}(1,\frac{1}{2})$. The existing bins can pack at most $16n$ new items, one item in each bin. For the remaining $8n$ items, $\mathcal{A}$ needs to open at least $4n$ new bins by packing at most two $\mathrm{T}(1,\frac{1}{2})$-items in each bin, thus $s_1 \geq 4n$.

Stage 2. In this stage, we aim at $s_2 \geq 2n$. We let all $\mathrm{T}(1,\frac{1}{2})$-items that have packed in the first $16n$ bins depart and keep one $\mathrm{T}(1,\frac{1}{2})$-item in each of the next $4n$ bins. The

		1-D bounds		2-D bounds		3-D bounds	
Algorithm	Input	Upper	Lower	Upper	Lower	Upper	Lower
Best-in-Class	General	2.788 [26]	2.75 [26] [*]	7.788 [70]	3.70301 [37]	22.788 [70]	4.85383 [37]
	UF	2.4842 [47]	2.45 [18]	**6.7850** [15]	3.70301 [37]	**21.6108** [15]	4.85383 [37]
	PF	2.4842 [47]	?	**6.2455** [15]	?	**20.0783** [15]	?
First-Fit	General	2.897 [26]	2.75 [26]	∞	∞	∞	∞
	UF	2.4842 [47]	2.45 [18]	?	**5.45**	?	**6.45**
	PF	2.4842 [47]	?	?	**5.4375**	?	**6.4375**
Best-Fit	General	3 [18, 26]	3 [18]	∞	∞	∞	∞
	UF	3 [18]	3 [18]	∞	∞	∞	∞
	PF	3 [18]	3 [18]	∞	∞	∞	∞
Worst-Fit	General	3 [18, 26]	3 [18]	∞	∞	∞	∞
	UF	3 [18]	3 [18]	∞	∞	∞	∞
	PF	3 [18]	3 [18]	∞	∞	∞	∞
Any-Fit	General	3 [18, 26]	**2.666** [71]	∞	∞	∞	∞
	UF	3 [18]	2.428 [18]	?	3.70301 [37]	?	4.85383 [37]
	PF	3 [18]	?	?	?	?	?
$\mathcal{A}$	General	-	**2.666** [71]	-	3.70301 [37]	-	4.85383 [37]
	UF	-	2.428 [18]	-	3.70301 [37]	-	4.85383 [37]
	PF	-	?	-	?	-	?

Table 7.1: Bounds for the Best-in-Class algorithms and the family of Any-Fit algorithms without classification of items. Bolded entries are results obtained in this thesis. For input, General, UF, and PF refer to general size, unit fraction, and power fraction items, respectively. $\mathcal{A}$ refers to any online algorithm. "∞" denotes unbounded competitive ratios and "?" denotes unknown bounds. [*] The lower bound construction allows the optimal offline algorithm to migrate items between bins.

configuration becomes $\{16n{:}\mathrm{T}(\frac{1}{2},\frac{1}{2}), 4n{:}\mathrm{T}(1,\frac{1}{2})\}$, with $20n$ bins and a total load of $6n$. We now release $20n$ items of type $\mathrm{T}(\frac{1}{2},1)$. $\mathcal{A}$ may pack at most $16n$ items in the first $16n$ bins and none in the second $4n$ bins. For the remaining $4n$ items, $\mathcal{A}$ needs to open at least $2n$ bins by packing at most two $\mathrm{T}(\frac{1}{2},1)$-items in each bin, thus $s_2 \geq 2n$.

Stage 3. In this final stage, we aim at $s_3 = 9n$. We let items depart such that the configuration becomes

$$\{16n{:}\mathrm{T}(\tfrac{1}{2},\tfrac{1}{2}), 4n{:}\mathrm{T}(1,\tfrac{1}{2}), 2n{:}\mathrm{T}(\tfrac{1}{2},1)\}\ ,$$

with $22n$ bins and a total load of $7n$. We finally release $9n$ items of type $\mathrm{T}(1,1)$ and $\mathcal{A}$ opens $9n$ new bins, i.e., $s_3 = 9n$. In total $\mathcal{A}$ uses $16n + 4n + 2n + 9n = 31n$ bins.

The optimal offline packing. In Stage 0, all $\mathrm{T}(\frac{1}{2},\frac{1}{2})$-items that never depart are packed in $4n$ bins and the remaining items in $12n$ bins. In Stage 1, the second $12n$ bins are emptied. All $\mathrm{T}(1,\frac{1}{2})$-items released that never depart are packed in $2n$ bins and the remaining items in $10n$ bins. In Stage 2, the last $10n$ bins are emptied. The $\mathrm{T}(\frac{1}{2},1)$-items that never depart are packed in n bins and the remaining items in $9n$ bins. Finally in Stage 3, the $\mathrm{T}(1,1)$-items are packed in the last $9n$ bins emptied at the beginning of the stage. Note that $\mathcal{O}$ uses at most $16n$ bins at any time and the lemma follows. $\square$

To provide some perspective and summarize our work in this thesis, we present in Table 7.1 state of the art results for online multi-dimensional dynamic bin packing of general size items, unit fraction items and power fraction items. The Best-in-Class algorithms refer to the following algorithms, all of which (apart from First-Fit) use classification of items. For 1-D general size items the algorithm is First-Fit Modified [26], while for 1-D unit and power fraction items the algorithm is First-Fit [47]. For 2-D general size, unit fraction, and power fraction items the algorithms are 2DDynamicPack [70], 2DDynamicPackUFS1 [15] (Chapter 4), and 2DDynamicPackPF [15] (Chapter 5), respectively. For 3-D general size, unit fraction, and power fraction items the algorithms are 3DDynamicPack [70], 3DDynamicPackUF [15] (Chapter 5), and 3DDynamicPackPF [15] (Chapter 5), respectively.

Apart from unknown results and closing gaps between upper and lower bounds in Table 7.1, there are yet many more directions for online dynamic bin packing, including rotation of items [40], allowing limited or full migration of items between bins [49, 50], resource augmentation [55], and packing of hypercubes [37] (items have equal lengths in all dimensions). All models are possible avenues of future work, especially for 2-D and 3-D unit and power fraction items. In terms of analysis future work may focus on introducing advice complexity analysis [10, 33], average case analysis [58], random order analysis [56], and relative worst order analysis [12] to online dynamic bin packing.

Bibliography

[1] S. Albers and J. Westbrook. Self-organizing data structures. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms*, volume 1442 of *LNCS*, pages 13–51. Springer Berlin Heidelberg, 1998.

[2] D. Applegate, L. S. Buriol, B. L. Dillard, D. S. Johnson, and P. W. Shor. The cutting-stock approach to bin packing: Theory and experiments. In R. E. Ladner, editor, *ALENEX '03*, pages 1–15. SIAM, 2003.

[3] G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1999.

[4] J. Balogh, J. Békési, and G. Galambos. New lower bounds for certain classes of bin packing algorithms. *Theoretical Computer Science*, 440–441:1–13, 2012.

[5] N. Bansal, A. Caprara, and M. Sviridenko. Improved approximation algorithms for multidimensional bin packing problems. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '06, pages 697–708, October 2006.

[6] N. Bansal, J. R. Correa, C. Kenyon, and M. Sviridenko. Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Mathematics of Operations Research*, 31(1):31–49, 2006.

[7] A. Bar-Noy and R. E. Ladner. Windows scheduling problems for broadcast systems. *SIAM J. Comput.*, 32(4):1091–1113, April 2003.

[8] A. Bar-Noy, R. E. Ladner, and T. Tamir. Windows scheduling as a restricted version of bin packing. *ACM Trans. Algorithms*, 3(3):28:1–28:22, August 2007.

[9] D. Blitz, A. van Vliet, and G. J. Woeginger. Lower bounds on the asymptotic worst-case ratio of online bin packing algorithms. Unpublished manuscript, 1996.

[10] H.-J. Böckenhauer, D. Komm, R. Královič, R. Královič, and T. Mömke. On the advice complexity of online problems. In *Proceedings of the 20th International Symposium on Algorithms and Computation*, ISAAC '09, pages 331–340. Springer-Verlag, 2009.

[11] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, New York, NY, USA, 1998.

[12] J. Boyar and L. M. Favrholdt. The relative worst order ratio for online algorithms. *ACM Trans. Algorithms*, 3(2):22:1–22:24, May 2007.

[13] A. R. Brown. *Optimum packing and depletion.* American Elsevier, New York, 1971.

[14] M. Burcea, W.-K. Hon, H.-H. Liu, P. W. H. Wong, and D. K. Y. Yau. Scheduling for electricity cost in smart grid. In P. Widmayer, Y. Xu, and B. Zhu, editors, *Combinatorial Optimization and Applications*, volume 8287 of *LNCS*, pages 306–317. Springer International Publishing, 2013.

[15] M. Burcea, P. W. H. Wong, and F. C. C. Yung. Online multi-dimensional dynamic bin packing of unit-fraction items. In P. G. Spirakis and M. Serna, editors, *Algorithms and Complexity*, volume 7878 of *LNCS*, pages 85–96. Springer Berlin Heidelberg, 2013.

[16] A. Caprara. Packing d-dimensional bins in d stages. *Math. Oper. Res.*, 33(1):203–215, February 2008.

[17] S. Caron and G. Kesidis. Incentive-based energy consumption scheduling algorithms for the smart grid. In *IEEE Smart Grid Comm.*, pages 391–396, 2010.

[18] J. W.-T. Chan, T.-W. Lam, and P. W. H. Wong. Dynamic bin packing of unit fractions items. *Theoretical Computer Science*, 409(3):521–529, 2008.

[19] J. 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*, 53(2):172–206, 2009.

[20] A. K. Chandra, D. S. Hirschberg, and C. K. Wong. Bin packing with geometric constraints in computer network design. *Operations Research*, 26(5):760–772, 1978.

[21] B. Chandra. Does randomization help in on-line bin packing? *Information Processing Letters*, 43(1):15–19, 1992.

[22] 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(3):38–402, 2000.

[23] E. G. Coffman, Jr., J. Csirik, G. Galambos, S. Martello, and D. Vigo. Bin packing approximation algorithms: Survey and classification. In P. M. Pardalos, D.-Z. Du, and R. L. Graham, editors, *Handbook of Combinatorial Optimization*, pages 455–531. Springer New York, 2013.

[24] 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*, pages 151–207. Kluwer Academic Publishers, 1998.

[25] E. G. Coffman, Jr., M. R. Garey, and D. Johnson. Bin packing with divisible item sizes. *Journal of Complexity*, 3(4):405–428, 1987.

[26] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Dynamic bin packing. *SIAM J. Comput.*, 12(2):227–258, 1983.

[27] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Bin packing approximation algorithms: A survey. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 46–93. PWS, 1996.

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

[29] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158. ACM, 1971.

[30] D. Coppersmith and P. Raghavan. Multidimensional on-line bin packing: Algorithms and worst-case analysis. *Operations Research Letters*, 8(1):17–20, 1989.

[31] J. Csirik and G. J. Woeginger. On-line packing and covering problems. In A. Fiat and G. J. Woeginger, editors, *On-line Algorithms—The State of the Art*, volume 1442 of *LNCS*, pages 147–177. Springer, 1996.

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

[33] S. Dobrev, R. Královič, and D. Pardubská. Measuring the problem-relevant information in input. *RAIRO - Theoretical Informatics and Applications*, 43(3):585–613, June 2009.

[34] G. Dósa, R. Li, X. Han, and Z. Tuza. Tight absolute bound for First Fit Decreasing bin-packing: $FFD(L) \leq 11/9\,OPT(L) + 6/9$. *Theoretical Computer Science*, 510:13–61, 2013.

[35] G. Dósa and J. Sgall. First Fit bin packing: A tight analysis. In N. Portier and T. Wilke, editors, *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20 of *LIPIcs*, pages 538–549. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2013.

[36] L. Epstein, L. M. Favrholdt, and J. S. Kohrt. Comparing online algorithms for bin packing problems. *Journal of Scheduling*, 15(1):13–21, 2012.

[37] L. Epstein and M. Levy. Dynamic multi-dimensional bin packing. *J. of Discrete Algorithms*, 8(4):356–372, December 2010.

[38] L. Epstein and R. van Stee. Optimal online algorithms for multidimensional packing problems. *SIAM J. Comput.*, 35(2):431–448, August 2005.

[39] L. Epstein and R. van Stee. Multidimensional packing problems. In T. F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, chapter 35, pages 35:1–35:15. Chapman & Hall/CRC, 2006.

[40] L. Epstein and R. van Stee. This side up! *ACM Trans. Algorithms*, 2(2):228–243, April 2006.

[41] L. Epstein and R. van Stee. Online bin packing with resource augmentation. *Discrete Optimization*, 4(3-4):322–333, 2007.

[42] X. Fang, S. Misra, G. Xue, and D. Yang. Smart grid – the new and improved power grid: A survey. *Communications Surveys Tutorials, IEEE*, 14(4):944–980, 2012.

[43] W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

[44] M. R. Garey, R. L. Graham, and J. D. Ullman. Worst-case analysis of memory allocation algorithms. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, STOC '72, pages 143–150. ACM, 1972.

[45] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco, 1979.

[46] X. Han, F. Y. L. Chin, H.-F. Ting, G. Zhang, and Y. Zhang. A new upper bound 2.5545 on 2D online bin packing. *ACM Trans. Algorithms*, 7(4):50:1–50:18, September 2011.

[47] X. Han, C. Peng, D. Ye, D. Zhang, and Y. Lan. Dynamic bin packing with unit fraction items revisited. *Inf. Process. Lett.*, 110(23):1049–1054, November 2010.

[48] R. Harren, K. Jansen, L. Prädel, and R. van Stee. A $(5/3 + \epsilon)$-approximation for strip packing. *Computational Geometry*, 47(2, Part B):248 – 267, 2014.

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

[50] Z. Ivković and E. L. Lloyd. Fully dynamic bin packing. In S. S. Ravi and S. K. Shukla, editors, *Fundamental Problems in Computing*, pages 407–434. Springer Netherlands, 2009.

[51] K. Jansen and R. Thöle. Approximation algorithms for scheduling parallel jobs: Breaking the approximation ratio of 2. In L. Aceto, I. Damgård, L. A. Goldberg,

M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *Automata, Languages and Programming*, volume 5125 of *LNCS*, pages 234–245. Springer Berlin Heidelberg, 2008.

[52] D. S. Johnson. *Near-optimal bin packing algorithms.* PhD thesis, MIT, Cambridge, MA, 1973.

[53] D. S. Johnson, A. J. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.*, 3(4):299–325, 1974.

[54] D. S. Johnson and M. R. Garey. A 71/60 theorem for bin packing. *Journal of Complexity*, 1(1):65 – 106, 1985.

[55] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.

[56] C. Kenyon. Best-fit bin-packing with random order. In *In 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 359–364, 1997.

[57] D. Knuth. *Fundamental Algorithms*, volume 1. Addison-Wesley, Reading, MA, USA, 3rd edition, 1997.

[58] L. A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, February 1986.

[59] F. Miyazawa and Y. Wakabayashi. Two- and three-dimensional parametric packing. *Computers & Operations Research*, 34(9):2589 – 2603, 2007.

[60] A.-H. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, and R. Schober. Optimal and autonomous incentive-based energy consumption scheduling algorithm for smart grid. In *Innovative Smart Grid Technologies (ISGT)*, pages 1–6, 2010.

[61] P. Ramanan, D. J. Brown, C. C. Lee, and D. T. Lee. On-line bin packing in linear time. *J. Algorithms*, 10(3):305–326, 1989.

[62] J. M. Robson. An estimate of the store size necessary for dynamic storage allocation. *J. ACM*, 18(3):416–423, July 1971.

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

[64] J. Sgall. On-line scheduling. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms*, volume 1442 of *LNCS*, pages 196–231. Springer Berlin Heidelberg, 1998.

[65] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, February 1985.

[66] A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997.

[67] R. van Stee. Combinatorial algorithms for packing and scheduling problems. Habilitation thesis, Universität Karlsruhe, June 2008. Available at `http://www.mpi-inf.mpg.de/~vanstee/habil.pdf`. Accessed August 2014.

[68] R. van Stee. SIGACT news online algorithms column 20: The power of harmony. *SIGACT News*, 43(2):127–136, June 2012.

[69] A. van Vliet. An improved lower bound for on-line bin packing algorithms. *Information Processing Letters*, 43(5):277–284, 1992.

[70] P. W. H. Wong and F. C. C. Yung. Competitive multi-dimensional dynamic bin packing via L-shape bin packing. In *Proceedings of the 7th International Conference on Approximation and Online Algorithms*, WAOA '09, pages 242–254. Springer-Verlag, 2009.

[71] P. W. H. Wong, F. C. C. Yung, and M. Burcea. An 8/3 lower bound for online dynamic bin packing. In K.-M. Chao, T.-s. Hsu, and D.-T. Lee, editors, *Algorithms and Computation*, volume 7676 of *LNCS*, pages 44–53. Springer Berlin Heidelberg, 2012.

[72] B. Xia and Z. Tan. Tighter bounds of the first fit algorithm for the bin-packing problem. *Discrete Applied Mathematics*, 158(15):1668–1675, 2010.

[73] X. Zhao and H. Shen. On the advice complexity of one-dimensional online bin packing. In J. Chen, J. E. Hopcroft, and J. Wang, editors, *Frontiers in Algorithmics*, volume 8497 of *LNCS*, pages 320–329. Springer International Publishing, 2014.