

# Balanced Allocations: The Heavily Loaded Case with Deletions

Nikhil Bansal\* William Kuszmaul†

## Abstract

In the 2-choice allocation problem,  $m$  balls are placed into  $n$  bins, and each ball must choose between two random bins  $i, j \in [n]$  that it has been assigned to. It has been known for more than two decades, that if each ball follows the GREEDY strategy (i.e., always pick the less-full bin), then the maximum load will be  $m/n + O(\log \log n)$  with high probability in  $n$  (and  $m/n + O(\log m)$  with high probability in  $m$ ). It has remained an open question whether the same bounds hold in the *dynamic* version of the same game, where balls are inserted/deleted with no more than  $m$  balls present at a time.

We show that, somewhat surprisingly, these bounds *do not* hold in the dynamic setting: already on 4 bins, there exists a sequence of insertions/deletions that cause the GREEDY strategy to incur a maximum load of  $m/4 + \Omega(\sqrt{m})$  with probability  $\Omega(1)$ —this is the same bound that one gets in the single-choice allocation model where each ball is assigned to a random bin!

This raises the question of whether *any* 2-choice allocation strategy can offer a strong bound in the dynamic setting. Our second result answers this question in the affirmative: we present a new strategy, called MODULATEDGREEDY, that guarantees a maximum load of  $m/n + O(\log m)$ , at any given moment, with high probability in  $m$ . We also show how to generalize MODULATEDGREEDY to obtain dynamic guarantees for the  $(1 + \beta)$ -choice setting, and for the setting of balls-and-bins on a graph.

Finally, we consider an extension of the dynamic setting in which balls can be *reinserted* after they are deleted, and where the pair  $i, j$  that a given ball uses is consistent across insertions. This seemingly small modification renders tight load balancing impossible: on 4 bins, any balls-and-bins strategy that is oblivious to the specific identities of balls being inserted/deleted *must* allow for a maximum load of  $m/4 + \text{poly}(m)$  at some point in the first  $\text{poly}(m)$  insertions/deletions, with high probability in  $m$ . This is a remarkable departure from the  $m = n$  case where the maximum load of  $O(\log \log n)$  holds independently of whether reinsertions are allowed or not.

---

\*University of Michigan, CS Dept., bansal@gmail.com. Supported in part by the NWO VICI grant 639.023.812.

†MIT CSAIL, kuszmaul@mit.edu. Funded by a Fannie and John Hertz Fellowship and an NSF GRFP Fellowship. This research was also partially sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

# 1 Introduction

Randomized balls-into-bins processes [MRS01, Wie17] serve as a useful abstraction for studying load-balancing problems, with applications such as scheduling, distributed systems, and data structures. The goal is to assign balls (e.g., tasks) to bins (e.g., machines) such that the balls are balanced as evenly as possible across the bins, where each individual ball may have only a few available random options for bins that it can be placed in.

It is well known that, if  $n$  balls are placed into  $n$  bins using the classical SINGLECHOICE rule, where each ball is placed independently in a uniformly random bin, then the maximum load is  $\Theta(\log n / \log \log n)$  with probability  $1 - 1/\text{poly}(n)$ .

**The power of 2-choices.** In a seminal 1994 paper, Azar, Broder, Karlin and Upfal [ABKU94] showed that under a seemingly minor modification, where for each ball *two* bins are chosen independently and uniformly at random, and the ball is placed *greedily* in the least loaded of the two bins, the maximum load reduces to  $\log \log n + O(1)$  with high probability in  $n$ . In the decades since, this *power of 2-choices* paradigm has been extremely influential, with both theoretical (e.g., [PR04, FNP04, BMP<sup>+</sup>06, FMMM09, HMZ11]) and empirical (e.g., [DB13, YYRC08, OWZS13, ORS<sup>+</sup>11, BM01]) applications, and with a large literature on generalizations; see e.g., [MRS01, Wie17] for some excellent surveys.

**The heavily-loaded case.** Azar et al.’s result [ABKU94] prompted researchers to consider the *heavily-loaded case*, where  $m \gg n$  balls are inserted into  $n$  bins. The early techniques that were developed for the lightly-loaded setting (i.e., layered induction [ABKU94], witness trees [Vöc99, CFM<sup>+</sup>98], and differential-equation approaches [Mit01, Mit99]) struggled to deliver strong bounds in the heavily-loaded setting, and for several years the best known bound stood at  $m/n + \log \log n + O(m/n)$  [CFM<sup>+</sup>98, Vöc99]. If we define the *overload* to be the amount by which the maximum load exceeds  $m/n$ , then this bound allows for an overload as large as  $\log \log n + O(m/n)$ —such a bound is useful if  $m \approx n$ , but when  $m \gg n \log n$ , the bound becomes worse even than the standard bound offered by SINGLECHOICE (i.e., an overload of  $O(\sqrt{(m/n) \log n})$ ).

In a breakthrough result, Berenbrink, Czumaj, Steger and Vöcking [BCSV00] showed how to use Markov-chain techniques to obtain a much stronger bound of  $\log \log n + O(1)$  on the overload, with probability  $1 - 1/\text{poly}(n)$ . Thus, somewhat remarkably, the gap between the maximum and average loads in the heavily-loaded case *is the same* as in the lightly-loaded case, with high probability in  $n$ .

When  $m \gg n$ , the  $O(\log \log n)$  overload bound does not, in general, extend to hold with probability  $1 - 1/\text{poly}(m)$  (i.e., w.h.p. in the number of *balls*). However, the known techniques can be used to achieve a quite strong (and, when  $n = O(1)$ , optimal) bound of  $O(\log m)$  on the overload in this case.

**The dynamic setting.** In typical load-balancing and data-structures applications, however, the items can be both inserted and deleted dynamically over time. Here two natural models have been studied: (i) the *insertion/deletion* model in which each insertion involves a new ball with independent random bin choices, and (ii) the *reinsertion/deletion* model in which a ball can be *reinserted* after being deleted, and has the same two random bin choices each time it is reinserted. Although these two models may seem quite similar at first glance, we shall see later that the distinction is significant.

Note that, whereas in the insertion-only setting,  $m$  is set to be the total number of insertions, in the dynamic setting,  $m$  is set to be an *upper bound* on the number of balls that are present at any given moment (and the sequence of insertions/deletions may be infinite). The objective is to minimize the *overload*, which

is now defined as the amount by which the maximum load exceeds  $m/n$  at any given moment.<sup>1</sup>

Azar et al. [ABKU94] considered the insertion/deletion model with  $m = n$  and with *random deletions*: that is,  $n$  balls are inserted initially, and then there is an infinite sequence of alternating insertions/deletions, where each deletion removes a *random* ball. They showed that, at any given moment, the GREEDY strategy achieves a maximum load of  $\log \log n + O(1)$ , with high probability in  $n$ .

Subsequent work has considered the more general setting where the insertions/deletions are determined by an *oblivious-adversary* (i.e., an adversary that does not know the random choices of the algorithm), and where the only constraint on the adversary is that the number of balls in the system can never exceed  $m$ . Using the witness tree technique, first introduced by [CMadH<sup>+</sup>98], Cole et al. [CFM<sup>+</sup>98] analyzed the reinsertion/deletion model with  $m = n$ , and established that the GREEDY strategy guarantees a maximum load of  $O(\log \log n)$  with high probability in  $n$ . Later, Vöcking [Vöc99] improved this to  $\log \log n + O(1)$ , which remarkably, matches the bound in the non-dynamic (insertion-only) case up to an additive  $O(1)$  term.

**What about the dynamic heavily-loaded case?** For more than two decades, it has remained an open question what the optimal bounds are in the *heavily-loaded case* if we wish to support both insertions and deletions performed by an oblivious adversary. Besides obvious theoretical interest, the question also arises naturally in practice—for example, as a scheduling problem in which jobs arrive and depart over time, the number of jobs (balls) at any moment is much larger than the number  $n$  of machines (bins), and the only guarantee on the arrivals/departures of jobs is an upperbound  $m/n$  on the average load at any moment.

The dynamic heavily-loaded setting was studied by Cole et al. [CFM<sup>+</sup>98] and Vöcking [Vöc99, Vöc03], who showed that GREEDY has overload  $\log \log n + O(m/n)$  with high probability in  $n$ . But again this bound is already worse for  $m \gg n \log n$  than the  $O(\sqrt{(m/n) \log n})$  overload bound for SINGLECHOICE (which also holds in the dynamic setting).

However, it is widely believed that GREEDY should also achieve similar bounds in the dynamic heavily-loaded case as in the non-dynamic heavily-loaded case (i.e., an overload of  $O(\log \log n)$  and  $O(\log m)$ , w.h.p. in  $n$  and  $m$ , respectively). The current limitation would seem to be a technical one: the witness-tree techniques that allow for us to analyze dynamic games with oblivious adversaries [CFM<sup>+</sup>98, Vöc03] are incompatible with the techniques (i.e., Markov-chain [BCSV00] and potential-function [PTW10a, LSS22, TW14] arguments) that achieve strong bounds in the heavily-loaded case.

In this work we prove new upper and lower bounds for the dynamic heavily-loaded case. We split our results into two parts, the first of which considers the insertion/deletion model, and the second of which considers the reinsertion/deletion model.

## 1.1 Results in the Insertion/Deletion Model

We begin by considering the insertion/deletion model, that is, an oblivious adversary performs an arbitrary sequence of insertions/deletions subject only to the constraint that no more than  $m$  balls are present at a time.

**A lower bound for GREEDY.** We show that, somewhat surprisingly, the GREEDY strategy actually *does not* offer strong bounds in the dynamic heavily-loaded setting. In particular, already for  $n = 4$  bins, there exists an oblivious sequence of insertions/deletions after which there is a maximum load of

$$m/n + \Omega(\sqrt{m})$$

---

<sup>1</sup>It is tempting to define the overload to be the amount by which the maximum load exceeds  $m(t)/n$ , where  $m(t)$  is the number of balls present at time  $t$ . However, the following (folklore) example demonstrates the flaw with such a definition: Suppose we insert  $m$  balls (using an arbitrary insertion strategy), and then we delete a random  $m/2$  of those balls. Since the  $m/2$  deletions are random, even if the system was perfectly balanced after the initial  $m$  insertions, the bin loads will typically be  $m/2n \pm \sqrt{m/2n}$ , and the maximum load will be  $m(t)/n + \tilde{O}(\sqrt{m/n})$ , which is no better than the bound trivially achieved by SINGLECHOICE.

with probability  $\Omega(1)$ . In other words, the GREEDY strategy is no better than SINGLECHOICE in this setting!

Our result represents a remarkable departure from the lightly-loaded  $m = n$  case, where GREEDY achieves an optimal bound of  $O(\log \log n)$  (even in the reinsertion/deletion model). The result also offers an explanation for why all previous attempts [CFM<sup>+</sup>98, Vöc03] to analyze GREEDY for large  $m$  have yielded only relatively weak bounds.

The high-level intuition behind our lower bound is as follows. Using GREEDY, if some bin  $i$  contains far fewer balls than the other bins, then there will be a contiguous time window during which all of the insertions are maximally biased towards bin  $i$ . But this means that, later on, the adversary can perform a sequence of deletions in which the balls being *deleted* exhibit a strong bias towards being from bin  $i$ . In other words, the biases that GREEDY exhibits during insertions can be thrown back at it by future deletions.

We present the full construction in Section 3. As a warmup, we first show a simpler (but already non-trivial) lower bound of  $m/n + \Omega(m^{1/4})$  for  $n = 4$  bins in Section 3.1, and then give the full lower bound of  $m/n + \Omega(m^{1/2})$  in Section 3.2. For ease of exposition we mostly focus on the case of  $n = 4$  — however, we also show how to use our techniques to obtain a lower bound of  $m/n + m^{1/4}/\text{poly}(n)$  for general  $n$ .

**The MODULATEDGREEDY algorithm.** Of course, the above phenomenon is not isolated to the GREEDY strategy. Any strategy that exhibits biases between bins is at risk of having those biases thrown back at it via future deletions. This raises a natural question: is it possible for *any 2-choice allocation strategy* to beat the bounds trivially achieved in the single-choice model?

Our second result is a new algorithm called MODULATEDGREEDY, in the insertion/deletion model, that at any time, with high probability in  $m$ , achieves a maximum load of

$$m/n + O(\log m).$$

This bound is optimal for any strategy that achieves high-probability bounds in  $m$  (see Section 2.3).

Given the choice between two bins  $i$  and  $j$ , the MODULATEDGREEDY algorithm chooses between the bins probabilistically, based on how their loads compare. In particular, it carefully modulates its biases between bins so that the adversary is unable to find any non-trivial correlations between how balls are inserted. Interestingly, the structure of MODULATEDGREEDY also allows for a direct combinatorial analysis, which proceeds by coupling the behavior of MODULATEDGREEDY to a seemingly different (and much simpler) randomized process that we call the *stone game*.

**Generalizations.** Our analysis of MODULATEDGREEDY extends to support a number of generalizations and applications. This includes a tight bound of  $m/n + O(\beta^{-1} \log m)$  for the  $(1 + \beta)$ -choice version of the game [PTW10b], where a  $(1 - \beta)$ -fraction of the balls are inserted using SINGLECHOICE and only a  $\beta$ -fraction of the balls get two choices; a bound of  $m/n + \text{polylog } m$  for the dynamic balls-and-bins game on an undirected well-connected regular graphs [BF22, KP06]; and a bound of  $m/n + O(\log M)$  for the setting in which  $m$  is permitted to increase over time, subject only to the constraint that  $m \leq M$ . In all of these settings, the previous states of the art were restricted to the insertion-only model.

To describe the main ideas as clearly as possible, we describe these results in two parts. In Section 2 we consider a simpler version of MODULATEDGREEDY that guarantees the  $m/n + O(\log m)$  bound for insertion/deletion sequences of  $\text{poly}(m)$  length. Later, in Section 5, we consider the general setting with unbounded request sequences and where  $m$  can increase over time. The extensions to the  $(1 + \beta)$ -choice and the graphical 2-choice processes are described in Section 5.3.

## 1.2 An Impossibility Result for the Reinsertion/Deletion Model

Finally, in Section 4, we turn our attention to the reinsertion/deletion model. That is, the adversary can perform an arbitrary sequence of insertions, deletions, and reinsertions (as long as the ball being reinserted is not currently present) subject only to the constraint that no more than  $m$  balls are present at a time.

Here we establish an impossibility result. Consider any 2-choice bin-allocation strategy that is *oblivious* to the specific *identities* of balls (i.e., when a ball is inserted, all that the strategy gets to see is the pair  $i, j$  of bins that the ball is assigned to). We show that, against any such strategy, it is possible for an oblivious adversary to force a maximum load of  $m/4 + \text{poly}(m)$  at some point in the first  $\text{poly}(m)$  insertions/deletions, with high probability in  $m$ .

This result reveals a fundamental (and perhaps unexpected) gap between the insertion/deletion model and the reinsertion/deletion model. In particular, in the lightly-loaded setting with deletions where  $m \leq n$ , both models yield the same  $O(\log \log n)$  bounds even for infinite sequences of reinsertions/deletions [CFM<sup>+</sup>98, Vöc03]. But, in the heavily-loaded setting, the cyclic dependencies that are introduced by reinsertions (i.e., a ball  $x$  being reinserted is being placed into a system whose state has *already* been affected by  $x$ 's bin choices in the past) end up being lethal to any ID-oblivious allocation strategy.

## 1.3 Other Related Work

Beyond research on the heavily-loaded and dynamic settings, there has been a large body of work on other ways to extend the 2-choice allocation framework—because the literature on this subject is so extensive, we give only a brief overview here. These extensions have included work on restricted classes of insertion strategies (e.g.,  $(1 + \beta)$ -choice strategies [PTW10b, PTW10c], thinning strategies [LSS22, FGG21, LS22], strategies with limited information [LS22], etc.), on balls with nonuniform sizes [TW14, BFHM08, PTW10c, TW07], on parallel settings in which balls arrive in batches [Ste96, LPY19, BCE<sup>+</sup>12, BFK<sup>+</sup>16, BFK<sup>+</sup>18], on settings in which bins correspond to vertices on a graph [BF22, KP06], on settings where balls can be relocated after insertion [AKT21, BFCKK22], etc. Another notable extension is Vöcking's asymmetric  $d$ -choice paradigm [Vöc03] which, in the lightly-loaded setting, chooses between  $d$  bins on each insertion to achieve a maximum load of  $O((\log \log n)/d)$ .

Another line of work, related to the current work on the dynamic setting, is on queuing models [Mit01, VDK96, MBVLW18, BLP10, LM06, BL12, EG16, LN05], where insertions and deletions are *stochastic*. Many of these focus on the so-called supermarket model, introduced by [Mit01, VDK96], in which customers (i.e., balls) arrive in a Poisson stream of rate  $\lambda n$ ,  $\lambda < 1$ , and are processed within each queue (i.e., bin) in FIFO order, where each customer requires processing time that is exponentially distributed with mean 1. In the case where  $\lambda$  is allowed to go to 1 (see, e.g., [BL12, EG16]), the number of balls in the system can become  $\omega(n)$  (this is analogous to the heavy case in standard balls and bins). However, because insertions/deletions are assumed to be stochastic, the analyses (and the flavors of the results) take a very different form than those in this paper (where deletions are performed by an oblivious adversary, and the number of balls in the system is deterministically bounded by a parameter  $m$ ).

In addition to the past work described above, there have also been recent efforts within the succinct-data-structure literature to obtain stronger bounds for the reinsertion/deletion model in specialized regimes, resulting in a 3-choice allocation scheme that achieves a bound of  $m/n + O(\log \log n) + O(\sqrt{m/n} \cdot \sqrt{\log(m/n)})$  on the maximum load at any given moment [BCFC<sup>+</sup>21b, BCFC<sup>+</sup>21a]. This bound is useful when  $m \leq O(n \log n)$ , but does not improve significantly on SINGLECHOICE when  $m \gg n$ .

## 1.4 Preliminaries

In the *dynamic 2-choice allocation problem*, an oblivious adversary performs a sequence of ball insertions and deletions subject to the constraint that the number of balls in the system can never exceed  $m$ . Whenever a ball  $x$  is inserted, a uniformly random pair  $h(x) = (h_1(x), h_2(x)) \in [n] \times [n]$  of distinct bins is selected, and the *insertion strategy* must choose which of the bins  $h_1(x)$  or  $h_2(x)$  the ball will be placed in. The pair  $h(x)$  is sometimes referred to as the *hash* of the ball  $x$ .

There are two models that we will consider for insertions and deletions. In the *insertion/deletion model*, each insertion  $\text{INSERT}(x)$  places a new ball  $x$  into the system that has never been present before. In the *reinsertion/deletion model*, each insertion  $\text{INSERT}(x)$  places a ball  $x$  into the system that is not *currently* present, but that may have been present in the past (each time  $x$  is inserted, its bin pair  $h(x)$  stays the same). In both models, the  $\text{DELETE}(x)$  operation selects a ball  $x$  that is currently present and removes it.

We are interested in bounding the maximum *load* (i.e., the number of balls) of any bin. Our algorithms will offer guarantees with high probability (w.h.p.) in  $m$ , meaning that the failure probability is  $1/\text{poly}(m)$  for a polynomial of our choice. Two basic insertion strategies that we will discuss frequently are **GREEDY**, which always selects the least full of the bins  $h_1(x), h_2(x)$ , and **SINGLECHOICE**, which always selects bin  $h_1(x)$ .

In our lower bound for the reinsertion/deletion model (Section 4), we will study the class of *ID-oblivious* insertion strategies—such a strategy makes each insertion decision based on the hash  $h(x)$  of the ball being inserted, rather than based on the specific identity  $x$  of the ball. Formally, an ID-oblivious strategy is one that can be implemented with operations  $\text{INSERT}(h_1(x), h_2(x))$  (indicating the pair of bins for the ball being inserted) and  $\text{DELETE}(r)$  (indicating a deletion of the  $r$ -th-most-recently-inserted ball of those present).

Finally, although  $h(x) = (h_1(x), h_2(x))$  is a uniformly random pair of *distinct* bins, any strategy in the insertion/deletion model can choose to view  $h(x)$  as a pair of *independent bins* by artificially resetting  $h_2(x) = h_1(x)$  with probability  $1/n$ . The strategies that we design in this paper will assume (without loss of generality) that they are given a uniformly random pair of (not necessarily distinct) bins for each insertion.

## 2 MODULATEDGREEDY: Handling $\text{poly}(m)$ Insertions/Deletions

In this section, we consider the insertion/deletion model, with  $n$  bins and up to  $m$  balls present at a time, and we describe an insertion strategy, called **MODULATEDGREEDY**, that achieves a strong bound on maximum load. Here, we describe the simplest possible version of the strategy, which supports any sequence of  $\text{poly}(m)$  insertions/deletions while guaranteeing a maximum load of  $m/n + O(\log m)$  with high probability in  $m$ . Later, in Section 5, we will extend **MODULATEDGREEDY** in various ways, such as supporting an infinite sequence of insertions/deletions, allowing  $m$  to increase over time, etc.

The main result of the section is the following:

**Theorem 1.** *Let  $m \geq n$ . Consider the insertion/deletion model with  $n$  bins and an upper bound of at most  $m$  balls present at a time. Consider a sequence of  $\text{poly}(m)$  insertions/deletions, where insertions are implemented using **MODULATEDGREEDY**. With high probability in  $m$ , **MODULATEDGREEDY** does not halt during any of the insertions/deletions, and no bin ever has load more than  $m/n + O(\log m)$ .*

When we describe the lower bound for **GREEDY** in Section 3, we will see that the main problem with **GREEDY** is that it is too aggressive. Given the choice between two bins  $i, j$ , as **GREEDY** always chooses the less loaded of the two—this creates correlations between balls that can be exploited to construct a bad sequence of insertions/deletions. In contrast, **MODULATEDGREEDY** will try to be as *unaggressive* as possible, while still guaranteeing an upper gap of  $O(\log m)$ . In particular, it carefully modulates its behavior

and only exhibits a strong bias between two bins  $i$  and  $j$  if (1) the two bins  $i$  and  $j$  have significantly different loads; and (2) the system is nearly saturated (i.e., there are nearly  $m$  balls present).

As we shall see, this modulated behavior also allows for a simple (but clever) combinatorial analysis, marking a departure from the (typically quite involved) potential-function and Markov-chain arguments used in past analyses of the heavily-loaded case.

## 2.1 The Algorithm

The MODULATEDGREEDY algorithm for allocating a bin to a ball is given below. We assume without loss of generality that  $m$  is a multiple of  $n$ .

---

**Algorithm 1** The MODULATEDGREEDY insertion strategy. Here,  $\ell_k$  is the number of balls in bin  $k$  prior to the insertion, and  $c$  is a large positive constant.

---

**procedure** MODULATEDGREEDY

    Select two bins  $i, j \in [n]$  independently and uniformly at random.

    Set  $T = m/n + c \log m - \sum_r \ell_r/n$ .

**if**  $(\max_k \ell_k) - (\min_k \ell_k) \leq T$  **then**

        Assign the ball to bin  $i$  with probability  $1/2 + \frac{\ell_j - \ell_i}{2T}$ , and otherwise assign it to bin  $j$ .

**else**

        Halt.

---

For  $k \in [n]$ , let  $\ell_k$  denote the load on bin  $k$  prior to the insertion, let  $\bar{\ell} = \sum_k \ell_k/n$  be the average bin load, and  $c$  be a (sufficiently large) fixed constant. When choosing between two bins  $i, j$ , the algorithm exhibits bias

$$(\ell_j - \ell_i)/2T$$

towards bin  $i$ , where

$$T = m/n + c \log m - \bar{\ell}.$$

Note that the algorithm is well-defined as long as  $|\ell_j - \ell_i| \leq T$  for all  $i, j \in [n]$ . One should think of  $T$  as representing the average amount of leftover space that each bin would have if each bin had a total capacity of  $m/n + c \log m$  balls. This means that the bias is proportional to the difference  $\ell_j - \ell_i$  between the loads of the bins, and is inversely proportional to the average amount  $T$  of space left in each bin.

The following lemma gives a closed-form solution for the probability of a given bin  $k$  being selected by MODULATEDGREEDY.

**Lemma 2.** Suppose that  $|\ell_i - \ell_j| \leq T$  for all bins  $i, j$ . Consider a bin  $k$ , and set  $T_k = m/n + c \log m - \ell_k$ . Upon an insertion, a bin  $k$  is selected with probability  $T_k/(nT) = T_k/(\sum_i T_i)$ .

*Proof.* Let  $i, j$  denote the random bin choices for the ball being inserted. The probability that a given bin  $k$  is selected is given by

$$\begin{aligned} & \Pr[i, j = k] + \sum_{s \neq k} \Pr[i = k, j = s] \left( \frac{1}{2} + \frac{\ell_s - \ell_k}{2T} \right) + \sum_{s \neq k} \Pr[i = s, j = k] \left( \frac{1}{2} + \frac{\ell_s - \ell_k}{2T} \right) \\ &= \frac{1}{n^2} + \frac{2}{n^2} \sum_{s \neq k} \left( \frac{1}{2} + \frac{\ell_s - \ell_k}{2T} \right) = \frac{2}{n^2} \sum_{s=1}^n \left( \frac{1}{2} + \frac{\ell_s - \ell_k}{2T} \right) \\ &= \frac{2}{n} \left( \frac{1}{2} + \frac{\bar{\ell} - \ell_k}{2T} \right) = \frac{T + \bar{\ell} - \ell_k}{nT} = \frac{T_k}{nT}. \end{aligned}$$

Finally we note that  $\sum_{i=1}^n T_i = \sum_{i=1}^n (m/n + c \log m - \ell_i) = m + nc \log m - n\bar{\ell} = nT$ . □

## 2.2 Analysis

To analyze MODULATEDGREEDY, we begin by describing a seemingly different process (which we call the stone game) that, by design, yields to a simple combinatorial analysis. We then show that the MODULATEDGREEDY algorithm and the stone game can be *coupled together* so that bounds on the behavior of the stone game directly imply bounds on the behavior of MODULATEDGREEDY.

**Stone Game.** In the  $(Q, n)$ -stone game, parameterized by  $Q$  and  $n$ , there are  $Qn$  stones which are distributed among two bags; an *inactive bag* and an *active bag*. Initially the active bag is empty, and all the stones are in the inactive bag.

The game supports two types of operations: the  $\text{ACTIVATE}()$  operation moves a *random* stone from the inactive bag to the active bag; and the  $\text{DEACTIVATE}(r)$  operation examines the stones in the active bag, selects the stone that was added the  $r$ -th most recently, and moves it back to the inactive bag. ( $\text{ACTIVATE}()$  can only be called if the inactive bag is non-empty, and  $\text{DEACTIVATE}(r)$  can only be called if the active bag contains  $r$  or more balls). The sequence of operations is generated by an oblivious adversary, independent of the random bits used by the game.

The stones are labeled  $x_{k,q}$  for  $k \in [n], q \in [Q]$ . We call  $k$  the *color* of the stone, so that there are  $Q$  stones of each color. However, the labels of the stone should be thought of as *hidden*, since the behaviors of  $\text{ACTIVATE}()$  and  $\text{DEACTIVATE}(r)$  do not depend on the labels of the stones.

We will now prove some lemmas establishing that the stone game is, by design, very well behaved. Our first lemma shows that, even though the adversary gets to perform activations/deactivations, it has no control over which specific stones are in the active bag.

**Lemma 3.** *At any given moment, if the active/inactive bag contains  $s$  stones, then these stones are a uniformly random subset of size  $s$  of the stones  $\{x_{k,q}\}_{k \in [n], q \in [Q]}$ .*

*Proof.* The point is that the activation/deactivation operations do not depend on the labels of the balls.

Formally, fix any sequence of activations/deactivations and the random choices of the  $\text{ACTIVATE}()$  operations, and let  $S$  be set of stones currently in the inactive bag (the argument for the active bag is identical). Then for any run of the game with a random permutation  $\pi$  applied to the  $Qn$  labels  $\{x_{k,q}\}_{k \in [n], q \in [Q]}$ , the set stones in the active bag will be  $\pi(S)$ . Thus, if the inactive bag contains  $s$  stones, every  $s$ -element subset of the  $nQ$  stones is equally likely.  $\square$

This implies that as long as the inactive bag contains a reasonably large number of stones (namely,  $\Omega(n \log(nQ))$ ), each color is guaranteed to have roughly equal representation in the bag.

**Lemma 4.** *Suppose at some given moment, the inactive bag contains  $s \geq cn \log(nQ)$  stones, for some large enough constant  $c$ . Let  $s_k$  be the number of these stones with color  $k$ . Then  $s_k \in [s/2n, 3s/2n]$  for each  $k \in [n]$ , with probability at least  $1 - 1/(Qn)^{\Omega(c)}$ .*

*Proof.* By Lemma 3, the balls  $S$  in the inactive bag are a random subset of size  $s$  of the  $Qn$  balls  $\{x_{k,q}\}$ . Let  $X_k = \{x_{k,1}, \dots, x_{k,Q}\}$  be the set of all color- $k$  balls. Then  $s_k = |X_k \cap S|$ , the number  $s_k$  of balls of color  $k$  in  $S$ , has the hypergeometric distribution  $H(Qn, Q, s)$ .

As the standard tail bounds on sampling without replacement at least as sharp as those given by Chernoff bounds for sampling with replacement [FK15] (Section 22.5), and as  $\mathbb{E}[s_i] = s/n$ , we get that

$$\Pr[|s_k - s/n| \geq \epsilon s/n] \leq 2 \exp(-\epsilon^2 s/3n). \quad (1)$$

Setting  $\epsilon = 1/2$ , and taking a union bound over the  $n$  colors, gives that  $s_k \in [s/2n, 3s/2n]$  for each  $k \in [n]$  with probability  $1 - 2n \exp(-\Omega(c \log Qn))$  which is  $1 - 1/(Qn)^{\Omega(c)}$  for large enough  $c$ .  $\square$



### 2.2.1 Relating the stone game to the balls-and-bins game

One can think of the stones in the stone game as being similar to balls in the balls-and-bins game—the active bag represents the set of balls that are present, the color of a stone dictates which “bin” a given ball is in, and activations/deactivations correspond to insertions/deletions.

However, there are several significant differences between the games. Notably, the whole point of the balls-and-bins game is to ensure that no single bin contains too many balls, but in the stone game, the active bag trivially (and deterministically) has at most  $Q$  stones of any given color. Nonetheless, we shall now see how to couple the two games together in such a way that our analysis of the stone game yields a bound for the balls-and-bins game.

**Mapping between instances.** We first giving a mapping between the sequence of insertions/deletions for balls-and-bins game and the input sequence for the stone game. For any sequence  $\mathcal{S}$  of insertions/deletions in balls-and-bins game, define  $\phi(\mathcal{S})$  to be a corresponding sequence of activations/deactivations, where each INSERT operation is replaced with an ALLOCATE operation, and where each DELETE( $x$ ) operation on a ball  $x$  is replaced with a DEACTIVATE( $r$ ) operation, where  $r - 1$  is the number of balls in the system that were inserted after  $x$ .

The following key lemma shows that the random choices in the two games can be coupled.

**Lemma 5** (Coupling). *Let  $n \leq m$  and let  $\Delta = c \log m$ , where  $c$  is the positive constant used by MODULATEDGREEDY. Consider a sequence  $\mathcal{S}$  of insertions/deletions in a balls-and-bins game on  $n$  bins, where there are never more than  $m$  balls present at a time. Let  $G_1$  be a balls-and-bins game with operation-sequence  $\mathcal{S}$  and let  $G_2$  be  $(Q, n)$ -stone game with  $Q = m/n + \Delta$  with operation sequence  $\phi(\mathcal{S})$ .*

*If  $G_1$  is implemented using MODULATEDGREEDY, then there exists a coupling between  $G_1$  and  $G_2$  with the following property: Up until MODULATEDGREEDY halts, the number of balls in a given bin  $k$  (in the balls-and-bins game) always equals the number of stones in the active bag with color  $k$  (in the stone game).*

*Proof.* Let  $\ell_1, \ell_2, \dots, \ell_n$  denote the loads of the bins at any given moment. By Lemma 2, we know that, on any given insertion in which MODULATEDGREEDY does not halt, each bin  $k$  is selected with probability

$$\frac{T_k}{nT} = \frac{T_k}{\sum_{i=1}^n T_i}. \quad (2)$$

Now suppose that, for each color  $k$  there are  $\ell_k$  stones with color  $k$  in the active bag (and hence  $Q - \ell_k$  such stones in the inactive bag) of the stone game. Then on any given activation, the probability of a ball with color  $k$  being moved into the active bag is

$$\frac{Q - \ell_k}{nQ - \sum_{i=1}^n \ell_i} = \frac{m/n + \Delta - \ell_k}{m + n\Delta - \sum_i \ell_i} = \frac{T_k}{\sum_{i=1}^n T_i}, \quad (3)$$

where the first equality uses that  $Q = m/n + \Delta$ . The two probabilities (2) and (3) are precisely equal. Thus, we can couple the games so that the bin selected by the insertion in the balls-and-bins game is the same as the stone color selected by the activation in the stone game.

If we implement the insertions/activations in this way, then the deletions/deactivations also become coupled: whenever a ball is deleted from a bin  $k$ , a stone with color  $k$  is removed from the active bag (in particular, the ball and stone were assigned to have the same bin/color when they were inserted/activated previously). Thus the proof of the lemma is complete.  $\square$

**Proof of Theorem 1.** Finally, we can use the coupling in Lemma 5 to bound the probability of MODULATEDGREEDY halting and prove Theorem 1.

*Proof.* (Theorem 1) Observe that, if MODULATEDGREEDY does not halt, then deterministically there are at most  $m/n + O(\log m)$  balls in any given bin. In particular, the condition  $\max_k \ell_k - \min_k \ell_k \leq T$  implies that  $\max_k \ell_k - \bar{\ell} \leq T$ . Plugging  $T = m/n + c \log m - \bar{\ell}$ , this gives that  $\max_k \ell_k \leq m/n + c \log m$ .

Thus, it suffices to analyze the probability of halting.

By Lemma 5, up until MODULATEDGREEDY halts, it can be coupled to a stone game on  $nQ = m + nc \log m$  balls, where the number of balls in the active bag never exceeds  $m$ . Under this coupling, the number of balls  $\ell_k$  in bin  $k$  satisfies  $\ell_k = Q - s_k$ , where  $s_k$  is the number of color- $k$  stones in the inactive bag.

The MODULATEDGREEDY algorithm halts only if

$$|\ell_i - \ell_j| > T = m/n + c \log m - \bar{\ell} = Q - \bar{\ell} \quad (4)$$

for some pair  $i, j$  of bins. For the stone game, denoting  $s = \sum_k s_k = \sum_k (Q - \ell_k) = n(Q - \bar{\ell})$ , and as  $|s_i - s_j| = |\ell_i - \ell_j|$ , condition (4) is equivalent to

$$|s_i - s_j| > s/n.$$

But we know by Lemma 4 that, w.h.p. in  $m$ , we have  $|s_i - s_j| \leq s/n$  at all times during the stone game (since the number of balls in the inactive bag is always at least  $nc \log m$ ). Thus, we have w.h.p. in  $m$  that MODULATEDGREEDY never halts.  $\square$

### 2.3 Tightness of the Bound

Clearly, the bound of  $m/n + O(\log m)$  is not optimal for all parameter regimes, since it is known that GREEDY achieves maximum load  $O(\log \log n)$  in the regime of  $n = m$ . We remark, however, that for parameter regimes where  $m$  is much larger than  $n$ , or when  $n$  is fixed, this bound is essentially optimal.

**Proposition 6.** *Consider  $m$  insertions into 4 bins using any sequential 2-choice insertion strategy. With probability at least  $1/\text{poly}(m)$ , some bin contains at least  $m/4 + \Omega(\log m)$  balls. More generally for  $n$  bins, some bin contains at least  $m/n + \Omega(\log_n m)$ .*

*Proof.* Let us consider the final  $\log m$  insertions  $x_1, \dots, x_{\log m}$ . Suppose, without loss of generality, that prior to those insertions being performed, bins 1, 2 contain at least as many total balls as bins 3, 4. With probability  $1/\text{poly}(m)$ , all of the insertions  $x_1, \dots, x_{\log m}$  are forced to choose between bins 1 and 2. No matter how they are assigned, this forces at least one of bins 1, 2 to have load  $m/4 + \Omega(\log m)$  balls at the end of the insertions.

Deleting all the balls, and repeating the instance again  $\text{poly}(m)$  times, this event will occur for one of the instances with high probability.

The same argument also implies a  $m/n + \Omega(\log_n m)$  lower bound; consider the final  $k = \log_n m$  balls, and note that with probability  $\Omega(n^{-2k}) = 1/\text{poly}(m)$  the only bin choices for these balls are 1 and 2.  $\square$

## 3 A Lower Bound for Greedy with Deletions

This section gives a lower bound for the GREEDY algorithm in the insertion/deletion model against an oblivious adversary, with up to  $m$  balls present at a time. Recall that the trivial SINGLECHOICE strategy achieves an overload of  $O(\sqrt{(m/n) \log n})$  (w.h.p. in  $m$ ) in this setting, so the natural question is whether GREEDY does any better. We show that, even for  $n = 4$  (meaning that SINGLECHOICE has an overload of  $O(\sqrt{n})$ ), it does not.

**Theorem 7.** *Consider the insertion/deletion model on  $n = 4$  bins, with the restriction that at most  $m$  balls can be present at any time, and suppose that insertions are implemented using GREEDY. There exists an oblivious sequence of  $\text{poly}(m)$  insertions/deletions such that, after the sequence is complete, we have with probability  $\Omega(1)$  that some bin contains  $m/4 + \Omega(\sqrt{m})$  balls.*

For ease of exposition, and to keep the main ideas as clear as possible, we focus our lower bound on  $n = 4$  bins. We will also see, however, that for general  $n$  and  $m$ , a similar construction gives an  $m/n + \Omega(\sqrt{m}/\text{poly}(n))$  lower bound on the maximum load.

### 3.1 A Simpler $\Omega(m^{1/4})$ Bound

Before proving Theorem 7, we first describe a simpler (but already surprisingly) lower bound of  $m/4 + \Omega(m^{1/4})$ . Later in Section 3.2 we build on these ideas to prove Theorem 7.

We first describe a construction with the property that if we ever reach a state where one of the bins (say, bin 1) contains significantly fewer balls (say,  $k$  fewer balls) than the other bins, then we can subsequently reach a state in which (with probability  $\Omega(1)$ ), some bin contains at least  $m/4 + \Omega(\sqrt{k})$  balls. As we shall see later in the subsection, this can be used to directly obtain the  $m/4 + \Omega(m^{1/4})$  bound.

**Proposition 8** (Gap to overload). *Consider the GREEDY algorithm on 4 bins, on instances where at most  $m$  balls can be present at a time. Suppose we begin in a state that contains at most  $m - k$  balls, and where bin 1 contains  $k + 1$  fewer balls than each of bins 2, 3, 4. Then there is an oblivious sequence of  $O(m)$  insertions/deletions such that, after the sequence is complete, we have the following property with probability  $\Omega(1)$ : some bin contains  $m/4 + \Omega(\sqrt{k})$  balls.*

*Proof.* Let  $X_0$  denote the initial state of the game. Consider the sequence with the following three steps.

1. Insert  $k$  balls  $x_1, x_2, \dots, x_k$  to get to a state  $X_1$ .
2. Then insert  $m - j$  balls  $y_1, y_2, \dots, y_{m-j}$ , where  $j$  is the number of balls in state  $X_1$ —this brings us to a state  $X_2$  with  $m$  balls in total.
3. Finally, delete the balls  $x_1, x_2, \dots, x_k$ , and insert new balls  $z_1, z_2, \dots, z_k$  to reach a state  $X_3$ .

We claim that, for at least one of the two states  $X_2$  and  $X_3$ , we have with probability  $\Omega(1)$  that some bin contains  $m/4 + \Omega(\sqrt{k})$  balls.

During the insertions of  $x_1, x_2, \dots, x_k$ , we are always in a state where bin 1 contains fewer balls than bins 2, 3, 4. Thus, each insertion  $x_i$  will go into bin 1 if and only if  $1 \in \{h_1(x_i), h_2(x_i)\}$  (this is where we are exploiting that the GREEDY algorithm is too aggressive). The number  $A$  of balls  $x_1, x_2, \dots, x_k$  that are placed in bin 1 is therefore given by

$$A = |\{i \mid 1 \in \{h_1(x_i), h_2(x_i)\}\}|.$$

Let  $\mu = \mathbb{E}[A]$ . As  $A$  is a binomial random variable with mean  $\mu = \Theta(k)$ , with probability  $\Omega(1)$  we have

$$A \geq \mu + \Omega(\sqrt{k}).$$

Now consider the number  $B$  of balls  $z_1, z_2, \dots, z_k$  that are placed into bin 1. We deterministically have that

$$B \leq |\{i \mid 1 \in \{h_1(z_i), h_2(z_i)\}\}|. \tag{5}$$

Since the right side of (5) is a binomial random variable with mean  $\mu$ , we have with probability  $\Omega(1)$  that

$$B \leq \mu.$$

Moreover, since  $A$  and  $B$  are independent, the above bounds on  $A$  and  $B$  hold simultaneously with probability  $\Omega(1)$ .

Finally, let us consider the number of balls in bins 2, 3, 4 once we reach state  $X_3$ . Assume that state  $X_2$  has maximum load  $m/4 + o(\sqrt{k})$ , otherwise we are already done. Then, since  $X_2$  contains  $m$  balls in total, bins 2, 3, 4 must contain a total of at least  $3m/4 - o(\sqrt{k})$  balls. By step 3 of the input sequence above, it follows that, in state  $X_3$ , the total number of balls in bins 2, 3, 4 is at least

$$3m/4 - o(\sqrt{k}) - (k - A) + (k - B).$$

Conditioning on the event above, and plugging in our bounds for  $A$  and  $B$ , we see that (with probability  $\Omega(1)$ ) this is at least  $3m/4 + \Omega(\sqrt{k})$ . Thus, at least one of bins 2, 3, 4 must contain  $m/4 + \Omega(\sqrt{k})$  balls, as desired.  $\square$

**The lower bound.** Using Proposition 8, the claimed lower bound follows quite easily. Consider the following input sequence, starting from an empty system. (1) Insert  $m$  balls into the system; (2) delete each ball independently and randomly with probability  $1/2$ ; and (3) apply the sequence in Proposition 8 with  $k = \sqrt{m}$ .

As the deletions are random in step (2), the precondition for Proposition 8 (i.e., the least loaded bin contain at least  $k = \sqrt{m}$  fewer balls than every other bins) holds with probability  $\Omega(1)$ . So by Proposition 8, we can achieve  $m/4 + \Omega(\sqrt{k}) = m/4 + \Omega(m^{1/4})$  balls in some bin, with probability  $\Omega(1)$ .

**General  $n$ .** For  $n$  bins, where  $n$  is arbitrary, the same approach gives a lower bound of

$$m/n + \Omega(m^{1/4} / \sqrt{n^3 \log n}). \quad (6)$$

In particular, Proposition 8 can be directly modified, in this setting, to achieve an overload of  $\Omega(k^{1/2}/n)$ : instead of using  $k$  balls in each of steps 1 and 3, use  $kn/100$  balls; then by the same argument as in the lemma, we have  $A - B = \Omega(k^{1/2})$  with constant probability; this means that bin 1 is under-loaded by at least  $\Omega(k^{1/2})$ , and thus that some other bin is over-loaded by at least  $\Omega(k^{1/2}/n)$ .

To achieve (6) using the modified Proposition 8, we just need to cause the smallest load to be  $k = \Theta(\sqrt{m/(n \log n)})$  smaller than the other loads—this can again be achieved again by performing  $m$  insertions and then deleting each ball independently with probability  $1/2$ . After the  $m$  insertions, every bin will have essentially the same load ( $\pm O(\log \log n)$  w.h.p. in  $n$ ). Conditioning on the loads, the number of balls deleted from each bin is a Gaussian with standard deviation  $\sigma = \Theta(\sqrt{m/n})$  (and the Gaussians are independent between bins). By standard estimates on order statistics, the difference in loads between the least loaded and the second least loaded bins is roughly the difference between the  $1/n$ -th and  $2/n$ -th percentile of the distribution, see e.g., [Roy82], which in expectation is  $\Theta(\sigma/\sqrt{\log n})$  for the Gaussian  $N(0, \sigma^2)$ —hence an imbalance of  $k = \Theta(m/(n \log n))$ .

### 3.2 The Stronger $\Omega(m^{1/2})$ Lower Bound

We now show how to achieve the stronger bound of  $m/4 + \Omega(\sqrt{m})$  balls in some bin. Given Proposition 8, to prove Theorem 7 it suffices to show how to achieve a gap of  $k = \Omega(m)$  between bin 1 and bins 2, 3, 4. This is accomplished in the following proposition.

**Proposition 9.** *Consider the GREEDY algorithm on 4 bins, with the restriction that at most  $m$  balls can be present at a time. There exists an oblivious sequence of  $\text{poly}(m)$  insertions/deletions such that, after the sequence is complete, we have the following property with probability  $\Omega(1)$ : Bin 1 contains  $\Omega(m)$  fewer balls than each of bins 2, 3, 4.*

The rest of the section is focused on the proof of Proposition 9.

Let  $0 < \varepsilon_1, \varepsilon_2, \varepsilon_3 < 1$  be constants, where  $\varepsilon_2$  is sufficiently small as a function of  $\varepsilon_1$ , and let  $\varepsilon_3$  is sufficiently small as a function of  $\varepsilon_2$ . Sometimes we will write  $\varepsilon_1, \varepsilon_2, \varepsilon_3$  inside the  $O(\cdot)$  notation, to make the dependence on them explicit, while hiding fixed constants that do not depend on  $\varepsilon_1, \varepsilon_2, \varepsilon_3$ .

### 3.2.1 Some basic gadgets

We begin with a basic technical lemma establishing that GREEDY has a tendency of eliminating imbalances over time. For brevity (and since the proof follows from standard arguments), we defer the proof of Lemma 10 to Appendix A.

**Lemma 10.** *Consider the GREEDY algorithm on 4 bins, and fix an arbitrary initial state in which the bins have loads within  $\varepsilon_2 m$  of each other. If  $\varepsilon_1 m$  insertions are performed, then after the sequence is complete, all of the bins have loads within  $O(\log m)$  of each other with high probability in  $m$ . Furthermore, with high probability in  $m$ , there is a point in time prior to the final insertion at which all of the bins have equal loads.*

Using Lemma 10, we now construct a simple strategy for forcing GREEDY to add a ball to a uniformly random bin.

**Lemma 11** (Uniform ball placement gadget). *Consider the GREEDY algorithm on 4 bins, and fix an arbitrary initial state in which the bins have loads within  $\varepsilon_2 m$  of each other. Suppose we insert balls  $x_1, \dots, x_{\varepsilon_1 m}$ , and then we delete balls  $x_1, \dots, x_{\varepsilon_1 m - 1}$  (all except the last insertion). With high probability in  $m$ , this is equivalent to placing the ball  $x_{\varepsilon_1 m}$  uniformly at random into one of the bins 1, 2, 3, 4.*

*Proof.* We have by Lemma 10 that, with high probability in  $m$ , there is some insertion  $x_i$ ,  $i \in [\varepsilon_1 m - 1]$ , after which the bins have equal loads. It follows that, from the perspectives of insertions  $x_{i+1}, \dots, x_{\varepsilon_1 m}$ , the four bins are symmetric. Thus the last insertion  $x_{\varepsilon_1 m}$  is equally likely to be placed into each of the bins, which establishes the lemma.  $\square$

Lemma 11 allows for us to place a ball into a random bin, but we can only do this  $O(m)$  times before there are too many balls ( $> m$ ) in the system. But for the purposes of Proposition 9, we will need to do this  $\Omega(m^2)$  times. Our next lemma provides a mechanism for reducing the number of balls that are present while having only a small effect on the relative loads of the bins.

**Lemma 12** (Almost equal load reduction gadget). *Consider the GREEDY algorithm on 4 bins, and fix an arbitrary initial state in which the bins 1, 2, 3, 4 have loads  $\ell_1, \ell_2, \ell_3, \ell_4$  within  $\varepsilon_2 m$  of each other. We can construct an oblivious sequence of  $O(\varepsilon_1 m)$  insertions/deletions such that, after this sequence, the total number of balls in the system is at most  $\varepsilon_1 m$ ; and such that, with high probability in  $m$ , the new bin loads  $\ell'_i$  for  $i \in [4]$  satisfy*

$$\ell'_i = \ell_i - r + Y^{(i)}, \quad (7)$$

where  $r \in \mathbb{N}$ ,  $|Y^{(i)}| \leq O(\log m)$ , and  $\mathbb{E}[Y^{(i)}] = 0$ .

*Proof.* Let us begin by describing is a sequence of  $O(\varepsilon_1 m)$  insertions/deletions after which (1) the total number of balls in the system is at most  $\varepsilon_1 m$ ; and (2) the new loads  $\ell'_i$  of the bins satisfy (w.h.p. in  $m$ )

$$\ell'_i = r - \ell_i + Y^{(i)}, \quad (8)$$

where  $r \in \mathbb{N}$ ,  $|Y^{(i)}| \leq O(\log m)$ , and  $\mathbb{E}[Y^{(i)}] = 0$ . (Note that (8) is the same as (7) but with  $r$  and  $\ell_i$  flipped).

The lemma would then follow by applying the above construction twice. That is, first we obtain  $\ell'_1, \ell'_2, \ell'_3, \ell'_4$  satisfying (8), and then apply it again to obtain  $\ell''_1, \ell''_2, \ell''_3, \ell''_4$  satisfying

$$\ell''_i = r' - \ell'_i + Y'^{(i)}, \quad (9)$$

where  $r' \in \mathbb{N}$ ,  $|Y^{(i)}| \leq O(\log m)$ , and  $\mathbb{E}[Y^{(i)}] = 0$ . Chaining together (8) and (9), we get relationship between  $\ell_1, \ell_2, \ell_3, \ell_4$  and  $\ell'_1, \ell'_2, \ell'_3, \ell'_4$  as desired by (7).

Our construction for achieving (8) is very simple: we perform  $\varepsilon_1 m$  insertions  $x_1, x_2, \dots, x_{\varepsilon_1 m}$ , and then we delete all of the *other elements* besides  $x_1, x_2, \dots, x_{\varepsilon_1 m}$ . Let  $\ell_i$  be the load of bin  $i$  before these insertions/deletions, let  $q_i$  be the load of bin  $i$  after the insertions are completed (but the deletions have not yet begun), and let  $\ell'_i$  be the load of bin  $i$  after the deletions have completed.

By Lemma 12, the quantities  $q_1, q_2, q_3, q_4$  are within  $O(\log m)$  of each other (w.h.p. in  $m$ ). Moreover, w.h.p. in  $m$ , there is some point during the insertions at which all of the bins have equal loads—if we condition on this, then we have  $\mathbb{E}[q_1] = \mathbb{E}[q_2] = \mathbb{E}[q_3] = \mathbb{E}[q_4]$  by symmetry. Defining  $Y^{(i)} = q_i - \mathbb{E}[q_i]$ , we have  $|Y^{(i)}| \leq O(\log m)$ , and  $\mathbb{E}[Y^{(i)}] = 0$ .

As  $\ell'_i = q_i - \ell_i$ , we have  $\ell'_i = \mathbb{E}[q_i] + Y^{(i)} - \ell_i$ . Setting  $r = \mathbb{E}[q_i]$ , it follows that (8) holds w.h.p. in  $m$ .  $\square$

### 3.2.2 Applying the gadgets

We say that an application of Lemma 11 or of Lemma 12 *fails* if either: the precondition of  $\ell_1, \ell_2, \ell_3, \ell_4$  being within  $\varepsilon_2 m$  of each other fails (this is a *precondition failure*); or the high-probability guarantee offered by the lemma fails (this is a *probabilistic failure*).

We now describe the sequence of insertions/deletions that we use to achieve Proposition 9. We perform  $\varepsilon_3 m$  phases, where phase  $a \in [\varepsilon_3 m]$  proceeds as follows:

- Apply Lemma 11  $m$  times, one after another. For  $b \in [m]$ , use  $Z_{m \cdot (a-1) + b}$  to denote the bin that the  $b$ -th application of the lemma adds a ball to. If the lemma fails, then for the sake of analysis, we redefine  $Z_{m \cdot (a-1) + b}$  to be uniformly random in  $[4]$ . This ensures that, regardless of whether the lemma fails, the  $Z_i$ 's are independently and uniformly random in  $[4]$ .
- Apply Lemma 12 once to reduce the loads almost equally. Let  $Y_a^{(1)}, Y_a^{(2)}, Y_a^{(3)}, Y_a^{(4)}$  denote the outcomes of  $Y^{(1)}, Y^{(2)}, Y^{(3)}, Y^{(4)}$  in that application of the lemma. If the lemma fails, then for the sake of analysis, we redefine  $Y_a^{(1)}, Y_a^{(2)}, Y_a^{(3)}, Y_a^{(4)}$  to be 0.

To analyze the sequence of insertions/deletions, we first argue that the  $Y_i^{(s)}$ 's have a negligible effect on the loads of the bins at any given moment.

**Lemma 13.** *Let  $s \in [4]$  and  $k \in [\varepsilon_3 m]$ . Then w.h.p. in  $m$ , it holds that for each  $k$ ,  $|\sum_{a=1}^k Y_a^{(s)}| \leq \tilde{O}(\sqrt{m})$ , where  $\tilde{O}(\cdot)$  hides polylogarithmic factors in  $m$ .*

*Proof.* The sequence of partial sums  $P_r = \sum_{a=1}^r Y_a^{(s)}$  for  $r = 0, \dots, k$  forms a martingale satisfying  $|P_r - P_{r-1}| = O(\log m)$  deterministically for each  $r \in [k]$ . The lemma follows from Azuma's inequality.  $\square$

Next we consider the effect of the  $\varepsilon_3 m^2$  insertions  $Z_i$  over the  $\varepsilon m$  phases, and show that with probability at least  $1 - \varepsilon_2$ , there is no point in time at which the  $Z_i$ 's cause an imbalance of more  $\varepsilon_2 m/2$ .

For  $k \in [\varepsilon_3 m^2]$  and  $s \in [4]$ , let

$$S(k, s) = |\{i \in [k] \mid Z_i = s\}|$$

denote the number insertions in bin  $s$  during the first  $k$  applications of Lemma 11.

**Lemma 14.** *Let  $s \in [4]$  and  $\varepsilon_2 = (2\varepsilon_3)^{1/3}$ . With probability at least  $1 - \varepsilon_2$ , it holds (simultaneously) for all  $k \in [\varepsilon_3 m^2]$  that*

$$|S(k, s) - k/4| \leq \varepsilon_2 m/2.$$

*Proof.* As  $Z_i$  is equal to  $s$  independently with probability  $1/4$ , the sequence  $S(k, s) - k/4$  for  $k = 0, 1, \dots, \epsilon_3 m^2$  forms a martingale with increments  $\{-1/4, 3/4\}$  (and hence variance at most 1). By the maximal inequality for martingales, for any  $\lambda > 0$ ,

$$\Pr \left[ \max_{k \in [\epsilon_3 m^2]} |S(k, s)| > \lambda \right] \leq 2 \frac{\text{Var}[S(\epsilon_3 m^2, s)]}{\lambda^2} \leq 2 \frac{\epsilon_3 m^2}{\lambda^2}.$$

Setting  $\lambda = m(2\epsilon_3/\epsilon_2)^{1/2}$  so that the right hand side above is  $\epsilon_2$ , and choosing  $\epsilon_2^3 \leq 2\epsilon_3$  so that  $\lambda \geq \epsilon_2 m$  gives the claimed result.  $\square$

Combining Lemmas 13 and 14, we can bound the probability of any failures occurring during our construction.

**Lemma 15.** *With probability at least  $1 - \epsilon_2 - 1/\text{poly}(m)$ , no failures (either precondition failures or probabilistic failures) occur during the construction.*

*Proof.* Probabilistic failures occur with probability only  $1/\text{poly}(m)$  per application of Lemma 11 or Lemma 12. Across the  $O(m^2)$  applications of the lemmas, the probability of a probabilistic failure ever occurring is at most  $1/\text{poly}(m)$ . For the rest of the proof, we condition on no probabilistic failures occurring.

We now bound the probability of any precondition failure. Before any particular application of Lemma 11 or Lemma 12 (during the input sequence of insertions/deletions), for bin  $s \in [4]$ , the amount by which its load differs from the mean can be expressed as

$$\left| \sum_{i=1}^{k_1} Y_i^{(s)} + S(k_2, s) - k_2/4 \right|$$

for some  $k_1, k_2$ . By Lemmas 13 and 14, the probability that this quantity ever exceeds  $\epsilon_2 m$  (and hence any precondition failure occurring) is at most  $\epsilon_2 + 1/\text{poly}(m)$ , which completes the proof.  $\square$

Finally, we argue that with probability at least  $\epsilon_1$ , the  $Z_i$ 's *do cause* an imbalance of  $\Omega(m)$  at the end of the construction. In particular, bin 1 contains  $\Omega(m)$  fewer balls than bins 2, 3, 4.

**Lemma 16.** *With probability at least  $\epsilon_1$ , we have that*

$$|S(\epsilon_3 m^2, 1)| < \max_{s \in \{2, 3, 4\}} |S(\epsilon_3 m^2, s)| - \Omega(\sqrt{\epsilon_3} m).$$

*Proof.* Let  $X_s$  denote the number of such balls inserted in bin  $s$ . Then  $X_1$  is a binomial random variable with mean  $\mu = \Theta(\epsilon_3 m^2)$ . Thus, with probability at least  $2\epsilon_1$ , we have that,  $X_1 \leq \mu - 10\sqrt{\mu}$ . On the other hand, if we condition on some value  $\leq \mu - 10\sqrt{\mu}$  for  $X_1$ , then the variables  $X_2, X_3, X_4$  become binomial random variables with means  $\mu' > \mu$ . Each  $X_i$  has probability at least 0.9 of satisfying  $X_i > \mu' - 5\sqrt{\mu'} \geq \mu - 5\sqrt{\mu}$ . Thus, if we condition on  $X_1 \leq \mu - 10\sqrt{\mu}$ , then the probability at least 0.7, we have  $X_2, X_3, X_4 > \mu - 5\sqrt{\mu}$ . Putting these together, the probability that  $\max\{X_2, X_3, X_4\} - X_1 > 5\sqrt{\mu}$  is at least

$$\Pr[X_1 \leq \mu - 10\sqrt{\mu}] \cdot \Pr[X_2, X_3, X_4 > \mu - 5\sqrt{\mu} \mid X_1 \leq \mu - 10\sqrt{\mu}] \geq 2\epsilon_1 \cdot 0.7 > \epsilon_1. \quad \square$$

We can now complete the proof of Proposition 9.

*Proof of Proposition 9.* We prove the proposition using the construction described in this section. Note that, by design, there are never more than  $m$  balls present at a time, as Lemma 12 brings the number of balls back down to  $\epsilon_1 m$  every  $O(\epsilon_1 m)$  operations.

By Lemma 15, with probability at least  $1 - \epsilon_2 - 1/\text{poly}(n)$ , all of the applications of Lemma 11 and Lemma 12 succeed. Conditioned on this, at the end of the construction, the gap of each bin  $s \in [4]$  can be expressed as

$$\sum_{i=1}^{\epsilon_3 m} Y_i^{(s)} + S(\epsilon_3 m^2, s) - \epsilon_3 m^2 / 4.$$

By Lemma 13, we have  $\left| \sum_{i=1}^{\epsilon_3 m} Y_i^{(s)} \right| \leq \tilde{O}(\sqrt{m})$  with high probability in  $m$ . On the other hand, by Lemma 16,

$$|S(\epsilon_3 m^2, 1)| < \max_{s \in \{2,3,4\}} |S(\epsilon_3 m^2, s)| - \Omega(\sqrt{\epsilon_3 m})$$

with probability at least  $\epsilon_1$ . It follows that, with probability at least  $\epsilon_1 - \epsilon_2 - 1/\text{poly}(n)$ , the load of bin 1 at the end of the construction is  $\Omega(\sqrt{\epsilon_3 m})$  smaller than the loads of bins 2, 3, 4.  $\square$

## 4 An Impossibility Result For The Deletions with Reinsertions

In this section we prove an impossibility result for the reinsertion/deletion model, namely, that no ID-oblivious insertion strategy can guarantee sub-polynomial overload.

**Theorem 17.** *Consider the reinsertion/deletion model with 4 bins, and with a limit of up to  $m$  balls present at a time. Against any ID-oblivious insertion strategy, it is possible for an oblivious adversary to force a maximum load of  $m/4 + m^{\Omega(1)}$  at some point in the first  $\text{poly}(m)$  operations, with high probability in  $m$ .*

The section splits the proof of Theorem 17 into two parts. First, in Subsection 4.1, we introduce and analyze the so-called *marble-splitting game*; then, in Subsection 4.2 we show how to perform a sequence of insertions/deletions that simulates an instance of the marble-splitting game and forces some bin to contain load  $m/4 + m^{\Omega(1)}$  with non-negligible probability.

### 4.1 The Marble-Splitting Game

In this section we present and analyze a simple game, which we call the *marble-splitting game*—the game plays an important role in our lower bound for balls-and-bins games with reinsertions.

In the marble-splitting game, there are two players Alice and Bob. The player Alice has two types of moves: she can perform an INSERT operation, which adds a new marble into the game, or she can perform a SPLIT( $x, y$ ) operation, which takes two marbles  $x$  and  $y$  and replaces them with new marbles  $x'$  and  $y'$ . Alice must decide her moves at the beginning of time (so she is an oblivious adversary).

The second player Bob gets to assign a *value*  $v_x$  to each marble  $x$ , according to the following rule: whenever Alice performs an INSERT, Bob can assign the new marble an arbitrary real-numbered value in the range  $[-1, 1]$ ; and whenever Alice performs a SPLIT( $x, y$ ) operation, Bob assigns  $x'$  and  $y'$  values  $v_{x'}$  and  $v_{y'}$  satisfying

$$v_{x'} + v_{y'} = v_x + v_y \pm o(R^{-2}), \quad \text{and} \quad v_{x'} - v_{y'} \geq 2/R. \quad (10)$$

Equivalently,  $v_{x'} = (v_x + v_y)/2 + \Delta \pm o(R^{-2})$  and  $v_{y'} = (v_x + v_y)/2 - \Delta \pm o(R^{-2})$  for some  $\Delta \geq 1/R$ .

Alice's goal is to force some marble (she need not know which one) to have a value greater than 1 at some point within the first  $O(R^3)$  steps of the game. Her disadvantage is that she does not know the precise values of marbles. Intuitively, she would like to perform split operations on marbles  $x$  and  $y$  that satisfy  $|v(x) - v(y)| = o(1/R)$ . But she might, for example, accidentally split two marbles  $x$  and  $y$  whose values differ considerably—this would result in  $x'$  and  $y'$  having values that are *closer together* than  $x$  and  $y$  had,



which is intuitively counterproductive for Alice. We shall see that, nonetheless, Alice can deterministically force a win within  $O(R^3)$  steps.

In constructing Alice's strategy, we will find it helpful for accounting purposes to artificially place the following additional constraints on Alice. We think of there as being *bags*  $0, 1, 2, \dots$ , each of which is capable of holding arbitrarily many marbles. Whenever a marble is inserted, we place it in bag 1. Whenever a split  $\text{SPLIT}(x, y)$  operation is performed, we require that the marbles  $x$  and  $y$  are currently in the *same* bag  $i \geq 1$  as each another, and after the split, we place the new marbles  $x'$  and  $y'$  into bags  $i+1$  and  $i-1$ , respectively. This restriction somewhat limits Alice's possible strategies, but, as we shall see, it also simplifies the task of analyzing Alice's "progress" over time.

The key result of this section is the following.

**Proposition 18.** *Alice can deterministically force some marble to have a value greater than 1 at some point within the first  $O(R^3)$  steps of the game. Moreover, the strategy performs only  $O(R^2)$  insertions.*

*Proof.* We begin by describing Alice's strategy. Let  $c$  be a large positive constant. She initially performs one insertion into bag 1. She then proceeds in  $cR$  phases, where at the beginning of phase  $i \in \{1, 2, \dots, cR\}$ , the state of the system is as follows: bag 0 contains some arbitrary number of marbles; bags  $1, 2, \dots, i$  each contain one marble; and bags  $i+1, i+2, \dots$  are empty.

The  $i$ -th phase consists of  $(i+1)$  sub-phases, where at the beginning of each subphase  $j \in \{1, 2, \dots, i+1\}$ , the state of the system is as follows: bag 0 contains some arbitrary number of marbles; and, with the exception of bag  $i-j+2$ , which is empty, all of bags  $2, 3, 4, \dots, i+1$  contain one marble (so bags  $1, 2, 3, 4, \dots, i-j+1$  each contain one marble; bag  $i-j+2$  is empty; and bags  $i-j+3, \dots, i+1$  each contain one marble).

The  $(i+1)$ -th subphase is special in that, all Alice does is perform one more insertion in order to reach the starting state for phase  $i+1$  (i.e., all of bags  $1, 2, \dots, i+1$  contain 1 marble).

For  $j < i+1$ , the  $j$ -th subphase of phase  $i$  is implemented as follows. Alice inserts one marble into bag 1. She then performs splits, one after another, on bags  $1, 2, 3, \dots, i-j+1$ . For each  $t \in \{1, 2, \dots, i-j\}$  (i.e., for every split but the final split), after she performs a split on bag  $t$ , the state of the system is that: bags  $1, 2, \dots, t-1$  contain one marble each; bag  $t$  is empty; bag  $t+1$  contains 2 marbles; and bags  $t+2, t+3, \dots$  are as they were at the beginning of the subphase. The final split that Alice performs (i.e., the split in  $i-j+1$ ) has the effect of placing a marble into the previously empty bags  $i-j$  and  $i-j+2$ , and leaving bag  $i-j+1$  as the solitary empty bag out of bags  $0, 1, 2, \dots, i+1$ . Thus we reach the starting state for the  $(j+1)$ -th subphase.

**Analysis of the strategy.** The analysis will need only the following basic facts about Alice's strategy: (1) it performs a total of  $O(c^2 R^2)$  INSERT operations and  $\Omega(c^3 R^3)$  SPLIT operations; (2) it only places marbles in bags  $i \leq cR+1$ ; and (3) at the end of the game, there is at most 1 marble in each bag  $i$  for  $i > 0$ .

Let  $B_i$  denote the marbles in bag  $i$  at any given moment, and define the potential function

$$\phi = \sum_{i=0}^{\infty} i \cdot \sum_{x \in B_i} v_x.$$

We will prove the proposition by analyzing how  $\phi$  evolves over time.

Each time that an INSERT is performed,  $\phi$  may decrease by up to 1, as  $v_x \in [-1, 1]$  and the marble is inserted in bag 1. During the entire game, this leads to a decrease of at most  $O(c^2 R^2)$ .

Each time that a SPLIT is performed, two marbles  $x$  and  $y$  in some bag  $i$  are replaced by  $x'$  and  $y'$  with values given by (10). Removing  $x$  and  $y$  decreases  $\phi$  by  $i \cdot (v_x + v_y)$  and inserting  $x'$  and  $y'$  increases  $\phi$  by

$$(i+1)v_{x'} + (i-1)v_{y'} = i \cdot (v_x + v_y) + (v_{x'} - v_{y'}) \pm o(iR^{-2}) \geq i \cdot (v_x + v_y) + 1/R.$$

The net effect of a split is therefore to increase  $\phi$  by at least  $1/R$ . As there are  $\Omega(c^3 R^3)$  split operations across the entire game, this increases  $\phi$  by  $\Omega(c^3 R^2)$ .

Combining the bounds for INSERT and SPLIT operations, we have that, at the end of the game,

$$\phi \geq \Omega(c^3 R^2) - O(c^2 R^2) = \Omega(c^3 R^2).$$

But this means that some bin  $i \leq cR + 1$  must satisfy  $i \cdot \sum_{x \in B_i} v_x > \Omega(c^2 R)$ , and thus that  $\sum_{x \in B_i} v_x > \Omega(c)$ . As  $|B_i| \leq 1$ , this implies that there is a ball  $x$  with  $v_x > 1$ .  $\square$

**Remark.** It is worth noting that, in the strategy in Proposition 18, we could have alternatively performed all of the insertions into bag 1 up front (i.e., at the beginning of the game), and then applied the appropriate SPLIT operations without performing any further insertions—each marble would simply remain in bag 1 until it was used for the SPLIT operations involving it. This perspective will be convenient in our application of marble splitting.

## 4.2 Proof of Theorem 17

We will now derive a sequence of insertions/deletions that can be used to establish Theorem 17.

As notation, let  $Q = \{(i, j) \mid i, j \in [4], i \neq j\}$ , and let  $h$  be a fully independent hash function mapping each ball  $x$  to a uniformly random pair  $h(x) = (h_1(x), h_2(x)) \in Q$ . Notice that  $|Q| = 12$ .

insertion

$$|\{x \in A \mid h(x) = (1, 2)\}| = k/12 + t \pm O(\sqrt{k}) \quad (11)$$

$$\text{and } |\{x \in A \mid h(x) = (3, 4)\}| = k/12 - t \pm O(\sqrt{k}). \quad (12)$$

Note that  $\mathcal{E}$  only depends on the hash values for balls in  $A$ .

We will show that, if we condition on  $\mathcal{E}$  occurring, and if  $t$  is moderately large (i.e.,  $c\sqrt{k}$  for some sufficiently large positive constant  $c$ ), then we can perform a sequence of insertions/deletions that make use of the sets  $A$  and  $B$  in order to defeat any ID-oblivious insertion strategy. While  $\mathcal{E}$  only has a small constant probability of occurring, this can be amplified by repeating the strategy multiple times.

As a final but crucial piece of notation, for any set  $S$  of balls present in the system, define the *value*  $v(S)$  to be the number of balls  $x \in S$  that reside in bins 1, 2. The ultimate structure of our analysis will be to show that, if an ID-oblivious algorithm guarantees a maximum load of  $m/4 + m^{o(1)}$  (with high probability), then we can construct a set  $S$  for which we can derive the clearly false assertion that  $\mathbb{E}[v(S)] > |S|$ .

### 4.2.1 Some basic gadgets

We will now prove a series of lemmas showing how to construct a malicious sequence of insertions/deletions using the sets  $A$  and  $B$  (and conditioned on  $\mathcal{E}$ ). We begin by observing what happens if we simply insert the elements  $A \cup B$  in a random order.

**Lemma 19.** *Consider a balls-and-bins game with 4 bins, starting from an arbitrary state. Suppose balls are allocated to bins using an arbitrary ID-oblivious insertion strategy that has already been shown the sets  $A, B$  (i.e., the algorithm can depend on the multisets  $\{h(x) \mid x \in A\}$  and  $\{h(x) \mid x \in B\}$ ). Condition on event  $\mathcal{E}$ , and suppose that we insert the balls  $A \cup B$  in a random order. Then, after the insertions are completed, we have*

$$\mathbb{E}[v(A) - v(B)] \geq t - O(\sqrt{k}).$$

The intuition behind Lemma 19 is quite simple. For  $(i, j) \in Q$ , define  $A_{i,j}$  (resp.  $B_{i,j}$ ) to be the set of balls in  $A$  (resp.  $B$ ) that hash to the bin pair  $(i, j)$ . Due to event  $\mathcal{E}$ , we have that  $\mathbb{E}[|A_{1,2}| - |B_{1,2}|] \geq t - O(\sqrt{k})$ , so this immediately gives  $A$  an extra  $t - O(\sqrt{k})$  balls (in expectation) in bins 1, 2 that  $B$  doesn't get. On the other hand, for each  $(i, j) \in Q \setminus \{(1, 2), (3, 4)\}$ , we expect the number of balls from  $A_{i,j}$  that are in bins 1, 2 to be roughly the same as the number of balls from  $B_{i,j}$  that are in bins 1, 2, hence the conclusion of the lemma. Formalizing this argument requires some care as the algorithm can try to distinguish the balls in  $A$  from those in  $B$  based on the differences between  $|A_{i,j}|$  and  $|B_{i,j}|$ , for  $(i, j) \in Q$ . Thus we defer the full proof of the lemma to Appendix B.

Our next lemma makes a simple observation about what happens when we remove a set  $X$  of balls and replace it with a set  $X'$  of balls, in a balls-and-bins game that is at capacity (i.e., contains  $m$  balls).

**Lemma 20.** *Consider a balls-and-bins game with 4 bins, starting with  $m$  balls in the system. Let  $X$  be a set of  $r$  balls that are present. Suppose that we delete the balls  $X$ , and then insert new balls  $X'$ , where  $|X'| = r$ . Then one of the following events must occur:*

- *there is some point in time at which some bin contains  $m/4 + \omega(\sqrt{k})$  balls;*
- *or,  $|v(X) - v(X')| = O(\sqrt{k})$ .*

*Proof.* Suppose that they are never more than  $m/4 + \Omega(\sqrt{k})$  balls in any given bin. This means that, whenever there are  $m$  balls in the system, the number of balls in bins 1, 2 must be within  $O(\sqrt{k})$  of  $m/2$ .

When we remove balls  $X$ , we decrease the number of balls in bins 1, 2 by  $v(X)$ . When we insert balls  $X'$ , we increase the number of balls in bins 1, 2 by  $v(X')$ . In total, we must change the load of bins 1, 2 by  $O(\sqrt{k})$ , meaning that  $|v(X) - v(X')| = O(\sqrt{k})$ .  $\square$

**Gadget for splitting.** By combining the previous two lemmas in the right way, we can construct a sequence for *splitting* a set  $X$  of size  $\text{poly}(k)$  into two sets  $Y$  and  $Z$  such that  $v(Y) + v(Z) = (1 \pm o(k^{-1}))v(X)$  and  $\mathbb{E}[v(Y) - v(Z)] \geq \Omega(|X|/\sqrt{k})$ .

**Lemma 21** (Splitting gadget). *Consider a balls-and-bins game with 4 bins, starting from an arbitrary state with  $m$  balls, and where balls are allocated to bins using an ID-oblivious insertion strategy that, as in Lemma 19, has already been shown the sets  $A, B$ , and that keeps the load of each bin below  $m/4 + O(\sqrt{k})$  w.h.p. in  $m$ . Finally, condition on event  $\mathcal{E}$  with  $t = c\sqrt{k}$  for some sufficiently large constant  $c > 0$ .*

*Let  $X$  be a set of  $q = k^{1.5} \log k$  balls that are currently present in the system. There exists a sequence of  $\text{poly}(k)$  insertions/deletions that (without ever placing more than  $m$  balls in the system at a time) replaces  $X$  with  $q/2$ -element sets  $Y, Z$  satisfying*

$$\mathbb{E}[v(Y) - v(Z)] \geq k \log k, \quad (13)$$

*and satisfying*

$$\mathbb{E}[v(Y) + v(Z)] = v(X) \pm O(\sqrt{k}). \quad (14)$$

*Proof.* Roughly speaking, the goal is to transfer the *imbalance* between the sets  $A$  and  $B$  (in how they allocate balls to bins 1, 2 vs. 3, 4) to the set  $X$ , so that the resulting sets  $Y$  and  $Z$  have similar *relative* imbalance to what  $A$  and  $B$  have. Of course,  $A$  and  $B$  have size  $k$  each, while  $X$  has size  $q = k^{1.5} \log k$ , so the imbalance between  $A$  and  $B$  needs to be amplified in order to get the same relative imbalance between  $Y$  and  $Z$ . As we shall see, this is where we crucially make use of the ability to delete and *reinsert*  $A \cup B$  multiple times.<sup>2</sup>

Let us partition  $X$  into sets  $X_1, X_2, \dots, X_{q/k}$  of size  $k$  each. For each  $i \in [q/2k]$ , we will replace  $X_{2i-1}$  by a new set  $Y_i$  and  $X_{2i}$  by a new set  $Z_i$ , in such a way that the relative imbalance between  $Y_i$  and  $Z_i$  is similar

<sup>2</sup>The other place where we make use of reinsertions is that, ultimately, we will apply Lemma 21 multiple times, and we will continue to reuse  $A$  and  $B$  across those multiple applications.

to that between  $A$  and  $B$ . This is accomplished by performing the following sequence of insertions and deletions:

1. Delete the balls  $X_{2i-1} \cup X_{2i}$ .
2. Insert the balls  $A \cup B$  in a random order.
3. Delete the balls of  $A$ , and replace them with a set  $Y_i$  of  $k$  elements.
4. Delete the balls of  $B$ , and replace them with a set  $Z_i$  of  $k$  elements.

By Lemma 19, we have after Step (2) that

$$\mathbb{E}[v(A) - v(B)] \geq t - O(\sqrt{k}).$$

By Lemma 20 (and since the insertion strategy keeps bin loads of  $m/4 + O(\sqrt{k})$  with high probability in  $m$ ), we then have that  $\mathbb{E}[v(Y_i)]$  and  $\mathbb{E}[v(Z_i)]$  are within  $O(\sqrt{k})$  of  $\mathbb{E}[v(A)]$  and  $\mathbb{E}[v(B)]$ , respectively. Thus

$$\mathbb{E}[v(Y_i) - v(Z_i)] \geq t - O(\sqrt{k}) \geq 2\sqrt{k},$$

where the final inequality uses the fact that  $t = c\sqrt{k}$  for a sufficiently large positive constant  $c$ .

Summing over  $i \in \{1, 2, \dots, q/(2k)\}$ , and denoting  $Y = \cup_i Y_i$  and  $Z = \cup_i Z_i$ , we get the claimed bound

$$\mathbb{E}[v(Y) - v(Z)] = \sum_i \mathbb{E}[v(Y_i) - v(Z_i)] \geq k \log k.$$

Next, applying Lemma 20 with  $X' = Y \cup Z$ , we have that either

$$v(Y) + v(Z) = v(X) \pm O(\sqrt{k}),$$

or that there is some point in time at which a bin has load  $m/4 + \omega(\sqrt{k})$ . Since the latter event is assumed to occur with probability at most  $1/\text{poly}(m)$ , this completes the proof of the lemma.  $\square$

#### 4.2.2 Connection to marble-splitting

We are now ready to prove Theorem 17. We begin by proving a slightly weaker version of the theorem, namely that no ID-oblivious insertion strategy can offer a high-probability guarantee of achieving overload  $m^{o(1)}$ .

**Proposition 22.** *Consider the reinsertion/deletion model with 4 bins, and with a limit of up to  $m$  balls present at a time. Suppose there is an ID-oblivious bin-allocation algorithm that, for the first  $\text{poly}(m)$  steps, bounds the load of each bin by  $m/4 + f(m)$  with high probability in  $m$ . Then  $f(m) = m^{\Omega(1)}$ .*

*Proof.* Set  $k = m^\epsilon$  for a positive constant  $\epsilon$  to be selected later in the proof, and suppose for contradiction that  $f(m) = O(\sqrt{k})$ .

Let  $A$  and  $B$  be disjoint sets of  $k$  balls each. Let  $c$  be a sufficiently large positive constant, and set  $t = c\sqrt{k}$ . Finally, let  $\mathcal{E}$  be the event that (11) and (12) hold. Note that  $\mathcal{E}$  occurs with probability  $\Omega(1)$ ; for the rest of the proof, condition on  $\mathcal{E}$ .

Let  $X_1, X_2, \dots, X_{ck}$  be disjoint sets of  $(k^{1.5} \log k)/2$  balls each. To begin, insert  $m$  balls into the system, where those balls include  $X_1, X_2, \dots, X_{ck}$ . The sets  $X_1, X_2, \dots, X_{ck}$  will act as *marbles* in a marble-splitting game. There are two types of operations that we will perform in this game: an INSERT operation, which adds one of the sets  $X_1, X_2, \dots, X_{ck}$  as a new marble in the game; and a SPLIT( $X, Y$ ) operation, which takes

two sets  $X$  and  $Y$  of size  $(k^{1.5} \log k)/2$  balls each, and applies Lemma 21 to replace them with sets  $X', Y'$  (also of  $(k^{1.5} \log k)/2$  balls each) satisfying

$$\frac{\mathbb{E}[v(X')]}{|X'|} - \frac{\mathbb{E}[v(Y')]}{|Y'|} \geq 2/\sqrt{k}, \quad (\text{by (13)})$$

$$\frac{\mathbb{E}[v(X)]}{|X|} + \frac{\mathbb{E}[v(Y)]}{|Y|} = \frac{\mathbb{E}[v(X')]}{|X'|} + \frac{\mathbb{E}[v(Y')]}{|Y'|} \pm o(1/k). \quad (\text{by (14)})$$

If we define  $v_X := \frac{\mathbb{E}[v(X)]}{|X|}$  for each set  $X$  of  $k^{1.5}/2$  balls, it follows that we are playing a marble-splitting game with  $R = \sqrt{k}$ , and where marbles correspond to sets of  $(k^{1.5} \log k)/2$  balls. By Proposition 18, there is an  $O(R^3) = O(k^{1.5})$ -step strategy that results in some marble  $X$  satisfying  $v_X > 1$ . This is a contradiction, since  $v_X$  must deterministically be in the range  $[0, 1]$ .

Note that the marble-splitting game requires  $O(R^2) = O(k)$  marbles at a time, each of which consists of  $O(k^{1.5} \log k)$  balls. Thus, the entire game uses  $O(k^{2.5} \log k)$  balls, meaning that we can set  $k = m^{1/2.5 - o(1)}$ . We can therefore conclude that  $f(m)$  must be at least  $m^{1/5 - o(1)}$ .  $\square$

Finally, we prove Theorem 17 by applying a basic amplification argument to Proposition 22.

*Proof of Theorem 17.* By Proposition 22, there exists a parameter  $s \in \text{poly}(m)$  such that, within  $\text{poly}(m)$  operations, an oblivious adversary can achieve maximum load  $m/4 + m^{\Omega(1)}$  with probability  $1/s$ . By independently repeating this construction  $\Theta(s \log n) = \text{poly}(m)$  times, the probability of achieving a load of  $m/4 + m^{\Omega(1)}$  at some point during the sequence becomes

$$1 - (1 - 1/s)^{\Theta(s \log n)} = 1 - 1/\text{poly}(n),$$

as desired.  $\square$

## 5 Generalizations of MODULATEDGREEDY

We now generalize the MODULATEDGREEDY algorithm from Section 2 in several interesting ways:

1. We give guarantees over an infinite time horizon, instead of  $\text{poly}(m)$  steps.
2. We allow  $m$  (the maximum number of balls present in the system) to increase with time, and only require an a-priori bound  $M$  on  $m$ .
3. We consider the more general  $(1 + \beta)$ -choice and the graphical 2-choice settings (defined in Section 5.3) and extend the previous results for these settings (which were insertion-only) to also handle deletions.

These generalizations require extending both the algorithm and the analysis techniques. We begin in Subsection 5.1 by describing the algorithm and giving an overview of the key ideas; we then present the analysis and applications in Subsections 5.2 and 5.3.

### 5.1 The Algorithm and Overview

The algorithm, which we call GENERALIZEDMODULATEDGREEDY, is described as Algorithm 2 below. Its key properties are summarized in the following theorem.

**Theorem 23.** *Consider the insertion/deletion model with  $n$  bins, and an arbitrarily long sequence of insertions/deletions, with no more than  $M$  balls present at a time. Suppose the parameters  $n, M, \epsilon$  are known to the algorithm. Then the GENERALIZEDMODULATEDGREEDY algorithm satisfies the following guarantees:*

- **Bounded Load:** At any given moment, every bin has load at most  $m/n + O(\epsilon^{-1} \log M)$  with high probability in  $M$ , where  $m$  is the largest number of balls that were ever present so far.
- **Bounded Bias:** For any given insertion, if  $i$  and  $j$  are the two bins being chosen between, then each bin is selected with a probability in the range  $[1/2 - \epsilon, 1/2 + \epsilon]$ .

---

**Algorithm 2** The GENERALIZEDMODULATEDGREEDY algorithm. The algorithm has parameters  $M$  (an upperbound on the number of balls that will ever be present) and  $\epsilon$ , and makes use of a sufficiently large constant  $c > 0$ . The algorithm outputs a bin and a color for the ball being inserted.

---

**procedure** GENERALIZEDMODULATEDGREEDY

For  $k \in [n]$ , let  $\ell_k$  denote # balls with color  $k$ . Let  $\bar{\ell} = \frac{1}{n} \sum_k \ell_k$ .

Let  $m$  be the largest number of balls that have been present in the system at once thus far.

Let  $\Delta = c\epsilon^{-2} \log M$ .

Set  $T = \lceil m/n \rceil + \Delta - \bar{\ell}$ .

Select two bins  $i, j \in [n]$  independently and uniformly at random.

**if**  $(\max_k \ell_k) - (\min_k \ell_k) \leq \epsilon T$  **then**

With probability  $1/2 + \frac{\ell_j - \ell_i}{2T}$ , assign the ball to bin  $i$  and assign it color  $i$ .

Otherwise, assign the ball to bin  $j$  and assign it color  $j$ .

**else**

Declare the ball to be *corrupted*.

Select  $\rho \in [n]$  such that, for each  $k \in [n]$ ,

$$\Pr[\rho = k] = \frac{\lceil m/n \rceil + \Delta - \ell_k}{n \cdot T}.$$

Assign the ball uniformly at random in  $\{i, j\}$  and assign it color  $\rho$ .

---

Notice that the algorithm assigns a ball both a bin and a color. Typically, the color is the same as the bin to which the ball is assigned, but occasionally a ball will get *corrupted*, in which case the bin and color may differ. Moreover, at any time, the maximum load is bounded with respect to  $m/n$  (instead of  $M/n$ ).

Before giving the detailed analysis, we briefly describe the new ideas we need over those in Section 2.

**Infinite time horizon.** A key feature of the algorithm is that it offers guarantees on an infinite time horizon. To achieve this we explicitly incorporate the coupling with the stone game into the design of the algorithm. In particular, whenever there is an insertion that MODULATEDGREEDY would have been at risk of halting on, GENERALIZEDMODULATEDGREEDY instead declares that ball to be *corrupted*. The algorithm then “fudges” its bookkeeping: it treats the corrupted ball as being placed into whichever bin is necessary to maintain the coupling with the stone game.

More concretely, we assign each ball both to a bin (where it truly resides) and to a color (which, if the ball is corrupted, may differ from the ball’s bin). The algorithm makes all of its decisions based on ball colors (and ignores the actual bins that balls reside in). This allows for the algorithm to maintain a coupling forever between the colors of its balls and the colors of the balls in the stone game.

**Increasing  $m$ .** Another interesting feature is that the algorithm allows for  $m$  to grow over time, subject only to the constraint  $m \leq M$ . To handle this, GENERALIZEDMODULATEDGREEDY bases its allocation decisions on the largest value of  $m$  that it has witnessed so far. At first glance, this seems to significantly break the relationship between the balls-and-bins game and the stone game, and indeed Lemma 3 no longer holds—however, as we shall see, the stone game and its analysis can be modified to also handle the incremental growth in  $m$  over time.

**Bias,  $(1 + \beta)$ -choice and graphical process.** Finally, a third feature of the algorithm is that it introduces a new variable  $\varepsilon$  that constrains the amount of bias that the algorithm is permitted to exhibit. We will see at the end of the section that this seemingly minor modification allows us to extend the algorithm to the  $(1 + \beta)$ -choice and the graphical 2-choice process, both of which are generalizations of the classical 2-choice process. Moreover, the guarantees of the resulting algorithms matches the previous known results for the insertion-only case for these settings.

## 5.2 Algorithm Analysis

We now turn to proving Theorem 23. We begin by defining the generalized stone game, which extends the stone game in Section 2. Then we show how this game is closely related to the balls and bins game and use this relationship to analyze GENERALIZEDMODULATEDGREEDY.

### 5.2.1 The generalized stone game

The  $\Delta$ -GENERALIZED STONE GAME has an inactive bag and an active bag. The inactive bag is initialized to contain  $\Delta \cdot n$  stones  $x_{k,j}$  for  $k \in [n]$  and  $j \in [\Delta]$ , and the active bag is initialized to be empty. We say that the ball  $x_{k,q}$  has *color*  $k \in [n]$ . The game supports two operations that are performed by an oblivious adversary: **ACTIVATE()** and **DEACTIVATE( $r$ )**.

The **ACTIVATE()** operation (described formally in Algorithm 3) takes two steps: First, the operation moves a random stone from the inactive bag to the active bag. Second, if there are fewer than  $\Delta \cdot n$  stones in the inactive bag, then it computes the number  $Q \cdot n$  of stones currently in the system (active and inactive bags), and it adds  $n$  new stones  $\{x_{k,Q+1}\}_{k \in [n]}$ , one of each color, to the inactive bag. This second step is different from the standard stone game in Section 2, and in particular, the total number of stones now can increase over time (in increments of  $n$ ).

The **DEACTIVATE( $r$ )** operation works exactly as before—it takes whichever stone was added to the active bag  $r$ -th most recently, and moves that stone back to the inactive bag.

---

**Algorithm 3** The **ACTIVATE** method for the generalized stone game. The algorithm has parameter  $\Delta$ . The moves a random stone from the inactive bag to the active bag, and then (possibly) adds additional stones to the inactive bag.

---

**procedure** **ACTIVATE**

Move a random stone from the inactive bag to the active bag.  
**if** Inactive bag contains fewer than  $\Delta \cdot n$  balls **then**  
    Let  $Q \cdot n$  be # stones currently in the system  
    Add a *batch*  $B_{Q+1} = \{x_{k,Q+1}\}_{k \in [n]}$  of  $n$  new balls to the inactive bag.

---

We begin by proving a basic fact about the generalized stone game.

**Lemma 24.** *Let  $c > 0$  be a sufficiently large constant, and let  $\varepsilon, M$  be parameters. Fix any time in the  $(c\varepsilon^{-2} \log M)$ -generalized stone game, and for  $k \in [n]$ , let  $s_k$  denote the number of stones with color  $k$  in the inactive bag. With probability  $M^{-\Omega(c)}$ , for each  $k \in [n]$ , we have that*

$$(1 - \varepsilon/2)\mathbb{E}[s_k] \leq s_k \leq (1 + \varepsilon/2)\mathbb{E}[s_k].$$

*Proof.* Let  $Q \cdot n$  be the number of stones currently in the system. For each  $q \in \{1, 2, \dots, Q\}$ , define  $B_q = \{x_{k,q}\}_{k \in [n]}$ . The  $n$  stones in  $B_q$  are all inserted into the system in the same instant and are indistinguishable from one another in terms of how they interact with the sequence of operations being performed. If there are  $a_k$  balls from  $B_k$  in the inactive set, then the probability that any of them have color  $i$  is simply  $a_k/n$ .

Thus, if we fix some outcome for the values of the  $a_k$ 's, then we can write  $s_k = \sum_{q=1}^Q A_k$ , where  $A_k$  are independent indicator random variables with  $\Pr[A_q = 1] = a_q/n$ . Using  $I$  to denote the set of balls in the inactive set, the expected value of  $s_k$  evaluates to

$$\mathbb{E}[s_k] = \sum_{q=1}^Q a_q/n = |I|/n.$$

By design, however, the inactive set always at least  $|I| \geq \Delta \cdot n = c\epsilon^{-2}n \log M$  balls, so that  $\mathbb{E}[s_k] \geq \Omega(c\epsilon^{-2} \log M)$ . Applying a Chernoff bound (and as  $c$  is a large constant), for each  $k \in [n]$ ,  $s_k$  lies between  $(1 - \epsilon/2)\mathbb{E}[s_k]$  and  $(1 + \epsilon/2)\mathbb{E}[s_k]$  with probability  $M^{-\Omega(c)}$ .  $\square$

### 5.2.2 Coupling with GENERALIZEDMODULATEDGREEDY

Next we establish the connection between the generalized stone game and the GENERALIZEDMODULATEDGREEDY algorithm.

First, as in Section 2, the oblivious sequences of insertion/deletions for the balls-and-bins game maps to an input sequence of the  $\Delta$ -generalized stone game as follows: each insertion in the balls-and-bins game causes an activation in the stone game, and each deletion  $\text{DELETE}(x)$  in the balls-and-bins game causes a deactivation  $\text{DEACTIVATE}(r)$ , where  $r - 1$  is the number of balls present in the balls-and-bins game that were inserted after  $x$ .

The following key lemma shows that the random choices in the two games can be coupled.

**Lemma 25** (Coupling). *Consider a sequence  $S$  of insertions/deletions in a balls-and-bins game on  $n$  bins, with no more than  $M$  balls present at a time. Let  $G_1$  be a balls-and-bins game with operation-sequence  $S$ , let  $\Delta = c\epsilon^{-2} \log M$ , and let  $G_2$  be  $\Delta$ -generalized stone game with operation sequence  $\phi(S)$ .*

*If  $G_1$  is implemented using the GENERALIZEDMODULATEDGREEDY algorithm with parameters  $M, c$  and  $\epsilon$ , then there exists a coupling between  $G_1$  and  $G_2$  such that: (1) the number of balls with a given color  $k \in [n]$  in  $G_1$  always equals the number of active-bag stones with color  $k$  in  $G_2$ ; and (2) the total number  $n \cdot Q$  of stones in  $G_2$  always satisfies  $Q = \lceil m/n \rceil + \Delta$ , where  $m$  is the largest number of balls ever present at once so far in the balls-and-bins game.*

*Proof.* Let  $\ell_k$  denote the number of balls with color  $k$  at any given moment and let  $\bar{\ell} = \sum_k \ell_k/n$ . By Lemma 2 (modified so that  $T = \lceil m/n \rceil + \Delta - \bar{\ell}$  and  $T_k = \lceil \frac{m}{n} \rceil + \Delta - \ell_k$ ), we know that, on any given insertion in which GENERALIZEDMODULATEDGREEDY does not create a corrupted ball, each color  $k$  is selected with probability

$$\frac{T_k}{n \cdot T} = \frac{\lceil \frac{m}{n} \rceil + \Delta - \ell_k}{n \cdot T}. \quad (15)$$

On the other hand, on insertions that do create corrupted balls, we have by design that (15) is still the probability of color  $k$  being selected. Thus, (15) is always the probability of any given color  $k$  being selected on any given insertion.

Next we turn our attention to the generalized stone game. By design, the number  $n \cdot Q$  of stones in the generalized stone game at any given moment satisfies  $Q = \lceil m/n \rceil + \Delta$ , where  $m$  is the largest number of balls that have ever been present at once in the balls-and-bins game. Suppose that, for each color  $k$  there are  $\ell_k$  stones with color  $k$  in the active set of the stone game. Then on any given activation, the probability of a ball with color  $k$  being moved into the active set is

$$\frac{Q - \ell_k}{n \cdot Q - \sum_i \ell_i} = \frac{\lceil \frac{m}{n} \rceil + \Delta - \ell_k}{n \cdot (\lceil \frac{m}{n} \rceil + \Delta - \bar{\ell})} = \frac{\lceil \frac{m}{n} \rceil + \Delta - \ell_k}{n \cdot T}. \quad (16)$$



The two probabilities (15) and (16) are precisely equal. Thus, we can couple the games so that the color selected by the insertion in the balls-and-bins game is the same as the stone color selected by the activation in the stone game.

If we implement the insertions/activations in this way, then the deletions/deactivations also become coupled: whenever a ball is deleted with a color  $k$ , a stone with color  $k$  is removed from the active bag (in particular, the ball and stone were assigned to have the same color when they were inserted/activated previously). Thus the proof of the lemma is complete.  $\square$

Combining Lemmas 25 and 29, we can bound the probability that a given ball is corrupted.

**Lemma 26** (Corruption probability). *Consider a sequence of insertions/deletions in a balls-and-bins game on  $n$  bins with no more than  $M$  balls ever present at a time, and suppose that insertions are implemented using the GENERALIZEDMODULATEDGREEDY algorithm with parameters  $M$  and  $\epsilon$ . For any given insertion, the probability that the ball being inserted is corrupted is at most  $1/\text{poly}(M)$ .*

*Proof.* For  $k \in [n]$ , let  $\ell_k$  denote the number of balls with color  $k$ . Let  $\bar{\ell} = \sum_k \ell_k/n$  and let  $\Delta = c\epsilon^{-2} \log M$ , where  $c$  is the constant used by GENERALIZEDMODULATEDGREEDY. In order for the inserted ball to be corrupted, we would need

$$\left( \max_k \ell_k \right) - \left( \min_k \ell_k \right) > \epsilon T = \epsilon(\lceil m/n \rceil + \Delta - \bar{\ell}). \quad (17)$$

If we couple the process to a  $\Delta$ -generalized stone game as in Lemma 25, then we have (1) that the number of balls with each color  $k$  in the active bag of the generalized stone game is  $\ell_k$ ; and (2) that the total number of stones in the generalized stone game is  $n(\lceil m/n \rceil + \Delta)$ . It follows by Lemma 29 that, w.h.p. in  $M$ ,

$$(1 - \epsilon/2)\mathbb{E}[s_k] \leq s_k \leq (1 + \epsilon/2)\mathbb{E}[s_k],$$

where  $s_k = \lceil m/n \rceil + \Delta - \ell_k$  and  $\mathbb{E}[s_k] = \lceil m/n \rceil + \Delta - \bar{\ell}$ . That is, each  $s_k$  deviates by at most  $\frac{1}{2}\epsilon(\lceil m/n \rceil + \Delta - \bar{\ell})$  from its mean. The same holds for each  $\ell_k$  (as  $\ell_k + s_k$  is fixed), which implies that (17) does not occur.  $\square$

Finally, we can prove Theorem 23.

*Proof of Theorem 23.* It suffices to prove the Bounded Load guarantee, since the Bounded Bias guarantee is hardcoded into the GENERALIZEDMODULATEDGREEDY algorithm by design. In particular, given the bin choices  $i, j$ , if the ball is not corrupted then  $|\ell_i - \ell_j| \leq \epsilon T$  and it is assigned to bin  $i$  with probability  $1/2 + (\ell_j - \ell_i)/2T \leq 1/2 + \epsilon/2$ . On the other hand if it is corrupted, then it is assigned uniformly.

Let  $\Delta = c\epsilon^{-2} \log M$ . Couple the balls-and-bins game to the  $\Delta$ -generalized stone game as in Lemma 25, and consider the state of both systems at some fixed point in time.

By Lemma 26, we have with high probability in  $M$  that there are no corrupted balls in the balls-and-bins game. Thus the number of balls in any given bin  $k$  (in the balls-and-bins game) is equal to the number of active-bag stones with color  $k$  (in the generalized stone game). Moreover, if  $m$  is the most balls that were ever present in the balls-and-bins game, the number of stones in the generalized stone game is  $\lceil m/n \rceil + \Delta$ .

Using  $\ell_k$  to be the number of active-bag stones with color  $k$ , and  $s_k$  to be the number of inactive-bag stones with color  $k$ , by Lemma 29 we have that  $s_k > (1 - \epsilon)\mathbb{E}[s_k] \geq (1 - \epsilon)\Delta$ , which gives the desired bound

$$\ell_k = \lceil m/n \rceil + \Delta - s_k \leq \lceil m/n \rceil + \epsilon\Delta = m/n + O(\epsilon^{-1} \log M). \quad \square$$

### 5.3 Extensions

We conclude the section with applications of GENERALIZEDMODULATEDGREEDY to several more general settings.

**$(1 + \beta)$ -choice process.** The  $(1 + \beta)$ -choice setting was proposed by Peres, Talwar, and Wieder [PTW10b] as a useful generalization of the 2-choice process, where each insertion selects a random bin with probability  $(1 - \beta)$ , and gets to choose between two random bins  $i, j$  with probability  $\beta$ . For any fixed  $\beta < 1$ , they showed that in the insertion-only case, the GREEDY algorithm achieves maximum load  $m/n + \Theta(\beta^{-1} \log n)$  with high probability in  $n$ ; this load becomes  $m/n + \Theta(\beta^{-1} \log m)$  if one wishes for a high-probability guarantee in  $m$ . They further proved that these bounds are optimal for any  $(1 + \beta)$ -choice insertion strategy.

We can directly use GENERALIZEDMODULATEDGREEDY to construct an optimal  $(1 + \beta)$ -choice insertion strategy for the insertion/deletion model.

**Theorem 27.** *Consider a balls-and-bins game with  $n$  bins and with no more than  $m$  balls present at a time. In the insertion/deletion model, there exists a  $(1 + \beta)$ -choice algorithm that at any given moment, with probability in  $m$ , has maximum load*

$$m/n + O(\beta^{-1} \log m).$$

*Proof.* If we set  $\varepsilon = \beta/2$ , then GENERALIZEDMODULATEDGREEDY selects between bins  $i, j$  with a probabilities in the range  $1/2 \pm \varepsilon$ ; this is equivalent to selecting a random bin (i.e., a random one of  $i, j$ ) with probability  $1 - 2\varepsilon = 1 - \beta$ , and then selecting between bins  $i, j$  with a probabilities in the range  $[0, 1]$ . □

**Graphical-Allocation.** Graphical allocation is another generalization of the 2-choice model, introduced by Kenthapadi and Panigrahy [KP06]. Here we are given an arbitrary fixed  $d$ -regular graph  $G$  on  $n$  vertices (i.e., bins). To assign a ball to a bin, we select a uniformly random edge  $e = (v_1, v_2)$  choose one of bins  $v_1, v_2$ . The classic 2-choice process corresponds to the complete graph  $G = K_n$ .

Bansal and Feldheim [BF22] showed that, in the insertion-only case, it is possible to guarantee a maximum load of  $m/n + O((d/k) \log^4 n \log \log n)$  w.h.p. in  $n$ , where  $k$  is the edge-connectivity of  $G$ . The linear dependence on  $(d/k)$  is necessary and the bound becomes  $m/n + O((d/k) \log m \log^3 n \log \log n)$  if one requires the bound to be w.h.p. in  $m$ .

Their algorithm reduces the problem, in a black-box manner, to that of constructing a  $(1 + \beta)$ -choice strategy on two bins (in particular, where the two “bins” represent sibling sets in a binary hierarchical decomposition of the vertices of  $G$ , and the different sibling pairs use different choices for  $\beta$ , see [BF22]). In the insertion-only case [BF22], they use the GREEDY  $(1 + \beta)$ -choice strategy—to extend this to handle deletions, we can simply use GENERALIZEDMODULATEDGREEDY instead (as in Theorem 27). Together with the framework developed in [BF22], this gives the following result.

**Theorem 28.** *Consider a graphical process where, given a  $k$ -edge-connected  $d$ -regular graph  $G$  on  $n$  vertices (i.e., bins), the two bin choices for each ball are given by the endpoints of a uniformly random edge  $e = (v_1, v_2)$  of  $G$ . Consider any sequence of insertions/deletions where the number of balls in the system never exceeds  $m$ . Then it is possible to guarantee a maximum load of  $m/n + O((d/k) \log m \log^3 n \log \log n)$  w.h.p. in  $m$ , at any given moment.*

## A Proof of Lemma 10

We prove Lemma 10, reformulated here to use a constant  $c$  in place of constants  $\varepsilon_1, \varepsilon_2$ , and to use a variable  $k$  in place of  $\varepsilon_2 m$ :

**Lemma 29** (Lemma 10 reformulated). *Let  $c > 0$  be a sufficiently large constant. Consider the GREEDY algorithm on 4 bins, and fix an arbitrary initial state in which the bins have loads within  $k$  of each other. If  $ck$  insertions are performed, then after the sequence is complete, all of the bins have loads within  $O(\log k)$*

of each other with high probability in  $k$ . Furthermore, with high probability in  $k$ , there is some intermediate point in time during which all of the bins have equal loads.

We break the proof of this lemma into a few simple claims.

**Claim 30.** *Given an arbitrary initial state with bin loads within  $k$  of each other, if  $j \geq ck$  insertions are performed, then at end of the sequence, the bin loads will be within  $O(\log k)$  of each other, w.h.p. in  $k$ .*

*Proof.* Let  $D_{i,j}$  be the difference between the loads of the  $i$ -th and  $j$ -th bins (where  $i \neq j$ ). It suffices to show that, after the insertions are complete,  $D_{i,j} \leq O(\log k)$  with high probability in  $k$ .

Notice that whenever  $D_{i,j} \neq 0$  and we insert a ball,  $D_{i,j}$  has a random increment with  $\Omega(1)$  bias towards 0 (it surely decreases by 1 when  $i, j$  are the two choices, which has  $\Omega(1)$  probability as  $n = 4$ , and has zero bias otherwise). So starting at  $|D_{i,j}| \leq k$ , w.h.p. in  $k$  that the random walk thus reaches 0 within  $O(k) \leq ck$  steps. Moreover, each time that the random walk hits 0, w.h.p. in  $k$  it will hit 0 again within  $O(\log k)$  steps. Thus, after the  $ck$  insertions are performed, we have  $|D_{i,j}| = O(\log k)$  w.h.p. in  $k$ .  $\square$

Next we show that, during the insertions, the loads become equal at some point with probability  $\Omega(1)$ .

**Claim 31.** *Given any arbitrary initial state the bin loads within  $k$  of each other, if  $2ck$  insertions are performed, then with probability at least  $\Omega(1)$  there is some time at which all the 4 bins have equal loads.*

*Proof.* This follows by iterated applications of Claim 30. After  $ck$  insertions, all the 4 the bins have loads within  $T_1 = O(\log k)$  of each other, w.h.p. in  $k$ . After  $cT_1$  further insertions, the bins have loads within  $T_2 = O(\log T_1)$  of each other, w.h.p. in  $T_1$ . After  $cT_2$  further insertions, the bins have loads within  $T_3 = O(\log T_2)$  of each other, w.h.p. in  $T_2$ . Continuing like this, after  $c(k + T_1 + T_2 + \dots + T_{O(\log^* n)}) = (c + o(1))k$  insertions, we reach a state where all bin loads are within  $O(1)$  of each other with probability  $\Omega(1)$ . Once this occurs, we have with probability  $\Omega(1)$  that during the next  $O(1)$  insertions after that, there is a point at which the 4 bins have equal loads.  $\square$

Finally, we amplify Claim 31 in order to achieve a high-probability bound.

**Claim 32.** *Given an arbitrary initial state with bin loads within  $k$  of each other, if  $ck$  insertions are performed, then w.h.p. in  $k$  there is some time when all the bins have equal loads.*

*Proof.* By Claim 30, w.h.p. in  $k$  the loads are within  $T = O(\log k)$  of each other during each of the final  $ck/2$  insertions. Break these insertions into  $\Omega(k/\log k)$  chunks of size  $2cT$ . Within each chunk, we have by Claim 31 that the loads equalize (at some point) with probability at least  $\Omega(1)$ . Thus, the probability that the loads stay unequal during all  $\Omega(k/\log k)$  chunks is  $\exp(-\Omega(k/\log k))$ .  $\square$

Combined, Claims 30 and 32 imply Lemma 29.

## B Proof of Lemma 19

For  $(i, j) \in Q$ , define  $A_{i,j}$  (resp.  $B_{i,j}$ ) to be the set of balls in  $A$  (resp.  $B$ ) that hash to the bin pair  $(i, j)$ . Let  $a_{i,j} = |A_{i,j}|$  and  $b_{i,j} = |B_{i,j}|$ . Let

$$p_{i,j} = \frac{v(A_{i,j} \cup B_{i,j})}{|A_{i,j} \cup B_{i,j}|}$$

denote the (random) fraction of balls in  $A_{i,j} \cup B_{i,j}$  that are placed into bins 1, 2.

We remark that there are two sources of randomness in this lemma: the first, which we denote by  $\mathcal{R}_1$ , is the outcome of the hashes of the balls in  $A$  and  $B$  (i.e., the random bits that determine  $\{a_{i,j}\}$  and  $\{b_{i,j}\}$ ); the second, which we denote by  $\mathcal{R}_2$ , is the random order in which the balls  $A \cup B$  are inserted into the system.

Note that, from the perspective of the ID-oblivious insertion strategy, the balls  $A_{i,j}$  are indistinguishable from the balls  $B_{i,j}$  (this is due to the randomness from  $\mathcal{R}_2$ ). Thus we have that, for any fixed outcome of  $\mathcal{R}_1$ ,

$$\mathbb{E}[v(A_{i,j}) - v(B_{i,j}) \mid \mathcal{R}_1] = \mathbb{E}[p_{i,j}(a_{i,j} - b_{i,j}) \mid \mathcal{R}_1].$$

Summing over  $(i,j) \in Q$ , we have that (again for any fixed outcome of  $\mathcal{R}_1$ )

$$\mathbb{E}[v(A) - v(B) \mid \mathcal{R}_1] = \sum_{(i,j) \in Q} \mathbb{E}[v(A_{i,j}) - v(B_{i,j}) \mid \mathcal{R}_1] = \sum_{(i,j) \in Q} \mathbb{E}[p_{i,j}(a_{i,j} - b_{i,j}) \mid \mathcal{R}_1].$$

Considering all outcomes for  $\mathcal{R}_1$  that satisfy  $\mathcal{E}$ , it follows that

$$\mathbb{E}[v(A) - v(B) \mid \mathcal{E}] = \sum_{(i,j) \in Q} \mathbb{E}[p_{i,j}(a_{i,j} - b_{i,j}) \mid \mathcal{E}].$$

Thus, to prove the lemma, it suffices to show that

$$\mathbb{E} \left[ \sum_{(i,j) \in Q} p_{i,j}(a_{i,j} - b_{i,j}) \mid \mathcal{E} \right] \geq t - O(\sqrt{k}).$$

Note that  $p_{(1,2)} = 1$  and  $p_{(3,4)} = 0$  deterministically. Moreover,

$$\mathbb{E}[a_{1,2} - b_{1,2} \mid \mathcal{E}] \geq \mathbb{E}[k/12 + t - O(\sqrt{k}) - b_{1,2}] = t - O(\sqrt{k}) - \mathbb{E}[b_{1,2} - k/12] = t - O(\sqrt{k}).$$

Thus

$$\begin{aligned} \mathbb{E} \left[ \sum_{(i,j) \in Q} p_{i,j}(a_{i,j} - b_{i,j}) \mid \mathcal{E} \right] &= \mathbb{E}[a_{1,2} - b_{1,2} \mid \mathcal{E}] + \mathbb{E} \left[ \sum_{(i,j) \in Q \setminus \{(1,2), (3,4)\}} p_{i,j}(a_{i,j} - b_{i,j}) \mid \mathcal{E} \right] \\ &= t - O(\sqrt{k}) + \mathbb{E} \left[ \sum_{(i,j) \in Q \setminus \{(1,2), (3,4)\}} p_{i,j}(a_{i,j} - b_{i,j}) \mid \mathcal{E} \right] \\ &\geq t - O(\sqrt{k}) - \sum_{(i,j) \in Q \setminus \{(1,2), (3,4)\}} \mathbb{E}[|a_{i,j} - b_{i,j}| \mid \mathcal{E}]. \end{aligned}$$

To complete the proof, it suffices to show that for each  $(i,j) \in Q \setminus \{(1,2), (3,4)\}$ , we have

$$\mathbb{E}[|a_{i,j} - b_{i,j}| \mid \mathcal{E}] \leq O(\sqrt{k}).$$

Let  $\alpha_{i,j} = \mathbb{E}[a_{i,j} \mid \mathcal{E}]$  and  $\beta_{i,j} = \mathbb{E}[b_{i,j} \mid \mathcal{E}]$ . By Chernoff bounds, we know that  $\mathbb{E}[|a_{i,j} - \alpha_{i,j}| \mid \mathcal{E}] \leq O(\sqrt{k})$  and  $\mathbb{E}[|b_{i,j} - \beta_{i,j}| \mid \mathcal{E}] \leq O(\sqrt{k})$ . Thus, it suffices to show that

$$|\alpha_{i,j} - \beta_{i,j}| = O(\sqrt{k}).$$

For each ball  $x \in A$  with  $h(x) \notin \{(1,2), (3,4)\}$ , we have that  $h(x)$  is random among the  $|Q| - 2 = 10$  pairs in  $Q \setminus \{(1,2), (3,4)\}$ ; and for each ball  $x \in B$ , we have that  $h(x)$  is random among the  $|Q| = 12$  pairs in  $Q$ . Thus  $\alpha_{i,j} = \mathbb{E}[\frac{1}{10}(k - a_{1,2} - a_{3,4}) \mid \mathcal{E}]$  and  $\beta_{i,j} = k/12$ . Finally, as  $a_{1,2} + a_{3,4} = k/6 \pm O(\sqrt{k})$  (conditioned on event  $\mathcal{E}$  occurring), we get

$$\alpha_{i,j} - \beta_{i,j} = \mathbb{E} \left[ \frac{1}{10}(k - a_{1,2} - a_{3,4}) \mid \mathcal{E} \right] - k/12 = \frac{1}{10}(k - k/6) - k/12 \pm O(\sqrt{k}) = \pm O(\sqrt{k}),$$

which completes the proof.

## References

- [ABKU94] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. In *Symposium on theory of computing (STOC)*, pages 593–602, 1994.
- [AKT21] Anders Aamand, Jakob Bæk Tejs Knudsen, and Mikkel Thorup. Load balancing with dynamic set of balls and bins. In *Symposium on Theory of Computing (STOC)*, pages 1262–1275, 2021.
- [BCE<sup>+</sup>12] Petra Berenbrink, Artur Czumaj, Matthias Englert, Tom Friedetzky, and Lars Nagel. Multiple-choice balanced allocation in (almost) parallel. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 411–422. Springer, 2012.
- [BCFC<sup>+</sup>21a] Michael A Bender, Alex Conway, Martín Farach-Colton, William Kuszmaul, and Guido Tagliavini. All-purpose hashing. *arXiv preprint arXiv:2109.04548*, 2021.
- [BCFC<sup>+</sup>21b] Michael A Bender, Alex Conway, Martín Farach-Colton, William Kuszmaul, and Guido Tagliavini. Tiny pointers. *arXiv preprint arXiv:2111.12800*, 2021.
- [BCSV00] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: the heavily loaded case. In *Symposium on Theory of Computing (STOC)*, pages 745–754, 2000.
- [BF22] Nikhil Bansal and Ohad Feldheim. Well-balanced allocation on general graphs. In *Symposium on Theory of Computing (STOC) (to appear)*, 2022.
- [BFCKK22] Michael A Bender, Martín Farach-Colton, John Kuszmaul, and William Kuszmaul. On the optimal time/space tradeoff for hash tables. In *Symposium on Theory of Computing (STOC) (to appear)*, 2022.
- [BFHM08] Petra Berenbrink, Tom Friedetzky, Zengjian Hu, and Russell Martin. On weighted balls-into-bins games. *Theoretical Computer Science*, 409(3):511–520, 2008.
- [BFK<sup>+</sup>16] Petra Berenbrink, Tom Friedetzky, Peter Kling, Frederik Mallmann-Trenn, Lars Nagel, and Christopher Wastell. Self-stabilizing balls & bins in batches: The power of leaky bins. In *Symposium on Principles of Distributed Computing (PODC)*, pages 83–92, 2016.
- [BFK<sup>+</sup>18] Petra Berenbrink, Tom Friedetzky, Peter Kling, Frederik Mallmann-Trenn, Lars Nagel, and Chris Wastell. Self-stabilizing balls and bins in batches. *Algorithmica*, 80(12):3673–3703, 2018.
- [BL12] Graham Brightwell and Malwina Luczak. The supermarket model with arrival rate tending to one. *arXiv preprint arXiv:1201.5523*, 2012.
- [BLP10] Maury Bramson, Yi Lu, and Balaji Prabhakar. Randomized load balancing with general service time distributions. *ACM SIGMETRICS performance evaluation review*, 38(1):275–286, 2010.
- [BM01] Andrei Broder and Michael Mitzenmacher. Using multiple hash functions to improve ip lookups. In *Conference on Computer Communications (INFOCOM)*, volume 3, pages 1454–1463. IEEE, 2001.
- [BMP<sup>+</sup>06] Flavio Bonomi, Michael Mitzenmacher, Rina Panigrahy, Sushil Singh, and George Varghese. An improved construction for counting bloom filters. In *European Symposium on Algorithms (ESA)*, pages 684–695. Springer, 2006.
- [CFM<sup>+</sup>98] Richard Cole, Alan Frieze, Bruce M. Maggs, Michael Mitzenmacher, Andréa W Richa, Ramesh Sitaraman, and Eli Upfal. On balls and bins with deletions. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 145–158. Springer, 1998.
- [CMadH<sup>+</sup>98] Richard Cole, Bruce M. Maggs, Friedhelm Meyer auf der Heide, Michael Mitzenmacher, Andréa W. Richa, Klaus Schröder, Ramesh K. Sitaraman, and Berthold Vöcking. Randomized protocols for low congestion circuit routing in multistage interconnection networks. In *Symposium on Theory of Computing (STOC)*, pages 378–388. ACM, 1998.
- [DB13] Jeffrey Dean and Luiz André Barroso. The tail at scale. *Communications of the ACM*, 56(2):74–80, 2013.
- [EG16] Patrick Eschenfeldt and David Gamarnik. Supermarket queueing system in the heavy traffic regime. short queue dynamics. *arXiv preprint arXiv:1610.03522*, 2016.

- [FGG21] Ohad N Feldheim and Ori Gurel-Gurevich. The power of thinning in balanced allocation. *Electronic Communications in Probability*, 26:1–8, 2021.
- [FK15] Alan Frieze and Michal Karonski. *Introduction to Random Graphs*. Cambridge University Press, 2015.
- [FMMM09] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and Shan Muthukrishnan. Online stochastic matching: Beating  $1-1/e$ . In *Symposium on Foundations of Computer Science (FOCS)*, pages 117–126. IEEE, 2009.
- [FNP04] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–19. Springer, 2004.
- [HMZ11] Bernhard Haeupler, Vahab S Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In *International Workshop on Internet and Network Economics*, pages 170–181. Springer, 2011.
- [KP06] Krishnaram Kenthapadi and Rina Panigrahy. Balanced allocation on graphs. In *Symposium on Discrete Algorithms (SODA)*, volume 6, pages 434–443, 2006.
- [LM06] Malwina J Luczak and Colin McDiarmid. On the maximum queue length in the supermarket model. *The Annals of Probability*, 34(2):493–527, 2006.
- [LN05] Malwina J Luczak and James Norris. Strong approximation for the supermarket model. *The Annals of Applied Probability*, 15(3):2038–2061, 2005.
- [LPY19] Christoph Lenzen, Merav Parter, and Eylon Yogev. Parallel balanced allocations: The heavily loaded case. In *Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 313–322, 2019.
- [LS22] Dimitrios Los and Thomas Sauerwald. Balanced allocations with incomplete information: The power of two queries. In *Innovations in Theoretical Computer Science Conference (ITCS)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.
- [LSS22] Dimitrios Los, Thomas Sauerwald, and John Sylvester. Balanced allocations: Caching and packing, twinning and thinning. In *Symposium on Discrete Algorithms (SODA)*, pages 1847–1874. SIAM, 2022.
- [MBVLW18] Debankur Mukherjee, Sem C Borst, Johan SH Van Leeuwen, and Philip A Whiting. Universality of power-of-d load balancing in many-server systems. *Stochastic Systems*, 8(4):265–292, 2018.
- [Mit99] Michael Mitzenmacher. Studying balanced allocations with differential equations. *Combinatorics, Probability and Computing*, 8(5):473–482, 1999.
- [Mit01] Michael Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, 2001.
- [MRS01] Michael Mitzenmacher, Andrea W. Richa, and Ramesh Sitaraman. The power of two random choices: A survey of techniques and results. *Combinatorial Optimization*, 9:255–304, 2001.
- [ORS<sup>+</sup>11] Diego Ongaro, Stephen M Rumble, Ryan Stutsman, John Ousterhout, and Mendel Rosenblum. Fast crash recovery in ramcloud. In *Symposium on Operating Systems Principles (SOSP)*, pages 29–41, 2011.
- [OWZS13] Kay Ousterhout, Patrick Wendell, Matei Zaharia, and Ion Stoica. Sparrow: distributed, low latency scheduling. In *Symposium on Operating Systems Principles (SOSP)*, pages 69–84, 2013.
- [PR04] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004.
- [PTW10a] Yuval Peres, Kunal Talwar, and Udi Wieder. The  $(1+\beta)$ -choice process and weighted balls-into-bins. In *Symposium on Discrete Algorithms (SODA)*, pages 1613–1619. SIAM, 2010.
- [PTW10b] Yuval Peres, Kunal Talwar, and Udi Wieder. The  $(1+\beta)$ -choice process and weighted balls-into-bins. In *Symposium on Discrete Algorithms (SODA)*, pages 1613–1619. SIAM, 2010.
- [PTW10c] Yuval Peres, Kunal Talwar, and Udi Wieder. The  $(1+\beta)$ -choice process and weighted balls-into-bins. In *Symposium on Discrete Algorithms (SODA)*, pages 1613–1619. SIAM, 2010.

- [Roy82] J. P. Royston. Expected normal order statistics (exact and approximate). *Journal of the Royal Statistical Society*, 31:161–165, 1982.
- [Ste96] Volker Stemmann. Parallel balanced allocations. In *Symposium on Parallel algorithms and Architectures (SPAA)*, pages 261–269, 1996.
- [TW07] Kunal Talwar and Udi Wieder. Balanced allocations: the weighted case. In *Symposium on Theory of Computing (STOC)*, pages 256–265, 2007.
- [TW14] Kunal Talwar and Udi Wieder. Balanced allocations: A simple proof for the heavily loaded case. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 979–990. Springer, 2014.
- [VDK96] Nikita Dmitrievna Vvedenskaya, Roland L’vovich Dobrushin, and Fridrikh Izrailevich Karpelevich. Queueing system with selection of the shortest of two queues: An asymptotic approach. *Problemy Peredachi Informatsii*, 32(1):20–34, 1996.
- [Vöc99] Berthold Vöcking. How asymmetry helps load balancing. In *Foundations of Computer Science (FOCS)*, page 131, 1999.
- [Vöc03] Berthold Vöcking. How asymmetry helps load balancing. *Journal of the ACM (JACM)*, 50(4):568–589, 2003.
- [Wie17] Udi Wieder. Hashing, load balancing and multiple choice. *Foundations of Computer Science (FOCS)*, 12:275–379, 2017.
- [YYRC08] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*, 38(2):17–29, 2008.