Charles University in Prague

Faculty of Mathematics and Physics

MASTER THESIS

Pavel Veselý

Online algorithms for variants of bin packing

Computer Science Institute of Charles University

Supervisor of the master thesis:	prof. RNDr. Jiří Sgall, DrSc.
Study programme:	Computer Science
Specialization:	Discrete Models and Algorithms

Prague 2014

I would like to thank my advisor Jiří Sgall for fruitful weekly meetings in which we discussed problems of Colored Bin Packing, Bin Stretching and some other, for much advice on research and on writing computer science papers and for proofreading the thesis. His PhD. student Martin Böhm was also on these meetings and helped with some ideas. He proofread the thesis, too. My brother Jan Veselý read the introduction and the first chapter.

On the conference MATCOS 2013 just after my presentation on Black and White Bin Packing, Gábor Galambos raised the question of what can be done for more than two colors after which we started to work on Colored Bin Packing, a generalization of Black and White Bin Packing to more colors. We also tried to generalize the lower bound of 1.721 to more colors with Jósef Békési, although without any success.

Last but not least, I would like to thank my family for their kind support during my whole bachelor and master studies and my girlfriend Ludmila for being my companion and for making me happy (besides that, she also read the introduction).

The work on this thesis was supported by the project 14-10003S of GA ČR and by the GAUK project 548214.

Název práce: Online algoritmy pro varianty bin packingu

Autor: Pavel Veselý

Ústav: Informatický ústav Univerzity Karlovy

Vedoucí diplomové práce: prof. RNDr. Jiří Sgall, DrSc., Informatický ústav Univerzity Karlovy

Abstrakt: Online algoritmus se musí rozhodovat okamžitě a nevratně podle části vstupu bez jakékoliv znalosti budoucí části vstupu. Představíme kompetitivní analýzu online algoritmů, což je standardní analýza nejhoršího případu, a hlavní výsledky této analýzy pro problém online bin packingu a pro některé jeho varianty. V bin packingu je úkolem naskládat posloupnost položek o maximální velikosti 1 do minimálního počtu košů jednotkové kapacity. Zaměříme se hlavně na barevný bin packing, v němž mají položky také barvu a je zakázáno mít v koši dvě položky stejné barvy vedle sebe. Vylepšíme některé předchozí výsledky pro problém omezený na dvě barvy a představíme první výsledky pro neomezený počet barev. Hlavním výsledkem je optimální 1.5-kompetitivní algoritmus pro důležitý případ, v němž mají všechny položky velikost 0. Pro položky jakékoliv velikosti dokážeme dolní odhad 2.5 a vytvoříme 3.5-kompetitivní algoritmus.

Klíčová slova: online algoritmy, bin packing, analýza nejhoršího případu, barevný bin packing, černobílý bin packing,

Title: Online algorithms for variants of bin packing

Author: Pavel Veselý

Institute: Computer Science Institute of Charles University

Supervisor: prof. RNDr. Jiří Sgall, DrSc., Computer Science Institute of Charles University

Abstract: An online algorithm must make decisions immediately and irrevocably based only on a part of the input without any knowledge of the future part of the input. We introduce the competitive analysis of online algorithms, a standard worst-case analysis, and present main results of this analysis on the problem of online Bin Packing and on some of its variants. In Bin Packing, a sequence of items of size up to 1 arrives to be packed into the minimal number of unit capacity bins. Mainly, we focus on Colored Bin Packing in which items have also a color and we cannot pack two items of the same color adjacently in a bin. For Colored Bin Packing, we improve some previous results on the problem with two colors and present the first results for arbitrarily many colors. Most notably, in the important case when all items have size zero, we give an optimal 1.5-competitive algorithm. For items of arbitrary size we present a lower bound of 2.5 and a 3.5-competitive algorithm.

Keywords: online algorithms, bin packing, worst-case analysis, colored bin packing, black and white bin packing,

Contents

Introduction

In computer science, one of major tasks is to design algorithms for different optimization problems. Typically, we are facing time complexity or memory consumption issues. Since in practice one cannot usually allow an algorithm to run in super-polynomial time, this is very important for NP-hard problems for which no polynomial time algorithm is known and moreover, there is none under the widely believed conjecture that P $\neq$ NP. One way to deal with complexity of NP-hard optimization problems is that we allow the algorithm to output some (slightly) worse solution, but it has to run in polynomial time. The field of approximation algorithms deals with such problems.

Another challenging issue which often arises in practice is a lack of information known to the algorithm before it must make some decisions. In other words, the algorithm does not have an access to the whole input or situation at the beginning; the input is rather revealed, while the algorithm is making decisions, or discovered by algorithm's actions. An example is the terrain exploration problem: Suppose that a robotic rover explores an unknown terrain on a planet, but does not have a map with enough details to compute an optimal path. Can it go through the crater which is over the horizon, or should it take a longer path around it that is surely safe?

In this thesis we consider another problem with incomplete information: the Bin Packing problem. In Bin Packing, a sequence of items of size up to one arrives to be packed into bins of unit capacity. The goal is to minimize the number of bins used. In other words, we need to partition items into the minimum number of subsets (called bins) such that the sum of sizes of items in each subset is at most one. Bin Packing is considered both when an algorithm has the whole input in advance and when items are coming one by one and each must be packed immediately and irrevocably into a bin without any knowledge of future items (not even their number or the total size). The former setting is called *offline* and the latter *online*.

Generally, *online computation* deals with situations when the input is revealed to the algorithm as the time goes on and the algorithm must deal with an incoming part of the input immediately after it comes without any knowledge of the future. On the other hand, in the *offline computation* an algorithm knows the whole future before making any decision.

An online problem can be described by a set of *parameters* known to an algorithm ALG in advance and a problem *instance* which can be viewed as a sequence of *events*. ALG starts in an *initial configuration*. When an event arrives, it must change the configuration somehow to deal with the event. There is also either a *cost function* which assigns a cost to each input and sequence of configurations that ALG visited, or a *payoff function* which represents the payoff that the algorithm obtains. The goal is to minimize the cost function, or to maximize the payoff function on a given instance. From now on we consider only minimization problems with a cost function.

In Bin Packing, there is no problem parameter, or only the capacity of bins is a parameter, but we rescale the capacity to one (together with item sizes). Items correspond to events. The configuration is the set of levels of bins where the level of a bin is the total size of all items in the bin. The cost function simply returns the number of bins used in a configuration, so it does not depend on the history.

Note that there are some other problems that are not online, but an algorithm has an incomplete information, for example the terrain exploration described above (the rover discovers the map by its own actions), or distributed computing in which no single node of a network has a global information, thus they need to communicate.

In the online computation, it is mostly impossible to behave as an optimal algorithm. We thus want to maintain as good solution as possible and moreover, we are typically not restricting or optimizing the running time of online algorithms. Nevertheless, nearly all online algorithms run in a time polynomial in the input size and in this thesis, all algorithms considered run in polynomial time.

For many problems, some randomization techniques, i.e., deciding according to some random coin flips, help to obtain better online algorithms (in expectation). Randomized online algorithms are sometimes even better than any deterministic algorithm can be. However, throughout the whole thesis we deal only with deterministic algorithms.

Competitive Analysis

Since we are interested in a quality of the solution produced by the algorithm rather than its running time, we need a measure on the algorithms that depends on the cost function. The *competitive analysis* is a standard worst-case measure in the online computation. We compare a solution produced by an algorithm to a solution by an offline optimal algorithm and we look for the maximal ratio between the cost of algorithm's solution and the optimum, i.e., the ratio on the worst instance. This idea is usually attributed to Sleator and Tarjan [57].

Another view of the competitive analysis is the following: We assume that there is a malicious *adversary* that decides on the next event according to what the algorithm did with the previous events. The adversary tries to maximize the cost function of the algorithm's solution, while keeping a relatively low cost of the offline optimal solution.

An alternative to the worst-case analysis is the *average case analysis* in which we take a probability distribution over input instances and we are counting the expected cost of an algorithm. This approach has some disadvantages, the main ones are the questions like "What distribution should we take?" and "How a distribution corresponds to the reality?"

Of course, the worst-case measure has a disadvantage, too. In reality there is often no malicious adversary and instances proving bad ratios of a particular algorithm might occur only rarely, so the average case ratio of the algorithm might be much better. An example from time complexity is the algorithm QuickSort for sorting numbers: It runs in $\mathcal{O}(n^2)$ in worst case (where n is the input size), but in $\mathcal{O}(n \log n)$ in average case and it is used commonly in practice. On the other hand, the worst-case analysis is useful when we need to guarantee that the performance of an algorithm cannot be bad. It is also interesting theoretically to find out the worst-case cost of not knowing the future.

We now formally define the competitive ratio of an algorithm ALG. For a particular instance L, we use the offline optimum denoted by $OPT(L)$ which is the cost of the solution by an offline optimal algorithm on the instance L, e.g., the number bins used. We use only OPT when the instance L is obvious from the context. Let $ALG(L)$ denote the cost of the solution by ALG. The algorithm is *absolutely r-competitive* if for any instance L

$$ALG(L) \leq r \cdot OPT(L)$$

and *asymptotically r-competitive* if for any instance L

$$ALG(L) \leq r \cdot OPT(L) + o(OPT(L)) .$$

Typically, the additive term is just a constant. We say that an algorithm has the (absolute or asymptotic) *competitive ratio r* (or that its *competitiveness* is r) if it is (absolutely or asymptotically) r-competitive and it is not r'-competitive for any $r' < r$.

Another way to define the absolute competitive ratio is

$$R_{ALG} = \sup_L \left\{ \frac{ALG(L)}{OPT(L)} \right\} .$$

Similarly, the asymptotic competitive ratio can be defined as

$$R_{ALG}^{\infty} = \limsup_{n\to\infty} \sup_L \left\{ \frac{ALG(L)}{OPT(L)} \middle| OPT(L) \geq n \right\} .$$

We note that the asymptotic ratio is often more reasonable than the absolute one, since an online algorithm does not need to deal with inputs with a small optimum and it can concentrate more on large lists for which a good behavior is more crucial. Also observe that in some sense the definition of the competitive ratio is similar to the definition of the approximation ratio.

For some problems, it is useful to restrict the behavior of an offline optimal algorithm, i.e., restrict the computation of OPT, otherwise all online algorithms would have really bad (non-constant) ratios and it would be hard to compare them. We shall see such an example in the Colored Bin Packing problem considered in detail in Chapter 2 and some other examples in Chapter 3.

More details about the online computation and the competitive analysis can be found in books *Online Algorithms: The State of the Art* by Fiat and Woeginger [29] and *Online Computation and Competitive Analysis* by Borodin and El-Yaniv [11].

Organization of the Thesis

We start with a brief survey of results for classical Bin Packing in Chapter 1, mainly for the online setting. Chapter 2 is the main part of the thesis; it introduces the new problem called Colored Bin Packing in which items have one of arbitrary many colors. (The problem restricted to two colors was studied earlier under the name Black and White Bin Packing.) We present first results on the problem with more colors in detail and we even solve the case when items have size zero. We also improve the upper bound on the competitiveness of Any Fit algorithms for the problem with two colors.

Finally, in Chapter 3 we survey some other variants of Bin Packing, such as Bounded Space Bin Packing, Bin Packing with Cardinality Constraints and Bin Stretching, for which we mention the current state-of-art results and some of the main historical results.

The results for Colored Bin Packing presented in Chapter 2 were obtained during discussions with Jiří Sgall and his PhD. student Martin Böhm. The full paper with all our results for Colored Bin Packing was accepted to the 12th Workshop on Approximation and Online Algorithms (WAOA 2014); it is also available on arXiv [10]. The abstract of the paper was accepted to the workshop Trends in Online Algorithms 2014 (TOLA 2014) without a publication in proceedings. The results for Black and White Bin Packing were presented at the conference MATCOS 2013 (also without a publication in proceedings).

1. Bin Packing

In the classical Bin Packing problem, we are given items with sizes in $(0, 1]$ and the goal is to assign them into the minimum number of unit capacity bins. See Figure 1.1 for an example of bins. Formally, Bin Packing is described as follows:

Input: A sequence of items I given either online one by one, or offline in advance. Each item i has size $S(i) \in (0, 1]$.

Output: Partitioning (packing) of I into m bins $B_1, B_2, \dots B_m$ which satisfies the capacity of bins, i.e., $\sum_{i \in B_j} S(i) \leq 1$ for all $j = 1, \dots m$.

Goal: Minimize the number of bins m.

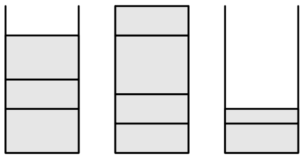

Figure 1.1: An example of three bins. The first has level (load) of 0.8, the second is full and the third has level of 0.3.

The problem was proposed by Ullman [59] and by Johnson [41] in early seventies and it was studied extensively since then in both the offline and online settings. In fact, Scheduling and Bin Packing were the first problems being investigated properly in both of these settings and moreover, Graham's work [38] on Scheduling from 1966 and Johnson's [41] on Bin Packing from 1973 introduced comparing the quality of the solution by an online algorithm against an optimal strategy.

We survey most important results for Bin Packing, mostly in the online setting, but we do not go into the details. See the survey of Coffman et al. [16] for many results on classical Bin Packing and its many variants, both in the online and offline settings. We also restrict the survey to the worst-case analysis only, see e.g. the survey of Coffman, Garey and Johnson [15] for average-case results.

Applications. Online Bin Packing arises in many real-world problems. For example, we put goods of different weights into trucks or containers with the same weight limit and we want to use as few of them as possible, while items are packed immediately as they arrive without a possibility to change their assignment later. Of course, one has to simplify the problem, mainly we are assuming that the weight is more restrictive than the size of containers.

Another applications follow: assigning newspaper articles into columns, adding network packets of different sizes into larger blocks of the same size, assigning commercials into breaks on a television station, etc.

Cutting pieces of a material is an example in which we are not adding something into a kind of containers. We have pieces of a material of a standard length, e.g., a cable or a sheet of paper, and we want to cut smaller pieces of different lengths from the material.

1.1 Offline Algorithms

Bin Packing is known to be NP-hard and moreover, there is no r-approximation algorithm for $r < 1.5$ unless P = NP. Both results can be proven by reducing the NP-hard partition problem: Given a list of n positive integers, is it possible to partition it into two lists with the same sum? The reduction first normalizes the numbers such that their total sum is two and items then correspond to numbers. If we have a polynomial-time algorithm for Bin Packing which outputs strictly less than $1.5 \cdot OPT$ bins, we can use the reduction to decide whether the instance can be packed into two bins or not. Therefore we can solve the original partition instance in polynomial time.

Another consequence is that there is no polynomial-time approximation scheme (PTAS) unless P = NP. On the other hand, in 1981 Fernandez de la Vega and Lueker [28] designed an asymptotic polynomial-time approximation scheme (APTAS), i.e., for any $0 < \varepsilon \leq 1$ there is a polynomial-time algorithm A_ε which outputs a packing into at most $(1 + \varepsilon)OPT + 1$ bins. The following year, Karmarkar and Karp [45] gave a seminal polynomial algorithm that uses no more than $OPT + \mathcal{O}(\log^2 OPT)$ bins, i.e., an asymptotic fully polynomial-time approximation scheme (AFPTAS). In 2013, Rothvoß [54] improved the result to $OPT + \mathcal{O}(\log OPT \cdot \log\log OPT)$.

1.2 Online Algorithms

We now briefly present key algorithms and techniques used in the online Bin Packing problem. We start with a bit of notation.

Preliminaries. The *level of a bin* means the cumulative size of all items in the bin. By OPT we denote the number of bins used by an optimal offline algorithm on a particular instance.

We analyze algorithms not only in the case when items have sizes in $(0, 1]$, but also in the so called *parametric case* in which items have sizes of at most $1/d$ for a real $d \geq 2$.

To analyze online algorithms we need to know something about the optimum. It is usually hard to characterize fully how an optimal packing of an instance looks, therefore we only use some lower bounds on the number of bins in any packing. Throughout this chapter, we use a single lower bound on the optimum number of bins which is the sum of sizes of all items, denoted by LB_1. There are of course some other bounds such as $L + \lceil H/2 \rceil$ where L is the number of items of size more than 0.5 and H is the number of items of size exactly 0.5.

1.2.1 Classical Algorithms

There are several well-known and often used online algorithms for classical Bin Packing. Algorithms from the *Any Fit* family (AF) pack an incoming item into some already open bin whenever it is possible with respect to the size constraints. The choice of the open bin where to put the item (if more are available) depends on the algorithm. AF algorithms thus open a new bin with an incoming item only when there is no other possibility. Among AF algorithms, these are the most important:

- *First Fit* (FF) packs an incoming item into the *first* bin where it fits (in the order by creation time),
- *Best Fit* (BF) chooses the bin with the *highest* level where the item fits,
- and *Worst Fit* (WF) packs the item into the bin with the *lowest* level where it fits.

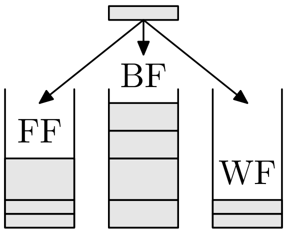

Figure 1.2: An illustration of how classical Any Fit algorithms treats with an incoming item which fits in any bin.

Next Fit (NF) is more restrictive than Any Fit algorithms, since it keeps only a single open bin and puts an incoming item into it whenever the item fits, otherwise the bin is closed and a new one is opened. Its advantages are that it may be useful to have only one open bin in the real world and that the running time on the input with n items is $\mathcal{O}(n)$, while FF, BF and WF run in $\mathcal{O}(n \log n)$.

The absolute competitive ratio of NF is 2 [42]. This can be proven by observing that any two consecutive bins (in the order by their creation time) have total volume strictly greater than 1. Then we match each bin with an odd index with the following bin with an even index. The last bin is not matched, if the number of bins is odd. Since the number of matched bins is strictly less than $2 \cdot OPT$, i.e., at most $2 \cdot OPT - 1$, and there is at most one bin not matched, we get that NF uses at most $2 \cdot OPT$ bins. Moreover, the list $(0.5, \varepsilon, 0.5, \varepsilon, \ldots 0.5, \varepsilon)$ of length $2n$ (for $0 < \varepsilon \leq 1/n$) shows that the bound is tight, since NF packs each item of size 0.5 into a new bin, opening n bins in total, but the optimum is $\lceil 0.5n \rceil + 1$.

Similarly we prove that Any Fit algorithms are at most absolutely 2-competitive: The sum of levels of any pair of bins is over 1 and if we sum total levels of each pair of consecutive bins (including the pair of the first and the last bin), we get that twice the total size of items is more than the number of bins. Hence for an input instance L we have $AF(L) < 2 \cdot LB_1 \leq 2 \cdot OPT$. One can use the same instance as for NF to show that for WF the bound of 2 is tight.

FF and BF work better and quite surprisingly, their competitive ratios are the same. It is known for a long time that they are asymptotically 1.7-competitive and that they use at most $1.7 \cdot OPT + 1$ bins [43]. Recently, Dósa and Sgall showed that even the absolute competitive ratio of both FF and BF equals 1.7 [23, 26].

Johnson [42] also proved that there is no Any Fit algorithm with a better asymptotic ratio than 1.7. He also dealt with *Almost Any Fit* algorithms (AAF), a subfamily of AF algorithms, which avoids the bin with the lowest level, unless there are more bins with the lowest level or there is no other bin before this bin where the incoming item fits. BF and FF belong to this class, while WF does not. Surprisingly, all AAF algorithms are asymptotically 1.7-competitive [42] — even *Almost Worst Fit* (AWF) which puts an incoming item into the second lowest bin if the item fits there, otherwise behaves like WF.

In the parametric case, i.e., when items have sizes in $(0, 1/d]$ for a real $d \geq 2$, the worst-case ratio of algorithms is getting better. The performance ratio of FF, BF and AWF is $(m+1)/m$ where $m \in \mathbb{N}$ such that $1/(m+1) < 1/d \leq 1/m$, while WF and NF reaches only $d/(d-1)$ [42, 43]. See Table 1.1 with competitive ratios for small values of d.

Some of these classical algorithms were also analyzed in their *decreasing variants* in which the algorithm assumes that the items are sorted by their size non-increasingly. This is a kind of *semi-online setting*, because an algorithm has some little knowledge of the future. In the offline setting, a decreasing variant of an online algorithm first sorts the input and then performs the online algorithm. *Next Fit Decreasing* (NFD) has the asymptotic ratio of $h_\infty \approx 1.691$ [4], but *First Fit Decreasing* (FFD) uses at most $11/9 \cdot OPT + 2/3$ bins and this is tight [24]. *Best Fit Decreasing* (BFD) shares the asymptotic ratio of 11/9 with FFD [43]. Due to Johnson and Garey [44], FFD was improved to the offline algorithm Modified First Fit Decreasing (MFFD) which packs items into at most $71/60 \cdot OPT + 1$ bins [63]. On the other hand, the lower bound on competitiveness of online algorithms on inputs sorted decreasingly is $54/47 \approx 1.149$ [7], while $71/60 \approx 1.183$.

1.2.2 Harmonic-type Algorithms

Currently best algorithms (in the worst-case behavior) are based on the *Harmonic(K)* algorithm H_K by Lee and Lee [50] which assigns items into K classes according to their size; the classes are $(0, \frac{1}{K}], (\frac{1}{K}, \frac{1}{K-1}], \dots (\frac{1}{3}, \frac{1}{2}], (\frac{1}{2}, 1]$. Each class of items is packed separately by NF.

The asymptotic competitive ratio of Harmonic tends to $h_\infty = 1.691\dots$ as K goes to infinity. The number h_∞ can be computed using integer sequence $2, 3, 7, 43, 1807\dots$ investigated first by Sylvester [58]. The sequence is defined as follows: Let $k_1 = 2$ and $k_{i+1} = k_i(k_i - 1) + 1$. Then

$$h_\infty = \sum_{i=1}^{\infty} \frac{1}{k_i - 1} = 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{42} + \frac{1}{1806} + \dots \approx 1.691$$

For $K = 7$, the algorithm has competitive ratio of 1.6944 and it is thus better than FF.

Using the Harmonic algorithm, an incoming item can only go into one of at most K active (open) bins, because the algorithm applies NF for each class. The other bins are closed and cannot receive further items. Algorithms with such a limited number of active bins are called *K-bounded-space*. We shall survey some results about bounded-space algorithms in Section 3.1.

The ratio of Harmonic was later improved a few times by modifying size classes. First, *Simplified Harmonic* (SH_K) by Woeginger achieved a better ratio for small values of K, but its ratio also tends to 1.691. It was designed from Harmonic by using only one bin for classes $(\frac{1}{6}, \frac{1}{5}]$ and $(\frac{1}{5}, \frac{1}{4}]$,

The first algorithm that crossed the ratio of h_∞ was *Revised First Fit* (RFF) by Yao [62] which is asymptotically 1.6-competitive. It uses classes $(0, \frac{1}{3}]$, $(\frac{1}{3}, \frac{2}{5}]$, $(\frac{2}{5}, \frac{1}{2}]$, $(\frac{1}{2}, 1]$ and utilizes FF on each instead of NF. Additionally, RFF treats every sixth item from the class $(\frac{1}{3}, \frac{2}{5}]$ in a special way.

Lee and Lee [50] also designed Refined Harmonic (RH) by breaking classes $(\frac{1}{3}, \frac{1}{2}]$ and $(\frac{1}{2}, 1]$ into two subclasses. This approach achieved the worst-case ratio of $373/228 \approx 1.639$. Ramanan, Brown, Lee and Lee [53] improved RH and their Modified Harmonic (MH) has the asymptotic ratio of approximately 1.616. They also showed that Harmonic-type algorithms cannot achieve a better performance than 1.583. The currently best online algorithm is Harmonic++ by Seiden [55] which uses more than 70 items classes to achieve the ratio of at most 1.58889.

1.2.3 Lower Bounds on Competitiveness

In this section we describe the main idea which leads to proofs that there is no r-competitive algorithm for some $r > 1$. First observe that if an algorithm is given big items at the beginning of the sequence it may pack them nearly optimally and then it is easy to add remaining small items. In fact, one of the best approximation algorithms, First Fit Decreasing, first sorts items in a non-increasing order and then packs them in this order by First Fit. Also, the asymptotic PTAS by Fernandez de la Vega and Lueker [28] finds an optimal packing for big items (bigger than $\varepsilon/2$) and adds small items by First Fit.

On the other hand, if small items come at the beginning, an online algorithm does not know whether some big items come and how big they are, so it must balance between a good solution for small items and enough place to add bigger items into some already open bins.

This idea leads to the lower bound of 1.5 by Yao [62]. His construction consists of three lists of items: the list L_1 with n items of size $1/7 + \varepsilon$, L_2 with n items of size $1/3 + \varepsilon$ and finally L_3 with n items of size $1/2 + \varepsilon$. Yao proved that every online algorithm must use at least 1.5 times the optimum either on the instance only with the list L_1, or with lists L_1 and L_2, or with all three lists.

Brown [12] and Liang [51] independently extended the construction and obtained the lower bound of 1.53635. They both added items of size $1/1807 + \varepsilon$ and $1/43 + \varepsilon$ at the beginning of the sequence. Observe that the denominators of the sequence $1/2, 1/3, 1/7, 1/43, 1/1807, \ldots$ form the Sylvester's sequence, also used to define the asymptotic competitive ratio of Harmonic.

Galambos [33] and later Galambos and Frenk [35] gave a simplified analysis of the result and used it to prove lower bounds in the parametric case; see Table 1.1 for the values. Van Vliet [60] then gave the bound of ≈ 1.54014 by a tighter analysis of the simplified construction. Chandra [13] showed that these bounds hold even for randomized online algorithms, i.e., for algorithms that are allowed to decide according to some random coin flips.

Finally in 2012, Balogh, Békési and Galambos [7] showed that there is no online algorithm which is better than $248/161 \approx 1.54037$-competitive. Note that the gap between the current lower and upper bounds is even smaller than 0.05.

Competitive ratios of certain algorithms, both in the general and parametric cases, together with currently best lower bounds, are given in Table 1.1.

For further references and other results see the survey of Coffman et al. [16] from 2013. There are also older surveys such as the one by Csirik and Woeginger in the book Online Algorithms: The State of the Art [29] from 1998, the survey by Conffman, Garey and Johnson [15] from 1997 and by Galambos and Woeginger [36] from 1995.

Algorithm	General case	Parametric case		
		$d = 2$	$d = 3$	$d = 4$
NF, WF	2	2	1.5	1.33333...
AAF (e.g., FF, BF, AWF)	1.7	1.5	1.33333...	1.25
Harmonic(∞)	1.69103...	1.42307...	1.30238...	1.23441...
Harmonic++	1.58889...	—	—	—
Lower bound	1.54037...	1.38966...	1.29144...	1.22986...

Table 1.1: Table with asymptotic competitive ratios of algorithms for Bin Packing and current lower bounds. The parametric case is when the items have size of at most $1/d$.

1.3 Relation to Scheduling

In the basic Scheduling problem we need to assign n jobs with processing times $p_1, p_2, \dots p_n$ into a fixed number of identical machines such that the *makespan*, i.e, the completion time of the last job, is minimized. In the definition, identical means that all machines are equal. There are many variants of Scheduling studied: machines can have different speeds, there are some dependencies between the jobs, a job can run in parallel on a couple of machines, or a processing of a job can be interrupted and continued on another machine (which is called preemption).

Bin Packing can be viewed as Scheduling in which items correspond to jobs, bins to machines and each machine can process jobs of at most unit total duration, but we can always add a new machine. We are thus optimizing the number of machines rather than their maximal load. Colored Bin Packing as a Scheduling problem adds the restriction that we cannot have two jobs from the same color class processed one after the other.

In Section 3.5 we survey the online Bin Stretching problem which is somehow between Scheduling and Bin Packing. We are given m, the number of bins, and a list of items with sizes in $(0, 1]$ that can be packed offline into m unit capacity bins. Items are arriving in an online fashion and an online algorithm can use m bins of capacity $R \geq 1$, i.e., the bins are stretched. The goal is to minimize the stretching factor R.

2. Colored Bin Packing

In this chapter we focus in detail on a generalization of Bin Packing in which each item has a color. We sketch previous results and then we present our results for the online problem together with proofs.

In the Black and White Bin Packing problem proposed by Balogh et al. [6, 5] as a generalization of classical Bin Packing, we are given a list of items of size in $[0, 1]$, each item being either black, or white. The items are coming one by one and need to be packed into bins of unit capacity. The items in a bin are ordered by their arrival time. The additional constraint to capacity is that the colors inside the bins are alternating, i.e., no two items of the same color can be next to each other in the same bin. The goal is to minimize the number of bins used.

Colored Bin Packing is a natural generalization of Black and White Bin Packing in which items can have more than two colors. As before, the only additional condition to unit capacity of bins is that we cannot pack two items of the same color next to each other in one bin. See Figure 2.1 for an example of bins.

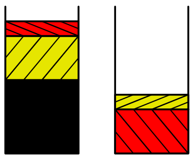

Figure 2.1: An example of two bins

Formally, Colored Bin Packing is defined as follows:

Input: A sequence of items I given either online one by one, or offline in advance. Each item i has size $S(i) \in [0, 1]$ and a color $c \in C$.

Output: Partitioning (packing) of I into m bins $B_1, B_2, \dots B_m$ which satisfies the capacity and color constraints, i.e., for all $j = 1, \dots m$ it holds that $\sum_{i \in B_j} S(i) \leq 1$ and no two consecutive items in B_j in the order by their arrival time have the same color.

Goal: Minimize the number of bins m.

Note that items of size zero make sense in this model, although they are useless in classical Bin Packing. For example, we are able to pack two green items of size 0.5 into the same bin by putting a red zero-size item between them.

Interestingly, the problem is not monotonous in the following sense: If we delete some items from the sequence, the optimum may increase, even by a non-constant factor. Consider for example an alternating sequence of $2n$ black and white zero-size items. They can be packed in a single bin, but if we delete all white items, the optimum becomes n. On the other hand, observe that deleting a single item increases the optimum by at most one: Consider a bin in an optimal packing and delete an item from it. At worst, we need to split the bin into two bins because of the color restriction. Therefore, if there was a packing of the instance into n bins, after deleting an item there is a packing into $n + 1$ bins.

There are three settings of Colored Bin Packing:

- In the *offline setting* we are given the items in advance and we can pack them in an arbitrary order.
- In the *restricted offline setting* we also know sizes and colors of all items in advance, but they are given as a sequence and they need to be packed in that order.
- In the *online setting* the items are coming one by one and we do not know what comes next or even the total number of items. An online algorithm has to pack each incoming item immediately and it is not allowed to change its decisions later.

Observe that optimal offline and restricted offline packings, i.e., packings with and without reordering the items, may differ in this model, unlike in standard Bin Packing. The packings may even differ by a non-constant factor: Let the input sequence have n yellow items and then n red items, all of size zero. The offline optimal number of bins is 1, but a restricted offline packing (or an online packing) needs n bins, since the first n yellow items must be packed into different bins; see Figure 2.2 for an illustration of the offline optima on this instance. Hence it is more relevant to use the restricted offline optimum in the worst-case analysis of online algorithms, otherwise no online algorithm would be constant competitive.

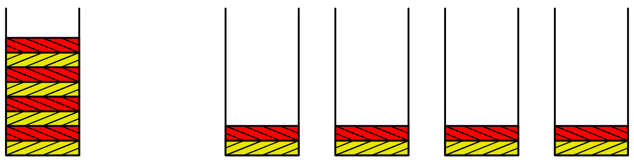

Figure 2.2: The unrestricted offline optimum of the sequence with n yellow items and n red items on the left and the restricted offline optimum on the right

We focus mostly on the online setting, both in general and parametric cases. (Recall that in the parametric case all items have size of at most $1/d$ for a real $d \geq 2$.) Apart from these two cases, we investigate also the case when all items have size zero which we call the case with zero-size items. Sometimes we refer to the general case as the case with items of arbitrary size.

We start with a summary of the results on Black and White Bin Packing which were known previously to our work and then we outline our results on the problem. In Section 2.5 we show two lower bounds on the optimal number of bins which we use in analyses of algorithms. In Section 2.6 we present briefly some results on the offline problem, namely a characterization of the offline optima for zero-size items and NP-completeness of the general case. The rest of this chapter consists of results on the online problem together with proofs, first for Black and White Bin Packing, then for Colored Bin Packing with zero-size items and finally for Colored Bin Packing with items of unrestricted size.

2.1 Previous Results

Balogh, Békési, Dósa, Kellerer and Tuza [6] introduced the Black and White Bin Packing problem in 2012 and later improved their results together with Epstein [5]. They present three lower bounds on the restricted offline optimum; we use two of them for Colored Bin Packing also. As the main result, they give an algorithm *Pseudo* with the absolute competitive ratio exactly 3 in the general case and $1 + d/(d-1)$ in the parametric case in which items have size of at most $1/d$ for a real d. They also proved that there is no deterministic or randomized online algorithm which asymptotic competitiveness is below $1+1/(2\ln 2) \approx 1.721$.

The idea of the algorithm Pseudo, on which we build as well, is that it first packs the items regardless of their size, i.e., treating their size as zero. This can be done optimally, and the optimum equals the maximal discrepancy in the sequence of colors (to be defined below). Then these bins are partitioned by Next Fit into bins of level at most 1.

Concerning classical algorithms used in Bin Packing, they proved that Any Fit algorithms are absolutely at most 5-competitive and even absolutely 1-competitive for zero-size items, i.e., they use OPT bins. They show input instances on which First Fit and Best Fit create asymptotically $3 \cdot OPT$ bins. For Worst Fit, there are sequences of items witnessing that it is at least 3-competitive and $(1 + d/(d-1))$-competitive in the parametric case for an integer $d \geq 2$. Furthermore, Next Fit and Harmonic are not constant competitive.

Balogh et al. [6] also gave a 2.5-approximation offline algorithm with time complexity of $\mathcal{O}(n \log n)$ and an asymptotic polynomial time approximation scheme, both when reordering is allowed.

Very recently and independent of us Dósa and Epstein [25] studied Colored Bin Packing (under the name Colorful Bin Packing). They improved the lower bound for online Black and White Bin Packing to 2 for deterministic algorithms, which holds for more colors as well. For at least 3 colors they proved an asymptotic lower bound of 1.5 for zero-size items. They designed an absolutely 4-competitive algorithm based on Pseudo and an absolutely 2-competitive balancing algorithm for zero-size items. They also showed that BF, FF and WF are not competitive at all (with non-zero sizes).

2.2 Our Results

We completely solve the case of Colored Bin Packing for zero-size items (in the worst-case behavior). As we have seen, this case is important for constructing general algorithms. The restricted offline optimum is actually not only lower bounded by the color discrepancy, but equal to it for zero-size items. For online algorithms, we give an (asymptotically) 1.5-competitive algorithm Balancing Any Fit (BAF) which is optimal. The algorithm mostly puts an incoming item into a bin of the most frequent other color.

In fact, BAF always uses at most $\lceil 1.5 \cdot OPT \rceil$ bins and we show a matching lower bound of $\lceil 1.5 \cdot OPT \rceil$ for any value of $OPT \geq 2$ using three colors. This is significantly stronger than the asymptotic lower bound of 1.5 of Dósa and Epstein [25], in particular it shows that the absolute ratio of our algorithm is 5/3, and this is optimal. Note introducing the third color makes the problem with zero-size items harder, as for two colors all Any Fit algorithms have absolute competitive ratio of 1 [6].

For items of unrestricted size and three colors, we show a lower bound of 2.5, which breaks the natural barrier of 2. We use the optimal algorithm for zero-size items and the algorithm Pseudo to design an absolutely 3.5-competitive algorithm Pseudo-BAF which is also asymptotically $(1.5 + d/(d-1))$-competitive in the parametric case, where the items have size of at most $1/d$ for a real $d \geq 2$. (Note that for $d < 2$ we have $d/(d-1) > 2$ and the bound for items of arbitrary size is better.)

We show that algorithms BF, FF and WF are not constant competitive, even for instances with only three colors and very small non-zero items, in contrast to their 3-competitiveness for two colors.

For Black and White Bin Packing, we present a lower bound of 1.5 on competitiveness of any online algorithm using arbitrarily small, yet non-zero items. More importantly, we improve the upper bound on the absolute competitive ratio of Any Fit algorithms in the general case to 3 which is tight for BF, FF and WF. For WF in the parametric case, we prove that it is absolutely $(1+d/(d-1))$-competitive for a real $d \geq 2$ which is tight for an integral d. Therefore, WF has the same competitive ratio as the algorithm Pseudo and these are surprisingly the best known algorithms for two colors in the worst-case behavior.

Algorithm	≥ 3 colors	≥ 3 colors param. case	2 colors	2 colors param. case
Pseudo-BAF	**3.5**	$\mathbf{1.5 + \frac{d}{d-1}}$	3	$1 + \frac{d}{d-1}$
Any Fit	[**2.5**, **unbounded**)	[2, **unbounded**)	$[2, \mathbf{3}]$	$[2, \mathbf{3}]$
Worst Fit	**unbounded**	**unbounded**	**3**	$\mathbf{1 + \frac{d}{d-1}}$
First Fit	**unbounded**	**unbounded**	**3**	**3**
Best Fit	**unbounded**	**unbounded**	**3**	**3**
Lower bound	**2.5**	**1.5**	2	**1.5**

Table 2.1: Table with asymptotic competitive ratios of algorithms for Colored Bin Packing and the current lower bounds. If we do not know a competitive ratio exactly, an interval with lower and upper bounds is shown. Our improvements are typeset in boldface. Note that Any Fit algorithms for three or more colors may have unbounded competitive ratio and it is open whether there is a constant competitive Any Fit algorithm for three or more colors and items of arbitrary size.

2.3 Applications

Suppose that a television or a radio station maintains several channels and wants to assign a set of programs to them. The programs have types like "documentary", "thriller", "sport" on TV, or music genres on radio. To have a fancy schedule of programs, the station does not want to broadcast two programs of the same type one after the other. Colored Bin Packing can be used to create such a schedule. Items here correspond to programs, colors to genres and bins to channels. Moreover, the programs can appear online and have to be scheduled immediately, e.g., when listeners send requests for music to a radio station via the Internet.

Another application of Colored Bin Packing comes from software which renders user-generated content (for example from the Internet) and assigns it to columns which are to be displayed. The content is in boxes of different colors and we do not want two boxes of the same color to be adjacent in a column, otherwise they would not be distinguishable for the user.

Moreover, Colored Bin Packing with all items of size zero corresponds to a situation in which we are not interested in loads of bins (lengths of the schedule, sizes of columns, etc.), but we just want some kind of diversity or colorfulness.

2.4 Notation

For Colored Bin Packing, let C be the set of all colors. For $c \in C$, the items of color c are called c-items and bins with the top (last) item of color c are called c-bins; see Figure 2.3 for an illustration. By a non-c-item we mean an item of color $c' \neq c$ and similarly a non-c-bin is a bin of color $c' \neq c$. A *non-zero item* is an item of size strictly greater than zero. The *level of a bin* means the cumulative size of all items in the bin (sometimes, it is called the *load of a bin*). A *non-zero bin* is a bin with a non-zero level, i.e., its level is greater than zero.

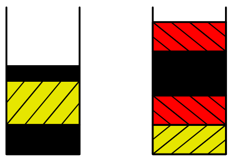

Figure 2.3: A black and a red bin

We denote a sequence of nk items consisting of n groups, each with k items of colors $c_1, c_2, \dots c_k$ and sizes $s_1, s_2, \dots s_k$, by $n \times \left(\substack{c_1 \\ s_1}, \substack{c_2 \\ s_2}, \dots \substack{c_k \\ s_k}\right)$.

2.5 Lower Bounds on the Restricted Offline Optimum

As for classical Bin Packing, it is usually hard to characterize fully how an optimal packing of an instance looks, but we need to know something about the optimum to analyze online algorithms. Therefore we only use some lower bounds on the number of bins in any packing. However, in the next section we also prove a characterization of both the restricted and unrestricted offline optima for the case with zero-size items.

For Colored Bin Packing, we present two lower bounds. The first bound LB_1 is the sum of sizes of all items — as bins have unit capacity, we cannot pack items of total size S into less than $\lceil S \rceil$ bins.

In Colored Bin Packing, LB_1 does not suffice in many cases. For an illustration, consider the sequence of ten white items, then two red items and finally ten white items, all of size zero. The first ten white items are packed into ten bins, the next two red items fit into two open bins and the last ten white items must open additional ten bins, resulting in 18 open bins. As $LB_1 = 0$, we need to consider also the colors of items.

Thus for the second bound LB_2 we use the maximal color discrepancy inside the input sequence. In Black and White Bin Packing, the color discrepancy introduced by Balogh et al. [5] is simply the difference of the number of black and white items in a segment of the input sequence, maximized over all segments. It is easy to see that it is a lower bound on the number of bins.

In the generalization of the color discrepancy for more than two colors we count the difference between c-items and non-c-items for all colors c and segments. It is easy to see that this is a lower bound as well. Formally, let $s_{c,i} = 1$ if the i-th item from the input sequence has color c, and $s_{c,i} = -1$ otherwise. We define

$$LB_2 = \max_{c \in C} \max_{i,j} \sum_{\ell=i}^{j} s_{c,\ell} \,.$$

For Black and White Bin Packing, equivalently

$$LB_2 = \max_{i,j} \left| \sum_{\ell=i}^{j} s_\ell \right|$$

where $s_i = 1$ if the i-th item is white, and $s_i = -1$ otherwise. Note that the absolute value replaces the maximization over colors.

We prove that LB_2 is a lower bound on the optimum similarly to the proof of Lemma 5 in [5]. First we observe that the number of bins in the optimum cannot decrease by removing a prefix or a suffix from the sequence of items.

Observation 1. *Let $L = L_1L_2L_3$ be a sequence of items partitioned into three subsequences (some of them can be empty). Then $OPT(L) \geq OPT(L_2)$.*

Proof. It is enough to show that the removal of the first or the last item does not increase the optimum. By iteratively removing items from the beginning and the end of the sequence we obtain the subsequence L_2 and consequently $OPT(L) \geq OPT(L_2)$.

The first item of the sequence is clearly the first item in a bin. By removing the first item from the bin we do not violate any condition. Hence any packing of L is a valid packing of L without the first item. A similar claim holds for the last item. $\square$

Lemma 2. *$OPT(L) \geq LB_2$.*

Proof. We prove that for all colors c that the optimum is at least $LB_{2,c} := \max_{i,j} \sum_{\ell=i}^{j} s_{c,\ell}$. Fix a color c and let i,j be $\arg\max_{i,j} \sum_{\ell=i}^{j} s_{c,\ell}$. (Recall that $s_{c,i}$ is 1 when the i-th item from the input sequence has color c, otherwise -1.) Let $d = LB_{2,c}$. We may assume that $d > 0$, otherwise d is trivially at most the optimum. By the previous observation we may assume $i = 1$ and $j = n$.

Consider any packing of the sequence and let k be the number of bins used. Any bin contains at most one more c-item than non-c-items, since colors must alternate between c and other colors. (In worst-case, the first and the last items and all odd items have color c and the other items must be non-c-items.) Since we have d more c-items than non-c-items, we get $k \geq d$. Therefore $OPT \geq LB_{2,c}$ $\square$

Note that if we consider a sequence with the maximal discrepancy LB_2 of a minimal length, i.e., removing a prefix or a suffix of the sequence decreases LB_2, the optimal restricted offline algorithm puts all non-c-items into c-bins whenever it is possible.

Balogh et al. [5] propose a third lower bound for Black and White Bin Packing based on the conflict graph for each color. We can generalize conflict graphs to more than two colors straightforwardly, but we do not use them to obtain new results, so we omit the generalization. We also note that on some hard instances for certain algorithms such as FF, BF, WF and Pseudo the third lower bound is actually equal to LB_1 or LB_2.

2.6 Offline Results

2.6.1 Offline Optima for Zero-size Items

In Black and White Bin Packing, when all the items are of size zero, all Any Fit algorithms create a packing into LB_2 bins [5] which is optimal. For more than two colors this is not true and in fact no deterministic online algorithm can have a competitive ratio below 1.5. However, in the restricted offline setting a packing into LB_2 bins is still always possible, even though this fact is not obvious. This shows that the color discrepancy fully characterizes the combinatorial aspect of the color restriction in Colored Bin Packing.

Theorem 3. *Let all items have size equal to zero. Then a packing into LB_2 bins is possible in the restricted offline setting, i.e., items can be packed into LB_2 bins without reordering.*

Proof. The proof is by contradiction. Consider a counterexample with a minimal number of items in the sequence. Let $d = LB_2$ be the maximal discrepancy in the counterexample and $n \geq d$ be the number of items. The minimality implies that the theorem holds for all sequences of length $n' < n$. Moreover, $d > 1$, since for $d = 1$ we can pack the sequence trivially into a single bin.

We define an *important interval* as a maximal interval of discrepancy d, more formally a subsequence from the i-th item to the j-th such that for some color c the discrepancy on the interval is d, i.e., $\sum_{\ell=i}^{j} s_{c,\ell} = d$, and we cannot extend the interval in either direction without decreasing its discrepancy. For an important interval, its *dominant color* c is the most frequent color inside it. At first we show that important intervals are just d items of the same color.

Observation 4. *Each important interval I contains only d items of its dominant color c in the minimal counterexample.*

Proof. Suppose there is a non-c-item in I and let a be the last such item in I. Then a must be followed by a c-item b in I, otherwise I without a would have higher discrepancy. We delete a and b from the sequence and pack the rest into d bins by minimality.

Consider the situation after packing the item prior to a. There must be a c-bin B, otherwise the subsequence of I from the beginning up to a (including a) has strictly more non-c-items than c-items (each c-item from I is under a non-c-item and a is the extra non-c-item). Hence the rest of I after a must have discrepancy more than d. By putting a and b into B we pack the whole sequence into d bins, thus it is not a counterexample. □

When two different important intervals I_1 and I_2 with dominant colors c_1 and c_2 intersect on an interval J, then $c_1 \neq c_2$, otherwise their union would have higher discrepancy than d. Let $|I_1| \geq |I_2|$, so I_1 must contain a c_2-item and we can pack the sequence into d bins by the previous observation. Therefore important intervals are disjoint in the minimal counterexample. Note that I_2 can be contained in I_1 completely, for example in a sequence of d black items, d white items and d black items.

Clearly, there must be an important interval in any non-empty sequence. Let $I_1, I_2, \ldots I_k$ be important intervals in the counterexample sequence and let $J_1, J_2, \ldots J_{k-1}$ be the intervals between the important intervals (J_i between I_i and I_{i+1}), J_0 be the interval before I_1 and J_k be the interval after I_k. These intervals are disjoint and form a complete partition of the sequence, i.e., $J_0, I_1, J_1, I_2, J_2, \ldots J_{k-1}, I_k, J_k$ is the whole sequence of items. Note that some of J_ℓ's can be empty.

If $k > 2$, we can create a packing P_1 of the sequence containing only intervals J_0, I_1, J_1, I_2 into d bins by minimality of the counterexample (I_3 is non-empty). Also there exists a packing P_2 of intervals $I_2, J_2, I_3, \ldots I_k, J_k$ into d bins. Any bin from P_1 must end with an item from the important interval I_2 and any bin from P_2 must start with an item from I_2, since important intervals are d items of the same color. Therefore we can merge both packings by items from I_2 and obtain a valid packing of the whole sequence into d bins. Hence $k \leq 2$.

In the case $k = 1$, there are four subcases depending on whether J_0 and J_1 are empty or not:

- J_0 and J_1 are non-empty: We create packings of J_0, I_1 and I_1, J_1 into d bins and merge them as before.
- J_0 is empty and J_1 non-empty: We delete the first item from I_1, pack the rest into $d-1$ bins (the maximal discrepancy decreases after deleting) and put the deleted item into a separate bin.
- J_0 is non-empty and J_1 empty: Similarly, we delete the last item from I_1 and pack the rest into $d-1$ bins.
- both are empty: I_1 can be trivially packed into d bins.

For $k = 2$, we first show that J_0 and J_2 are empty and J_1 is non-empty in the counterexample. If J_0 is non-empty, we merge packings of J_0, I_1 and I_1, J_1, I_2, J_2, and if J_2 is non-empty, we put together packings of J_0, I_1, J_1, I_2 and I_2, J_2. When J_1 is empty, the sequence consists only of intervals I_1 and I_2 which must have different dominant colors. Thus they can be easily packed one on the other into d bins.

The last case to be settled has only I_1, J_1 and I_2 non-empty. If the dominant colors c_1 for I_1 and c_2 for I_2 are different, we delete the first item from I_1 and the last item from I_2, so the discrepancy decreases. We pack the rest into $d-1$ bins and put the deleted items into a separate bin, so the whole sequence is in d bins again.

Otherwise c_1 is equal to c_2 and let c be c_1. Since the important intervals are maximal, there must be at least $d+1$ more non-c-items than c-items in J_1. Also any prefix of J_1 contains strictly more non-c-items than c-items, thus at least the first two items in J_1 have colors different from c.

We delete the first c-item p from I_1, the first non-c-item q from J_1 and the last c-item r from I_2. Suppose for a contradiction that there is an interval I of discrepancy d in the rest of the sequence. As I has lower discrepancy in the original sequence (we deleted an item from each important interval of the original sequence), it must contain q and thus intersect I_1 and J_1, hence its dominant color is c. If I intersects also I_2, we add the items p, q and r into I (and possibly some other items from I_1 or I_2) to obtain an interval of discrepancy at least $d+1$ in the original sequence which is a contradiction. Otherwise I intersects only I_1 and J_1, but any prefix of the rest of J_1 still contains at least as many non-c-items as c-items, so $I \setminus J_1$ has discrepancy at least d. But $I \setminus J_1$ is contained in the rest of I_1 that has only $d-1$ items and we get a contradiction. Therefore the maximal discrepancy decreases after deleting the three items, so we can pack the rest into $d-1$ bins and the items p, q and r are put into a separate bin. Note that important intervals of discrepancy $d-1$ may change after deleting the three items.

In all cases we can pack the sequence into d bins, therefore no such counterexample exists. □

The previous theorem immediately yields a polynomial algorithm for computing the restricted offline optimum of an instance with zero-size items: For each interval from the i-th to the j-th item, find the most frequent color c in it (in linear time) and compute $\sum_{\ell=i}^{j} s_{c,\ell}$. The running time of this algorithm is $\mathcal{O}(n^3)$ where n is the number of items. There is also an algorithm with the time complexity of $\mathcal{O}(|C|n)$ where C is the set of colors in the input sequence.

The previous theorem has also some combinatorial applications such as the following: Let $(a_i)_{i=1}^{n}$ be a sequence of numbers (colors, strings, etc.). Partition the sequence into the minimum number of subsequences such that each two consecutive numbers in a chosen subsequence are different.

We now show an analysis of the unrestricted offline optimum of an instance with zero-size items which is a bit easier than in the restricted case. Also, the restricted offline optimum is more useful, since we use it for the competitive analysis of online algorithms.

Theorem 5. *Let all items have size equal to zero. Let c be one of the most frequent colors in the list with $n > 0$ items and let n_c be the number of c-items in the list. Then the offline optimum is equal to* $\max(1, 2n_c - n)$ *where $2n_c - n = n_c - (n - n_c)$ is the difference between the number of c-items and non-c-items.*

Proof. We start with the case $n_c - (n - n_c) \leq 1$ in which we pack all items into a single bin. First we put all c-items in the bin and we insert items of other colors between any two consecutive c-items (possibly also after the last one if necessary). Let i be the index of the current c-item in the bin (we index only c-items) after which we put the next non-c-item. Let $c_1, c_2, c_3 \dots c_k$ be an arbitrary ordering of colors other than c and let j be the index of the current color which items we are inserting in the bin. We pack non-c-items with the following algorithm:

1. Let $i = 1$ and $j = 1$.
2. Put a c_j-item in the bin just after the i-th c-item.
3. If all c_j-items are inserted to the bin, then let $j = j + 1$. If $j > k$, stop (we assigned all items into the bin).
4. Let $i = i + 1$. If $i > n_c$, then $i = 1$.
5. Go to the step 3.

Since $n_c - (n - n_c) \leq 1$, i.e., we have at most one more c-item than non-c-items, there is at least one non-c-item between any two c-items. Moreover, two non-c-items of the same color are not between the same two c-items, because c is the most frequent color. Hence the packing created by the algorithm is correct.

Otherwise, if $2n_c - n > 1$, we first show that there is a packing into $2n_c - n$ bins and then we prove that there cannot be a better packing. To create the packing, assign $2n_c - n = n_c - (n - n_c)$ of c-items into separate bins. It remains to pack $n - n_c$ of c-items and $n - n_c$ of non-c-items. We put these items into the first bin such that they alternate between a non-c-item and a c-item, starting with a non-c-item which is put on the c-item packed previously. Therefore there are $n - n_c + 1$ of c-items and $n - n_c$ of non-c-items in the first bin and the other bins contain a single c-item.

We show a matching lower bound on the number of bins. Observe that there is at most one more c-item than non-c-items in each bin. Hence the number of bins is at least the number of c-item minus the number of non-c-items, i.e., $n_c - (n - n_c) = 2n_c - n$. □

The previous theorem gives us a simple linear-time algorithm for computation of the offline optimum of an instance with zero-size items; it just needs to count the number of the most frequent color c and only in the case $n_c > n - n_c$ which can be checked in linear time.

2.6.2 NP-completeness

As we discussed in Chapter 1, Bin Packing is a classical NP-complete problem. We now show that Colored Bin Packing is a generalization of classical Bin Packing by a simple reduction that uses only two colors. In other words, if we are able to solve offline Colored Bin Packing efficiently even only for two colors, we can also solve offline Bin Packing efficiently.

Proposition 6. *A decision version of both unrestricted and restricted offline versions of* Colored Bin Packing *is NP-complete. I.e., it is NP-complete to decide whether it is possible to pack a list of items into k unit capacity bins.*

Proof. First we show that it belongs to NP. As for classical Bin Packing, the certificate is the number of a bin for each item and the verifier just verifies the constraints on the packing of each bin, i.e., that items in each bin does not exceed the capacity and that they can be packed with alternating colors — in the restricted offline case it checks that colors of items in the order of the input sequence alternate in each bin and in the unrestricted offline case it tries to use the algorithm from Theorem 5.

For NP-hardness we reduce Bin Packing to Colored Bin Packing as follows: In an instance of Bin Packing we replace each item of size s by a white item of size s and a black zero-size item. If there exists a packing of the BP instance into k bins, we simply replace each item by the white and black items and obtain a packing of the CBP instance into k bins. On the other hand we create a packing of the Bin Packing instance from a packing of the CBP instance just by considering white items only and deleting their colors. Hence the BP instance has a packing into k bins if and only if the CBP instance has a packing into k bins. □

Note that this reduction preserves inapproximability results for Bin Packing, hence there is no absolutely r-approximation algorithm for any $r < 1.5$. On the other hand, for Black and White Bin Packing, Balogh et al. [6] gave a 2.5-approximation offline algorithm with time complexity of $\mathcal{O}(n \log n)$ and an asymptotic polynomial time approximation scheme (APTAS), both when reordering is allowed. Their APTAS exactly means that for any $\varepsilon > 0$ there is a polynomial algorithm A_ε which produces a packing into at most $(1+\varepsilon)OPT + 1$ bins. It is based on APTAS of Fernandez de la Vega and Lueker [28] developed for classical Bin Packing, yet APTAS for BWBP is more complicated.

2.7 Black and White Bin Packing

We now focus on Colored Bin Packing with two colors, i.e., Black and White Bin Packing, studied previously by Balogh et al. [6, 5]. First we present a simple lower bound of 1.5 on competitiveness of any online algorithm using arbitrarily small, yet non-zero items. We show that Worst Fit performs well for items of size at most $1/d$ (for $d \geq 2$) as it is $(1 + d/(d-1))$-competitive in this case. Then we improve the upper bound on the absolute competitive ratio of Any Fit algorithms from 5 to 3.

Both bounds are tight, since there are instances for FF and BF on which the ratio between the number of bins used and the optimum is asymptotically 3 and an instance for WF in the parametric case on which WF uses asymptotically $(1 + d/(d-1)) \cdot OPT$ bins for an integer $d \geq 2$ [5] (more precisely, the bound for WF is tight only for an integer $d \geq 2$). Note that for infinitesimally small items WF is 2-competitive, while BF and FF are 3-competitive.

We observe a similarity between classical Bin Packing and Black and White Bin Packing: In both problems FF and BF have the same absolute competitive ratio and moreover, in Black and White Bin Packing the instance from [5] proving tightness of the ratio is the same for both algorithms.

2.7.1 Lower Bound on Competitiveness of Any Online Algorithm for Small Items

The lower bound of 2 by Dósa and Epstein [25] and also the preceding lower bound of approximately 1.721 by Balogh et al. [6] are using large items of size up to 1, respectively up to 0.5. We now show a simple construction that uses arbitrarily small items and only two colors to obtain a lower bound of 1.5. Note that this is a little bit surprising, since for zero-size items, all Any Fit algorithms are 1-competitive.

Theorem 7. *For* Black and White Bin Packing *and for any* $\varepsilon > 0$*, there is no deterministic online algorithm with an asymptotic competitive ratio less than* 1.5 *even for items of size at most* ε*.*

Proof. Let k be an integer such that $1/(2k) \leq \varepsilon$ and let $n > 1$ be a large integer (note that the value of n is independent on ε, we may even have $n < k$). The malicious adversary sends $2nk$ items of size $1/(2k)$ and of colors alternating between black and white, starting with a black item. The items have total size of

n, thus an algorithm must use at least n bins. Also the number of black bins is at least $n/2$, or the number of white bins is at least $n/2$. In the former case the adversary sends n black zero-size items and in the latter case he sends n white zero-size items. This forces the algorithm to use at least $n + n/2$ bins in both cases.

We show that $OPT \leq n + 1$, thus the ratio between the number of bins used by the algorithm and OPT tends to 1.5 as n goes to infinity. If the adversary sends n black zero-size items, we pack the first $2nk$ items greedily by an Any Fit algorithm which creates n white bins, thus the n black zero-size items fit in them. Otherwise if the adversary sends n white zero-size items, we put the first black item of size $1/(2k)$ into a separate bin, pack the next $2nk - 1$ items greedily creating a nearly-full white bin and $n - 1$ full black bins in which the $n - 1$ white zero-size items fit, while the last zero-size item is put on the first black item of size $1/(2k)$. Overall we have $n + 1$ bins. Note that $OPT \geq n$. $\square$

2.7.2 Competitiveness of Worst Fit

Theorem 8. *Suppose that all items in the input sequence have size of at most $1/d$, for a real $d \geq 2$. Then Worst Fit is absolutely $(1 + d/(d - 1))$-competitive for* Black and White Bin Packing.

Proof. Let OPT be the number of bins used in an optimal packing. We divide bins created by WF into sets B (big bins) and S (small bins). Each *big bin* has level at least $(d - 1)/d$, thus $|B| \leq d/(d - 1) \cdot OPT$. *Small bins* are smaller than $(d - 1)/d$, thus they can receive any item of the right color. We show that $|S|$ is bounded by the maximal color discrepancy LB_2 and we obtain that WF is $(1 + d/(d - 1))$-competitive.

As items are arriving, we count the number of small black bins, i.e., bins with a black item on the top and with level less than $(d - 1)/d$. Let b_i be the number of small black bins after adding the i-th item from the sequence. Similarly let w_i be the number of white bins with level less than $(d - 1)/d$ after adding the i-th item.

If $b_n = 0$ and $w_n = 0$, i.e., there is no small bin at the end, WF created at most $d/(d - 1) \cdot OPT$ bins. Otherwise suppose without loss of generality that the last created bin has a black item at the bottom. Let t be the index of the black item that created the last bin. It holds that $w_t = 0$, since otherwise the t-th item would go into a small white bin.

Let k be the last index smaller than t for which $b_k = 0$ (if $b_i > 0$ for all $i \geq 1$, we set $k = 0$). The $(k + 1)$-st item must be black. We observe that any bin created after this point has a black item at the bottom, otherwise $b_i = 0$ for some i such that $k < i < t$. Note that w_k can be greater than 0, i.e., there can be some small white bins and the $(k + 1)$-st item goes into one of them. Let W be the set of these bins. See Figure 2.4 for an example of the situation after packing the k-th item. Before adding the t-th item and creating the last bin, all bins in W must have a black item on the top, or become big bins (thus $k \leq t - |W|$).

Let *new items* be items with an index i such that $k < i \leq t$. We want to bound the number of small bins after adding the t-th item by the color discrepancy. We already observed that all these bins must have a black item on the top. Hence for small bins with a black item at the bottom the number of black items is greater

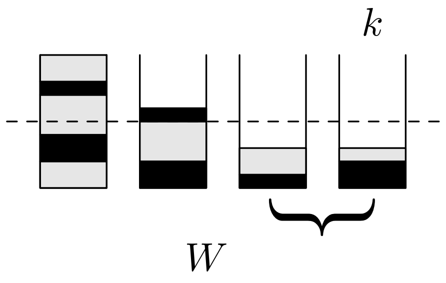

Figure 2.4: An example of the situation after packing the k-th item for $d = 2$. Bins under the dashed line at height of 0.5 are small.

by one than the number of white items. Small bins from the set W can have the same number of black and white items, but in each such bin there is one more new black item than new white items, since the first such item is black.

Now we look at new items which are packed into bins that are big after the t-th item comes. It suffices to show that the number of such new black items is at least the number of such new white items. We observe that WF packs any new white item into a small bin, otherwise $b_\ell = 0$ for some ℓ such that $k < \ell < t$. Hence any new white item must be packed into a bin created after the k-th item (therefore with a new black item at the bottom), or into a bin from the set W. Since the first item that falls into a bin from W after the k-th item is black, our claim holds. See Figure 2.5 for an example of the situation after packing the t-th item.

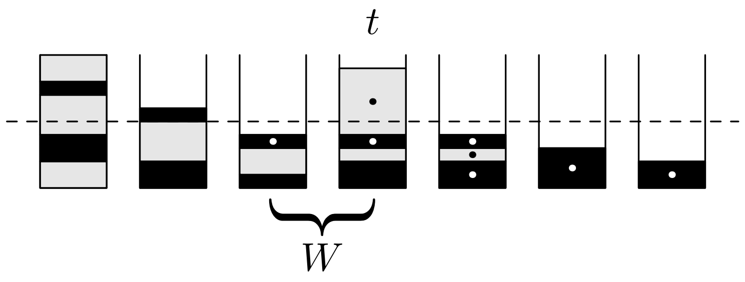

Figure 2.5: The situation after packing the t-th item into a new bin. New items, i.e., items with an index i such that $k < i \le t$, are marked with a dot.

Note that this matching of black and white items in big bins would fail for algorithms like Best Fit or First Fit, since they can put a white item into a big bin created before the k-th item and not contained in W.

We found out that when we focus on new items, i.e., items with an index i such that $k < i \le t$, there is one more such black item than such white items in all small bins and at least as many such black items as such white items in all big bins. Moreover, after the t-th item comes the number of small bins $|S|$ can only decrease, since no bin is created. So we bound $|S|$ from above by the color discrepancy between the $(k+1)$-st and the t-th item:

$$|S| \le \left| \sum_{\ell=k+1}^{t} s_\ell \right| \le LB_2$$

where s_i is 1 when the i-th item is white and -1 otherwise.

Note that the last bin is already counted in the color discrepancy, since its bottom item is black and has index t. □

2.7.3 Competitiveness of Any Fit Algorithms

We can use a similar technique to the one in the proof of Theorem 8 to obtain a 3-competitiveness of Any Fit algorithms, although it gets more complicated, since the number of small bins can be higher than LB_2.

Theorem 9. *Any algorithm in the Any Fit family is absolutely* 3*-competitive for* BLACK AND WHITE BIN PACKING.

Proof. We use the following notation: an item is *small* when its size is less than 0.5 and *big* otherwise. Similarly *small bins* have level less than 0.5 and *big bins* have level at least 0.5.

We assign bins into *chains* — sequences of bins in which all bins except the last must be big. If there is only one bin in a chain it must be big. Moreover, the bottom item in the i-th bin of a chain cannot be added into the $(i-1)$-st bin — even if it would have the right color, i.e., it is too big to be put into the $(i-1)$-st bin.

A bin is contained in at most one chain. We call a bin that is not in a chain a *separated* bin. We create chains such that all big bins are in a chain and only as few small bins as possible remains separated, still ensuring that the average level of bins in each chain is at least 0.5.

Claim 10. *The total size of items in a chain with m bins is at least $m/2$.*

Proof. For $m = 1$ the claim holds, because the only bin in the chain must be big. For $m > 1$ the sum of levels of the last two bins in a chain is at least 1 from the definition of the chain (it holds also for every two consecutive bins) and every bin except the last must be big. Thus the total size of items in a chain with m bins of levels $\ell_1, \ell_2, \dots \ell_m$ is at least:

$$\sum_{i=1}^{m} \ell_i = \sum_{i=1}^{m-2} \ell_i + (\ell_{m-1} + \ell_m) \geq \sum_{i=1}^{m-2} \frac{1}{2} + 1 = \frac{m}{2}$$

□

It follows that the total number of bins in all chains is bounded from above by $2 \cdot OPT$. We want to bound the number of separated bins from above by the maximal color discrepancy LB_2 which yields the 3-competitiveness of AF.

We define a process of assigning bins into chains. We simply try to put as many bins into chains as possible, but we add a bin into a chain only when the last bin in the chain has another color than the bottom item of the added bin.

Formally, when an item from the input sequence is added we do the following:

- The item is added into a bin in a chain: nothing happens with chains or separated bins.
- The item is added into a small separated bin: if the bin becomes big we create a chain from the bin, otherwise the bin stays separated.

- The item is big and creates a new bin: the newly created bin forms a new chain.
- The item is small and creates a new bin: if there is a chain in which the last bin has an item of another color on the top, i.e., black for a white incoming item and white for a black incoming item, we add the newly created bin into the chain. (Note that the last bin in the chain must be big, otherwise the item would fit into the last bin.) If there is no such chain, the new bin is separated.

Moreover, whenever a chain has two big bins we split it into two chains, each containing one big bin. Therefore each chain is either one big bin, or a big bin and a small bin. The intuitive reason for splitting chains is that we can put more newly created small bins into chains.

If there is no separated bin at the end (after the last item is added), we have created at most $2 \cdot OPT$ bins. Otherwise we define k and t as indexes of incoming items and show that the color discrepancy of items between the k-th and the t-th item is at least the number of separated bins at the end.

Let t be the index of an item that created the last bin that is separated when it is created (the t-th item must be small). Suppose without loss of generality that the t-th item is black. Note that a small item that comes after the t-th item can create a bin, but we put the bin into a chain immediately, therefore the number of separated bins can only decrease after adding the t-th item.

Let b_i be the number of small black bins, i.e., bins with a black item on the top, and w_i be the number of small white bins after adding the i-th item from the sequence. From the definition of t we know that $w_t = 0$.

We define k as the biggest $i \leq t$ such that $b_i = 0$, i.e., there is no small black bin (if $b_i > 0$ for all $i \geq 1$ we set $k = 0$). Clearly the $(k+1)$-st item must be small and black. Note that there can be some separated white bins and possibly some other small white bins in chains, but there is no separated black bin. Let W be the set of white bins that are separated after adding the k-th item. See Figure 2.6 for an example of the situation after packing the k-th item. Before adding the t-th item and creating the last bin, all bins in W must have a black item on the top, or become big bins in chains (thus $k \leq t - |W|$).

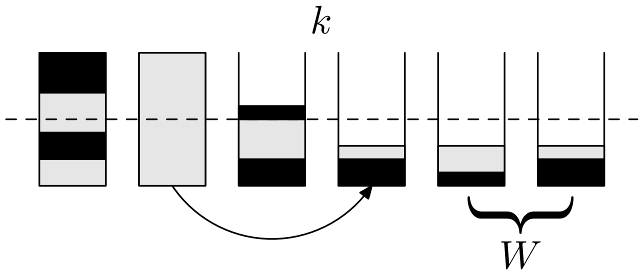

Figure 2.6: An example of the situation after packing the k-th item. Chains with two bins are depicted by arrows (the arrow goes from the first to the second bin). The dashed line is at height 0.5.

Let *new items* be items with an index i such that $k < i \leq t$. We want to bound the number of separated bins after adding the t-th item by the color discrepancy. Note that these bins are small by the process of assigning bins into chains. We observe that all separated bins must have a black item on the top before adding the t-th item and also all chains have a black item on the top in the last bin, otherwise the bin created by the t-th item would be added in a chain.

Hence for separated bins with a black item at the bottom the number of black items is greater by one than the number of white items. Separated bins created with a new item must have a black item at the bottom, since otherwise there cannot be a small black bin and $b_i = 0$ for $k < i < t$.

Separated bins from the set W can have the same number of black and white items before adding the t-th item, but in each such bin there is one more new black item than new white items, since the first and the last such items are black.

Now we look at new items which are packed into bins that are in chains after adding the t-th item. We call such an item a *link*. Note that some links can be at first packed into separated bins, but these bins are put into chains before adding the t-th item. It suffices to show the following claim.

Claim 11. *In each chain the number of black links is at least the number of white links after adding the t-th item.*

Proof. When the t-th item comes and creates a new separated bin, the last item in each chain must be black. Therefore the claim holds for the chains with only one bin.

For the chains with two bins (the first big and the second small) we observe that a bin created with a link has either a black item, or a big white item at the bottom. If it would have a small white link at the bottom, there cannot be a small black bin and $b_i = 0$ for $k < i < t$ which is a contradiction with the definition of k. Since a big white item starts a new chain, the second bin in a chain cannot have a white link at the bottom.

Moreover, the first link in the second bin of a chain must be black, because either the second bin was created after the k-th item and we use the observation from the previous paragraph, or it was created before the k-th item and then it must had a white item on the top when the k-th item came, since there was no small bin with a black item on the top.

So it cannot happen in a chain with two bins that there are two white links next to each other, or separated by some items that are not links. Note that for black links this situation can happen. Since there must be a link in the second bin and the last such link is black, the claim holds for chains with one big and one small bin.

The process of assigning bins into chains does not allow chains with more than two bins or with two big bins. Hence in each chain the number of black links is at least the number of white links. □

See Figure 2.7 for an example of the situation after packing the t-th item.

Let s be the number of separated bins. We found out that when we focus on new items, i.e., items with an index i such that $k < i \leq t$, there is one more such black item than such white items in all separated bins and at least the same number of such items of both colors in bins in all chains, i.e., links. Moreover, after the t-th item comes s can only decrease, since no separated bin is created. So we have bounded the value of s at the end from above by the color discrepancy

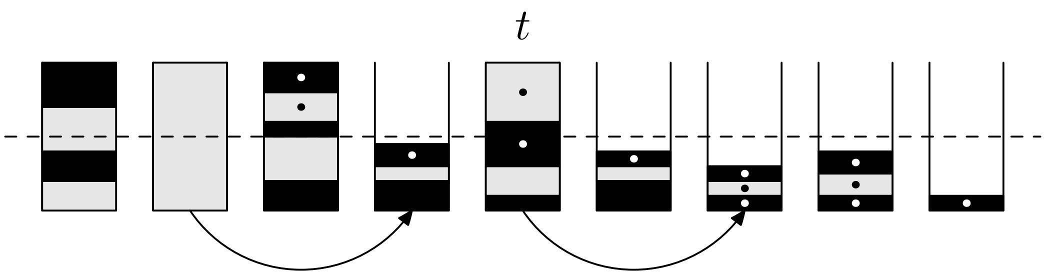

Figure 2.7: The situation after packing the t-th item into a new bin. New items, i.e., items with an index i such that $k < i \leq t$, are marked with a dot.

between the $(k+1)$-st and the t-th item:

$$s \leq \left| \sum_{\ell=k+1}^{t} s_\ell \right| \leq LB_2$$

where s_i is 1 when the i-th item is white and -1 otherwise.

Observe that some items after the t-th item can create a bin, but such bins are put into chains. □

We note that the definition of the chain is not fully used in the previous proof as chains have at most two bins.

2.8 Algorithms for Zero-size Items

We now move to Colored Bin Packing with at least three colors and first show results for the combinatorial case of zero-size items.

2.8.1 Lower Bound on Competitiveness of Any Online Algorithm

It is not hard to see that all Any Fit algorithms behave optimally for zero-size items and only two colors, because the only choice for an incoming item is whether to put it into an open bin of another color, or into a new bin; we observe that the latter is useless. The formal proof is by Balogh et al. [6]. On the other hand, introducing a third color raises the question whether to put a red item into a black bin, or into a white bin. However, it is still useless to put an incoming item into a new bin, unless there is no other possibility as we shall see from the optimal algorithm in Section 2.8.3.

The next theorem indeed shows that the online problem with zero-size items gets harder even for three colors, but a fourth color cannot raise the lower bound which is a consequence of our optimal algorithm.

Theorem 12. *For zero-size items of at least three colors, there is no deterministic online algorithm with an asymptotic competitive ratio less than* 1.5*. Precisely, for each $n > 1$ we can force any deterministic online algorithm to use at least $\lceil 1.5n \rceil$ bins using three colors, while the optimal number of bins is n.*

Proof. The proof by an adversarial construction. We show that if an algorithm uses less than $\lceil 1.5n \rceil$ bins, the adversary can send some items and force the algorithm to increase the number of black bins or to use at least $\lceil 1.5n \rceil$ bins, while the maximal discrepancy stays n. Applying Theorem 3 we know that $OPT = n$, but the algorithm is forced to open $\lceil 1.5n \rceil$ bins using finitely many items as the number of black bins is increasing. Moreover, the adversary uses only three colors throughout the whole proof, denoted by black, white and red and abbreviated by b, w and r in formulas.

We introduce the current discrepancy of a color c which basically tells us how many c-items have come recently and thus how many c-items may arrive without increasing the overall discrepancy. Formally, we define the current discrepancy after packing the k-th item as $CD_{c,k} = \max_{i \leq k+1} \sum_{\ell=i}^{k} s_{c,\ell}$, i.e., the discrepancy on an interval which ends with the last packed item (the k-th). Note that $CD_{c,k}$ is at least zero as we can set $i = k + 1$. We omit the k index in $CD_{c,k}$ when it is obvious from the context.

Initially the adversary sends n black items, then he continues by phases and ends the process whenever the algorithm uses $\lceil 1.5n \rceil$ bins at the end of a phase. When a phase starts, there are less than $\lceil 1.5n \rceil$ black bins and possibly some other white or red bins. We also guarantee $CD_{\mathrm{w}} = 0$, $CD_{\mathrm{r}} = 0$, and $CD_{\mathrm{b}} \leq n$. Let N_{b} be the number of black bins when a phase starts. In each phase the adversary forces the algorithm to use $\lceil 1.5n \rceil$ bins or to have more than N_{b} black bins, while $CD_{\mathrm{w}} = 0$, $CD_{\mathrm{r}} = 0$, and $CD_{\mathrm{b}} \leq n$ at the end of each phase in which N_{b} increases.

We now present how a phase works. Let new items be items from the current phase and old items be items from previous phases. The adversary begins the phase by sending n new items of colors alternating between white and red, starting by white, so he sends $\lceil n/2 \rceil$ white items and $\lfloor n/2 \rfloor$ red items. After these new items, the current discrepancy is one either for red if n is even, or for white if n is odd, and it is zero for the other colors.

If some new item is not put on an old black item, the adversary sends n black items. Since the new items are packed into less than n black bins (more precisely, black at the beginning of the phase), the number of black bins increases. Moreover, $CD_{\mathrm{w}} = 0$, $CD_{\mathrm{r}} = 0$, and $CD_{\mathrm{b}} = n$, hence the adversary finishes the phase and continues with the next phase if there are less than $\lceil 1.5n \rceil$ black bins. See Figure 2.8 for an example of such situation.

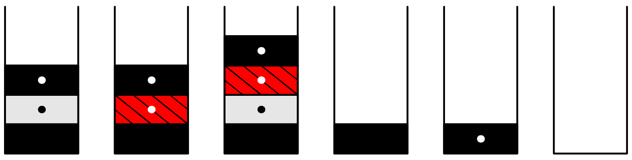

Figure 2.8: One possible end of a phase for $n = 4$. New items are depicted with a dot. Note that an algorithm packed first n new items into less than n black bins, thus the adversary sent n black items and forced an increase of the number of black bins.

Otherwise all new red and white items are put on old black items. If n is even, $CD_{\mathrm{w}} = 0$ and the adversary sends additional n white items. After that there are at least $1.5n$ white bins, so the adversary reaches his goal.

If n is odd, $CD_{\mathrm{w}} = 1$ and the adversary can send only $n - 1$ white items forcing $\lceil 1.5n \rceil - 1$ white bins. This suffices to prove the result in the asymptotic sense, but for the precise lower bound of $\lceil 1.5n \rceil$ for an odd n we need a somewhat more complicated construction.

Therefore if all new red and white items are put on old black items and n is odd, the adversary sends a black item e. If e does not go on a new white item, he sends n white items forcing $\lceil n/2 \rceil + n$ white bins and it is done. Otherwise the black item e is put on a new white item. White and red have $\lfloor n/2 \rfloor$ new items on the top of bins, $CD_{\mathrm{w}} = 0$, and $CD_{\mathrm{r}} = 0$. The adversary sends another black item f. Since red and white are equivalent colors (considering only new items), without loss of generality f goes into a red bin or into a newly opened bin.

Next he sends a white item g and a red item h. After packing g there are $\lceil n/2 \rceil$ bins with a new white item on the top and at least one bin with a new black item on the top. Moreover, after packing the red item h we have $CD_{\mathrm{b}} = 0$ and $CD_{\mathrm{w}} = 0$. So if h is not put on a new white item (i.e., it is put into a black bin, a new bin or on an old white item), the adversary sends n white items and the algorithm must use $\lceil 1.5n \rceil$ bins; see Figure 2.9 for an example of such situation. Otherwise h is packed on a new white item and he sends n black items. The number of black bins increases, because the adversary sent $n + 2$ new black items and at most $n + 1$ new non-black items were put into a black bin (at most n items at the beginning of the phase plus the item g). Since $CD_{\mathrm{w}} = 0$, $CD_{\mathrm{r}} = 0$, and $CD_{\mathrm{b}} = n$, the adversary continues with the next phase. □

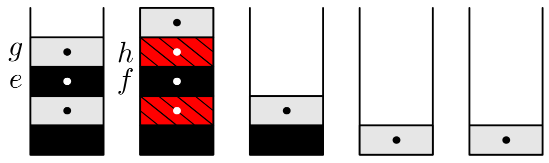

Figure 2.9: One possible end of a phase for $n = 3$. New items are depicted with a dot. Note that an algorithm packed first n new items into n black bins and that the adversary needed just one phase to force opening $\lceil 1.5n \rceil = 5$ bins.

The lower bound has additional properties that we use later in our lower bound for items of arbitrary size. Most importantly, we have at least $\lceil 1.5 \cdot OPT \rceil$ of c-bins at the end (and possibly some additional bins of other colors).

Lemma 13. *After packing the instance from Theorem 12 by an online algorithm there is a color c for which we have $\lceil 1.5 \cdot OPT \rceil$ of c-bins and $CD_c = OPT$, while $CD_{c'} = 0$ for all other colors $c' \neq c$. Moreover, in each restricted offline optimal packing of the instance all the bins have a c-item on the top.*

Proof. Let $n = OPT$ as in the previous proof. The adversary stops sending items when he finishes the last phase. In the last phase either the number of black bins increases to $\lceil 1.5n \rceil$, or the adversary forces $\lceil 1.5n \rceil$ white or red bins by sending n white or red items. In the former case the requirements of the lemma are satisfied, because the proof guarantees $CD_{\mathrm{w}} = 0$ and $CD_{\mathrm{r}} = 0$ at the end of each phase in which the number of black bins increases. Moreover $CD_{\mathrm{b}} = n$, since n black items are sent just before the end of such phase. In the latter case, the last n

white items cause $CD_\mathrm{b} = 0$, $CD_\mathrm{r} = 0$, and $CD_\mathrm{w} = n$, the case of n red items is symmetric.

Since an optimal packing uses n bins and the last n items are of the same color (in each case of the construction), they must go into different bins. Hence each bin of a restricted offline optimal packing has a c-item on the top. □

Note that our lower bound works also when all items have size of at most ε for a sufficiently small ε.

2.8.2 Simple 2-Competitive Algorithm

In Black and White Bin Packing we can use any algorithm from the Any Fit family and it packs zero-size items optimally. For more than two colors we know that no algorithm can be better than asymptotically 1.5-competitive, basically because the algorithm must make decisions of type "Should I put the white item on a red item or on a black item?" Such decisions are done badly for example by First Fit, since it packs items regardless the colors of bins. We shall see a bad behavior of FF, BF and WF even for small items in Section 2.9.3.

We address the problem by balancing the colors of top items in bins – we put an incoming c-item into a bin of the most frequent other color. If we have more choices of bins where to put an item we use First Fit. More formally, we define *Simple Balancing Any Fit* (SBAF) for items of size zero as follows.

Simple Balancing Any Fit (SBAF):

1. Let N_c be the number of c-bins. We put an incoming c-item into a bin of color $c' = \arg\max_{c'' \neq c} N_{c''}$. If more colors have the same maximal number of bins, we can choose color c' arbitrarily among them, e.g., by First Fit. Among c'-bins choose again arbitrarily.

Theorem 14. *Simple Balancing Any Fit algorithm is absolutely 2-competitive for items of size zero and arbitrary number of colors. Moreover, 2-competitiveness is asymptotically tight.*

Proof. We show that the number of bins created by SBAF is at most twice the maximal color discrepancy by a similar argument as in the analyses of Any Fit and Worst Fit for two colors in Section 2.7.

Let n be the number of bins created by SBAF and let t be the index of an item which opened the last bin. The t-th item is black without loss of generality. Note that before adding the t-th item all bins are black.

Let $k < t$ be the maximal index such that the number of black bins is strictly less than $n/2$ after adding the k-th item. The $(k+1)$-st item must be clearly black. We call an item between the $(k+1)$-st and the t-th item a *new item* (including both the $(k+1)$-st and the t-th item). We observe that all new non-black items are added into a black bin, therefore new items are alternating between black and other colors in each bin.

Let B be the set of bins which have a black item on top after adding the k-th item and let W be the complement of B, i.e., bins that are not black after packing the k-th item and bins created after the k-th item. It holds that $n = |B| + |W|$ and that $|B| < n/2$ from the definition of k. In each bin in B new items are alternating starting from another color than black and ending with black, so there is the same number of black and non-black new items. For a bin in W the situation is similar, but the first new item in the bin is black. Hence there is one more new black item than new non-black items in each bin in W which proves that there are overall $|W|$ more new black items than new non-black items.

Using the lower bound LB_2 (the maximal color discrepancy) we obtain the following:

$$LB_2 \geq \sum_{\ell=k+1}^{t} s_{\text{black},\ell} = |W| = n - |B| > \frac{n}{2}$$

Therefore $n < 2 \cdot LB_2$ which proves that SBAF is absolutely 2-competitive.

We now give an instance which shows that our analysis is tight. Let n be a big integer. We send $n-1$ groups, each consisting of n black items, then $n-1$ white items and finally one red item. SBAF does repeatedly the following mistake when there are less than $2n-2$ open bins: When the red item from a group comes, it is put into a white bin, since there are more white bins than black bins. Then SBAF has $n-1$ bins that are not black, hence it opens a new bin for one of the n following black items. Thus each group opens a new bin and the first group opens n bins, therefore SBAF creates $2n-2$ bins.

An optimal packing into n bins can be done by putting red items always into a black bin. Letting n go to infinity, the ratio between SBAF and the optimum tends to two. $\square$

2.8.3 Optimal Algorithm for Zero-size Items

We show that, if we modify the algorithm SBAF to treat with certain situations in another way, we obtain an algorithm for items of size zero that opens at most $\lceil 1.5 LB_2 \rceil$ bins which is optimal in the worst case by Theorem 12. We call the algorithm *Balancing Any Fit* (BAF). In the next section we combine BAF with the algorithm Pseudo by Balogh et al. [5] for items of arbitrary size and prove that the resulting algorithm is absolutely 3.5-competitive.

As in SBAF we still mostly put an incoming c-item into a bin of the most frequent other color, but we have to deal specifically with situations when two colors have relatively many bins.

After packing the k-th item from the sequence, let D_k be the maximal discrepancy so far, i.e., the discrepancy on an interval before the $(k+1)$-st item, and let $N_{c,k}$ be the number of c-bins after packing the k-th item. As in the proof of Theorem 12, we define the current discrepancy as $CD_{c,k} = \max_{i \leq k+1} \sum_{\ell=i}^{k} s_{c,\ell}$, i.e., the discrepancy on an interval which ends with the last packed item (the k-th). Note that $CD_{c,k} \leq D_k$ and that $CD_{c,k}$ is at least zero as we can set $i = k+1$. The current discrepancy basically tells us how many c-items have come recently and thus how many c-items may arrive without increasing the overall discrepancy.

Let $\alpha_{c,k} = N_{c,k} - \lceil D_k/2 \rceil$ be the difference between the number of c-bins and the half of the maximal discrepancy so far. Observe that $\lceil D_k/2 \rceil$ is the number of bins which BAF may use in addition to OPT bins. We omit the index k in D_k, $N_{c,k}$, $CD_{c,k}$ and $\alpha_{c,k}$ when it is obvious from the context.

While processing the items, if D is the maximal discrepancy so far, the adversary can send $D - CD_c$ of c-items without increasing the maximal discrepancy, while forcing the algorithm to use $N_c + D - CD_c$ bins. Hence, to end with at most $\lceil 1.5D \rceil$ bins we try to keep $N_c - CD_c \leq \lceil D/2 \rceil$ for all colors c. For simplicity, we use an equivalent inequality of $\alpha_c = N_c - \lceil D/2 \rceil \leq CD_c$. If we can keep that and there is a color c with $N_c > \lceil 1.5D \rceil$, we get $CD_c \geq N_c - \lceil D/2 \rceil > \lceil 1.5D \rceil - \lceil D/2 \rceil = D$ which contradicts $CD_c \leq D$. Let the **main invariant** for a color c be

$$\alpha_c = N_c - \left\lceil \frac{D}{2} \right\rceil \leq CD_c. \tag{2.1}$$

As $CD_c \geq 0$, keeping the invariant is easy for all colors with at most $\lceil D/2 \rceil$ bins. Also when there is only one color c with $N_c > \lceil D/2 \rceil$, we just put all non-c-items into c-bins. Therefore, if a non-c-item comes, the number of c-bins N_c decreases and the current discrepancy CD_c decreases by at most one. (CD_c stays the same when it is zero.) Since both increase with an incoming c-item, we are keeping our main invariant (2.1) for the color c.

Moreover, there are at most two colors with strictly more than $\lceil D/2 \rceil$ bins, given that we have at most $\lceil 1.5D \rceil$ open bins. Thus we only have to deal with two colors having $N_c > \lceil D/2 \rceil$. We state the algorithm Balancing Any Fit for items of size zero.

Balancing Any Fit (BAF):

1. For an incoming c-item, if there are no bins or c-bins only, open a new bin and put the item into it.
2. Otherwise, if there is at most one color with the number of bins strictly more than $\lceil D/2 \rceil$, put an incoming c-item into a bin of color $c' = \arg\max_{c'' \neq c} N_{c''}$. If more colors have the same maximal number of bins, choose color c' arbitrarily among them, e.g., by First Fit. Among c'-bins, choose again arbitrarily.
3. Suppose that there are two colors b and w such that $N_{\mathrm{b}} > \lceil D/2 \rceil$ and $N_{\mathrm{w}} > \lceil D/2 \rceil$. If $c = \mathrm{w}$, put the item into a bin of color b. If $c = \mathrm{b}$, put the item into a bin of color w. Otherwise $c \notin \{\mathrm{b}, \mathrm{w}\}$; if $N_{\mathrm{b}} - \lceil D/2 \rceil < CD_{\mathrm{b}}$, put the item into a bin of color w, otherwise into a bin of color b.

As we discussed, keeping the main invariant (2.1) is easy in the first and the second case of the algorithm. Therefore we can conclude the following claim.

Claim 15. *Suppose that the main invariant holds for all colors before packing the t-th item and that there is at most one color c with $N_{c,t-1} > \lceil D_{t-1}/2 \rceil$ before the t-th item, i.e., the t-th item is packed using the first or the second case of the algorithm. Then the main invariant holds for all colors also after packing the t-th item.*

Most of the proof of 1.5-competitiveness of BAF thus deals with two colors having more than $\lceil D/2 \rceil$ bins. Without loss of generality, let these two colors be black and white in the following and let us abbreviate them by b and w.

In the third case of the algorithm we have to choose either black or white bin for items of other colors than black and white, but the current discrepancy decreases for both black and white, while the number of bins stays the same for the color which we do not choose. So if $\alpha_{\mathrm{b}} = CD_{\mathrm{b}}$ and $\alpha_{\mathrm{w}} = CD_{\mathrm{w}}$, the adversary

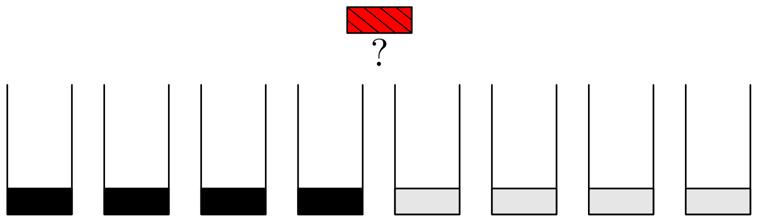

Figure 2.10: Example with $D = 5$ and $\lceil 1.5 \cdot D \rceil = 8$. Suppose that $CD_{\mathrm{b}} = 1$ and $CD_{\mathrm{w}} = 1$, thus $N_{\mathrm{b}} = CD_{\mathrm{b}} + \lceil D/2 \rceil$ and $N_{\mathrm{w}} = CD_{\mathrm{w}} + \lceil D/2 \rceil$, i.e., the main invariant does not hold strictly for both black and white. If the next incoming red item goes into a black bin, then the adversary sends five white items, since CD_{w} decreases to zero. Hence the adversary forces nine open bins. The case in which the red item is packed into a white bin is symmetric.

can force the algorithm to open more than $\lceil 1.5D \rceil$ bins. See Figure 2.10 for an example of such situation.

Therefore we need to prove that in the third case, i.e., when $N_{\mathrm{b}} > \lceil D/2 \rceil$ and $N_{\mathrm{w}} > \lceil D/2 \rceil$, at least one of inequalities $\alpha_{\mathrm{b}} \leq CD_{\mathrm{b}}$ and $\alpha_{\mathrm{w}} \leq CD_{\mathrm{w}}$ is strict. This motivates the following **secondary invariant:**

$$2\alpha_{\mathrm{b}} + 2\alpha_{\mathrm{w}} \leq CD_{\mathrm{b}} + CD_{\mathrm{w}} + 1\,. \tag{2.2}$$

If the secondary invariant holds, it is not hard to see that in the third case of the algorithm the choice of the bin maintains the main invariant. The tricky part of the proof is to prove the *base case* of the inductive proof of the secondary invariant. A natural proof would show the base case whenever b and w become the two colors with $N_{\mathrm{b}}, N_{\mathrm{w}} > \lceil D/2 \rceil$. However, we are not able to do that. Instead, we prove that the secondary invariant holds already at the moment when b and w become the two strictly most frequent colors on top of the bins, i.e., $N_{\mathrm{b}} > N_c$ and $N_{\mathrm{w}} > N_c$ for all other colors c, which may happen much earlier, when the number of their bins is significantly below $D/2$. After that, maintaining both invariants is relatively easy.

Theorem 16. *Balancing Any Fit algorithm is* 1.5*-competitive for items of size zero and an arbitrary number of colors. Precisely, it uses at most* $\lceil 1.5 \cdot OPT \rceil$ *bins.*

Proof. First we show that keeping the main invariant (2.1) for each color c, i.e., $\alpha_c \leq CD_c$, is sufficient for the algorithm to create at most $\lceil 1.5D \rceil$ bins. This implies both that the algorithm is well defined since there are at most two colors with $N_c > \lceil D/2 \rceil$, and that the algorithm is 1.5-competitive, since the maximal discrepancy equals the optimum.

Claim 17. *After packing the t-th item, if we suppose that $N_{c,i} - \lceil D_i/2 \rceil \leq CD_{c,i}$ for all colors c and for all $i < t$, the algorithm uses at most $\lceil 1.5D_t \rceil$ bins.*

Proof. We prove the claim by contradiction: Suppose that BAF opens a bin with the k-th item in the sequence (for $k \leq t$) and we exceed the $\lceil 1.5D_k \rceil$ limit, but before the k-th item there were at most $\lceil 1.5D_{k-1} \rceil$ bins. Thus $D_k = D_{k-1}$, since if $D_k = D_{k-1} + 1$, then the bound also increases with the k-th item.

Let c be the color the k-th item. Let the ℓ-th item be the last non-c-item before the k-th, so only c-items come after the ℓ-th item. None of c-items from the $(\ell+1)$-st to the k-th increase the maximal discrepancy D, otherwise if one such item increases D, then all following such items also do. Thus $D_\ell = D_k$.

The algorithm must have received $\lceil 1.5D_\ell \rceil + 1 - N_{c,\ell}$ of c-items after the ℓ-th item to open $\lceil 1.5D_\ell \rceil + 1$ bins, but then

$$CD_{c,k} = CD_{c,\ell} + \lceil 1.5D_\ell \rceil + 1 - N_{c,\ell} \geq N_{c,\ell} - \left\lceil \frac{D_\ell}{2} \right\rceil + \lceil 1.5D_\ell \rceil + 1 - N_{c,\ell} = D_\ell + 1$$

where we used the main invariant for the inequality which holds, because $\ell < k \leq t$. We get a contradiction, since $CD_{c,k} \leq D_k = D_\ell$. $\square$

We have to deal with the case in which $N_\mathrm{b} > \lceil D/2 \rceil$ and $N_\mathrm{w} > \lceil D/2 \rceil$. We show that we can maintain the secondary invariant (2.2), while black and white are the two strictly most frequent colors of bins (even if $N_\mathrm{b} \leq \lceil D/2 \rceil$ or $N_\mathrm{w} \leq \lceil D/2 \rceil$). Then we prove that the secondary invariant starts to hold when black and white become the two strictly most frequent colors, i.e., $N_c < N_\mathrm{b}$ and $N_c < N_\mathrm{w}$ for all other colors c; this step must precede the time when the number of bins for the second color gets over the $\lceil D/2 \rceil$ limit. Therefore we prove by induction that the secondary invariant holds in certain intervals of the input sequence.

Claim 18. *Suppose that black and white are the two strictly most frequent colors of bins before packing the t-th item and that the main invariant (2.1) holds for all colors and the secondary invariant (2.2) also holds before packing the t-th item, i.e., $N_{c,t-1} - \lceil D_{t-1}/2 \rceil \leq CD_{c,t-1}$ for all colors c and $2\alpha_{\mathrm{b},t-1} + 2\alpha_{\mathrm{w},t-1} \leq CD_{\mathrm{b},t-1} + CD_{\mathrm{w},t-1} + 1$. Then the main invariant for all colors and the secondary invariant for black and white hold also after packing the t-th item.*

Proof. First we suppose that the maximal discrepancy D is not changed by the t-th item. We start by showing that the main invariant holds after packing the t-th item. If the t-th item is packed using the second case of BAF, the main invariant holds by Claim 15. (Note that the t-th item cannot be packed using the first case of the algorithm, since $N_{\mathrm{b},t-1} > 0$ and $N_{\mathrm{w},t-1} > 0$.)

Otherwise, if the t-th item is packed using the third case, it holds that $\alpha_{\mathrm{b},t-1} > 0$ and $\alpha_{\mathrm{w},t-1} > 0$. The main invariant holds for a color c other than black and white, because $N_{c,t-1} < \lceil D_{t-1}/2 \rceil$ which implies $N_{c,t} \leq \lceil D_t/2 \rceil$.

To prove the main invariant for black and white, we show by contradiction that the secondary invariant (2.2) guarantees that $\alpha_{\mathrm{b},t-1} < CD_{\mathrm{b},t-1}$ or $\alpha_{\mathrm{w},t-1} < CD_{\mathrm{w},t-1}$. Otherwise, if $\alpha_{\mathrm{b},t-1} \geq CD_{\mathrm{b},t-1}$ and $\alpha_{\mathrm{w},t-1} \geq CD_{\mathrm{w},t-1}$, the secondary invariant becomes $2\alpha_{\mathrm{b},t-1} + 2\alpha_{\mathrm{w},t-1} \leq CD_{\mathrm{b},t-1} + CD_{\mathrm{w},t-1} + 1 \leq \alpha_{\mathrm{b},t-1} + \alpha_{\mathrm{w},t-1} + 1$ which is a contradiction. Note that we used that $\alpha_{\mathrm{w},t-1}$ and $\alpha_{\mathrm{b},t-1}$ are integral.

We now distinguish three cases according to the color of the t-th item:

- The t-th item is black: Then it is packed into a white bin. The main invariant for black holds after packing the item, because both N_b and CD_b increase, and the main invariant for white holds, since N_w decreases and CD_w decreases by at most one. (CD_w stays the same when it is zero.)
- When the t-th item is white, the situation is symmetric to the previous case.

- We pack the t-th item of another color into a white bin if $N_{\mathrm{b},t-1} - \lceil D_{t-1}/2 \rceil < CD_{\mathrm{b},t-1}$, otherwise into a black bin. If it is packed into a white bin, N_{w} decreases and CD_{w} decreases by at most one, thus the main invariant holds for white. The main invariant holds for black too, since N_{b} stays the same and CD_{b} decreases by at most one, but the main invariant held strictly for black before packing the t-th item.
  If the t-th item is packed into a black bin, we have $N_{\mathrm{w},t-1} - \lceil D_{t-1}/2 \rceil < CD_{\mathrm{w},t-1}$ and the situation is symmetric as if the t-th item is packed into a white bin.

It remains to show that the t-th item does not violate the secondary invariant. There are again three cases according to the color of the t-th item:

- The t-th item is black: Then it is packed into a white bin in both second and third cases of the algorithm. Thus α_{b} increases and α_{w} decreases, so the left-hand side of the inequality stays the same. Also the right-hand side does not change or even increases as CD_{b} increases and CD_{w} decreases by at most one. (CD_{w} stays the same when it is zero.)
- When the t-th item is white, the situation is symmetric to the previous case.
- The t-th item has another color than black and white: Then it is packed into a white or black bin in both second and third cases of the algorithm. Thus one of α_{w} and α_{b} decreases and the other one stays the same, while both CD_{b} and CD_{w} decrease by at most one. The secondary invariant holds as the left-hand side decreases by two and the right-hand side decreases by at most two.

Otherwise D increases with an incoming c-item, thus also CD_c increases and both $N_{c'}$ and $\alpha_{c'}$ for each color c' decrease if D becomes odd. We follow the same proof as if D stays the same, and the eventual additional decrease of $N_{c'}$ and $\alpha_{c'}$ can only decrease the left-hand sides of the main and secondary invariants. □

Note that in the previous proof, α_{b} or α_{w} can be negative in the secondary invariant. We show the base case of the secondary invariant, i.e., that it starts to hold when two colors become the two strictly most frequent colors of bins.

Claim 19. *Suppose that after packing the k-th item it starts to hold that $N_c < N_{\mathrm{b}}$ and $N_c < N_{\mathrm{w}}$ for all other colors c, i.e., black and white become the two strictly most frequent colors. Suppose also that the main invariant holds all the time before packing the k-th item. Then $2\alpha_{\mathrm{b},k} + 2\alpha_{\mathrm{w},k} \leq CD_{\mathrm{b},k} + CD_{\mathrm{w},k} + 1$, i.e., the secondary invariant holds after packing the k-th item.*

Proof. Assume without loss of generality that $N_{\mathrm{b},k} \geq N_{\mathrm{w},k}$. If $N_{\mathrm{b},k} = N_{\mathrm{w},k}$, we also suppose without loss of generality that $N_{\mathrm{b},k-1} \geq N_{\mathrm{w},k-1}$.

First we show by contradiction that always $N_{\mathrm{b},k-1} \geq N_{\mathrm{w},k-1}$. Otherwise if $N_{\mathrm{b},k-1} < N_{\mathrm{w},k-1}$, then $N_{\mathrm{b},k} > N_{\mathrm{w},k}$ (note that $N_{\mathrm{b},k} = N_{\mathrm{w},k}$ would imply $N_{\mathrm{b},k-1} \geq N_{\mathrm{w},k-1}$). This can happen only when a black item goes into a white bin, but then the numbers of black and white bins are swapped, hence black and white were already the two strictly most frequent colors before the k-th item which contradicts the assumption of the claim. We conclude that $N_{\mathrm{b},k} \geq N_{\mathrm{w},k}$ and $N_{\mathrm{b},k-1} \geq N_{\mathrm{w},k-1}$.

Before the k-th item the number of non-black bins is at most $\lceil 1.5D_{k-1} \rceil - N_{\mathrm{b},k-1} = D_{k-1} - \alpha_{\mathrm{b},k-1}$, since there are at most $\lceil 1.5D_{k-1} \rceil$ bins by Claim 17 (we use that the main invariant holds before packing the k-th item). As we have $N_{\mathrm{b},k-1} \geq N_{\mathrm{w},k-1}$ and black and white are not the two strictly most frequent colors before the k-th item, there must be a color $\mathrm{r} \notin \{\mathrm{b}, \mathrm{w}\}$ such that $N_{\mathrm{r},k-1} \geq N_{\mathrm{w},k-1}$ (let the color be red without loss of generality). Therefore the number of white bins is at most half of the number of non-black bins, i.e., $N_{\mathrm{w},k-1} \leq (D_{k-1} - \alpha_{\mathrm{b},k-1})/2$.

We show by contradiction that the k-th item must be packed using the second case of the algorithm. (Note that BAF cannot use the first case, since otherwise all bins would have the same color after packing the k-th item.) If the item is packed using the third case, it must hold that $N_{\mathrm{b},k-1} \geq \lceil D_{k-1}/2 \rceil + 1$ and $N_{\mathrm{r},k-1} \geq \lceil D_{k-1}/2 \rceil + 1$. Since there are at most $\lceil 1.5 D_{k-1} \rceil$ bins by Claim 17, we get $N_{\mathrm{w},k-1} \leq \lfloor D_{k-1}/2 \rfloor - 2$, but then the k-th item cannot cause $N_{\mathrm{w},k} > N_{\mathrm{r},k}$.

Therefore BAF packs the k-th item using the second case and it follows that the main invariant holds after packing the k-th item for all colors by Claim 15.

Observe that by packing the k-th item, the number of white bins must increase, or the number of red bins must decrease, or both. Note that the k-th item can have any color, not only white. We distinguish two cases: the k-th item is white and the k-th item is not white.

If the k-th item is white, we have $\alpha_{\mathrm{b},k} \leq \alpha_{\mathrm{b},k-1}$, as the number of black bins does not increase (note that there is an inequality because of a possible increase of D or a decrease of N_{b}). We get

$$\begin{aligned}\alpha_{\mathrm{w},k} = N_{\mathrm{w},k} - \left\lceil \frac{D_k}{2} \right\rceil = N_{\mathrm{w},k-1} + 1 - \left\lceil \frac{D_k}{2} \right\rceil \leq \frac{D_{k-1} - \alpha_{\mathrm{b},k-1}}{2} + 1 - \left\lceil \frac{D_k}{2} \right\rceil \\ \leq \frac{D_k - \alpha_{\mathrm{b},k}}{2} + 1 - \left\lceil \frac{D_k}{2} \right\rceil \leq -\frac{\alpha_{\mathrm{b},k}}{2} + 1.\end{aligned}$$

where we used $N_{\mathrm{w},k-1} \leq (D_{k-1} - \alpha_{\mathrm{b},k-1})/2$ for the first inequality and $D_{k-1} - \alpha_{\mathrm{b},k-1} \leq D_k - \alpha_{\mathrm{b},k}$ for the second inequality which follows from $\alpha_{\mathrm{b},k} \leq \alpha_{\mathrm{b},k-1}$.

We know that $\alpha_{\mathrm{w},k} \leq -\alpha_{\mathrm{b},k}/2 + 1$. Therefore

$$2\alpha_{\mathrm{w},k} + 2\alpha_{\mathrm{b},k} \leq -\alpha_{\mathrm{b},k} + 2 + 2\alpha_{\mathrm{b},k} = \alpha_{\mathrm{b},k} + 2 \leq CD_{\mathrm{b},k} + 2 \leq CD_{\mathrm{w},k} + CD_{\mathrm{b},k} + 1$$

where we use the main invariant (2.1) for black color for the second inequality and $CD_{\mathrm{w},k} \geq 1$ for the third inequality which holds, because the k-th item is white.

Otherwise the k-th item is not white and it is packed into a bin of another color than black and white, otherwise N_{b} or N_{w} decreases, thus black and white cannot become the two strictly most frequent colors. After packing the k-th item we have $\alpha_{\mathrm{b},k} \leq \alpha_{\mathrm{b},k-1} + 1$, as the k-th item may be black, therefore $D_{k-1} - \alpha_{\mathrm{b},k-1} \leq D_k - \alpha_{\mathrm{b},k} + 1$. Since the number of white bins does not change, we get

$$\begin{aligned}\alpha_{\mathrm{w},k} = N_{\mathrm{w},k} - \left\lceil \frac{D_k}{2} \right\rceil = N_{\mathrm{w},k-1} - \left\lceil \frac{D_k}{2} \right\rceil \leq \frac{D_{k-1} - \alpha_{\mathrm{b},k-1}}{2} - \left\lceil \frac{D_k}{2} \right\rceil \\ \leq \frac{D_k - \alpha_{\mathrm{b},k} + 1}{2} - \left\lceil \frac{D_k}{2} \right\rceil \leq -\frac{\alpha_{\mathrm{b},k}}{2} + 0.5.\end{aligned}$$

In this case we have $\alpha_{\mathrm{w},k} \leq -\alpha_{\mathrm{b},k}/2 + 0.5$. Therefore

$$2\alpha_{\mathrm{w},k} + 2\alpha_{\mathrm{b},k} \leq -\alpha_{\mathrm{b},k} + 1 + 2\alpha_{\mathrm{b},k} = \alpha_{\mathrm{b},k} + 1 \leq CD_{\mathrm{b},k} + 1 \leq CD_{\mathrm{w},k} + CD_{\mathrm{b},k} + 1$$

where we use the main invariant (2.1) for black color for the second inequality. Hence the secondary invariant (2.2) holds. □

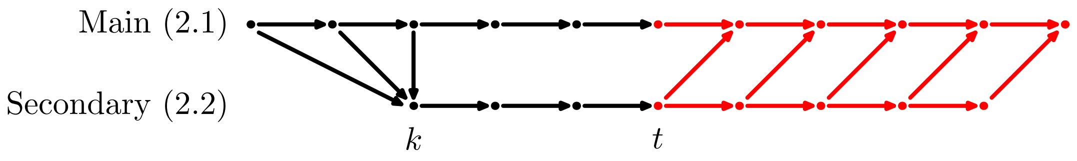

Figure 2.11: An illustration of dependencies of the main and secondary invariants. The horizontal axis represents time. An invariant at a certain time represented by point P follows from invariants from which there is an arrow to P. After packing the k-th item (time k) black and white become the two strictly most frequent and after the t-th item (time t) it starts to hold that $N_\mathrm{b} > \lceil D/2 \rceil$ and $N_\mathrm{w} > \lceil D/2 \rceil$. Thus in the black part of the figure, BAF uses the first or the second case of the algorithm, while in the red part BAF uses the third case of the algorithm.

We now complete the proof of the theorem by putting everything together. Precisely, we prove that the main invariant holds during the whole run of the algorithm by induction. The main invariant for each color holds trivially at the beginning before any item comes. When the t-th item is packed, there are two cases:

- No two colors were the strictly most frequent before the t-th item: BAF keeps the main invariant for all colors by Claim 15, since it must pack the t-th item with the first or the second case of the algorithm. If two colors become the two strictly most frequent after packing the t-th item, the secondary invariant starts to hold by Claim 19; otherwise the secondary invariant is irrelevant in this case.
- Two colors were the strictly most frequent: Let these two colors be black and white without loss of generality. Then the main invariant for all colors and the secondary invariant for black and white are kept by Claim 18 (even if black and white are not the two strictly most frequent after the t-th item). It may happen that the two strictly most frequent colors change after packing the t-th item (e.g., to black and red). The main invariant for all colors still follows by Claim 18, but the secondary invariant for the new strictly most frequent colors follows by Claim 19.

See Figure 2.11 for an illustration of dependencies of the invariants.

Therefore we can keep the main invariant $N_c - \lceil D/2 \rceil \le CD_c$ for all colors c during the whole run of the algorithm and the theorem follows by Claim 17. $\square$

2.9 Algorithms for Items of Arbitrary Size

2.9.1 Lower Bound on Competitiveness of Any Online Algorithm

We use the construction by Dósa and Epstein [25] proving the lower bound 2 for two colors to get a lower bound 2.5 using three colors. We combine it with the hard instance that shows the lower bound 1.5 for zero-size items. This is an evidence that Colored Bin Packing with at least three colors is harder than Black and White Bin Packing even for items of arbitrary size, but it is not proven — still the best possible competitive ratio for both problems may be the same, although it seems unlikely to be true.

Theorem 20. *For items of at least three colors, there is no deterministic online algorithm with an asymptotic competitive ratio less than* 2.5.

Proof. Throughout the whole proof the adversary uses only three colors denoted by black, white and red and abbreviated by b, w and r in formulas. Let $n > 1$ be a large integer. The adversary starts with the hard instance for zero-size items from the proof of Theorem 12 with the optimum equal to n. By Lemma 13 there are at least $\lceil 1.5n \rceil$ bins of the same color, without loss of generality white. Let W be the set of bins that are white after the first part of the instance ($|W| \geq \lceil 1.5n \rceil$). We also know that $CD_{\mathrm{w}} \leq n$, $CD_{\mathrm{b}} = 0$, and $CD_{\mathrm{r}} = 0$.

The second part of the instance is a slightly simplified construction by Dósa and Epstein [25]. Their idea goes as follows: The adversary sends the instance in phases, each starting with white and black small items. If the black item is put into an already opened bin, we send a huge white item that can be put only on the small black item. Therefore the algorithm has to put the huge white item in a new bin, but an optimal offline algorithm puts the small black item into a new bin and the huge white item on it. Otherwise, if the small black item is put into a new bin, the phase is finished — the online algorithm opened a bin in the phase, while an optimal offline algorithm does not need to. This way an online algorithm is forced to behave oppositely to an optimal offline algorithm.

We formalize this idea by the following adversarial algorithm. Let $\varepsilon = 1/(6n)$ and $\delta_i = 1/(5^i \cdot 6n)$. The adversary uses the items of the following types:

- regular white items of size ε,
- regular black items of size δ_i for some $i \geq 1$,
- special black items of size $3\delta_i$ for some $i \geq 1$,
- huge white items of size $1 - 2\delta_i$ for some $i \geq 1$.

Let i be the index of the current phase and let j be the number of huge white items in the instance so far. The adversarial algorithm is as follows:

1. Let $i = 0$ and $j = 0$.
2. If $j = n$ or if $i = \lceil 2.5n \rceil$, then stop.
3. Let $i = i + 1$. Send $\left(\substack{\text{white} \\ \varepsilon}, \substack{\text{black} \\ \delta_i}\right)$.
4. If the regular black item goes to a new bin or to a bin with level zero, go to the step 2 (continue with the next phase).
5. Let $j = j + 1$. Send $\left(\substack{\text{black} \\ 3\delta_i}, \substack{\text{white} \\ 1-2\delta_i}, \substack{\text{black} \\ \delta_i}\right)$. Then go to the step 2 (continue with the next phase).

See Figure 2.12 for an example of the situation after two phases of the adversarial algorithm.

First we show that we can pack the whole list of items into $n + 1$ bins and then that no huge white item can be packed by an online algorithm into a bin from the set W, i.e., one of $\lceil 1.5n \rceil$ bins which are white after the first part with zero-size items.

Lemma 21. $OPT = n + 1$.

Proof. The first part of the items, i.e., the hard instance for zero-size items, has the optimum exactly n. Moreover, all bins are white by Lemma 13. Each of $j \leq n$ huge white items is packed with the two regular black items from the same phase, thus creating j full bins with a black item at the bottom. All these bins thus can be combined with the bins from the first part of the instance. The remaining

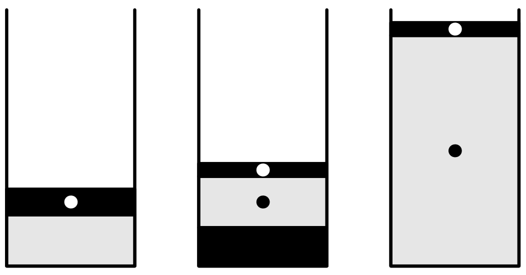

Figure 2.12: A situation after two phases of the algorithm (for simplicity, zero-size items are not shown). Items from the second phase are marked with a dot. In the first phase, the second item went into a bin of level zero, so the phase ended immediately. But in the second phase the second item went into a non-zero bin, so a huge white item came.

items have alternating colors and total size of at most $i \cdot 2\varepsilon \leq \lceil 2.5n \rceil \cdot 2/(6n) \leq 1$ (recall that i is the index of the current phase and that each black item is smaller than ε), so they can be put into an additional bin.

Note that the maximal discrepancy LB_2 is $n + 1$, since we end the first part by n white items and start the second part by a regular white item. Hence $OPT = n + 1$. □

We now analyze how an online algorithm behaves on the second part of the instance.

Lemma 22. *After the i-th phase the number of bins with a non-zero level is at least i. Moreover, no huge white item goes into a bin from the set W.*

Proof. We use the fact that $3\delta_i < \varepsilon$, i.e., all black items are smaller than ε, and that a huge white item of size $1 - 2\delta_i$ cannot be packed with a black item of size at least δ_j for any $j < i$.

We show that in each phase the number of bins with a non-zero level increases by at least one. This holds trivially, if the second item in a phase, denoted by s, goes to a new bin or to a bin with level zero. Otherwise, if s is put into a bin of non-zero level, the adversary continues the phase by sending three other items, most importantly a huge white item h. The item s is the only one sent before h that is sufficiently small to be packed into a single bin with h, but s is in a bin with another non-zero item. Therefore h must be packed into a new bin or a bin with level zero. This proves the first statement of the lemma.

For the second statement, note that if h goes to a bin with zero-size items only, the bin cannot be white, but all the bins from the set W that have still level zero, while packing h, are white. As we already observed, h also does not go to a bin from W that has a non-zero level. □

By the previous lemma we know that if the second part of the instance ends with $i = \lceil 2.5n \rceil$, there are $\lceil 2.5n \rceil$ non-zero bins. If the instance stops by $j = n$, the online algorithm has at least $|W| + n \geq \lceil 2.5n \rceil$ open bins, since it opens bins in W on the first part of the instance and it must put n huge white items into other bins. As $OPT = n + 1$, we get that the ratio between the online algorithm and the optimum tends to 2.5 as n goes to infinity. □

Note that if we replace the first part of the instance by n white items, we get the lower bound of 2 using only two colors, i.e., the lower bound for Black and White Bin Packing.

We tried to modify the second part of the instance to work in the parameter setting and we were able to get only a lower bound of $d/(d-1)$ using only two colors. The bound equals 2 for $d = 2$, i.e., for items of size at most 0.5, but for $d \geq 3$ it drops to at most 1.5, but the Theorem 7 already proves this bound for arbitrarily small items.

2.9.2 3.5-competitive Algorithm

We now show that there is a constant competitive online algorithm even for items of size between 0 and 1. We combine algorithms Pseudo from [5] and our algorithm BAF that is 1.5-competitive for zero-size items. The algorithm Pseudo uses *pseudo bins* which are bins of unbounded capacity; see Figure 2.13 for examples of pseudo bins.

```
Pseudo-BAF:
  1. First pack an incoming item into a pseudo bin using the algorithm
     BAF (treat the item as a zero-size item).
  2. In each pseudo bin, items are packed into unit capacity bins using
     Next Fit.
```

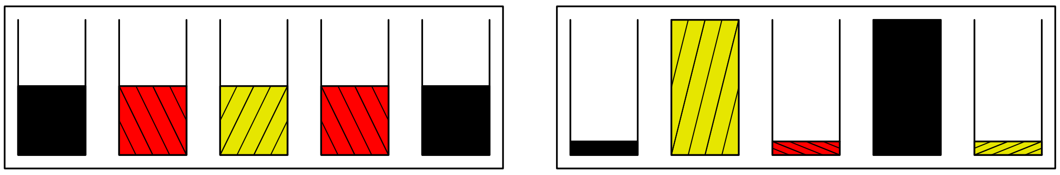
Figure 2.13: An example of two pseudo bins, both split into five unit capacity bins.

Theorem 23. *The algorithm Pseudo-BAF for* Colored Bin Packing *is absolutely* 3.5*-competitive. In the parametric case when items have size at most* $1/d$*, for a real* $d \geq 2$*, it uses at most* $\lceil (1.5 + d/(d-1))OPT \rceil$ *bins. Moreover, the analysis is asymptotically tight.*

Proof. In the general case for items between 0 and 1 we know that two consecutive bins in one pseudo bin have total size strictly more than one, since no two consecutive items of the same color are in a pseudo bin. In each pseudo bin we match each bin with an odd index with the following bin with an even index, therefore we match all bins except at most one in each pseudo bin. Moreover, the total size of a pair of matched bins is more than one. Therefore the number of matched bins is strictly less than $2 \cdot LB_1 \leq 2 \cdot OPT$, i.e., at most $2 \cdot OPT - 1$. (Recall that LB_1 is the sum of the sizes of all items in the sequence, hence a lower bound on the optimum.) The number of not matched bins is at most the number of pseudo bins created by the algorithm BAF which uses at most $\lceil 1.5 \cdot LB_2 \rceil \leq \lceil 1.5 \cdot OPT \rceil \leq 1.5 \cdot OPT + 0.5$ bins as the maximal discrepancy LB_2

is also a lower bound on the optimum. Summing both bounds, the algorithm Pseudo-BAF creates at most $3.5 \cdot OPT$ bins.

For the parametric case, inside each pseudo bin all real bins except the last one have level strictly more than $(d-1)/d$, so their number is strictly less than $d/(d-1) \cdot OPT$. The number of pseudo bins is still bounded by $\lceil 1.5 \cdot OPT \rceil$, thus the algorithm Pseudo opens at most $\lceil (1.5 + d/(d-1))OPT \rceil$ bins.

We show the tightness of the analysis by combining hard instances for Pseudo by Balogh et al. [5] and for BAF from the proof of Theorem 12. More concretely, for n (a big integer) let $\varepsilon = 1/(2n)$. The adversary sends $n-1$ groups of three items, specifically $(n-1) \times \left(\substack{\text{white}\\ \varepsilon}, \substack{\text{black}\\ 1}, \substack{\text{black}\\ \varepsilon}\right)$.

The algorithm creates one pseudo bin containing every first and second item from each group and $n-1$ pseudo bins, each containing only the third item from a group. Moreover, the first pseudo bin is split into $2 \cdot (n-1)$ unit capacity bins (each item is in a separate bin), so there are $3 \cdot (n-1)$ bins. The optimum for $n-1$ groups is n, because we can pack all tiny items together in one bin and $LB_1 = n$.

Then the adversary sends the hard instance with zero-size items from the proof of Theorem 12 (excluding initial n black items, since all pseudo bins are already black) and BAF creates additional $\lceil (n-1)/2 \rceil$ pseudo bins, while the optimum on the instance is $n-1$. Pseudo-BAF now has $\lceil 3.5 \cdot (n-1) \rceil$ bins. Observe that the optimal packing for $n-1$ groups does not need to be changed to put there zero-size items (an optimal packing of $n-1$ groups of non-zero items has also all bins black), thus $OPT = n$.

We now show a modification of the first part of the hard instance for the parametric case, also by Balogh et al. [5]. The adversary sends $(d-1)n$ groups of $4d-1$ items. Let $\delta = 1/(4d^2(d-1)n)$. Items in a group are of three types: big of size $1/d - (d-1)\delta$, small of size $2d\delta$ and tiny of size $d\delta$.

Items in a group are

$$\left(\substack{\text{black}\\ \text{big}}, \substack{\text{black}\\ \text{small}}, \substack{\text{white}\\ \text{small}}, \substack{\text{black}\\ \text{small}}, (d-2) \times \left(\substack{\text{white}\\ \text{tiny}}, \substack{\text{black}\\ \text{big}}, \substack{\text{white}\\ \text{small}}, \substack{\text{black}\\ \text{small}}\right), \substack{\text{white}\\ \text{tiny}}, \substack{\text{black}\\ \text{big}}, \substack{\text{white}\\ \text{small}}\right)$$

.

In one group there are d big black items, $d-1$ tiny white items and $2d$ small items of alternating colors starting with a black item and ending with a white item. An optimal packing puts all small items from all groups in one bin of level $(d-1)n \cdot 2d \cdot 2d\delta = (d-1)n \cdot 4d^2 \cdot \frac{1}{4d^2(d-1)n} = 1$. Big and tiny items alternate in each group, so we can pack big and tiny items from one group into one bin of level $d \cdot (\frac{1}{d} - (d-1)\delta) + (d-1) \cdot d\delta = 1$. In total, an optimal packing has $(d-1)n + 1$ full bins.

The algorithm Pseudo packs all items into one pseudo bin except the second items in all groups that are put into separate pseudo bins. The second items in groups thus create $(d-1)n$ bins. Moreover, the first pseudo bin must be split into at least dn bins, because there are totally $(d-1)nd$ big items and any bin has at most $d-1$ big items which we show by contradiction.

Suppose that a bin B contains d big items. We observe that between any two big items in the first pseudo bin there is always at least one small item. Therefore the bin B contains items of total size at least $d \cdot (1/d - (d-1)\delta) + (d-1) \cdot 2d\delta = 1 + (d-1) \cdot d\delta > 1$ and we get a contradiction.

Hence Pseudo packs these $(d-1)n$ groups into at least $(d-1)n+dn$ bins and the optimum is $(d-1)n+1$, so the ratio is $(2d-1)n/((d-1)n+1)$ which tends to $1+d/(d-1)$ as n goes to infinity.

Similarly to the general case all $(d-1)n+1$ pseudo bins except one are black and the adversary proceeds with the hard instance for BAF with zero-size items. Note that the maximal discrepancy on the first part of the instance is $(d-1)n$, since there is one more black item than white items in each group. The optimum for the hard instance for BAF is now $(d-1)n$, so BAF creates additional $\lceil 0.5 \cdot (d-1)n \rceil$ pseudo bins.

In the optimal packing we put zero-size items into bins containing big and tiny items, since these bins are black and the adversary starts the hard instance from the proof of Theorem 12 by $(d-1)n$ non-black items. Thus the whole hard instance has optimum still $(d-1)n+1$ and Pseudo ends up with asymptotically $(1.5+d/(d-1)) \cdot OPT$ bins. $\square$

Note that for two colors the previous proof yields the absolute 3-competitiveness of Pseudo and $(1+d/(d-1))$-competitiveness of Pseudo in the parametric case. This follows from the fact that BAF behaves optimally for zero-size items of two colors (even all Any Fit algorithms are 1-competitive in this case). The hard instances proving tightness of the result for two colors are the first parts of the instances from the previous proof, i.e., we just delete the second parts with zero-size items.

We tried to modify the algorithm Pseudo to obtain an algorithm with a better competitive ratio, e.g., by splitting pseudo bins (they can be viewed as chains from the proof of Theorem 9) and reusing bins at the beginning of pseudo bins, but it seems that it does not lead to a better than 3.5-competitive algorithm (or 3-competitive for two colors). Therefore, a different approach must be probably carried out. We also conjecture that there is a better than 3.5-competitive algorithm, since Pseudo is quite simple. For two colors, Worst Fit is even simpler and together with Pseudo they are the best known algorithms up to now.

2.9.3 Classical Any Fit Algorithms

We analyze classical Any Fit algorithms, namely First Fit, Best Fit and Worst Fit and we find that they are not constant competitive. Their competitiveness cannot be bounded by any function of the number of colors even for only three colors, in contrast to their good performance for two colors.

Proposition 24. *First Fit and Best Fit are not constant competitive.*

Proof. The adversary sends an instance of $4n$ items which can be packed into two bins, but FF and BF create $n+1$ bins where n is an arbitrary integer.

Let $\varepsilon = 1/(4n)$. The instance is $n \times \binom{\text{black}}{\varepsilon}, \binom{\text{black}}{\varepsilon}, \binom{\text{white}}{\varepsilon}, \binom{\text{red}}{\varepsilon}$. An optimal packing can be obtained by putting black items from each group into the first and the second bin, the white item into the first bin and the red item into the second bin.

FF and BF pack the first group into two bins, both with a black bottom item, while white and red items fall into the first bin. The first black item, the white item and the red item from each following group are packed into the first bin, while the second black item is packed into a new bin. Therefore these algorithms create one bin with all white and red items and all first black items from each group and n bins with a single black item.

Hence FF and BF create $(n+1)/2OPT$ bins. □

Note that WF on such instance creates an optimal packing, but the instance can be modified straightforwardly to obtain a bad behavior for WF.

Proposition 25. *Worst Fit is not constant competitive.*

Proof. The instance is similar to the one in the previous proof, but sizes of items are different in each group. Let $\varepsilon = 1/(2n)$ and let $\delta = 1/(6n^2+1)$. The instance is $n \times \left(\substack{\text{black}\\ \delta}, \substack{\text{black}\\ \varepsilon}, \substack{\text{white}\\ \delta}, \substack{\text{red}\\ \delta} \right)$.

We observe that the optimal packing does not need to be changed due to other sizes. However, WF packs all δ-items into the first bin, i.e., first black items from each group and all white and red items, since the level of the first bin stays at most $3n/(6n^2+1)$, which is less than $1/2n$ as $\varepsilon/\delta > 3n$. Therefore all second black items are packed into separate bins and WF creates $n+1$ bins, while the optimum is two. □

2.10 Notes

2.10.1 Other Restrictions on Packing Items with Colors

In Colored Bin Packing we do not allow two items of the same color to be adjacent in a bin, but there can be other, more restrictive models of packing items with more than two colors:

- **Graph model.** We fix a graph G over the set of colors C. No two items of the same color can be packed one on the other and moreover, if there is an edge between colors c_1 and c_2, then c_1-items and c_2-items cannot be packed next to each other, otherwise such a packing is possible.

- **Permutation model.** We fix a permutation P of colors. We need to pack items in the order of colors given by the permutation. The bottom item in a bin can be arbitrary.

- **Natural model.** No two items of the same color can be packed next to each other in a bin which is exactly how we defined Colored Bin Packing.

Note that the natural model and the graph model are equivalent for a graph G without edges. For two colors and a graph without edges all three models are equivalent. Colored Bin Packing with a complete graph G is trivial: no two items can be packed in a single bin.

See Chapter 3 for some other variants of Bin Packing in which items have colors.

3. Other Variants of Bin Packing

Many variants of Bin Packing restricting the packing were studied since seventies. We survey some of them which are important or interesting. At the end of this chapter we list definitions of some other Bin Packing variants (without any results).

3.1 Bounded Space Bin Packing

Of particular interest is Bounded Space Bin Packing where an algorithm can have only $K \geq 1$ open bins (also called active) in which it is allowed to put incoming items. When a bin is closed, an algorithm cannot pack any further item in the bin or open it again. Such algorithms are called *K-bounded-space* and in addition to a packing rule we need to specify their closing rule which is used whenever we have K open bins and the packing rule yields opening a new bin.

Bounded space algorithms finds many applications in the real world, too. For example when packing trucks at a loading dock, there can be only a limited number of positions for trucks, i.e., it is impossible to simultaneously load more than K trucks. In stock cutting there can be a limited number of papers of a standard size (or generally unit pieces of a material) from which we are cutting pieces of various sizes. Or suppose that a communication channel transmits blocks of a fixed size that are filled with arriving packets of various sizes, but the transmitter can keep only K blocks in the memory.

Next Fit is the only reasonable 1-bounded-space algorithm, since closing the active bin when the current item can go into it cannot lead to a better solution than adding the item into the active bin. As we argued in Chapter 1 its competitiveness equals 2. In the following we therefore assume $K \geq 2$.

Next-K Fit keeps the last K created bins open and puts an incoming item into the active bin with the lowest index where it fits. It also closes the lowest indexed active bin, if an item does not fit into any active bin. For $K \geq 2$, Next-K Fit is the first studied bounded space algorithm with the asymptotic competitive ratio of exactly $1.7 + 3/(10(K-1))$ [19, 52].

The champion among bounded-space variants of Any Fit algorithms is K-Bounded Best Fit (BBF-K), i.e., Best Fit with at most K open bins which closes the fullest open bin, if an item does not fit into any open bin. It is surprisingly (asymptotically) 1.7-competitive for all $K \geq 2$ [20].

Lee and Lee [50] presented Harmonic(K) which is K-bounded-space with the asymptotic ratio tending to $h_\infty = 1.691\ldots$ as K goes to infinity (see Section 1.2.2 for the definition of h_∞). After that, *Simplified Harmonic* (SH$_K$) by Woeginger designed from Harmonic achieved a better ratio for small values of K, but its ratio tends to h_∞, too. Lee and Lee [50] also proved that there is no bounded space algorithm with a better asymptotic ratio. Still, there is no online bounded-space algorithm which achieves this ratio (Harmonic(K) and SH$_K$ only approaches h_∞ in the limit).

The Bounded Space Bin Packing is an especially interesting variant in the context of Colored Bin Packing due to the fact that it matters whether we allow the optimum to reorder the input instance, or not. If we allow reordering for Bounded Space Bin Packing, we get the same optimum as classical Bin Packing. In fact, all the bounds on online algorithms in a few previous paragraphs hold, if the optimum with reordering is considered, which is a stronger statement than comparing to the optimum without reordering. This is a very different situation than for Colored Bin Packing where no online algorithm can be constant competitive against the optimum with reordering, as we have noted at the beginning of Chapter 2.

The bounded space offline optimum without reordering was studied by Chrobak et al. [14]. It turns out that the computational complexity is very different: There exists an offline $(1.5+\varepsilon)$-approximation algorithm for 2-bounded-space Bin Packing with running time polynomial for every constant $\varepsilon > 0$, but exponential in ε. No polynomial time 2-bounded-space algorithm can have its approximation ratio better than $5/4$ (unless $P = NP$). In the online setting it is open whether there exists a better algorithm than K-Bounded Best Fit which is at most 1.7-competitive when compared to the optimum without reordering; the current lower bound is $3/2$.

3.2 Bin Packing with Cardinality Constraints

Another interesting variant with restrictions on the contents of a bin is Bin Packing with Cardinality Constraints which restricts the number of items in a bin to at most k for a parameter $k \geq 2$. It was introduced by Krause, Shen and Schwetman [49].

We can view BPCC as a multiprocessor Scheduling with k processors such that items correspond to tasks with a unit execution time and the size of an item represents the amount of a unit resource that the task requires. In this settings, bins are integral starting times. The constraints for a packing ensures that there are at most k tasks running at each time and any k tasks that are processed simultaneously require together at most one unit of the resource.

Krause et al. [49] showed that Cardinality Constrained First Fit has the asymptotic ratio of at most $2.7 - 2.4/k$ and at least $2.7 - 3.7/k$ for $k \geq 3$. Note that both bounds tends to 2.7 as k goes to infinity, while FF is 1.7-competitive when the number of items in a bin is unrestricted. The upper bounds on First Fit were recently improved to $2.5 - 2/k$ for $k = 2, 3, 4$, to $8/3 - 8/(3k)$ for k from 4 to 10 and to $2.7 - 3/k$ for $k \geq 10$ [22].

For $k \geq 3$, there is an asymptotically 2-competitive online algorithm by Babel, Chen, Kellerer and Kotov [3] who also designed an approximately 1.447-competitive algorithm for $k = 2$. Better than 2-competitive algorithms are known for small values of k [27], see Table 3.1. Regarding the absolute competitive ratio there is a tight bound of 2 for any $k \geq 4$ [22].

Interestingly, the lower bound for the asymptotic competitive ratio for a large k is 1.540 [7], i.e., the same as for standard Bin Packing. Similarly, the current lower bound is 1.5 for $k = 6$ [62] and 1.539 for k from 42 to 293 [60], both using a construction developed for standard Bin Packing. The lower bounds for small values of k are presented in Table 3.1. The current lower bounds for k from 7 to 11 are due to Dósa and Epstein [22] and the ones for k from 12 to 41 are due to Fujiwara and Kobayashi [30].

k	Lower bound	Upper bound
2	1.428 [30]	1.447 [3]
3	1.5 [3]	1.75 [27]
4	1.5 [30]	1.868 [27]
5	1.5 [22]	1.937 [27]
6	1.5 [62]	1.993 [27]
7	1.517 [22]	2 [3]
8	1.524 [22]	2 [3]
9	1.524 [22]	2 [3]
10	1.509 [30]	2 [3]

Table 3.1: The current lower and upper bounds for small values of k in Bin Packing with Cardinality Constraints

3.3 Vector Packing

BPCC can be generalized to the Vector Packing problem in which items are d-dimensional vectors and bins have capacity 1^d, i.e., the sum of all vectors in a bin is at most one for every coordinate. For BPCC, items are 2-dimensional where the size of an item goes to the first coordinate and the second coordinate is always $\frac{1}{k}$.

Vector Packing models for example the resource allocation problem in which we have servers with d resources (all servers have the same capacity for each resource) and jobs corresponding to items demand some part of each resource. Another application comes from the virtual machine placement which currently gains attention due to the virtualization in data centers. In this problem, an algorithm assigns virtual machines to a set of real machines and must ensure that it does not exceed the capacity of CPU, memory, or a disk in each machine. The goal is to use as few real machines as possible.

Kou and Markowsky [48] generalized Any Fit algorithms to *reasonable algorithms* which output a packing such that the contents of any pair of bins cannot be combined into a single bin. They showed that any reasonable algorithm has the asymptotic competitive ratio of at most $d + 1$. Next Fit or Harmonic are not reasonable, but an extension of First Fit to d dimensions is a reasonable algorithm. The bound of $d + 1$ for FF was improved only once in 1976 by Garey, Graham, Johnson and Yao [37] to $d + 0.7$ (which yields a tight bound of 1.7 for one dimensional problem). It is open whether there is an algorithm with a better competitive ratio.

As for the lower bound, Yao [62] proved that there is no offline r-approximation algorithm with a running time $o(n \log n)$ for $r < d$ (more precisely, with a decision tree of height $o(n \log n)$ where n is the number of vectors). Hence there are no time efficient online algorithms which are better than d-competitive. Regarding the offline setting, even for $d = 2$ there is no asymptotic PTAS [61].

For many years, the lower bounds for all online algorithms were only slightly larger than 1.540, i.e., the lower bound for standard Bin Packing. Particularly, Blitz, van Vliet and Woeginger [40] gave bounds of 1.671, 1.758, 1.833, 1.873, 1.898 for $d = 2, 3, 4, 5, 6$, respectively; these bounds tends to 2 as d goes to infinity.

The wide gap between the lower bound of 2 and the upper bound of $d + 0.7$ was narrowed in 2013 by Azar, Cohen, Kamara and Shepherd [1] who showed that for any $\varepsilon > 0$ any deterministic online algorithm has a competitive ratio of $\Omega(d^{1-\varepsilon})$. They also analyzed Vector Packing with bins of the same size $B \geq 1$ in each coordinate, while items are still at most 1 in each coordinate.

3.4 Box Packing

Another, more geometric way of generalizing Bin Packing into more dimensions is Box Packing. Items are now d-dimensional boxes with sides of at most one and bins become d-dimensional unit cubes. The algorithm tries to pack boxes into the minimum number of cubes such that boxes are not overlapping and cannot be rotated. This problem models loading trucks closely to a real-world situation, although it seems to be harder for analysis than one dimensional case.

Though most of the results are for offline algorithms, there are a few online algorithms. Coppersmith and Raghavan [17] designed the first online 2-dimensional algorithm CRA which rounds up one side of an incoming box and then packs it using an algorithm similar to Harmonic, i.e., it assigns items into classes according to their size and packs classes separately. Both rounding and packing wastes only a constant factor, hence their algorithm has a constant competitive ratio of at most 3.25. The analysis of CRA was improved by Csirik, Frenk and Labbé [18] who showed that the bound of 3.25 is tight when CRA uses Next Fit to pack items from a class, but it drops to at most 3.06 if CRA utilizes First Fit.

In 2002, Seiden and van Stee [56] designed a Harmonic-based algorithm that achieves 2.660 and Han et al. [39] improved the result to 2.555 by using a better underlying one dimensional algorithm.

Csirik and van Vliet [21] gave the first online algorithm which works in higher dimensions. They used a similar, but more complex procedure than CRA and the resulting algorithm has the competitive ratio of at most $h_\infty^d \approx 1.691^d$.

For the lower bounds, we can trivially generalize the lower bound for one dimensional Bin Packing. Galambos [34] first improves the bound of 1.540 to 1.6 for two dimensions. The current lower bound of 1.907 for two dimensions is due to Blitz, van Vliet and Woeginger [8] who also showed that for 3D Box Packing there is no online algorithm which is better than 2.111-competitive. On the other hand, for a dimension $d \geq 4$ their method can lead only to bounds smaller than 3, so there is a large gap between the lower bound and the upper bound of 1.691^d.

A similar problem is Strip Packing: Two dimensional objects need to be packed into a single vertical strip such that the total height of the strip is minimized. In general, rotations are also forbidden.

3.5 Bin Stretching

In the online Bin Stretching problem, we are given $m \geq 2$, the number of bins that we can use. A sequence of items of size between 0 and 1 arrives and each item needs to be packed immediately. Moreover, the whole sequence of items can be packed into m bins with unit capacity by an offline algorithm which makes the problem rather semi-online. Since this is not always possible for an online algorithm, it is allowed to use stretched bins with capacity $R \geq 1$. The goal is to minimize the stretching factor R.

Since we are minimizing the levels of bins, the problem can be also thought of as a semi-online Scheduling on m identical machines in which we have a guarantee that an optimal offline algorithm can schedule all jobs with makespan 1.

Bin Stretching was proposed by Azar and Regev [2] who showed a lower bound of 4/3 for any number of bins $m \geq 2$. The lower bound goes as follows: Send two items of size 1/3. If they are packed into the same bin, send m items of size 2/3. Otherwise both the items of size 1/3 goes into different bins and then send $m-1$ items of size one. In both cases, any online algorithm must have a bin with level at least 4/3, while all items can be packed into m unit capacity bins.

Although this construction is quite simple, there is still no better lower bound for a general m. Only for three bins, Gabay, Kotov and Brauner [31] proved that the stretching factor of any online algorithm is at least 19/14 by describing the lower bound as a two-player zero-sum game between an algorithm and the adversary and solving the game by the Alpha-beta search (one has to limit the moves of the adversary, otherwise the game would not be finite).

On the algorithmic side, Azar and Regev [2] achieved a stretching factor 1.625 for any number of bins. Kellerer and Kotov [46] designed a two-phase algorithm with a stretching factor $11/7 \approx 1.571$ and Gabay, Kotov and Brauner [32] improved their result to $26/17 \approx 1.529$. Together with Böhm, Sgall, and van Stee [9] we presented an algorithm with a stretching factor 1.5 which takes the similar two-phase algorithms to their limit, i.e., another approach must be very likely applied to obtain a better algorithm.

Better algorithms are known for some small values of m. Even before the paper by Azar and Regev [2] that introduced this problem, there was an optimal algorithm for $m = 2$ by Kellerer et al. [47]. The algorithm goes as follows: First pack all items greedily into the first bin, until an item s comes and does not fit. If there are items of size at least 2/3 in the first bin, all remaining items including s fit into the second bin. Otherwise the size of s is more than 2/3 and s is packed into the second bin, whilst all items except s fit into the first bin.

Azar and Regev [2] also gave an algorithm achieving a factor $(5m-1)/(3m+1)$ which is 1.4 for $m = 3$. Together with Böhm, Sgall, and van Stee [9] we recently improved the result for three bins to $11/8 = 1.375$.

3.6 Other Variants of Bin Packing

We sketched results for several types of Bin Packing problems, but over more than 40 years of study of Bin Packing problems, researchers investigated many more variants of one dimensional Bin Packing. We list a few more:

- In Bin Packing with Variable Sized Bins there are bins of types $B_1, B_2, \ldots B_k$ such that $1 = sz(B_1) > sz(B_2) > \cdots > sz(B_k)$ where $sz(B)$ means the size of a bin of the type B. An online algorithm has unlimited number of bins of each type and its objective is to pack items into the bins of the smallest total capacity.
- Instead of minimizing the number of bins, one can fix the number of bins and tries to minimize their level, similarly to Bin Stretching, but without

a known optimum. This is equivalent to multiprocessor Scheduling as we noted in Section 1.3.

- Another objective is to maximize the number of items packed to m unit capacity bins where m is fixed. In other words, the objective is to choose the largest subset of a given list of items such that the subset can be packed into m bins.
- One can also maximize the number of full bins where a bin is full if it has level of at least T. For $T = 1$, this is called Bin Covering.
- In Dynamic Bin Packing, each item has its arrival time a_i and departure time d_i (besides its size as usual), i.e., it must be packed at time a_i and it disappears from the bin at time d_i. The algorithm aims to minimize the maximum number of bins used over the time.
- The Bin Packing can be also modeled by a game of selfish agents which correspond to items. Each item tries to minimize its cost which is proportional to the level of its bin and an item can decrease its cost by moving into a more full bin. The aim of the study is whether the game reaches a stable state in which no item wants to move and how well the number of bins in a stable state approximates the optimum. There are also other game theoretic models of some variants of Bin Packing.
- In Bin Packing with Rejection an items has its size and a rejection cost. An algorithm either packs the item, or rejects it, while it aims to minimize the number of bins used plus the sum of rejection costs of items that are not packed.
- Similarly to Colored Bin Packing, in Class Constrained Bin Packing each item has a size and a color, but we bound the number of different colors in a bin by a positive integer Q. We allow a packing of two items of the same color adjacently.
- Another Bin Packing problem with item colors (which can be viewed also as types or classes) is Bin Packing with Item Types in which the only additional constraint to capacity is that there cannot be two items of the same type in a bin. This problem is thus more restrictive than Colored Bin Packing.
- The Bin Packing with LIB Constraints (Largest Item in Bottom) restricts the packing so that an online algorithm is not allowed to put an item on a smaller item. As in Colored Bin Packing, items are ordered in a bin by their arrival time, so the LIB constraint is that the items in a bin are sorted decreasingly by their size. Moreover, the optimal offline packings with and without reordering the items differ in this model.
- In the very general problem of Bin Packing with Conflicts, we are given a graph over items (in addition to item sizes). If there is an edge between two items, they cannot be packed into a single bin. Since this problem generalizes graph coloring which is hard to approximate, it was studied only for special graphs, such as perfect graphs or interval graphs.

See the survey of Coffman et al. [16] for results on these problems and for further references.

Conclusions

We have surveyed state-of-art results of the worst-case analysis for the Bin Packing problem and some of its variants. We have focused on one of its generalizations, the Colored Bin Packing problem, which we have investigated in more detail and for which we have advanced the current state-of-art results.

Colored Bin Packing for zero-size items is now completely solved (in the worst-case scenario) — we have fully characterized the optimum in both restricted and unrestricted offline settings and we have given an optimal 1.5-competitive algorithm which uses at most $\lceil 1.5 \cdot OPT \rceil$ bins together with a matching lower bound on competitiveness of all online algorithms using only three colors.

For items of arbitrary size, our online algorithm still leaves a gap between our lower bound of 2.5 and our upper bound of 3.5. The upper bounds are only 0.5 higher than for two colors (Black and White Bin Packing) where a gap between 2 and 3 remains for general items. Since the best algorithm is relatively simple (when compared to the best algorithm for classical Bin Packing), a better than 3.5-competitive algorithm seems more plausible than a higher lower bound.

Interestingly, it seems that introducing the fourth color does not make the problem harder than for three colors, in contrast to introducing the third color. At least for zero-size items, it is proven by the optimal algorithm and the matching lower bound that the additional fourth color does not increase the competitiveness.

Classical algorithms First Fit, Best Fit and Worst Fit, although they maintain a constant approximation for two colors, start to behave badly when we introduce the third color. For two colors, we now know their exact behavior. In fact, all Any Fit algorithms are absolutely 3-competitive which is a tight bound for FF, BF and WF. However, for items of size up to $1/d$ (for a real $d \geq 2$), FF and BF remain 3-competitive, while WF has the absolute competitive ratio $1 + d/(d-1)$. Thus we know that even the simple Worst Fit algorithm matches the performance of Pseudo, the online algorithm with the best competitive ratio known so far. It is also an interesting question whether it holds that Any Fit algorithms cannot be better than 3-competitive for two colors.

Bibliography

[1] Y. Azar, I. R. Cohen, S. Kamara, and B. Shepherd. Tight bounds for online vector bin packing. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 961–970. ACM, 2013.

[2] Y. Azar and O. Regev. On-line bin-stretching. *Theoretical Computer Science*, 268(1):17–41, 2001.

[3] L. Babel, B. Chen, H. Kellerer, and V. Kotov. Algorithms for on-line bin-packing problems with cardinality constraints. *Discrete Applied Mathematics*, 143:238–251, 2004.

[4] B. S. Baker and E. G. Coffman. A tight asymptotic bound for Next-Fit-Decreasing bin-packing. *SIAM J. on Algebraic and Discrete Methods*, 2:147–152, 1981.

[5] J. Balogh, J. Békési, G. Dósa, L. Epstein, H. Kellerer, and Z. Tuza. Online results for black and white bin packing. *Theory of Computing Systems*, pages 1–19, 2014.

[6] J. Balogh, J. Békési, G. Dósa, H. Kellerer, and Z. Tuza. Black and white bin packing. In *Approximation and Online Algorithms*, volume 7846 of *Lecture Notes in Computer Science*, pages 131–144. Springer, 2013.

[7] 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.

[8] 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.

[9] M. Böhm, J. Sgall, R. van Stee, and P. Veselý. Better algorithms for online bin stretching. *To appear in proceedings of 12th Workshop on Approximation and Online Algorithms (WAOA 2014)*, 2014. Also ArXiv 1404.5569.

[10] M. Böhm, J. Sgall, and P. Veselý. Online colored bin packing. *To appear in proceedings of 12th Workshop on Approximation and Online Algorithms (WAOA 2014)*, 2014. Also ArXiv 1404.5548.

[11] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[12] D. J. Brown. A lower bound for on-line one-dimensional bin packing algorithms. Technical Report R-864, Coordinated Sci. Lab., Urbana, Illinois, 1979.

[13] B. Chandra. Does randomization help in online bin packing? *Inform. Process. Lett.*, 43:15–19, 1992.

[14] M. Chrobak, J. Sgall, and G. J. Woeginger. Two-bounded-space bin packing revisited. In *Algorithms - ESA 2011*, volume 6942 of *Lecture Notes in Computer Science*, pages 263–274. Springer Berlin Heidelberg, 2011.

[15] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In *Approximation algorithms.* PWS Publishing Company, 1997.

[16] E. Coffman Jr., J. Csirik, G. Galambos, S. Martello, and D. Vigo. Bin packing approximation algorithms: Survey and classification. In *Handbook of Combinatorial Optimization*, pages 455–531. Springer New York, 2013.

[17] D. Coppersmith and P. Raghavan. Multidimensional online bin packing: Algorithms and worst case analysis. *Oper. Res. Lett.*, 8:17–20, 1989.

[18] J. Csirik, J. B. G. Frenk, and M. Labbe. Two dimensional rectangle packing: On line methods and results. *Discrete Appl. Math.*, 45:197–204, 1993.

[19] J. Csirik and B. Imreh. On the worst-case performance of the NKF bin-packing heuristic. *Acta Cybern.*, 9:89–105, Dec. 1989.

[20] J. Csirik and D. S. Johnson. Bounded space on-line bin packing: Best is better than first. *Algorithmica*, 31(2):115–138, 2001.

[21] J. Csirik and A. van Vliet. An online algorithm for multidimensional bin packing. *Oper. Res. Lett.*, 13:149–158, 1993.

[22] G. Dósa and L. Epstein. Online bin packing with cardinality constraints revisited. *ArXiv e-prints 1404.1056*, Apr. 2014.

[23] G. Dósa and J. Sgall. First Fit bin packing: A tight analysis. In *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 538–549, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[24] G. Dósa. The tight bound of First Fit Decreasing bin-packing algorithm is $FFD(i) \leq 11/9OPT(i) + 6/9$. In *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, volume 4614 of *Lecture Notes in Computer Science*, pages 1–11. Springer Berlin Heidelberg, 2007.

[25] G. Dósa and L. Epstein. Colorful bin packing. In *Algorithm Theory – SWAT 2014*, volume 8503 of *Lecture Notes in Computer Science*, pages 170–181. Springer International Publishing, 2014. Also ArXiv 1404.3990.

[26] G. Dósa and J. Sgall. Optimal analysis of Best Fit bin packing. In *Automata, Languages, and Programming*, volume 8572 of *Lecture Notes in Computer Science*, pages 429–441. Springer Berlin Heidelberg, 2014.

[27] L. Epstein. Online bin packing with cardinality constraints. *SIAM Journal on Discrete Mathematics*, 20, 2006.

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

[29] A. Fiat and G. J. Woeginger, editors. *Online Algorithms: The State of the Art.* Springer, 1998.

[30] H. Fujiwara and K. Kobayashi. Improved lower bounds for the online bin packing problem with cardinality constraints. *Journal of Combinatorial Optimization*, pages 1–21, 2013.

[31] M. Gabay, N. Brauner, V. Kotov, et al. Computing lower bounds for online optimization problems: Application to the bin stretching problem. 2013.

[32] M. Gabay, V. Kotov, N. Brauner, et al. Semi-online bin stretching with bunch techniques. 2013.

[33] G. Galambos. Parametric lower bounds for online bin packing. *SIAM J. on Algebraic and Discrete Methods*, 7:362–367, 1986.

[34] G. Galambos. A 1.6 lower bound for the two-dimensional online rectangle bin packing. *Acta Cybernet.*, 10:21–24, 1991.

[35] G. Galambos and J. B. G. Frenk. A simple proof of Liang's lower bound for online bin packing and the extension to the parametric case. *Discrete Appl. Math.*, 41:173–178, 1993.

[36] G. Galambos and G. J. Woeginger. Online bin packing - a restricted survey. *ZOR Math. Methods Oper. Res.*, 42:25–45, 1995.

[37] M. R. Garey, R. L. Graham, D. S. Johnson, and A. C. C. Yao. Resource constrained scheduling as generalized bin packing. *J. Combin. Theory Ser. A*, 21:257–298, 1976.

[38] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical J.*, 45:1563–1581, 1966.

[39] X. Han, F. Y. Chin, H.-F. Ting, G. Zhang, and Y. Zhang. A new upper bound 2.5545 on 2D online bin packing. *ACM Transactions on Algorithms (TALG)*, 7(4):50, 2011.

[40] U. Hoffman. A class of simple stochastic online bin packing algorithms. *Computing*, 29:227–239, 1982.

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

[42] D. S. Johnson. Fast algorithms for bin packing. *J. Comput. Systems Sci.*, 8:272–314, 1974.

[43] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.*, 3:256–278, 1974.

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

[45] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on*, pages 312–320, Nov 1982.

[46] H. Kellerer and V. Kotov. An efficient algorithm for bin stretching. *Operations Research Letters*, 41(4):343–346, 2013.

[47] H. Kellerer, V. Kotov, M. G. Speranza, and Z. Tuza. Semi on-line algorithms for the partition problem. *Operations Research Letters*, 21(5):235–242, 1997.

[48] L. T. Kou and G. Markowsky. Multidimensional bin packing algorithms. *IBM J. Res. Dev.*, 21:443–448, 1977.

[49] K. L. Krause, V. Y. Shen, and H. D. Schwetman. Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. *J. ACM*, 22:522–550, Oct. 1975.

[50] C. C. Lee and D. T. Lee. A simple on-line bin-packing algorithm. *J. ACM*, 32:562–572, July 1985.

[51] F. M. Liang. A lower bound for online bin packing. *Inform. Process. Lett.*, 10:76–79, 1980.

[52] W. Mao. Tight worst-case performance bounds for Next-k-fit bin packing. *SIAM J. Comput.*, 22:46–56, Feb. 1993.

[53] P. Ramanan, D. J. Brown, C. C. Lee, and D. T. Lee. Online bin packing in linear time. *J. Algorithms*, 10:305–326, 1989.

[54] T. Rothvoss. Approximating bin packing within O(log OPT * Log Log OPT) bins. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 20–29, Oct 2013.

[55] S. S. Seiden. On the online bin packing problem. *J. ACM*, 49:640–671, Sept. 2002.

[56] S. S. Seiden and R. van Stee. New bounds for multi-dimensional packing. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 486–495. Society for Industrial and Applied Mathematics, 2002.

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

[58] J. J. Sylvester. On a point in the theory of vulgar fractions. *American Journal of Mathematics*, 3(4):332–335, 1880.

[59] J. Ullman. The performance of a memory allocation algorithm. *Technical Report 100*, 1971.

[60] A. van Vliet. An improved lower bound for online bin packing algorithms. *Inform. Process. Lett.*, 43:277–284, 1992.

[61] G. J. Woeginger. There is no asymptotic PTAS for two-dimensional vector packing. *Information Processing Letters*, 64(6):293–297, 1997.

[62] A. C. C. Yao. New algorithms for bin packing. *J. ACM*, 27:207–227, 1980.

[63] M. Yue and L. Zhang. A simple proof of the inequality $MFFD(l) \leq 71/60\,OPT(l) + 1, l$ for the MFFD bin-packing algorithm. *Acta Mathematicae Applicatae Sinica*, 11(3):318–330, 1995.

List of Figures

List of Tables

List of Abbreviations

- AAF – Almost Any Fit
- AF – Any Fit
- AFPTAS – Asymptotic Fully Polynomial-time Approximation Scheme
- APTAS – Asymptotic Polynomial-time Approximation Scheme
- AWF – Almost Worst Fit
- BF – Best Fit
- BP – Bin Packing
- BPCC – Bin Packing with Cardinality Constraints
- BWBP – Black and White Bin Packing
- CBP – Colored Bin Packing
- FF – First Fit
- PTAS – Polynomial-time Approximation Scheme
- WF – Worst Fit