Abstract

Lars Nagel – Randomised Load Balancing

Due to the increased use of parallel processing in networks and multi-core architectures, it is important to have load balancing strategies that are highly efficient and adaptable to specific requirements. Randomised protocols in particular are useful in situations in which it is costly to gather and update information about the load distribution (*e.g.* in networks).

For the mathematical analysis randomised load balancing schemes are modelled by balls-into-bins games, where balls represent tasks and bins computers. If m balls are allocated to n bins and every ball chooses one bin at random, the gap between maximum and average load is known to grow with the number of balls m. Surprisingly, this is not the case in the multiple-choice process in which each ball chooses $d \geq 2$ bins and allocates itself to the least loaded. Berenbrink *et al.* proved that then the gap remains $\frac{\ln \ln(n)}{\ln(d)}$.

This thesis analyses generalisations and variations of the multiple-choice process. For a scenario in which batches of balls are allocated in parallel, it is shown that the gap between maximum and average load is still independent of m. Furthermore, we look into a process in which only predetermined subsets of bins can be chosen by a ball. Assuming that the number and composition of the subsets can change with every ball, we examine under which circumstances the maximum load is one. Finally, we consider a generalisation of the basic process allowing the bins to have different capacities. Adapting the probabilities of the bins, it is shown how the load can be balanced over the bins according to their capacities.

Randomised Load Balancing

Lars Nagel

School of Engineering and Computing Sciences

University of Durham

A thesis submitted for the degree of

Doctor of Philosophy

June 2011

Contents

List of Figures

List of Tables

Declaration

Parts of the work presented in this thesis have been published in preliminary form [7, 6]. Although these results were developed in collaboration, my contributions to the discussion and development of all results were significant.

No part of the material presented in this thesis has previously been submitted for a degree.

Copyright

Acknowledgment

First of all my thanks go to my supervisors Dr. Tom Friedetzky and Dr. Magnus Bordewich for their support and guidance during my doctoral studies, and to my co-authors Prof. Dr. Petra Berenbrink, Prof. Dr. Andre Brinkmann and Dr. Stefan Dantchev for interesting discussions and fruitful collaboration.

I would like to thank the University of Durham and EPSRC for their support during the last three and a half years, and also the Simon Fraser University where I spent seven months of my studies. I very much enjoyed my time in Durham, Vancouver and at summer schools in Bristol and Lipari where I was fortunate to get to know many great people: Dr. Steven Bergner, Dr. Catarina Carvalho, Dr. Dominic Dumrauf, James Gate, Dr. Ross Kang, Ioannis Lignos, Dr. Barnaby Martin, Dr. Mark Rhodes, Allison Roohi, Dr. Jens Schmidt, Zoe Robertson and no doubt many others.

Finally, I would like to thank my examiners Prof. Dr. Leslie Ann Goldberg and Dr. Matthew Johnson.

Dedicated to my parents, Gisela and Rolf Nagel.

1 Introduction

Load balancing has become a very important subject in computer science. The increasing use of distributed and parallel computing in networks and multi-core architectures make it desirable to further optimise the distribution of workload over computers or processors.

Despite all efforts and research, the widespread use of parallel systems had for a long time been impeded by the exponential growth in processing speed. “Moore’s Law” – that the number of transistors on an integrated circuit doubles every two years [67] – turned out to be accurate: Parallel architectures quickly became obsolete, supercomputers of the preceding decade were suddenly outrun by desktop computers [41]. The end of Moore’s Law was predicted several times, but until today it has stubbornly prevailed for more than four decades.

Yet, now the voices are growing louder that the limit of downsizing chip components will categorically be reached in the 2020s [41]. At some point it must inevitably come to an end because the size of a transistor will certainly not fall below the size of an atom. An additional obstacle is the increasing heat dissipation of processors [39]. Possible solutions like quantum computers or reversible computing appear far away and would drastically change the current computing model [70, 52, 5].

Therefore, multi-core processors and computer clusters will dominate the (near) future, and in fact they already dominate if one considers the widespread use of server clusters in the worldwide web and the sales figures on the computer market. Most (desktop / laptop) PCs currently sold have multiple CPUs and tablets and smartphones follow this trend [81]. There remains a lack of software, however, that makes use of several cores. For this, programs need to be split into processes or threads which are then allocated to the different processors – a new challenge for software developers.

As mentioned, another application area of efficient load balancing schemes are large networks like the internet. Looking at the client-server model, the requirements have changed over the years. Until the mid-90s a single server could handle all requests sent to a popular website while nowadays “there are often dozens or hundreds of servers operating behind a single URL” [16]. Any imbalance in the load distribution will affect the system’s throughput and latency.

A further challenge is to find strategies that also cope with additional requirements. Load balancing in distributed and cloud computing, for instance, may be complicated by the structure of the network. If the computers are widespread over the network, the latency between client and server or peers, respectively, is an issue.

The subject of this thesis is randomised load balancing. This type of load balancing can be applied if there is not enough information available about the global state of the system or if it is difficult or expensive to gather it. A lack of information is usually not a problem in multi-core PCs so that deterministic load balancing schemes should be applied there. Nevertheless, this does not necessarily imply that an optimal strategy can be found. In his famous paper "Reducibility among combinatorial problems" Karp showed that deterministic load balancing of weighted tasks is an NP-hard problem, even in case of only two processors (see PARTITION problem in [44]).

In networks, on the other hand, it can be costly to gain global information on the load of each server or peer and keep it updated. It would bind resources and lead to a communication overhead which should be avoided in networks. Even if the dispatcher reduced its load enquiries to a minimum, it would still be a weak point in the system and could easily become a bottleneck.

The randomised allocation of requests to servers can help to overcome such difficulties and communication overhead. The dispatcher or the client itself needs only a list of server addresses from which it picks one at random. If the server structure is not too dynamic, this list can even be stored locally so that the client does not have to retrieve it for every request. The crucial question is whether random load balancing schemes do their job sufficiently well and whether they are adaptable to additional requirements if necessary.

For the mathematical analysis load balancing schemes are modelled by balls-into-bins games. Balls-into-bins games are also known as allocation processes or occupancy models and have a long history in mathematics and natural sciences [40]. In the client-server scenario the servers are represented by bins and the requests by balls. The allocation of the balls to the bins follows a randomised protocol. The analysis focuses on the load distribution and often merely on the maximum load.

Let m denote the number of balls and n the number of bins. In the most basic setting (i) all balls and bins have unit size, (ii) balls are only added, but not deleted, and (iii) each of the m balls simply chooses one of the n bins at random and allocates itself to it. It is well-known (see *e.g.* [74]) that then the gap between maximum and average load grows with the number of balls m. Surprisingly, if every ball has two random choices and selects the bin of lower load, the number of balls above the average is $\mathcal{O}(\ln \ln(n))$ and, thus, independent of m [8].

The idea of allowing balls multiple choices was first described and analysed by Karp *et al.* [45]

and Azar *et al.* [3]. Their work sparked new interest in balls-into-bins games and led to many variations modelling diverse applications. Load balancing is only one of them.

The problem of the multiple-choice approach is that it is basically sequential. The allocation of the current ball depends on the positions of the previous balls. Since many applications modelled by balls-into-bins games – especially load balancing in networks – are in fact parallel, several attempts were made to parallelise the algorithm [2, 61, 82, 12, 1]. In Chapter 5 we will investigate how the standard multiple-choice algorithm performs in a parallel environment. There we will assume that the balls arrive in batches of size n and that the balls of each batch are allocated concurrently.

Another generalisation of the basic process assumes that the balls are weighted. Translated into the client-server model, this means that requests can have varying sizes or running times. Evidently, the same could hold true for the servers. When, for example, a cluster of servers is extended by new machines, then the new servers will often have differing properties such as memory size and processing speed. It is then desirable to assign more work to the better computers. In Chapter 3 we will look into the related balls-into-bins game in which the bins have different capacities. We will show how altering the bins' probabilities helps to balance the load.

While games with different bin capacities have previously not been analysed, there are a few papers regarding allocation processes in which the bins' probabilities are not uniform. Byers *et al.* [17, 18], for example, apply the multiple-choice paradigm to improve consistent hashing as it is used in peer-to-peer networks like *Chord*, where some probabilities deviate from the average $\frac{1}{n}$ by a factor of $\ln(n)$.

In Chapter 4 we will consider another scenario in which the bins' probabilities are non-uniform and dependent. In this model the servers are grouped in possibly overlapping clusters. A request is not sent to a single bin, but to a random cluster which then allocates the ball to its least loaded bin. We assume that the number and composition of the clusters can change with every ball and show bounds on the maximum load.

1.1 Notation and Terminology

In this section we provide basic definitions and lemmas that will be used throughout the thesis. More special notation will be given where it is needed in the individual chapters.

We assume that the reader is familiar with the basic concepts of combinatorics and probability

theory that are used in the analysis of randomised algorithms. Good introductions into this field are [78, 68, 66].

1.1.1 Basic Definitions

As customary we denote the *real numbers* with $\mathbb{R}$ and the *positive real numbers* including 0 with $\mathbb{R}_+ := \{x \mid x \in \mathbb{R} \wedge x \geq 0\}$. Let $a, b \in \mathbb{R}$. The *open interval* is denoted with $(a, b) := \{x \mid x \in \mathbb{R} \wedge a < x < b\}$, the *closed interval* with $[a, b] := \{x \mid x \in \mathbb{R} \wedge a \leq x \leq b\}$ and the *half-closed intervals* with $[a, b) := \{x \mid x \in \mathbb{R} \wedge a \leq x < b\}$ and $(a, b] := \{x \mid x \in \mathbb{R} \wedge a < x \leq b\}$, respectively. The set of natural numbers $\mathbb{N}$ contains all integer numbers strictly greater than 0. For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \ldots, n\}$.

In Definition 1.2.7 and in Chapter 3 we will refer to normalised vectors:

Definition 1.1.1 (Normalised vector)**.** *Let $v = (v_1, ..., v_n)$ be any vector in $\mathbb{R}^n$. Then the* normalised *vector $\bar{v} = (\bar{v}_1, ..., \bar{v}_n)$ consists of the elements of v in decreasing order.*

In the analysis of the protocols we will use the big-O-notation (see *e.g.* [78]).

Throughout the thesis we say that an event A occurs *with high probability*, or *w.h.p.*, if $\mathbf{Pr}[A] \geq 1 - n^{-\alpha}$ for some constant $\alpha > 0$, and it occurs with *with very high probability*, or *w.v.h.p.*, if $\mathbf{Pr}[A] \geq 1 - n^{-\alpha}$ for any constant $\alpha > 0$.

Since the binomial distribution will be frequently used, the following lemma summarises a few well-known properties (see *e.g.* [78, 66]):

Lemma 1.1.2 (Binomial distribution)**.** *Let X_i, $i \in [n]$, be binary random variables (or Bernoulli experiments) that have the same success probability $p := \mathbf{Pr}[X_i = 1]$ and failure probability $1-p = \mathbf{Pr}[X_i = 0]$. Then $X = \sum_{i=1}^{n} X_i$ is binomially distributed, and we write $X \sim B(n, p)$.*

The expected value is $\mathbb{E}[X] = n \cdot p$, the variance $\mathbf{Var}[X] = n \cdot p \cdot (1 - p)$.

The probability for at least k successes is

$$\mathbf{Pr}[X \geq k] = \sum_{j=k}^{n} \binom{n}{j} \cdot p^j \cdot (1-p)^{n-j} \leq \binom{n}{k} \cdot p^k \leq \left(\frac{e \cdot n \cdot p}{k}\right)^k.$$

We will also make use of the following well-known lemmas:

Lemma 1.1.3 (Geometric series; see [50], page 59)**.** *If $x \in \mathbb{R}$ and $|x| < 1$, then*

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}.$$

Lemma 1.1.4 (Property of the exponential function; see [50], page 111)**.** *For all* $x \in \mathbb{R}$ *and* $n > 0$*,*

$$\left(1 + \frac{x}{n}\right)^n \leq e^x.$$

Lemma 1.1.5. *For all* $x > 0$*,*

$$\frac{1}{x+1} < \ln(x+1) - \ln(x) < \frac{1}{x}.$$

Proof. Since $\ln$ is a strictly concave function, it holds for $h \neq 0$ that

$$\ln(x+h) - \ln(x) < h \cdot (\ln(x))' = h \cdot \frac{1}{x}$$

(see *e.g.* [50], page 166). The first inequality follows from setting $h = -1$ and multiplying by -1, the second inequality from setting $h = 1$. □

1.1.2 Graphs

Occasionally we will mention undirected graphs and hypergraphs. Only definitions relevant for this thesis are given here; for a comprehensive introduction see [25, 78]. Most of the definition in this Section are based on [25].

A (simple) *graph* $G = (V, E)$ is an ordered pair of sets V and E where V is the non-empty set of *vertices* and E is the set of *edges*. An edge is a set $\{u, v\}$ of two vertices $u, v \in V$. If $\{u, v\} \in E$, then u and v are called *adjacent*. Define $e := \{u, v\}$, then we say that u and v are *incident* to e. The *order* of a graph is the number of vertices, usually denoted by $n := |V|$. A *multigraph* is an extension of the simple graph that allows for multiple edges; *i.e.*, E is a multiset.

The *neighbourhood* $N(v)$ of a vertex $v \in V$ is the set of vertices that v is adjacent to. The *degree* $d(v)$ of vertex v is the number edges v is incident to. In simple graphs the degree equals the size of the neighbourhood. A graph $G = (V, E)$ is called *regular* if all vertices $v \in V$ have the same degree. In a Δ*-regular graph* all vertices have degree Δ.

$G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subset V$, $E' \subset E$ and if $\{u, v\} \in E'$ implies $u, v \in V'$. G' is an *induced subgraph* if additionally $\{u, v\} \in E$ and $u, v \in V'$ imply $\{u, v\} \in E'$. A *walk* is an alternating sequence of vertices and edges, starting and ending with a vertex, such that each edge connects the preceding vertex with the succeeding vertex. The *length* of a walk is the number of edges in its sequence. A *path* is a walk in which the vertices (and edges) are pairwise distinct. A *cycle* is defined as a path whose sequence is extended by an edge that connects the start and end vertex of the path. Two vertices $u, v \in V$ are *connected* if there

exists a path between u and v. The *distance* between two vertices is the length of the shortest path between them. A graph is *connected* if each two vertices are connected.

A *tree* is a cycle-free, connected graph. Vertices of degree 1 are called leaves; all other vertices are *inner vertices* of the tree. If one fixed vertex of a tree is labelled as the *root*, then the tree is also called a *rooted tree*. Consider the path from the root to any leaf and fix any vertex v on this path. The vertex preceding v is called the *parent* of v, the parent's parent the grandparent of v and so on. The vertex succeeding v is a *child* of v, the vertex succeeding the child v's grandchild, and so on. As there is only one path from the root to v (otherwise the graph would not be cycle-free), v has exactly one parent, unless v itself is the root (then it has no parent). On the other hand, v can have any number of children. If v has no children, then v is a leaf. The *depth* of a vertex is the length of the path from the root to this vertex. The *height* of a tree is the maximum depth in the tree. A rooted tree is a *k-ary tree* if every inner vertex has exactly k children.

A graph is *bipartite* if the vertex set V can be partitioned into two sets V_1 and V_2 such that the vertices of every edge $\{u, v\} \in E$ are in different sets. Thus, either $u \in V_1$ and $v \in V_2$ or $u \in V_2$ and $v \in V_1$.

A *hypergraph* $G = (V, E)$ is a generalisation of a graph and specified by a set V of vertices and a set E of hyperedges. A *hyperedge* is a non-empty set of vertices whose size is not restricted. Thus, it can connect any number of (distinct) vertices. Similar to multigraphs, in a *multi-hypergraph* E is a multiset. A hypergraph or multi-hypergraph is *d-uniform* if every hyperedge in E consists of exactly d vertices.

1.1.3 Randomised Algorithms

A *randomised algorithm* is an algorithm that makes decisions during its execution that depend on random numbers (or bits). Therefore, these algorithms are non-deterministic in nature, and, if run on the same input twice, their behaviour can differ. This can concern the sequence of operations, the running time and, dependent on the type of algorithm, even the return value.

Ideally the random numbers are provided by a device or program that generates perfectly random numbers without delay. Hardware solutions that generate random numbers from physical processes approximate this idea. However, for the majority of applications pseudo-random number generators, implemented in software, are also sufficient. In the analytical sections we assume perfect randomness, whereas the simulations described in Section 3.4 have been implemented and executed using the pseudo-random number generator "Mersenne Twister" [59].

A great advantage of randomised algorithms lies in their robustness against unfavourable in-

puts which leads to expected running times that are often much better than the worst case running times of comparable deterministic algorithms. Even though the concept of randomisation appears more complicated at first glance, randomised algorithms are often not only more efficient than deterministic algorithms, but also easier to understand and to implement [68]. Unfortunately the same does not hold for the analysis of these algorithms which may be more complicated than in the deterministic case.

In the subsequent chapters we investigate randomised protocols for balls-into-bins processes in which balls are randomly allocated to a set of bins. Here, we are not interested in running times, but in the distribution of the balls over the bins.

1.1.4 Balls-into-bins Games

A *balls-into-bins game* is defined by a set of balls, a set of bins and a (randomised) protocol that describes how the balls are allocated to the bins. Balls-into-bins games are also known as *occupancy models* or *allocation processes*. The number of balls is denoted by m and the number of bins by n. We will consider the basic setting first in which all balls and bins have unit size. Further below we will explain processes with weighted balls and bins.

The *load* of a bin is the number of balls it contains. Assuming that balls are allocated sequentially, a ball's *height* or *level* is the load of the selected bin right after the allocation. Thus, one can picture the bin as a stack of balls and every new ball is simply put on top of the stack. If balls arrive at the same time, then we nevertheless assume that they are added to the stack one after the other (in an arbitrary order) so that each ball has a unique height.

The *load distribution* describes how the balls are distributed over the bins. The *load vector* of an allocation of balls into n bins is a vector $L = (\ell_1, ..., \ell_n)$ where ℓ_i is the load of bin i. The *normalised load vector* $\bar{L}$ consists of the loads of L in decreasing order. This complies with sorting the array and renaming the bins according to their new positions.

In the *d-choice game* each ball chooses d bins at random and commits itself to one of the least loaded. In case $d = 1$, we will also speak of a *single-choice game*, in case $d \geq 2$, of a *multiple-choice game*. As there can be more than one least loaded bin, a *tie-breaking mechanism* must be defined. If not stated otherwise, we assume that bins are chosen independently and uniformly at random (*i.u.r.*) and that ties are broken arbitrarily. We call this the *standard d-choice game* and the protocol GREEDY$[d]$, following Azar *et al.* [3].

The *access graph* depicts the possible choices of the balls [12, 30]: The bins are represented by the vertices of the graph, the balls by the (hyper)edges, each connecting the d chosen bins of a

ball. The access graph is *labelled* if the vertices and edges are labelled with the IDs or properties of the bins and balls, respectively. Otherwise it is *unlabelled*.

A protocol is *sequential* if one ball is thrown after the other and the loads are updated after each ball. Different *parallel models* are described in the literature [2, 61, 82, 12, 1, 29]. Some of them assume that all balls arrive at the same time (*static model*), others that the balls arrive in *batches* of a fixed size (*dynamic model*). Some of them allow for a possibly limited number of additional *communication rounds* in which the balls and bins can exchange messages, others do not. During a communication round every ball can send one message to each chosen bin, and every bin can send one message to each ball it was chosen by. The permitted communication lines are represented by the *communication graph*, a bipartite graph whose two vertex sets represent the balls and bins, respectively, and whose edges connect balls with their chosen bins.

A distributed load balancing strategy is *non-adaptive* if all choices are made before any communication takes place. A strategy is *symmetric* if all balls and bins run the same protocol and if all choices are made *i.u.r.* It is *asynchronous* if a ball or bin does not have to wait for other balls and bins to receive a message. A *synchronous* protocol has at least one *synchronisation point* when all balls and bins have to wait for a round to finish. This implies some kind of coordination, some notion of global time [2].

We speak of a *finite* or *fixed time* process if the number of balls is limited; otherwise it is *infinite* [3, 1]. The finite process is analysed for a fixed time that is known in advance whereas the infinite process performs arrivals and (possibly) deletions of balls over an infinite time line. Note that the bins do not have to be empty at the beginning and that it can help during the analysis to divide a process into finite sub-processes, especially if the balls are already grouped in batches.

The *waiting time* of a ball is the time (*i.e.*, number of rounds) between its arrival in the system and its processing / deletion. The *service time* defines how much time a bin needs to process / delete one ball. Normally it takes one time step.

In the d-choice game, as we have described it, each bin has the same uniform probability $\frac{1}{n}$ to be the i-th choice of a ball, for all $i \in [d]$. (Naturally the bins' probabilities to *receive* the ball are different in the multiple-choice game because they depend on the current load distribution.) One evident generalisation of the d-choice game allows for different probability distributions over the bins.

Another variation of balls-into-bins game allows that the balls have integer *weights*. The *load* of a bin is then defined as the added weights of the balls in it. As before, the height of a ball is defined as the bin's load immediately after the allocation.

In Chapter 3 we will consider the game in which the balls have unit size, but the bins have

different integer *capacities*. The *load* of a bin with capacity c is the number of balls in the bin divided by c. The probabilities of the bins will be adapted so that bins with higher capacities are more likely to receive balls. For a more detailed description see Section 3.2.

1.2 Proof Techniques

For the analysis of balls-into-bins games several proof techniques have been employed. Most of them are widely used in probability theory and computer science and certainly not restricted to allocation processes. We give an overview and describe the ones in detail that we use in the following chapters.

1.2.1 Tail Bounds

Parameters like the running time of a randomised algorithm or the maximum load in an allocation process are random variables. In many cases it is desirable to prove that, *w.h.p.*, a parameter takes a value that is smaller or greater than a certain bound. Such results are possible because very often the parameters are "concentrated" around their expected value. This means that, for a parameter X, there exists an interval I with $\mathbb{E}[X] \in I$ such that the probability $\mathbf{Pr}\,[\,X \in I\,]$ is (very) high. This phenomenon is referred to as the *concentration of measure* [27].

Several tools have been developed to prove *tail bounds*, *i.e.*, bounds on $\mathbf{Pr}\,[\,X \notin I\,]$. The *tails* are the areas far from the expected value: If $I = (a, b)$, then $\{x \mid x \in \mathbb{R} \wedge x \leq a\}$ is referred to as the *lower tail* and $\{x \mid x \in \mathbb{R} \wedge x \geq b\}$ as the *upper tail* [80]. Accordingly, bounds on the probabilities $\mathbf{Pr}\,[\,X \leq a\,]$ and $\mathbf{Pr}\,[\,X \geq b\,]$ are called *lower* and *upper tail bounds*.

Among others available tools are *Chernoff bounds* and the *inequalities* of *Markov*, *Chebyshev*, *Hoeffding* and *Azuma* (see *e.g.* [68, 66]). Even though all these bounds are based on *Markov's inequality*, they have different requirements. The inequalities of Markov and Chebyshev are relatively weak, but generally applicable. The other bounds are stronger, but restricted to martingales (Azuma), sums of independent and bounded random variables (Hoeffding) or sums of independent (or negatively correlated [72]) Bernoulli distributed random variables (Chernoff).

Tail bounds are frequently used in the analysis of balls-into-bins games; for instance, martingales in [18], Chernoff bounds in [3] and Markov's inequality in [9]. We will apply different types of Chernoff bounds, also a variation for geometrically distributed random variables.

Lemma 1.2.1 (Chernoff bounds; Theorem 4.4 and 4.5 in [66]). *Consider n independent random Bernoulli variables $X_1, \ldots, X_n$ with $\mathbf{Pr}[X_i = 1] = p_i$ and $\mathbf{Pr}[X_i = 0] = 1 - p_i$ for all $1 \leq i \leq n$. Let $X := \sum_{i=1}^{n} X_i$ and $\mu := \mathbb{E}[X] = \sum_{i=1}^{n} p_i$. Then, for $0 < \epsilon \leq 1$,*

$$\mathbf{Pr}[X \geq (1+\epsilon) \cdot \mu] \leq \left(\frac{e^{\epsilon}}{(1+\epsilon)^{1+\epsilon}} \right)^{\mu} \leq e^{-\epsilon^2 \cdot \mu / 3}$$

and, for $0 < \epsilon < 1$,

$$\mathbf{Pr}[X \leq (1-\epsilon) \cdot \mu] \leq \left(\frac{e^{-\epsilon}}{(1-\epsilon)^{1-\epsilon}} \right)^{\mu} \leq e^{-\epsilon^2 \cdot \mu / 2}.$$

As mentioned before, the Bernoulli random variables do not have to be independent. It suffices if they are negatively correlated.

Definition 1.2.2 (Negative correlation [72]). *Let $X_1, \ldots, X_n$ be binary random variables. They are negatively correlated if*

$$\mathbf{Pr}\left[\bigwedge_{i \in I} X_i = 1 \right] \leq \prod_{i \in I} \mathbf{Pr}[X_i = 1]$$

for all $I \subseteq [n]$.

Lemma 1.2.3 (Chernoff bounds, negative correlation [72]). *Let $X_1, \ldots, X_n$ be binary random variables, let $X := \sum_{i=1}^{n} X_i$ and $\mu := \mathbb{E}[X]$. If $X_1, \ldots, X_n$ are negatively correlated, then, for $\epsilon \in (0, 1)$,*

$$\mathbf{Pr}[X \geq (1+\epsilon) \cdot \mu] \leq \left(\frac{e^{\epsilon}}{(1+\epsilon)^{1+\epsilon}} \right)^{\mu} \leq e^{-\epsilon^2 \mu / 3}.$$

The following lemma states a variant of Chernoff-type bounds for geometrically distributed variables (see *e.g.* Theorem 3.2 in [36]). For completeness we give the proof of this result.

Lemma 1.2.4 (Chernoff bounds for geometrically distributed variables, [27, 36]). *Let $Y_1, Y_2, \ldots, Y_t$ be a collection of t independent geometrically distributed random variables with $\mathbf{Pr}[Y_i = j] = (1-p)^{j-1} \cdot p$ and constant parameter p, $0 < p < 1$. Then*

$$\mathbf{Pr}[Y > (1+\delta) \cdot \mathbb{E}[Y]] \leq e^{-\frac{t}{2} \cdot \frac{\delta^2}{1+\delta}}$$

where $\delta > 0$ and $Y = Y_1 + Y_2 + \cdots + Y_t$.

Proof. Let X_i, $i \in \mathbb{N}$, be binary random variables with $\mathbf{Pr}[X_i = 1] = p$ and $\mathbf{Pr}[X_i = 0] = 1 - p$. It is well-known, see *e.g.* [27], that for all $j \in \mathbb{N}$,

$$\mathbf{Pr}\left[\sum_{i=1}^{t} Y_i \leq j\right] = \mathbf{Pr}\left[\sum_{i=1}^{j} X_i \geq t\right]. \tag{1.1}$$

This will allow us to use Chernoff bounds.

The expected value of Y is $\mathbb{E}[Y] = \sum_{i=1}^{t} \mathbb{E}[Y_i] = \frac{t}{p}$. Let $X = \sum_{i=1}^{\lceil (1+\delta)\mathbb{E}[Y] \rceil} X_i$. Then $\mathbb{E}[X] = \lceil (1+\delta)\mathbb{E}[Y] \rceil \cdot p \geq (1+\delta) \cdot t$. Furthermore, let $\epsilon := 1 - \frac{t}{\mathbb{E}[X]}$. Then $0 < \epsilon < 1$ and $\mathbf{Pr}[X < t] = \mathbf{Pr}[X < (1-\epsilon) \cdot \mathbb{E}[X]]$.

Applying (1.1) and Lemma 1.2.1, we get:

$$\begin{aligned}
\mathbf{Pr}[Y > (1+\delta) \cdot \mathbb{E}[Y]] &= \mathbf{Pr}[X < (1-\epsilon) \cdot \mathbb{E}[X]] \\
&\leq \exp\left(-\frac{\mathbb{E}[X]}{2} \cdot \epsilon^2\right) = \exp\left(-\frac{\mathbb{E}[X]}{2} \cdot \left(1 - \frac{t}{\mathbb{E}[X]}\right)^2\right) \\
&\leq \exp\left(-\frac{(1+\delta) \cdot t}{2} \cdot \left(1 - \frac{t}{(1+\delta) \cdot t}\right)^2\right) \\
&= \exp\left(-\frac{(1+\delta) \cdot t}{2} \cdot \left(\frac{\delta}{1+\delta}\right)^2\right) \\
&= \exp\left(-\frac{t}{2} \cdot \frac{\delta^2}{1+\delta}\right)
\end{aligned}$$

□

1.2.2 Coupling and Majorisation

In this thesis we will only consider balls-into-bins processes in which the allocation of a ball depends on the current state of the load vector, but not on previous states. Therefore we can regard these processes as Markov chains[1]: The state space is the set of load vectors, and the probabilities of the transitions are given by the load balancing protocol.

The *coupling* of Markov chains is a technique to compare random processes with each other. It has many applications in computer science and probability theory (broad expositions are given in [53, 86]). We will couple allocation processes and compare their load vectors. In all instances our aim is to show that the maximum load of one process is stochastically dominated by the maximum load of the other process.

[1] A Markov chain is a stochastic process $(X_t) = \{X_t \mid t \in \mathbb{N}\}$ in which the transition probabilities only depend on the current state (see definitions given in [68, 66]).

Definition 1.2.5 (Stochastic dominance)**.** *Let X and Y be random variables whose co-domain is some topological space E on which a partial order is defined. Then X is* stochastically dominated *by Y if $\mathbb{E}[f(X)] \leq \mathbb{E}[f(Y)]$ for all monotonically increasing functions $f : E \to \mathbb{R}$.*

In the special case that X and Y are real-valued random variables, X is stochastically dominated *by Y if $\mathbf{Pr}[X \leq a] \leq \mathbf{Pr}[Y \leq a]$ for all $a \in \mathbb{R}$.*

Definition 1.2.6 (Coupling, order-preserving coupling [54])**.** *A* coupling *of two Markov chains (X_t) and (Y_t) with state space S is a Markov chain $(Z_t) = ((X_t), (Y_t))$ on the state space $S \times S$ such that:*

$$\begin{aligned} \mathbf{Pr}[X_{t+1} = x' \mid Z_t = (x, y)] &= \mathbf{Pr}[X_{t+1} = x' \mid X_t = x] \\ \mathbf{Pr}[Y_{t+1} = y' \mid Z_t = (x, y)] &= \mathbf{Pr}[Y_{t+1} = y' \mid Y_t = y] \end{aligned}$$

Let $\preceq_*$ be a partial order relation on S and let $K := \{(x, y) \in S \times S \mid x \preceq_* y\}$. Further, let $\mathbf{Pr}_z[Z_t \in K]$ denote the probability for $Z_t \in K$ given that (Z_t) starts from $z \in K$. The coupling (Z_t) is order-preserving *if $\mathbf{Pr}_z[Z_t \in K] = 1$ for all $t > 0$.*

After coupling (X_t) and (Y_t), both Markov chains behave exactly as before. The only difference is that their random decisions are coupled. Regarded as randomised algorithms, it is like using the same random number in both chains to determine the next step. Thus, in order to define a coupling we only have to specify a bijective mapping between the random choices for each pair of states. We will do this in Lemma 3.3.3 and Observation 4.3.4.

After establishing a method to couple allocation processes, we still need a method to compare load vectors with each other. We will use *majorisation* [58] which defines a partial order relation on $\mathbb{R}^n$ based on the *normalised* representation of the vectors (Definition 1.1.1):

Definition 1.2.7 (Majorisation $\succeq$)**.** *Given two vectors $\mathbf{p} = (p_1, \ldots, p_n)$ and $\mathbf{q} = (q_1, \ldots, q_n)$ in $\mathbb{R}^n$, we say that $\mathbf{p}$ majorises $\mathbf{q}$ if and only if for all $k = 1, \ldots, n$*

$$\sum_{i=1}^{k} \bar{p}_i \geq \sum_{i=1}^{k} \bar{q}_i$$

where $\bar{p}_i$ and $\bar{q}_i$ are the i-th entries of the normalised vectors $\bar{\mathbf{p}}$ and $\bar{\mathbf{q}}$, respectively[2]. We then write $\mathbf{p} \succeq \mathbf{q}$.

Observation 1.2.8. *Let L^A and L^B be the load vectors of the allocation processes A and B. Then $L^A \succeq L^B$ implies that the maximum load in A is not smaller than the maximum load in B.*

[2]Note: Our definition actually complies with the definition of *weak majorisation* in the literature. According to [58] q is (properly) majorised by p only if additionally $\sum_{i=1}^{n} \bar{p}_i = \sum_{i=1}^{n} \bar{q}_i$. We ignore this condition.

Observation 1.2.9. *Majorisation is a partial order relation on* $\mathbb{R}^n$.

Lemma 1.2.10 (by Kamae *et al.* [42, 54])**.** *Let* (X_t) *and* (Y_t) *be two Markov chains with state space* S *where* S *is a partially ordered* $\mathbb{R}^n$. *Let* $(Z_t) = ((X_t), (Y_t))$ *be a coupling starting from* $z = (x, y)$ *and* $\preceq_*$ *the partial order relation on* S. *If* $x \preceq_* y$ *and if* (Z_t) *is order-preserving, then* X_t *is stochastically dominated by* Y_t *for all* $t > 0$.

In the context of balls-into-bins games: (X_t) and (Y_t) are allocation processes, $S = \mathbb{R}^n$, and x and y are the initial load vectors. If x is majorised by y and if (Z_t) is order-preserving, then X_t is stochastically dominated (under majorisation) by Y_t for all $t > 0$. In particular, this implies that the maximum load is also stochastically dominated.

In Lemma 3.3.3 we will use the following claim to show that a coupling is order-preserving.

Claim 1.2.11 (by Wieder, Claim 2.4 in [94])**.** *Let* x *and* y *be two normalised integer vectors such that* $x \succeq y$. *If* $i \leq j$ *then* $x + e_i \succeq y + e_j$ *where* e_i *is the* i*-th unit vector and* $x + e_i$ *and* $y + e_j$ *are normalised.*

1.2.3 Layered Induction

Layered induction is a technique that was first used in [3] by Azar, Broder, Karlin and Upfal. They consider the d-choice game in which n balls are thrown into n bins and show that the maximum load is upper-bounded by $\frac{\ln \ln(n)}{\ln(d)} + \mathcal{O}(1)$, *w.h.p.* In the proof they bound the number μ_k of bins that have at least k balls for all levels $k \geq 1$ using an induction on k. The induction base, for $k = 1, ..., 5$, is easily established by setting the bounds to n. For the induction step $k \to k + 1$, they assume that the bound holds for μ_k and use this to estimate μ_{k+1}.

Later in [8, 63] this technique was termed *layered induction*. It was applied in [19, 8, 18]. A detailed description is given in [63].

1.2.4 Witness Trees

Witness trees are employed in different contexts. Concerning load balancing they are especially useful in the analysis of d-choice balls-into-bins games in which each ball allocates itself to the least loaded of d randomly chosen bins. Generally the technique embodies the following idea: In order to show that a certain event does not occur or only occurs with low probability, one proves that the event implies certain requirements and that it is impossible or unlikely that these requirements are fulfilled. In our example the event is that the load of any bin exceeds a certain

threshold ℓ. It is then to show that such an event implies the presence of a witness tree and that the probability for the existence of such a tree is small, given that ℓ is large enough.

Pursuing the example, we define a *witness tree* $\mathcal{T}$ to be a full d-ary rooted tree of height ℓ in which each vertex represents a ball. Let b be any inner vertex. The d children of b are the topmost balls of the d bins probed by b, *i.e.*, the topmost balls before the allocation of b.

Given the event that one bin contains (at least) $\ell+1$ balls, it is easy to show that such a witness tree $\mathcal{T}$ exists: As the root b_r we choose the (or a) ball on level $\ell + 1$. The balls represented by children of b_r must have height at least ℓ because they were the topmost balls in their bins when b_r was allocated and b_r was thrown into the least loaded bin. Accordingly, the balls represented by b_r's grandchildren must have height at least $\ell - 1$, and so on. As the balls represented by the leaves have height 1, this recursive argument yields that $\mathcal{T}$ has indeed height at least ℓ.

While taking care of the dependencies, it remains to show that the probability for the existence of a witness tree is small. Roughly, this is achieved by multiplying the number of all possible witness trees with the probability that a particular tree exists. For a detailed analysis see [88, 63].

Witness trees were introduced in [60] by Meyer auf der Heide, Scheideler and Stemann. Mitzenmacher *et al.* assess them as the most "challenging" technique, but also as the one that tends "to provide the strongest results" [63]. Witness trees are used in [22, 19, 20, 88]. A summary of examples and results is provided in [63].

1.2.5 Further Techniques

For the analysis of balls-into-bins games further techniques have been adopted. Beside layered induction and witness trees Mitzenmacher *et al.* count *fluid limits via differential equations* to "the three major techniques" [63]. Assuming $n \to \infty$, this technique aims to describe the behaviour of the system by differential equations, to solve them and, so, to gain results – which hopefully also hold for the finite case.

Another technique that is particularly useful in the analysis of load balancing schemes is the *Poisson approximation* for binomially distributed random variables: The binomial distribution plays a prominent role in d-choice balls-into-bins games. So is, for example, the number of times that a fixed bin is probed binomially distributed. As for sufficiently small p and large n the Poisson distribution is the limit of the binomial distribution, it can often be used instead and help to avoid dependencies [66]. Good accounts of the Poisson approximation are given in [80, 66]. It is employed in [35, 2, 62, 23, 29], among other things for bounding the maximum load in the single-choice balls-into-bins game [35, 62]. (However, Raab and Steger claim in [74] that the

Poisson approximation can only be applied if $\frac{m}{n}$ is constant. They utilise the *first* and the *second moment method* to derive tight bounds for the remaining cases.)

1.3 Thesis Overview

The subject of this thesis is *randomised load balancing*. Chapter 3 and 4 are based on research papers [7, 6] while the results of Chapter 5 have not been published before. Each of these chapters begins with an introduction that illuminates the context of the results and provides more specific definitions not covered in Section 1.1. Open problems and summaries are given in a conclusion at the end of each chapter.

Before the new results are discussed in the Chapters 3 to 5, Chapter 2 will provide an extensive overview of the work published in this area. We will point out the key results that are relevant for this thesis.

Chapter 3 considers a generalisation of the standard multiple-choice process in which the bins have different capacities (or speeds) and their loads are defined as the number of balls divided by the capacity. Although Wieder mentions such a game in [94] and suggests to choose the bins' probabilities proportional to their capacities, this type of balls-into-bins game has previously not been analysed.

The balls-into-bins game in Chapter 4 assumes the bins to be clustered. A ball chooses one random cluster of size d (instead of d single bins) and is then allocated to a bin of lowest load within the cluster. The model was introduced by Godfrey in [34] and allows the cluster set to change with every ball – provided that it fulfils certain requirements. We relax these requirements, improve Godfrey's results and simplify the proof.

Chapter 5 investigates GREEDY$[d]$ in the dynamic parallel model. The difference to the standard d-choice game is that the balls arrive in batches of size n and that the loads of the bins are not updated before all n balls have been allocated. For $m \leq poly(n)$ we show that, *w.v.h.p.*, the maximum load above the average is $\mathcal{O}(\ln(n))$ and therefore independent of the number of balls.

2 Related Work

This chapter provides an overview of the related work and describes applications. We will focus on balls-into-bins games in various settings and give an overview of their (recent) history. This sums up the work that is related to the results in the subsequent chapters.

Like urn problems where balls are drawn from urns, balls-into-bins games belong to the urn models. They are also known as allocation processes or occupancy problems and model the random allocation of m balls into n bins under various conditions. These conditions include different probability distributions over the bins as well as weighted balls or bins. In the following sections we will describe a few of these modifications and extensions.

It is difficult to date back the origins of balls-into-bins games, but there are references in de Moivre's "Doctrine of Chances" as early as 1718 [24, 40]. As a model in theoretical physics they were studied in more detail in the 19th century [40]. A combinatorial foundation for urn models in general and balls-into-bins games in particular was provided by Whitworth and MacMahon [93, 56]. In computer science balls-into-bins games were found to be a powerful model to analyse hashing and load balancing strategies. In this context the aim is to find strategies that balance the balls evenly over the bins and produce small maximum loads. Karp *et al.* [45] and Azar *et al.* [3] sparked new interest when they showed a considerable improvement in the maximum load that is achieved by letting each ball choose two random bins instead of one and allocate itself to the lesser loaded.

We will briefly describe *urn problems*, before we solely concentrate on balls-into-bins games. Due to the results in the subsequent chapters the focus lies on the maximum load in multiple-choice games. Most of all we are interested in weighted bins (Chapter 3), non-uniform probabilities (Chapter 3 and 4) and parallel settings (Chapter 5).

2.1 Urn Problems

Urn models offer a natural and easy way of describing a great variety of random experiments and processes. They are used in combinatorics and provide an instrument to develop probability

theory, more flexible than dice, cards or coins [33]. We distinguish between *urn problems* and *occupancy models* (or *balls-into-bins games*). The former cover all games in which (possibly coloured) balls are drawn from urns, the latter all games in which balls are randomly allocated to urns (or bins).

According to Heubeck [37] and Johnson and Kotz [40], Huygens analysed urn problems as early as 1665 [38]. Occupancy problems were mentioned by de Moivre in 1718 [24, 40].

Urn problems are described by a set of coloured urns, each containing a set of coloured balls, and a protocol that defines how balls are drawn from and placed into the urns. Balls (urns) of the same colour are not distinguishable. Generally, when a ball is drawn from an urn that contains s balls, each ball in this urn has the same probability $\frac{1}{s}$ to be selected. Dependent on the model balls are drawn from one or several urns, with or without replacement, dependent or independent of previous balls. The colour of the current ball can, for instance, determine the next urn(s) from which balls are drawn and the colour and number of balls that are (re)placed into the urn.

Johnson and Kotz [40] show how basic ideas of probability theory can be derived from urn models and develop a distribution theory based on them. They also list applications in other sciences like physics, chemistry and biology and describe in detail how the spread of contagious diseases can be modelled by Pólya-Eggenberger distributions.

A well-known problem that can be expressed as a relatively simple urn problem is the birthday paradox [91] that is described in the following example.

Example 2.1.1. *Suppose a year has exactly $n = 365$ days, and every day has the same probability $\frac{1}{365}$ to be the birthday of a randomly selected person. How many people need to be gathered in one room so that there is a 50% probability that at least two of them share the same birthday? A similar question is: What is the expected number of people needed so that at least two of them share the same birthday.*

Rewritten as an urn problem this problem reads: Suppose we have a single urn with $n = 365$ coloured balls which are pairwise distinct. Balls are drawn from the urn with replacement, that is, the selected ball is immediately put back into the urn. How many balls need to be drawn so that there is a 50% probability for at least one ball being chosen twice? What is the expected number of samples until a ball is drawn that has been chosen before?

2.2 Single-choice Balls-into-bins Games

The urn models we are actually interested in are the occupancy models or balls-into-bins games. A balls-into-bins game is defined by a set of m balls, a set of n bins (or urns) and a protocol that specifies how the balls are (randomly) allocated to the bins.

We classify the games by the number of choices per ball which we denote by d. In single-choice games each ball chooses $d = 1$ bin *i.u.r.* and allocates itself to it. If $d \geq 2$, we speak of multiple-choice games. Each ball looks at d randomly chosen bins, selects one of them and allocates itself to it. Generally, for any $d \in \mathbb{N}$, we will also use the term d-choice game.

The birthday paradox [91] that we expressed as a classical urn model in Example 2.1.1 can also be written as a single-choice balls-into-bins game. We identify the days with bins and the persons with balls that are randomly allocated to the bins. If $n = 365$ is the number of bins, the question is: What is the expected number of balls that need to be thrown, until one bin contains two balls? We answer this question in the following lemma:

Lemma 2.2.1 (*e.g.* [32, 48])**.** *Let $n \in \mathbb{N}$ be the number of bins and let X count the allocated balls until one bin contains two balls. Then the expected value is*

$$E[X] = 1 + Q(n) = \mathcal{O}(\sqrt{n})$$

where

$$Q(n) \sim \sqrt{\frac{\pi \cdot n}{2}} - \frac{1}{3} + \sqrt{\frac{\pi}{288 \cdot n}} - ... = \mathcal{O}(\sqrt{n})$$

is derived from Ramanujan's Θ-function [75, 76, 32]. For $n = 365$ in particular,

$$E[X] = 1 + Q(365) \approx 24.62.$$

All new results in this thesis that are related to balls-into-bins games target the load distribution: Placing m balls into n bins according to a given protocol, how evenly will the balls be distributed over the bins? What is the maximum load? Therefore, we will concentrate on these questions in the remainder of this chapter.

In [35] Gonnet shows for the single-choice balls-into-bins game and $m = n$ that, *w.h.p.*, the fullest bin contains $\Gamma^{-1}(n) \cdot \left(1 + \mathcal{O}\left(\frac{1}{\ln(\Gamma^{-1}(n))}\right)\right)$ balls[1] which implies a maximum load of $\frac{\ln(n)}{\ln\ln(n)} \cdot (1 + o(1))$ [74]. Mitzenmacher provides an easier proof for the less precise state-

[1] $\Gamma(n)$ is the gamma function which extends the factorial to non-integer arguments. For $n \in \mathbb{N}$ it is defined as $\Gamma(n) = (n-1)!$, for $x \in \mathbb{R}_+$ as $\Gamma(x) = \int_0^\infty t^{x-1} \cdot e^{-t} dt$ (see *e.g.* [50], pp. 342-348).

ment $\Theta\left(\frac{\ln(n)}{\ln\ln(n)}\right)$ and also shows that, in case $m < \frac{n}{\ln(n)}$, the maximum load is $\Theta\left(\frac{\ln(n)}{\ln(n/m)}\right)$ *w.h.p.* [62].

Raab and Steger give a more elementary proof of Gonnet's result in [74] and extend it to all $m \geq \frac{n}{polylog(n)}$. They achieve very tight bounds by applying the first and second moment method[2]:

Theorem 2.2.2 (by Raab *et al.*, Theorem 1 in [74])**.** *Let M be the random variable that counts the maximum number of balls in any bin, if we throw m balls independently and uniformly at random into n bins. Then $\mathbf{Pr}[M > k_\alpha] = o(1)$ if $\alpha > 1$ and $\mathbf{Pr}[M > k_\alpha] = 1 - o(1)$ if $0 < \alpha < 1$, where*

$$k_\alpha = \begin{cases} \frac{\ln(n)}{\ln(\frac{n\cdot\ln(n)}{m})} \cdot \left(1 + \alpha \cdot \frac{\ln\ln(\frac{n\cdot\ln(n)}{m})}{\ln(\frac{n\cdot\ln(n)}{m})}\right) & \text{if } \frac{n}{polylog(n)} \leq m \ll n \cdot \ln(n) \\ (d_c - 1 + \alpha) \cdot \ln(n) & \text{if } m = c \cdot n \cdot \ln(n) \\ \frac{m}{n} + \alpha \cdot \sqrt{2 \cdot \frac{m}{n} \cdot \ln(n)} & \text{if } n \cdot \ln(n) \ll m \leq n \cdot polylog(n) \\ \frac{m}{n} + \sqrt{2 \cdot \frac{m}{n} \cdot \ln(n) \cdot \left(1 - \frac{1}{\alpha} \cdot \frac{\ln\ln(n)}{2\cdot\ln(n)}\right)} & \text{if } m \gg n \cdot \ln^3(n) \end{cases}$$

Here, c is a constant and d_c is the unique solution of

$$f_c(x) := 1 + x \cdot (\ln(c) - \ln(x) + 1) - c = 0$$

for which $d_c > c$.

Single-choice balls-into-bins games with weighted balls are analysed in [79, 51, 10, 11]. As we will only consider allocation processes with unit-sized balls in the following chapters, we skip these results and continue with multiple-choice games.

2.3 The Power of Two Choices

Around 1990 researchers started to investigate how an increase in the number of choices d affects the load distribution in allocation processes. The surprising discovery was that, compared to the single-choice game, the maximum load in the 2-choice game is exponentially lower in the case $m = n$. A further increase of d, however, yields only a constant improvement.

According to [63], this effect was first observed by Eager, Lazowska and Zahorjan in [28]. The first mathematical analysis appeared in [45]. There Karp, Luby and Meyer auf der Heide investigate how a *parallel random access machine* (PRAM) with concurrent read / write oper-

[2] Most of the results were "folklore" though and known for decades; see references in [40, 49, 15].

ations (CRCW) on shared memory can be simulated by a *distributed memory machine* (DMM) where parallel access to the same memory module is not possible. The *delay* of such a simulation is measured by the "time needed to simulate a parallel memory access of the PRAM" [45]. The authors suggest to store data in more than one memory module of the DMM (using several hash functions) and show that the delay can be reduced from $\Theta\left(\frac{\ln(n)}{\ln\ln(n)}\right)$ to $\mathcal{O}(\ln\ln(n) \cdot \ln^*(n))$, where n is the number of processors and memory modules. (In the context of PRAM simulations this idea was further discussed in [26, 55, 22, 60].)

Azar, Broder, Karlin and Upfal [3, 4] consider the standard d-choice balls-into-bins game. They introduce the protocol GREEDY$[d]$ (see Section 1.1.4) and the *layered induction* technique to analyse it. For the general case $m \geq n$, they prove that the maximum load is $(1 + o(1)) \cdot \frac{\ln\ln(n)}{\ln(d)} + \Theta\left(\frac{m}{n}\right)$ *w.h.p.* In particular this implies:

Theorem 2.3.1 (by Azar *et al.*, Theorem 1.1 in [4])**.** *Suppose that n balls are sequentially placed into n bins. Each ball allocates itself to the least loaded of $d \geq 2$ bins that are chosen* i.u.r. *Then,* w.h.p.*, the maximum load is*

$$\frac{\ln\ln(n)}{\ln(d)} + \Theta(1) \tag{2.1}$$

Berenbrink, Czumaj, Steger and Vöcking [8, 9] improve on the result for $m > n$ and show that, *w.h.p.*, the expected maximum load above the average $\frac{m}{n}$ is independent of the number of balls:

Theorem 2.3.2 (by Berenbrink *et al.*, Corollary 1.4 in [9])**.** *Suppose that m balls are sequentially placed into n bins. Each ball allocates itself to the least loaded of $d \geq 2$ bins that are chosen* i.u.r. *Then,* w.h.p.*, the maximum load is*

$$\frac{m}{n} + \frac{\ln\ln(n)}{\ln(d)} + \Theta(1).$$

In the proof they first show that the result holds for $m = poly(n)$ and then extend it by applying a *Short Memory Lemma*. In Chapter 5 we will describe their approach.

In [3] Azar *et al.* also consider the infinite process assuming that in each step one ball is allocated with GREEDY$[d]$ and one random ball is removed. They show that, if the game starts with n balls arbitrarily placed into the bins, the maximum load will be $\frac{\ln\ln(n)}{\ln(d)} + \mathcal{O}(1)$ *w.h.p.* after a recovery time of $c \cdot n^2 \cdot \ln\ln(n)$ steps. The work was continued and extended in [23, 21, 20, 19].

The multiple-choice game with weighted balls is addressed in [10, 85, 73].

2.3.1 Slight Modifications of the Multiple-Choice Game

The *power of two choices* paradigm was a breakthrough in randomised load balancing, but also raised the question whether other small changes could possibly lead to better results.

A somewhat surprising improvement was presented in [88] by Vöcking. For his GoLeft$[d]$ strategy he assumes that the bins are partitioned into d sets of size $\frac{n}{d}$. Every ball chooses exactly one bin from each set *i.u.r.* and allocates itself to the least loaded. Contrary to Greedy$[d]$, ties are *not* broken arbitrarily, but the ball chooses the leftmost of the least loaded bins.

For $m = n$, Vöcking shows that, *w.v.h.p.*, the maximum load is

$$\frac{\ln \ln(n)}{d \cdot \ln(\Phi_d)} + \mathcal{O}(1) \tag{2.2}$$

where $\Phi_d = \lim_{k\to\infty} \sqrt[k]{F_d(k)} < 2$ and $F_d(k)$ are the d-ary Fibonacci numbers[3] [88]. Similar to Greedy$[d]$, this bound also holds for the gap between maximum and average load if $m > n$ [9]. Note that, for all $d \geq 2$, (2.2) is strictly better than (2.1). And simulations in [89] suggest that GoLeft$[d]$ results in better maximum loads in practice as well – seemingly independent of n.

Kenthapadi and Panigrahy add in [47] that, at least in case $m = n$, one can save random bits during the process as they show that "such bounds can be achieved by making only two random accesses and querying $\frac{d}{2}$ contiguous bins in each access" [47]. More precisely, the n bins are divided into $\frac{2 \cdot n}{d}$ clusters of size $\frac{d}{2}$. Every ball chooses two clusters and places the ball into the least loaded bin of the lesser loaded cluster. If the clusters have equal total load, then the tie is broken to the left.

In [90] Vöcking shows a matching lower bound of

$$\frac{\ln \ln(n)}{d \cdot \ln(\Phi_d)} - \mathcal{O}(1).$$

Interestingly this bound is valid for all multiple-choice strategies whose d choices are (possibly) non-uniform and (possibly) dependent. In Section 2.5 and 2.6 we will review more such processes, yet, processes that do not benefit from non-uniformity.

Mitzenmacher, Prabhakar and Shah consider in [65] a 2-choice model that allows for reusing one of the previous choices. As usual each ball allocates itself to the lesser loaded of two bins, but only one of these two bins is newly chosen. The other bin is memorised by the system as the lesser loaded bin of the previous round *after* the allocation of the ball.

[3] $F_d(k) = 0$ for $k \leq 0$, $F_d(1) = 1$ and $F_d(k) = \sum_{i=1}^{d} F_d(k-i)$ for $k \geq 2$.

For the maximum load the authors show the same upper bound as in Vöcking's approach, *i.e.* (2.2) for $d = 2$.

2.4 Parallelisation

The disadvantage of GREEDY$[d]$ is that it is a sequential algorithm – in which one ball is allocated after the other – and that it loses its powers in a parallel setting. The single-choice protocol can be run in parallel because the allocation of a ball does not depend on the positions of the other balls. In the multiple-choice game, however, this is not the case as it is essential for the performance that the loads are updated before the next ball is allocated. Since many applications like load balancing and hashing assume a parallel environment, it is desirable to adapt GREEDY$[d]$ to these settings.

Not long after the publication of GREEDY$[d]$ and its analysis, different attempts were made to parallelise it [2, 61, 62, 82, 12, 1, 13]. Most strategies involve additional rounds of communication, some are also adaptive and allow for rechoosing bins. Recently Even and Medina reviewed these models and suggested corrections in [29, 30]. In order to describe the models and results we will use the vocabulary defined in Section 1.1.4 which is adopted from [2, 82, 1, 30]. In what follows, r denotes the (maximum) number of communication rounds.

The first parallel multiple-choice protocols were introduced by Adler, Chakrabarti, Mitzenmacher and Rasmussen in [2]. Their algorithms PGREEDY, MPGREEDY and THRESHOLD address the static model with $m = n$ and allow parallel message exchange based on the communication graph. Stemann and Even *et al.* consider the same model and add the k-COLLISION and the RETRY protocol, respectively [82, 29]. Most of these protocols achieve a maximum load of

$$\mathcal{O}\left(\sqrt[r]{\frac{\ln(n)}{\ln\ln(n)}}\right). \tag{2.3}$$

The algorithm PGREEDY$[d]$ is based on GREEDY$[d]$ and adds only one round of communication. After each ball has chosen and informed its d random bins, the bins send their current *height* back, that is, the number of answered requests. Messages arriving simultaneously at the same bin are processed one by one (in arbitrary order) so that each reply contains a different height. As soon as a ball has received all d answers, it will commit to the bin of lowest height. PGREEDY$[d]$ is symmetric, non-adaptive and asynchronous. The maximum load achieved equals (2.3) for $r = 2$ *w.h.p.* [2].

The protocol MPGREEDY$[d]$ works similar, but adds a number of communication rounds that

is not fixed beforehand. When a ball informs the d chosen bins, it attaches its identification number (ID), randomly chosen from a large enough set. These IDs are then used by the bins to sort the list of requests. In every (synchronous) communication round each bin sends its current load to the ball whose ID comes next in the list. If a ball receives at least one message, it will allocate itself to a bin of lowest load and cancel all its remaining requests. MPGREEDY$[d]$ is also symmetric and non-adaptive, but it is not asynchronous because, contrary to PGREEDY$[d]$, a ball cannot deduce on its own from the (number of) returned messages whether a round is complete. According to [2], the expected maximum load as well as the number of communication rounds is

$$\frac{\ln\ln(n)}{\ln(d)} + 2 \cdot d + \mathcal{O}(1).$$

The synchronous protocol THRESHOLD$[T]$ diverges from GREEDY$[d]$ more considerably as it allows for rechoosing bins. In each round every ball not yet allocated chooses a random bin. Each bin accepts up to T balls and rejects all balls above this threshold. If the balls include their round numbers in their messages, this protocol works completely asynchronously. A bin must only memorise how many balls it has already accepted from each round. Adler *et al.* prove that, *w.h.p.*, THRESHOLD$[1]$ terminates after at most $\ln\ln(n) + \mathcal{O}(1)$ steps which implies that the maximum load is also at most $\ln\ln(n) + \mathcal{O}(1)$. It approximately matches the maximum load of GREEDY$[d]$ and is achieved in $\mathcal{O}(\Omega(\ln\ln(n)))$ asynchronous rounds whereas GREEDY$[d]$ requires n synchronous rounds. The authors show that THRESHOLD$[T]$ terminates after r rounds *w.h.p.* if the threshold T equals (2.3). This implies a maximum load of $r \cdot T$ which is again (2.3) if r is constant.

The k-COLLISION protocol (by Stemann [82], based on [26]) has the same upper bounds as THRESHOLD$[T]$, with the difference that they hold for all $r \in \mathcal{O}(\ln\ln(n))$. The protocol starts with each ball choosing two bins. After all bins have received the requests (synchronisation point), each bin checks if it has got at most k requests (where k equals (2.3)). If this is the case, it sends acknowledgments back to the balls. Otherwise it waits until the number of requests drop to k. When a ball receives at least one acknowledgment, it allocates itself to one of the according bins and cancels all other requests. The k-COLLISION protocol is symmetric, non-adaptive and almost asynchronous as it has only one synchronisation point. The loops can run asynchronously.

Adler *et al.* also provide a matching lower bound of

$$\Omega\left(\sqrt[r]{\frac{\ln(n)}{\ln\ln(n)}}\right) \tag{2.4}$$

for non-adaptive, symmetric protocols given that d and r are constant and that all decisions are based on the unlabelled access graph [2]. Berenbrink, Meyer auf der Heide and Schröder generalise this lower bound to $r \leq \ln\ln(n)$ in [12] matching Stemann's k-COLLISION protocol.

However, in [29] Even *et al.* point out that the lower bound does not hold for PGREEDY$[d]$ and THRESHOLD$[T]$ as they base their decisions on loads or round numbers, respectively. The access graph cannot be regarded as unlabelled because vertices (*i.e.* bins) with lower loads or round numbers are preferred. In [29] they show explicit lower bounds for both algorithms, and in [30] they generalise the lower bound of (2.4) to all aforementioned algorithms by rendering the symmetry assumption and some restrictions on the access graph unnecessary.

The first results for $m > n$ appeared in [82]. Allowing $\Omega\left(\frac{\ln\ln(n)}{\ln(m/n)}\right)$ rounds of communication, the k-COLLISION protocol achieves a maximum load of $\mathcal{O}(\frac{m}{n})$. Berenbrink *et al.* [12] adapt the k-COLLISION protocols to weighted balls. The only change is that balls attach their weights to their requests and that the threshold k is now compared to the sum of weights (instead of the number of balls). If W^A and W^M are the average and maximum weight, respectively, then the maximum load is shown to be

$$\gamma \cdot \left(\frac{m}{n} \cdot W^A + W^M\right) \quad \text{for} \quad \frac{\ln\ln(n)}{\ln\left(\frac{\gamma}{4} \cdot \left(\frac{m \cdot W^A}{n \cdot W^M} + 1\right)\right)} + 1$$

rounds of communication. In case of uniform weights this result conforms to the bound in [82].

Besides the static model, Stemann [82] also investigates a finite dynamic model in which n players have $\tau = \frac{m}{n}$ balls each and allocate them to n bins – one after the other, but without waiting for other players. Message exchange is restricted to the communication graph and to copies of the same ball. This so-called τ-ALLOCATION strategy lets a ball choose two bins and send a copy of itself to each of them. Every bin has two queues, one for first copies, one for second copies, and processes one ball from every queue per time step (provided that they are not empty). When a ball is processed, its copy is removed from the system as well. The according player is informed whereupon it initiates the allocation of the next ball.

The τ-ALLOCATION protocol is non-adaptive, symmetric and asynchronous. If $\tau = \ln(n)$, then, *w.h.p.*, all balls will be allocated in time $\mathcal{O}(\ln(n))$. The maximum waiting time is bounded by $\mathcal{O}(\ln\ln(n))$.

Another finite dynamic model is the *supermarket model* that was introduced by Mitzenmacher in [61]. Even though the allocation of the balls follows GREEDY$[d]$ (or THRESHOLD$[T]$), the model differs significantly from the standard process as the *service times* of the bins and the *arrival rate* of the balls are not regular. The service times are exponentially distributed with

mean 1, and the balls arrive in a Poisson stream with *average arrival rate* $\lambda < 1$ which implies that the intervals between the arrivals are also exponentially distributed. Assuming that all balls base their decision solely on the bin loads (but not on the service times), Mitzenmacher shows that the expected waiting time is constant for $n \to \infty$. The maximum waiting time is $\mathcal{O}(\ln \ln(n))$.

The supermarket model is further analysed in [62, 92, 64].

Adler, Berenbrink and Schröder [1] investigate the INFINITE ALLOCATION process in the dynamic model. It is inspired by the infinite, sequential d-choice process [4, 23] as well as Stemann's τ-ALLOCATION protocol [82]. The authors assume that the balls arrive in batches of size s and that each ball sends a copy of itself to $d \geq 2$ randomly chosen bins. The balls are stored in FIFO queues and, thus, processed in the order of arrival, one ball per bin in every round (unless the queue is empty). When a ball is processed, all its copies are informed and removed from the queues.

The INFINITE ALLOCATION protocol is non-adaptive, symmetric and asynchronous. If $s \leq \frac{n}{9}$, then the expected waiting time is constant and, thus, most of the balls are processed very quickly. However, the maximum waiting time is $\frac{\ln \ln(n)}{\ln(d)} + \mathcal{O}(1)$ *w.v.h.p.*

In Chapter 5 we will analyse the bare GREEDY$[d]$ protocol in a dynamic model. The balls arrive in batches of size n and each batch is allocated in parallel. This is modelled by updating the bin loads only between the batches. We will show that, if $m \leq poly(n)$, the gap between average and maximum load is $\mathcal{O}(\ln(n))$ and, thus, independent of the number of balls.

2.5 Non-uniform Probabilities

All load balancing schemes considered so far are based on the assumption that the probabilities for bins to be probed are uniform; *i.e.*, each bin has the same probability $\frac{1}{n}$. In many applications, however, this cannot generally be assumed. One prominent example is load balancing in distributed hash tables (see below). Byers, Considine and Mitzenmacher [17, 18] suggest to apply the multiple-choice paradigm and analyse it for $m = n$. Wieder [94] generalises their approach to the heavily-loaded case $m \gg n$. In this section we describe their work and finish with an outlook on Chapter 3.

We consider consistent hashing as it is used in some implementations of distributed hash tables [43, 84]. The task is to hash keys (items, requests, balls) to a set of peers (nodes, computers, bins) in such a way that (i) the keys are sufficiently balanced over the peers and that (ii) changes in the network structure do not overly affect the performance. In a basic approach,

used in *Chord* [84], the peers as well as the keys are mapped to points on a one-dimensional ring. Each peer is assigned all keys that are mapped to the arc bounded by itself and by the next peer on the ring in clockwise direction. Whenever a peer leaves the network, its arc is simply added to the arc of the predecessor (on the ring) which receives all the stored items. When a peer joins the network, it is assigned a new arc and takes over all items from its predecessor that belong to this arc.

The advantage of this hashing technique is that it behaves well in dynamic environments. Compared to other strategies, the addition and removal of peers induce only a relatively small number of relocations. The disadvantage of consistent hashing is the discrepancy in the arc lengths that leads to a considerable imbalance in the load distribution. Let n denote the number of peers. It can be shown that the maximum arc length is $\Theta\left(\frac{\ln(n)}{n}\right)$ *w.h.p.* and, thus, about $\ln(n)$ times greater than the average arc length $\frac{1}{n}$ [17].

To solve this problem different solutions were developed (*e.g.* [83, 57, 14, 69]). Stoica *et al.* suggest to map every peer to r points on the ring instead of only one so that it receives all keys mapped to the according r arcs [83, 46]. They call the additional points *virtual peers* and recommend $r = \Theta(\ln(n))$. Even though this does reduce the deviation of the arc lengths, the overhead due to the additional edges in the overlay graph[4] and messaging costs is considerable [83, 17].

In [14] Bienkowski *et al.* introduce a protocol that achieves a constant deviation in the arc lengths by allowing some redistribution of the intervals. Triggered by *join* and *leave* operations a constant number of peers can migrate to other points on the ring. The obvious disadvantages are the additional communication and the increased number of items to relocate.

Byers *et al.* point out in [17] that the load would still be unbalanced even if all intervals had the same length because this case would conform to the standard single-choice balls-into-bins game and result in a maximum load of $\frac{m}{n} + \mathcal{O}\left(\sqrt{\frac{m \cdot \ln(n)}{n}}\right)$ for m items and n peers (Theorem 2.2.2). Hence, the authors suggest to apply the multiple-choice paradigm and to run GREEDY$[d]$ on the network: For each item that is to be allocated, $d \geq 2$ hash functions are used to choose d points on the ring. The according peers send their current loads back and the item commits itself to the peer of lowest load; ties are broken arbitrarily.

Surprisingly, for $m = n$ the maximum load is shown to be at most $\frac{\ln \ln(n)}{\ln(d)} + \mathcal{O}(1)$ *w.h.p.* – which is optimal because it matches the lower bound of the standard game (Theorem 2.3.1). In addition, simulations suggest that this strategy is superior to the virtual peers approach for

[4]The overlay graph describes how the peers are interconnected. The design of the overlay graph governs the search time for items in the network. For details see [84].

arbitrary m [17]. It is also remarkable that the running time for search operations does not significantly increase with the number of hash functions. When an item is allocated, the other $d-1$ peers are provided with *redirection pointers* pointing to the peer that got the item. Thus, when an item is requested, it suffices to call only one hash function and to follow the redirection pointer.

In [18] the same authors extend the analysis to the case $m > n$. Based on [4], they show a maximum load of $\mathcal{O}\left(\frac{m}{n}\right) + \mathcal{O}\left(\frac{\ln\ln(n)}{\ln(d)}\right)$. Furthermore, they prove that their results hold even if the one-dimensional ring is replaced by a two-dimensional torus. Here, the peers and items are mapped to points in the torus, and the nearest peer is determined by the Euclidean distance.

The question – raised in [18] – whether there exists a (significantly) stronger bound on the maximum load for $m > n$ and (fixed) $d \geq 2$ is negated by Wieder in [94]. For the d-choice game with uniform probabilities Berenbrink *et al.* [8] proved that the gap between maximum and average load does not change with the number of balls m. But, as Wieder shows by example, this result is not transferable to the consistent hashing scenario of [17, 18].

However, if d is allowed to slowly grow with the deviation in the probability distribution, then it is possible to show such a result [94]. Let a distribution be (α, β)-*biased* if for all probabilities p_i, $i \in [n]$, it holds that $\frac{1}{\alpha \cdot n} \leq p_i \leq \frac{\beta}{n}$. If the probability distribution over the bins is (α, β)-biased and if

$$d \geq (1+\epsilon) \cdot \frac{\ln\left(\frac{\alpha\cdot\beta-1}{\alpha-1}\right)}{\ln\left(\frac{\alpha\cdot\beta-1}{\alpha\cdot\beta-\beta}\right)},$$

for $\epsilon > 0$, then, *w.h.p.*, the maximum load is at most

$$\frac{m}{n} + \frac{\ln\ln(n)}{\ln(1+\epsilon)}.$$

The presented bounds are tight in such a way that, for any $\epsilon < 0$, there exists a (α, β)-biased distribution that leads to a deviation of the load linear in m.

As a motivation for his paper [94], Wieder considers balls-into-bins games in which the bins have heterogeneous capacities. A bin with storage capacity (or speed) 3, for example, can store (process) three times as many balls as a bin with capacity 1. Wieder assumes that such a game can be reduced to a game with heterogeneous probabilities. In order to compensate for the different abilities of the bins, he suggests to set the probabilities proportional to their capacities. In Chapter 3 we will analyse this problem in detail and prove bounds on the maximum load.

2.6 Graph-based Models

Further examples for balls-into-bins games with non-uniform choices are the (hyper)graph-based models by Kenthapadi *et al.* and Godfrey [47, 34]. They provide the basis for Chapter 4.

In [47] Kenthapadi and Panigrahy study the standard 2-choice game (of Azar *et al.* [3]) with the restriction that only pairs of bins can be chosen that are connected by an edge in an underlying graph. Each edge in the graph has the same probability to be chosen. Given that the graph is Δ-regular, the authors upper-bound the maximum load by

$$\ln\ln(n) + \mathcal{O}\left(\frac{\ln(n)}{\ln(\Delta/\ln^4(n))}\right) + \mathcal{O}(1)$$

w.h.p. and also provide a nearly matching lower bound. In particular, if the graph is n^ϵ-regular, the maximum load is $\ln\ln(n) + \mathcal{O}(\epsilon^{-1}) + \mathcal{O}(1)$ and, thus, not worse than in the standard game.

Godfrey [34] extends the model of [47] from graphs to multi-hypergraphs in which each hyperedge connects d of the n vertices that represent the bins. The hypergraph is allowed to be different for every ball. In order to allocate itself, each ball first chooses a hyperedge *i.u.r.* and then a bin of lowest load within the hyperedge.

Assume $d = c \cdot \ln(n)$ for a suitable constant c. For the hypergraph of any ball j, let $\deg_j(b_i)$ denote the degree of vertex (bin) b_i and $p_{ij} := \frac{\deg_j(b_i)}{|E|\cdot d}$ the relative frequency of b_i. If, for all j, the p_{ij} are (β,β)-biased (or β-balanced as Godfrey terms it), then, after throwing $m \leq \frac{n}{\Theta(\beta)}$ balls, the maximum load will be 1 *w.h.p.*

In Chapter 4 we will discuss Godfrey's model and results in more detail. There we will improve the bound for m, relax the notion of balancedness and present a simpler proof. Moreover, we will show matching lower bounds and similar results for an extended model.

3 Bins with Different Capacities

In this chapter we consider a variant of the *balls-into-bins game* with multiple choices. In our model the balls have unit size as usual, but the bins have different capacities. We consider different probability distributions over the bins and analyse the load distribution.

Preliminary versions of some results presented in this chapter were published in:

[7] Petra Berenbrink, André Brinkmann, Tom Friedetzky, and Lars Nagel. Balls into non-uniform bins. In *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium*, IPDPS '10, pages 1–10. IEEE, 2010

The results are improved and new findings have been added.

3.1 Introduction

In the standard balls-into-bins game m unit-sized balls are allocated to n unit-sized bins. It is assumed that every ball independently, uniformly and randomly chooses d bins and that it commits itself to the least loaded bin. The goal of this strategy is to balance the load over the bins and to minimise the maximum number of balls allocated to any bin.

In the variant of the game that we consider here, we assume that the bins are not uniform, but that they come with an integer capacity c_i. The load $\ell_i = \frac{m_i}{c_i}$ is defined as the ratio of the number of balls m_i assigned to it and the capacity c_i. Again, every ball has d random choices, and the goal is to minimise the maximum load.

Let $C = \sum_{i=1}^{n} c_i$ be the sum of the capacities of all bins. The natural probability for a bin to be chosen would be either $\frac{1}{n}$, that is uniform, or $\frac{c_i}{C}$, proportional to the bin's capacity. Analysing the latter case, we prove, for $m = C$ and $d = 2$, that the maximum load is at most $\frac{\ln\ln(n)}{\ln(2)} + \mathcal{O}(1)$ *w.h.p.* (Theorem 3.3.5) and under certain conditions even constant (Theorem 3.3.5 and 3.3.6). Additionaly we investigate games with differing probability distributions and show that significantly better results can be achieved in some cases (Theorem 3.3.9).

This generalisation of the standard d-choice game is especially valuable for load balancing in heterogeneous networks because it allows for modelling computers with different speeds or

storage capacities. In such scenarios it is desirable to assign more requests (balls) to the better computers (bins with higher capacities). Our approach demonstrates how this can be realised by adapting the probabilities – in theory *and* in practice as simulations suggest well-balanced load distributions even for small n and large m (Section 3.4).

3.1.1 Related Work

Heterogeneous bin sizes have been considered in the related field of selfish load balancing [31], but to our knowledge nobody has analysed it for balls-into-bins games. Such processes are only mentioned by Wieder in [94] to motivate his work about multiple-choice games with heterogeneous probabilities. He suggests to choose the bins' probabilities proportional to their capacities. In this chapter we will analyse this particular case and variations of it.

For a broader view of the related work see Chapter 2. Especially relevant for this chapter are the Sections 2.3, 2.4 and 2.5.

3.1.2 Contributions

All previous results assume that each bin has capacity 1 and that the balls should be distributed as evenly as possible. In contrast, we assume that the system consists of heterogeneous bins where each bin b_i can have an arbitrary integral capacity c_i and where its load is defined as the number of balls divided by c_i. The objective is to balance the load over the bins according to their abilities. If not stated otherwise, we assume that a bin's probability to be chosen is $\frac{c_i}{C}$ where $C = \sum_{i=1}^{n} c_i$.

In the analytical part of this chapter, we show that under these circumstances the maximum load is at most $\frac{\ln\ln(n)}{\ln(2)} + \mathcal{O}(1)$ *w.h.p.* if $d = 2$ and $m = C$ (Theorem 3.3.5). The maximum load stays constant if $C \geq n^2$ or if almost all bins have capacity $\Omega(\ln(n))$ (Theorem 3.3.5 and 3.3.6). Provided that we can choose a different probability distribution over the bins, a constant maximum load can be achieved even if there is only a constant fraction of $\Theta(\ln\ln(n))$-sized bins (Theorem 3.3.9). The proof of this theorem uses Observation 3.3.7 which states that, if all bins have the same capacity $\bar{c}$, the maximum load is bounded by $\frac{1}{\bar{c}} \cdot \left(\frac{m}{n} + \mathcal{O}(\ln\ln(n))\right)$ *w.v.h.p.* This bound is based on [9] and holds even in the heavily loaded case ($m \gg C$).

Based on a simulation environment, we arrange and simulate bin arrays with varying parameters in Section 3.4 and compare our analytical results with the experiments. In this simulations section, we also consider settings that we have not analysed, most notably the general heavily

loaded case and systems with a small number of bins. The outcomes of the latter suggest the practicability of the analytical results.

3.1.3 Motivation

One can look at the problem in three different ways: (1) The bin sizes and probabilities are fixed. The task is to analyse the distribution of the balls in general and the maximum load in particular. (2) The capacities of the bins are given and the aim is to find the probability distribution that achieves the most balanced system. (3) The probabilities are fixed and the question is how to alter the capacities in order to balance the load more evenly.

(1) The first approach is the one we mainly focus on, and we usually assume that the probabilities are proportional to the bin sizes. This is a natural assumption: If there were actual bins covering the floor and a ball was dropped from a random point above them, then this would be the resulting probability distribution. Even though this approach does not (necessarily) result in the optimal load distribution, it works fairly well and might, for instance, be the best choice for a dynamic network that is frequently joined and left by servers (bins) of different speeds (capacities). This way the revaluation of the probabilities and the alterations to the selection algorithm could be kept simple.

(2) Computer networks, especially peer-to-peer (P2P) environments, are often heterogeneous in terms of speed and storage capacity. In such a case it makes sense to prefer computers that have faster processors or more memory and increase their probability to receive requests. What is the optimal strategy to balance the load evenly? Special cases of this problem are examined in Theorem 3.3.9 and in the experiment described in Section 3.4.4, but generally this question remains open.

(3) The inverse scenario is also imaginable, an environment in which the non-uniform probabilities are given, but not the capacities. P2P environments like Chord or CAN [84, 77] are examples in which the probabilities can considerably deviate from the average – though in such dynamic systems the increase of a peer's processing or storage capacity is not a good option. A better application would be a static, widely distributed network of servers in which clients tend to choose servers close-by. *E.g.*, the network could be run by a company that offers music or software to download. If certain servers were more frequented than others, then the overall processing time could be balanced by enhancing their computing or storage capacities.

3.2 Model and Definitions

Algorithm 1 Load Balancing Protocol

1: **for all** balls **do**
2: Choose d bins *i.u.r.*
3: Among the bins of lowest load place the ball in an arbitrary bin
4: **end for**

We assume bins to be non-uniform. All bins come with positive integer capacities (which could also reasonably be referred to as speeds or compression factors). Denote the capacities of the n bins as $c_1, \ldots, c_n$ and the total capacity as $C := \sum_{i=1}^{n} c_i$. If not stated otherwise, then we consider the process that allocates $m = C$ balls into n bins, each ball having $d \geq 2$ random choices and committing itself to a bin of smallest load among the chosen bins. We say that, if m_i balls are allocated to a bin b_i of capacity $c_i \geq 1$, then this bin's load is $\ell_i = \frac{m_i}{c_i}$. Usually we will assume that the probability of bin b_i with capacity c_i being chosen is $\frac{c_i}{C}$ and therefore proportional to c_i. If we use other probability distributions, we will clearly point this out.

To make our proofs more accessible we will occasionally imagine that each bin of capacity c does actually consist of c many unit-sized *slots* (the protocol is entirely unaware of this). Hence, the total number of slots equals the total capacity C of the bins. For a fixed slot $i \in \{1, \ldots, C\}$ let $b(i)$ denote the unique bin to which slot i belongs. The *height* of a ball is the load of the bin it is allocated to directly after its allocation. Thus, if a ball falls into a bin of load ℓ_i and capacity c_i, then its height will be $\ell_i + \frac{1}{c_i}$.

The terms *load vector* $L = (\ell_1, \ldots, \ell_n)$ and *normalised load vector* $\bar{L} = (\bar{\ell}_1, \ldots, \bar{\ell}_n)$ are used as defined in Section 1.1.4. For (possibly) non-uniform bins with total capacity C we also define the *slot load vector* $S = (s_{1,1}, \ldots, s_{1,c_1}, s_{2,1}, \ldots, s_{2,c_2}, \ldots, s_{n,1}, \ldots, s_{n,c_n})$ where $s_{i,j}$ is the j-th slot of the i-th bin. The load of a slot is the number of balls it contains. Let b be a bin of capacity c containing r balls. We assume that slots are filled in a round-robin fashion and therefore that b's first (leftmost) $r \bmod c$ slots contain one ball more than the remaining slots. The *normalised slot load vector* is denoted by $\bar{S}$. For convenience, we specify that, if two slots contain the same number of balls, then the slot that belongs to the more loaded bin comes first. (We may drop the two-dimensional indices and instead use $1, \ldots, C$ as any correspondence of position within $\bar{S}$.)

If we allocate m balls into n bins, L_i ($\bar{L}_i, S_i, \bar{S}_i$) is defined as the load vector (normalised load vector, slot load vector, and normalised slot load vector, respectively) after the allocation of the i-th ball.

3.3 Analysis

The structure of this section is as follows: The main contribution is Theorem 3.3.5, upper bounding the maximum load of any bin in a system with bins of variable capacities. We start by proving Observation 3.3.2 that bounds the load of big bins as well as the height of balls that have at least one big bin among their choices. Lemma 3.3.3 shows that load distributions achieved by systems with solely unit-sized bins dominate those achieved by systems with heterogeneous bins with the same total capacity. This lemma will then be used to prove Theorem 3.3.5. Theorem 3.3.6 analyses under which circumstances (that is, number of small bins vs. number of big bins) we may achieve constant maximum load. Observation 3.3.7 bounds the maximum load for uniform bin arrays in the heavily loaded case. Finally, Theorem 3.3.9 uses this observation to show that much better results than Theorem 3.3.6 are possible provided that one can choose the bins' probabilities oneself.

Definition 3.3.1 (Big bin, $\mathcal{B}_b$, $\mathcal{B}_s$, $\ell_{max}^{(b)}$, $\ell_{max}^{(s)}$)**.** *A bin is called* big *if its capacity is at least* $r \cdot \ln(n)$ *(where* $r \geq 2$ *is a constant), otherwise it is* small*.*

With $\mathcal{B}_b$ *we denote the set of balls that have at least one big bin among their choices and with* $\mathcal{B}_s$ *the remaining balls that probe only small bins.* $\ell_{max}^{(b)}$ *(*$\ell_{max}^{(s)}$*) is the maximum number of balls from* $\mathcal{B}_b$ *(*$\mathcal{B}_s$*) in any bin.*

First we bound $\ell_{max}^{(b)}$:

Observation 3.3.2. *Consider the* 2*-choice game in which* $m = C = \sum_{i=1}^{n} c_i$ *balls are thrown into* n *bins with total capacity* C*. Let* k *be a positive constant. If* $k \leq \frac{2}{3} \cdot r - 1$*, the load in every big bin is at most* 4 *and* $\ell_{max}^{(b)} \leq 4$ *with probability at least* $1 - n^{-k}$*.*

Proof. This is a simple application of Chernoff bounds. The probability that a ball commits to a big bin b_i is at most $\frac{2 \cdot c_i}{C} = \frac{2 \cdot c_i}{m}$. The expected number of balls hitting the big bin b_i after m balls is at most $2 \cdot c_i$. Let m_i be the number of balls that have bin b_i as one of their random choices. Then, using Chernoff bounds (Lemma 1.2.1 with $\epsilon = 1$), we obtain

$$\begin{aligned} \mathbf{Pr}\left[\, m_i \geq 4 \cdot c_i \,\right] &= \mathbf{Pr}\left[\, m_i \geq (1+\epsilon) \cdot 2 \cdot c_i \,\right] \leq e^{-\epsilon^2 \cdot 2 \cdot c_i/3} \leq e^{-2 \cdot r \cdot \ln(n)/3} \\ &= n^{-2 \cdot r/3} \leq n^{-k-1} \end{aligned}$$

Hence, for r chosen suitably, with probability at least $1 - n^{-k-1}$ the bin is chosen by at most $4 \cdot c_i$ many balls, which is certainly an upper bound on the total number of balls in the bin.

W.h.p. the load is

$$\ell_i \leq \frac{m_i}{c_i} \leq \frac{4 \cdot c_i}{c_i} = 4.$$

Since there are at most n big bins, the probability that (at least) one of them exceeds load 4 is bounded by $n \cdot n^{-k-1} = n^{-k}$.

Note that this observation still holds if all balls that choose a big bin are allocated to the big bin. Since under these circumstances the maximum load of big bins is still *w.h.p.* at most 4, no ball of $\mathcal{B}_b$ will choose a small bin unless its load is smaller than 4. Hence, no ball of $\mathcal{B}_b$ will have a height of more than 4. This implies $\ell_{max}^{(b)} \leq 4$. □

Lemma 3.3.3. *Let A be a d-choice process on n non-uniform bins with total capacity C, and let B be a d-choice process on C unit-sized bins. Then the maximum load in A is stochastically dominated by the maximum load in B.*

Proof. We show this result by coupling. Since the number of bins is different in both processes, we define the state space as the set of slot load vectors (instead of load vectors). The slot load vectors in A and B have equal length because the total capacity C is the same in both processes. We let the balls choose slots rather than bins setting the probability for each slot to $\frac{1}{C}$. (Note that the slot probabilities for a bin with capacity c sum up to $\frac{c}{C}$.) However, the protocol stays the same for each ball as we map its slot choices to the according bins and allocate the ball to the best bin.

As the process starts with empty slot load vectors, S^A is majorised by S^B in the beginning. Lemma 1.2.10 states that S^A will remain stochastically dominated by S^B if an order-preserving coupling of the two processes exists. This would already imply the statement of the lemma – that the maximum load $\bar{\ell}_1^A$ in A is dominated by the maximum load $\bar{\ell}_1^B$ in B – because $\bar{\ell}_1^A \leq \bar{s}_1^A$ and $\bar{\ell}_1^B = \bar{s}_1^B$.

Let S_j^A and S_j^B denote the slot load vectors after the j-th ball. For the coupling we have to show that for every ball j there exists a bijection between the random bin choices of A and B such that $S_j^A \preceq S_j^B$ implies $S_{j+1}^A \preceq S_{j+1}^B$. Let b be any bin in process A that has capacity c and let $i_1, ..., i_c$ be the slots in $\bar{S}_j^A$ that belong to b. Choose the order of the slots so that i_c is the slot that will get the next ball that is allocated to b. (This is possible because i_c must be among the least loaded slots of b.) Then we couple the slots $i_1, ..., i_c$ in $\bar{S}_j^B$ with slot i_c in $\bar{S}_j^A$.

Let $h_1 \leq h_2 \leq ... \leq h_d$ be the d random slot choices in B. Each coupled slot in A has either the same or a higher index. In both systems we choose the rightmost slot (or a slot with the same properties so that we can swap it with the rightmost slot). Since the index of the rightmost slot in A is not smaller than the one in B, it follows from Claim 1.2.11 that $S_{j+1}^A \preceq S_{j+1}^B$.

It remains to prove that we really choose the least loaded bin in system A by allocating the ball to the rightmost slot: (i) Note that we compare the least loaded slots of the bins at question and that the slots of a bin are filled in a round-robin fashion. Therefore, if one slot has a strictly smaller load than another slot, then the same is true for the according bins. (ii) Recall that we added to the definition of the normalised slot load vector that slots of the same load are ordered by the loads of the according bins in decreasing order. Hence, even if two slots have the same load, then the slot with the higher index belongs to the bin of lesser (or equal) load. □

Lemma 3.3.4. *Consider the d-choice game in which $m = C$ balls are allocated into n bins with total capacity C. Let c and h be positive constants and m_s the total capacity of all small bins. Then, for any constant κ, $\ell_{max}^{(s)} \in \mathcal{O}(1)$ with probability at least $1 - n^{-\kappa}$ if either*

(1) $m \geq n^2$ or

(2) $m \geq h \cdot n \cdot \ln(n)$ and $m_s \leq c \cdot (n \cdot \ln(n))^{2/3}$.

Proof. We will consider six cases, distinguished by different bounds on m_s and m, and prove for each of them that $\ell_{max}^{(s)} \in \mathcal{O}(1)$. The first three cases imply statement (1), the last three cases statement (2). Since the analysis of each case follows the same method, we outline the two steps of this method before we insert any values. In two cases, step 1 will already suffice; in all other cases we will carry out both steps.

Let s be the number of small bins in the system, and recall that $\mathcal{B}_s$ denotes the set of balls that have all d choices among small bins.

Step 1: Bounding $X_s := |\mathcal{B}_s|$. The probability for a ball to be in $\mathcal{B}_s$ is

$$p_s = \left(\frac{m_s}{m}\right)^d \leq \left(\frac{m_s}{m}\right)^2.$$

For the number X_s of such balls we obtain therefore

$$\mathbf{Pr}\left[X_s \geq k\right] = \mathbf{Pr}\left[B(m, p_s) \geq k\right] \overset{(L.1.1.2)}{\leq} \left(\frac{e \cdot m \cdot p_s}{k}\right)^k \leq \left(\frac{e \cdot m_s^2}{k \cdot m}\right)^k.$$

We will choose k so that $\mathbf{Pr}\left[X_s \geq k\right] \leq n^{-\alpha}$ for any constant α (provided that n is large enough). In two cases we will be able to choose k as a (small) multiple of α which already implies a constant maximum load. In the other cases we continue with step 2.

Step 2: Bounding the maximum load. We assume $X_s \leq k$ (where k is taken over from step 1). The remaining task is to bound the maximum load of the game in which k balls are

allocated into s small bins with a total capacity of m_s. Lemma 3.3.3 states that the maximum load of this process is dominated by the maximum load of the process $\mathcal{P}$ that allocates k balls to m_s unit-sized bins. Therefore, it is sufficient to show a constant bound for the maximum load in $\mathcal{P}$.

Let X_c count the number of collisions, that is, the number of times in which a ball falls into a non-empty bin. For each case, we will show that X_c is constant *w.v.h.p.* which already implies a constant maximum load. Let $X_{c,i}$, $i \in [k]$, denote binary random variables such that $X_{c,i} = 1$ if the i-th ball collides with a previous ball and $X_{c,i} = 0$ otherwise. Observe that $X_c = \sum_{i=1}^{k} X_{c,i}$. The collision probability $p_c := \mathbf{Pr}[X_{c,i} = 1]$ for ball i is upper-bounded by

$$p_c = \mathbf{Pr}[X_{c,i} = 1] \leq \left(\frac{i-1}{m_s}\right)^d \leq \left(\frac{i-1}{m_s}\right)^2 < \left(\frac{k}{m_s}\right)^2.$$

provided that $i - 1 \leq m_s$.

For the number of collisions X_c we obtain

$$\mathbf{Pr}[X_c \geq \lambda] \leq \mathbf{Pr}[B(k, p_c) \geq \lambda] \overset{(L.1.1.2)}{\leq} \left(\frac{e \cdot k \cdot p_c}{\lambda}\right)^{\lambda} \leq \left(\frac{e \cdot k^3}{\lambda \cdot m_s^2}\right)^{\lambda}.$$

The six cases. Now we apply the described method to bound $\ell_{max}^{(s)}$ in six cases that are specified by different bounds on m and m_s. (Note that a small bin has size less than $r \cdot \ln(n)$ and that therefore $m_s < n \cdot r \cdot \ln(n)$.)

Case: $m \geq n^2$ and $m_s \in [1, n^{3/4}]$.

$$\mathbf{Pr}[X_s \geq k] \leq \left(\frac{e \cdot m_s^2}{k \cdot m}\right)^k \leq \left(\frac{e \cdot n^{3/2}}{k \cdot n^2}\right)^k = \left(\frac{e}{k \cdot n^{1/2}}\right)^k$$

From this we can directly derive that $\ell_{max}^{(s)} \leq X_s \leq 2 \cdot \alpha = \mathcal{O}(1)$ with probability $1 - n^{-\alpha}$ (for any constant $\alpha \geq \frac{e}{2}$).

Case: $m \geq n^2$ and $m_s \in [n^{3/4}, n]$.

We choose $k = \ln(n)$.

$$\begin{aligned}
\mathbf{Pr}[X_s \geq \ln(n)] &\leq \left(\frac{e \cdot m_s^2}{\ln(n) \cdot m}\right)^{\ln(n)} \leq \left(\frac{e \cdot n^2}{\ln(n) \cdot n^2}\right)^{\ln(n)} = n^{-\ln\ln(n)+1} \\
\mathbf{Pr}[X_c \geq \lambda] &\leq \left(\frac{e \cdot k^3}{\lambda \cdot m_s^2}\right)^{\lambda} \leq \left(\frac{e \cdot \ln^3(n)}{\lambda \cdot n^{3/2}}\right)^{\lambda}
\end{aligned}$$

Thus, $X_s < \ln(n)$ holds with probability $1 - n^{-\alpha}$ for any given α and $X_c < \lambda$ with probability at least $1 - n^{-\lambda}$. The maximum load is constant *w.v.h.p.*

Case: $m \geq n^2$ **and** $m_s \in [n, n \cdot r \cdot \ln(n)]$**.**

We choose $k = (r \cdot \ln(n))^3$.

$$\begin{aligned}
\mathbf{Pr}\left[\, X_s \geq (r \cdot \ln(n))^3 \,\right] &\leq \left(\frac{e \cdot m_s^2}{(r \cdot \ln(n))^3 \cdot m}\right)^{(r \cdot \ln(n))^3} \leq \left(\frac{e \cdot (n \cdot r \cdot \ln(n))^2}{(r \cdot \ln(n))^3 \cdot n^2}\right)^{(r \cdot \ln(n))^3} \\
&= \left(\frac{e}{r \cdot \ln(n)}\right)^{(r \cdot \ln(n))^3} = \left(\frac{n}{r^{ln(n)} \cdot n^{\ln \ln(n)}}\right)^{r^3 \cdot \ln^2(n)} \\
\mathbf{Pr}\left[\, X_c \geq \lambda \,\right] &\leq \left(\frac{e \cdot k^3}{\lambda \cdot m_s^2}\right)^{\lambda} \leq \left(\frac{e \cdot (r \cdot \ln(n))^9}{\lambda \cdot n^2}\right)^{\lambda}
\end{aligned}$$

$X_s < (r \cdot \ln(n))^3$ holds with probability $1 - n^{-\alpha}$ for any given α and $X_c < \lambda$ with probability at least $1 - n^{-\lambda}$. Therefore, *w.v.h.p.*, the maximum load is constant.

Case: $m \geq h \cdot n \cdot \ln(n)$ **and** $m_s \in [1, (n \cdot \ln(n))^{5/12}]$**.**

$$\mathbf{Pr}\left[\, X_s \geq k \,\right] \leq \left(\frac{e \cdot m_s^2}{k \cdot m}\right)^k \leq \left(\frac{e \cdot (n \cdot \ln(n))^{5/6}}{k \cdot h \cdot n \cdot \ln(n)}\right)^k = \left(\frac{e}{k \cdot h \cdot (n \cdot \ln(n))^{1/6}}\right)^k$$

This immediately yields $\ell_{max}^{(s)} \leq X_s \leq 6 \cdot \alpha = \mathcal{O}(1)$ with probability $1 - n^{-\alpha}$ (for any constant α).

Case: $m \geq h \cdot n \cdot \ln(n)$ **and** $m_s \in [(n \cdot \ln(n))^{5/12}, (n \cdot \ln(n))^{7/12}]$**.**

We choose $k = (n \cdot \ln^2(n))^{1/6}$.

$$\begin{aligned}
&\mathbf{Pr}\left[\, X_s \geq (n \cdot \ln^2(n))^{1/6} \,\right] \leq \left(\frac{e \cdot m_s^2}{(n \cdot \ln^2(n))^{1/6} \cdot m}\right)^{(n \cdot \ln^2(n))^{1/6}} \\
&\leq \left(\frac{e \cdot (n \cdot \ln(n))^{7/6}}{(n \cdot \ln^2(n))^{1/6} \cdot h \cdot n \cdot \ln(n)}\right)^{(n \cdot \ln^2(n))^{1/6}} = \left(\frac{e}{(\ln(n))^{1/6} \cdot h}\right)^{(n \cdot \ln^2(n))^{1/6}} \\
&\mathbf{Pr}\left[\, X_c \geq \lambda \,\right] \leq \left(\frac{e \cdot k^3}{\lambda \cdot m_s^2}\right)^{\lambda} \leq \left(\frac{e \cdot (n \cdot \ln^2(n))^{1/2}}{\lambda \cdot (n \cdot \ln(n))^{5/6}}\right)^{\lambda} = \left(\frac{e \cdot \ln^{1/6}(n)}{\lambda \cdot n^{1/3}}\right)^{\lambda}
\end{aligned}$$

So, $X_s < (n \cdot \ln^2(n))^{1/6}$ with probability $1 - n^{-\alpha}$ for any given α and $X_c < 4 \cdot \lambda$ with probability $1 - n^{-\lambda}$. Hence, the maximum load is constant *w.v.h.p.*

Case: $m \geq h \cdot n \cdot \ln(n)$ **and** $m_s \in [(n \cdot \ln(n))^{7/12}, c \cdot (n \cdot \ln(n))^{2/3}]$**.**

We choose $k = (n \cdot \ln^2(n))^{1/3}$.

$$\mathbf{Pr}\left[\, X_s \geq (n \cdot \ln^2(n))^{1/3} \,\right] \leq \left(\frac{e \cdot m_s^2}{(n \cdot \ln^2(n))^{1/3} \cdot m}\right)^{(n \cdot \ln^2(n))^{1/3}}$$

$$\leq \left(\frac{e \cdot c^2 \cdot (n \cdot \ln(n))^{4/3}}{(n \cdot \ln^2(n))^{1/3} \cdot h \cdot n \cdot \ln(n)}\right)^{(n \cdot \ln^2(n))^{1/3}} = \left(\frac{e \cdot c^2}{\ln^{1/3}(n) \cdot h}\right)^{(n \cdot \ln^2(n))^{1/3}}$$

$$\mathbf{Pr}\left[\, X_c \geq \lambda \,\right] \leq \left(\frac{e \cdot k^3}{\lambda \cdot m_s^2}\right)^{\lambda} \leq \left(\frac{e \cdot n \cdot \ln^2(n)}{\lambda \cdot (n \cdot \ln(n))^{7/6}}\right)^{\lambda} = \left(\frac{e \cdot \ln^{5/6}(n)}{\lambda \cdot n^{1/6}}\right)^{\lambda}$$

$X_s < (n \cdot \ln^2(n))^{1/3}$ holds with probability $1 - n^{-\alpha}$ for any given α and $X_c < 7 \cdot \lambda$ with probability $1 - n^{-\lambda}$. This implies that, *w.v.h.p.*, the maximum load is constant. □

Theorem 3.3.5. *Consider the* 2*-choice game in which* $m = C$ *balls are allocated into* n *bins with total capacity* C*. Then,* w.h.p.*, the maximum load is bounded by*

$$\frac{\ln\ln(n)}{\ln(2)} + \mathcal{O}(1).$$

In case $m \geq n^2$*, the maximum load is constant* w.h.p.

Proof. Here we consider two cases for different values of m and n.

$\mathbf{m \geq n^2}$**.** Observation 3.3.2 states that the maximum load in big bins is constant *w.h.p.*, and from Observation 3.3.2 and Lemma 3.3.4 it follows that the same holds for the small bins since

$$\ell_{max} \leq \ell_{max}^{(s)} + \ell_{max}^{(b)} = \mathcal{O}(1).$$

$\mathbf{m < n^2}$**.** Lemma 3.3.3 compares the process in which m balls are allocated into n bins of total capacity C with the process that throws m balls into C unit-sized bins and states that the maximum load of the former is stochastically dominated by the maximum load of the latter. By applying Theorem 2.3.1 on the standard game with m balls and $m = C$ bins, we obtain a bound on the maximum load that is also valid for the first process. *W.h.p.*, the maximum load is

$$\ell_{max} \leq \frac{\ln\ln(m)}{\ln(2)} + \mathcal{O}(1) \leq \frac{\ln\ln(n^2)}{\ln(2)} + \mathcal{O}(1) = \frac{\ln\ln(n)}{\ln(2)} + \mathcal{O}(1).$$

□

The upper bound of $\frac{\ln\ln(n)}{\ln(2)} + \mathcal{O}(1)$ coincides with the upper bound for the standard multiple-choice game (Theorem 2.3.1) which is a special case of our problem. So, the matching lower bound of $\frac{\ln\ln(n)}{\ln(2)} - \mathcal{O}(1)$ (Theorem 2.3.1) is also valid for our problem.

The last theorem also shows that under certain conditions better bounds on the maximum load are possible. In the following we will consider more such cases. The next theorem observes that a constant maximum load is achieved if almost all bins are big.

Theorem 3.3.6. *Consider the* 2*-choice game in which* $m = C$ *balls are allocated into* n *bins with total capacity* C*. Assume that there are* s *small bins with total capacity* m_s *and* $n - s$ *big bins with total capacity* $m - m_s$*. If* $m_s \leq c \cdot (n \cdot \ln(n))^{2/3}$*, then the maximum load is constant* w.h.p.

Proof. Again, we apply Observation 3.3.2 which states that, *w.h.p.*, the height of all balls in $\mathcal{B}_b$ is at most 4 so that the load of the big bins is constant. The small bins additionally receive balls from $\mathcal{B}_s$. We will use Lemma 3.3.4 to estimate the extra load, but in order to apply it, we first have to bound the total capacity m. Note that $s \leq m_s \leq c \cdot (n \cdot \ln(n))^{2/3} = o(n)$ so that, for any constant $h < r$,

$$h < r \cdot \frac{n-s}{n} = r \cdot \frac{n - o(n)}{n}.$$

As there are at least $n - s$ big bins, we obtain

$$m \geq (n-s) \cdot r \cdot \ln(n) > h \cdot n \cdot \ln(n).$$

The bounds on m and m_s allow us to apply Lemma 3.3.4 which states that, *w.v.h.p.*, the maximum load due to balls from $\mathcal{B}_s$ is also bounded by a constant. □

The next observation bounds the maximum load for arbitrary m and n, but it is only applicable if all bins have the same capacity. The result is derived from [9].

Observation 3.3.7. *Consider the game in which all bins have the same capacity* $\bar{c}$*,* m *balls are thrown into* n *bins and each ball comes with* d *choices. Then the maximum load equals the maximum load of the standard game (in which all bins have capacity* 1*) divided by* $\bar{c}$*.*

In case $d = 1$ *we can apply Theorem 2.2.2 that provides different bounds for different values of* m*. If* $d \geq 2$*, we can apply Theorem 2.3.2 so that,* w.v.h.p.*, the maximum load is*

$$\ell_{max} = \frac{1}{\bar{c}} \cdot \left(\frac{m}{n} + \mathcal{O}(\ln \ln(n)) \right).$$

For $m = C = n \cdot \bar{c}$ *in particular we obtain*

$$\ell_{max} = \frac{1}{\bar{c}} \cdot \left(\frac{n \cdot \bar{c}}{n} + \mathcal{O}(\ln \ln(n)) \right) = 1 + \frac{\mathcal{O}(\ln \ln(n))}{\bar{c}}.$$

Proof. Since all capacities are the same, the loads are computed in the same way for all bins and every ball adds the same load to the total load regardless of where it is allocated. Therefore the allocation process equals that of the standard game in which all bins have capacity 1. For the number of balls in the fullest bin the bounds given in [74, 9] can be applied. Finally, we get the load by dividing by the bin's capacity $\bar{c}$. □

Corollary 3.3.8. *If $\bar{c} \in \Omega(\ln\ln(n))$ and if $m = k \cdot n \cdot \bar{c}$ for some arbitrary k, the maximum load is $k + \mathcal{O}(1)$,* w.v.h.p.

Theorem 3.3.9. *Let k and α ($0 < \alpha \leq 1$) be constants. Consider the game in which $\alpha \cdot n$ bins have capacity $q(n)$ and all other bins have capacity smaller $q(n)$. If $q(n) \in \Omega(\ln\ln(n))$, then there is a probability distribution over the bins such that the maximum load will be constant* w.v.h.p. *after the allocation of $m = k \cdot C$ balls.*

Proof. Assign probability $\frac{1}{\alpha \cdot n}$ to all bins with capacity $q(n)$ and probability 0 to all others. Ignoring the bins with probability 0, we may consider this a game of $m = k \cdot C \leq k \cdot n \cdot q(n)$ balls and $\alpha \cdot n$ bins. Applying Observation 3.3.7 we obtain

$$\begin{aligned} \ell_{max} &\leq \frac{1}{q(n)} \cdot \left(\frac{m}{\alpha \cdot n} + \mathcal{O}(\ln\ln(\alpha \cdot n)) \right) \leq \frac{1}{q(n)} \cdot \left(\frac{k \cdot n \cdot q(n)}{\alpha \cdot n} + \mathcal{O}(\ln\ln(n)) \right) \\ &\leq \frac{k}{\alpha} + \frac{\mathcal{O}(\ln\ln(n))}{q(n)} \leq \frac{k}{\alpha} + \mathcal{O}(1) = \mathcal{O}(1). \end{aligned}$$

□

The last result implies that in some cases much better results for the maximum load are possible if one can choose the probabilities oneself.

3.4 Simulations

The purpose of the simulations in this section is two-fold. On the one hand we consider the games analysed in the previous section and demonstrate that the asymptotic bounds behave well in practice. On the other hand we look at special settings not covered previously in this chapter and evaluate the performance of our approach.

Whereas the main focus is on the maximum load in the analytical section, we often consider complete distributions here. In order to obtain more precise results and smoother curves, the experiments are usually repeated $10,000$ times and the values plotted the average values. If not stated otherwise, the probabilities are proportional to the capacities and the number of balls

equals the total capacity. Each ball has two choices and allocates itself to the lesser loaded bin; ties are broken arbitrarily.

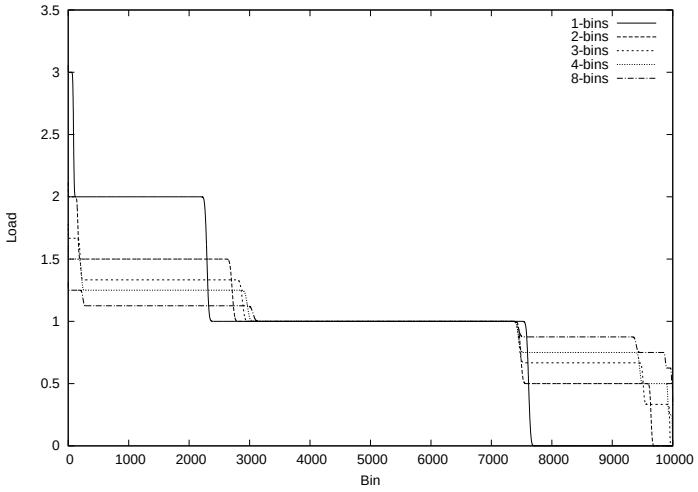

Figure 3.1: Uniform bins.

Among others we will present experiments indicating that our results also hold for a very small number of bins, which is, of course, a setting important for many practical applications.

3.4.1 Uniform Bins

In the first experiment we consider the completely uniform case, that is, all bins have the same capacity. We have $n = 10,000$ bins, and the capacities range around $\ln\ln(n) \approx 2.22$. In Figure 3.1 we plot the normalised load distribution of the entire bin vector for five different capacities, $c = 1, 2, 3, 4, 8$. (In the figure, "x-bins" refers to bins of capacity x.)

According to Observation 3.3.7 the maximum load is $1 + \frac{\mathcal{O}(\ln\ln(n))}{c}$ for $d \geq 2$ and $m = C = c \cdot n$. And in fact in our simulations the maximum load is very close to $1 + \frac{\ln\ln(n)}{c}$ for $c = 2, 3, 4, 8$ and close to $\frac{\ln\ln(n)}{\ln(2)}$ for $c = 1$ (see Theorem 2.3.1 or Theorem 3.3.5).

In Figure 3.2 on page 51 we consider uniform bins and observe how an increase of the number of balls m affects the load distribution for different capacities, $c = 1, 2, 3, 4$. The four plots in Figure 3.2 show, left to right, top to bottom, the load distributions over the entire array of $n = 32$ bins, for $m = C, 10 \cdot C, 100 \cdot C, 1,000 \cdot C$ respectively.

Notice how the deviation from the average load $\frac{m}{n}$ remains constant. In fact the curves for $m = 10 \cdot C, 100 \cdot C, 1,000 \cdot C$ look identical and suggest that the deviation is independent of the number of balls.

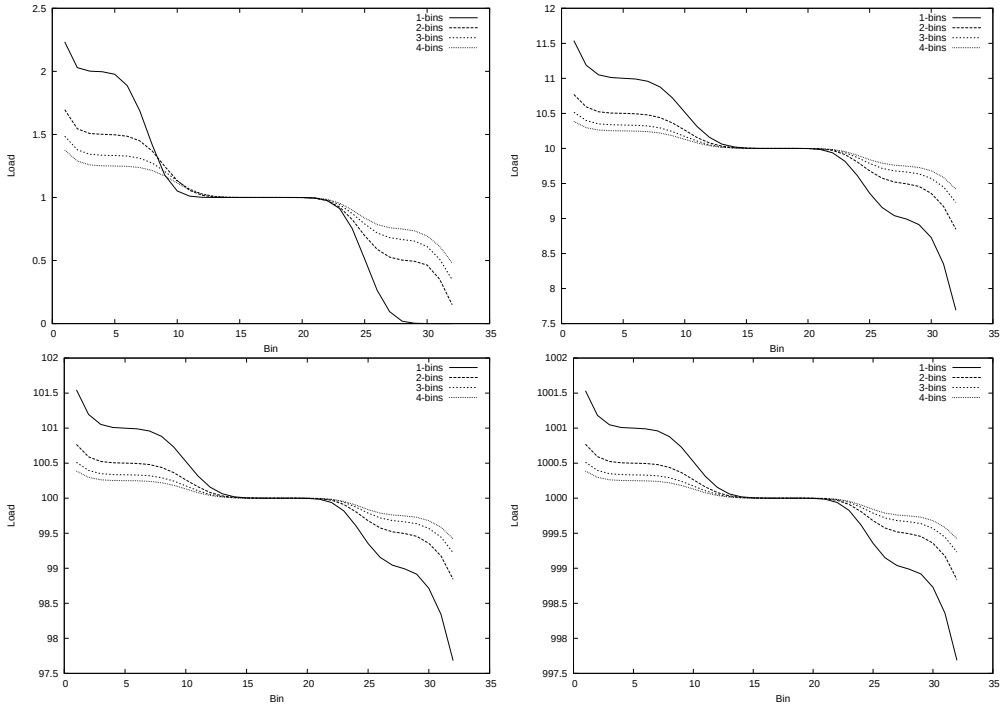

Figure 3.2: Uniform bins, heavily loaded case, varying number of balls.

3.4.2 Distribution in Mixed Arrays

In this section we look at heterogeneous bin arrays. As before we assume that the number of balls m equals the total capacity C and that the bins' probabilities are proportional to the capacities.

Under these circumstances it seems plausible that an increase of the total capacity leads to a decrease in the maximum load because the bigger bins draw balls and a ball in a big bin adds little to the total load. We will present a few simulations that substantiate this assumption. Moreover, we will analyse which type of bins are likely to hold the biggest load.

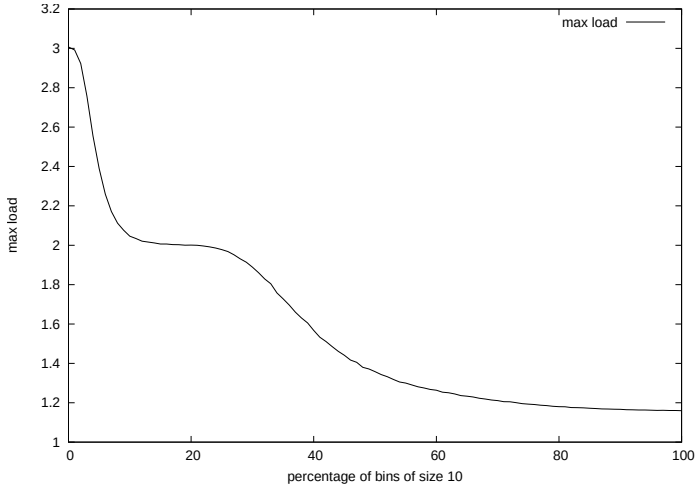

Figure 3.3: Maximum load as a function of the total capacity.

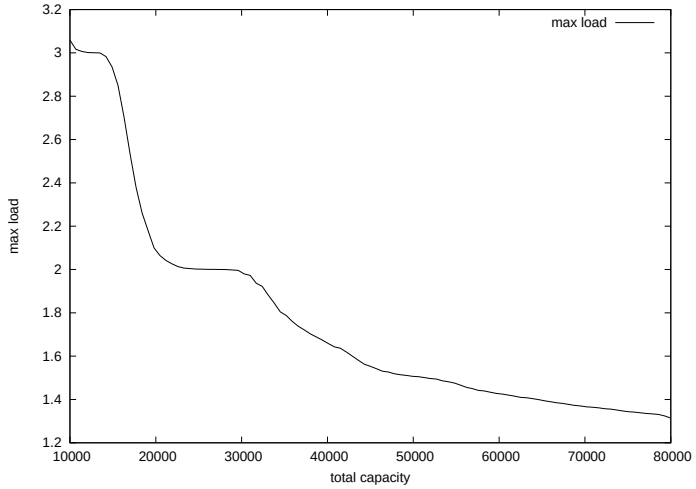

Figure 3.4: Maximum load as a function of the total capacity.

Figures 3.3 and 3.4 show how the maximum load changes when the total capacity is increased. In the first experiment (Figure 3.3) we mix small bins of capacity 1 with large bins of capacity 10. We maintain a fixed number of $n = 1,000$ bins and vary the fraction of large bins on the x-axis from 0% to 100%. We can see clearly that, as expected, the maximum load decreases as the proportion of large bins increases.

In the second experiment (Figure 3.4) we consider the maximum load as a function of the total capacity, but the latter is not obtained by gradually increasing the fraction of large fixed-size bins. Instead, we determine each bin's capacity using a random process in which, for a desired total capacity $C = c \cdot n$ (with c between 1 and 8) the size of each bin is determined by $1+X$ where X is a binomially distributed random variable with $X \sim B(7, \frac{c-1}{7})$. Notice that the total capacity will in general not be precisely equal to $c \cdot n$, but it can be shown, theoretically and experimentally, that it will be very close to it with large probability. The result is very similar to the previous experiment. While increasing the total capacity, the maximum load rapidly decreases.

Note that the slow decrease between 10% and 30% results from a typical effect happening in standard balls-into-bins games (uniform balls and uniform bins) with multiple choices. In these games the number of bins with maximum load of, say ℓ, increases for a long time when the number of allocated balls is increased. The maximum load increases by one only if a sufficient number of bins with load ℓ exist. At first glance it does not seem to be right because the number of balls grows with the number of large bins. But the maximum load is in the small bins (in the settings between 0% and, say, 40%), and from the viewpoint of the small bins the number of balls decrease due to the pull of the large bins.

The plots of Figure 3.5 and 3.6 on page 53 show load distributions for different ratios of small bins and large bins. We consider two cases: In Figure 3.5 we have only 32 bins, and the bin sizes

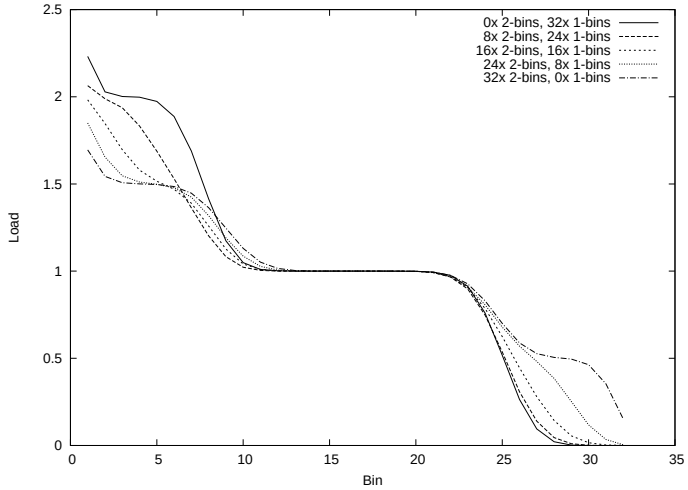

Figure 3.5: Load distributions, two distinct capacities.

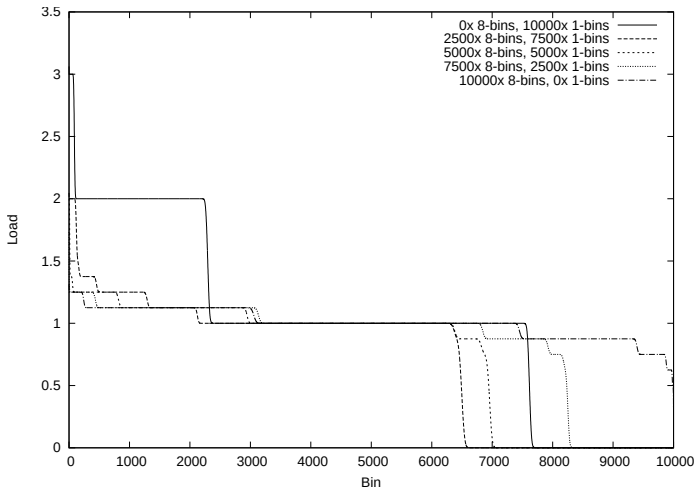

Figure 3.6: Load distributions, two distinct capacities.

are 1 and 2. In Figure 3.6 there are $10,000$ bins, and the bin sizes are 1 and 8. We observe in both plots: The more large bins we have, the more even the load distribution becomes.

The experiment charted in Figure 3.7 and 3.8 on page 54 equals the one in Figure 3.6 as we consider the same ratios of small and large bins; size 1 and 8 respectively. Yet, now we show the results in two separate plots that complement each other. The left part shows only the bins of size 8, the right part only the bins of size 1. (Notice that the curves do not generally span the entire width of the figures as there are simply not in general $n = 10,000$ bins of a given size available.)

Observation 3.3.2 and Theorem 3.3.5 predict a constant load in the large bins and higher loads in some small bins. We can observe that the asymptotical bounds behave very well in our experiment.

Naturally it is almost impossible to draw any valuable conclusions from the (averaged) load

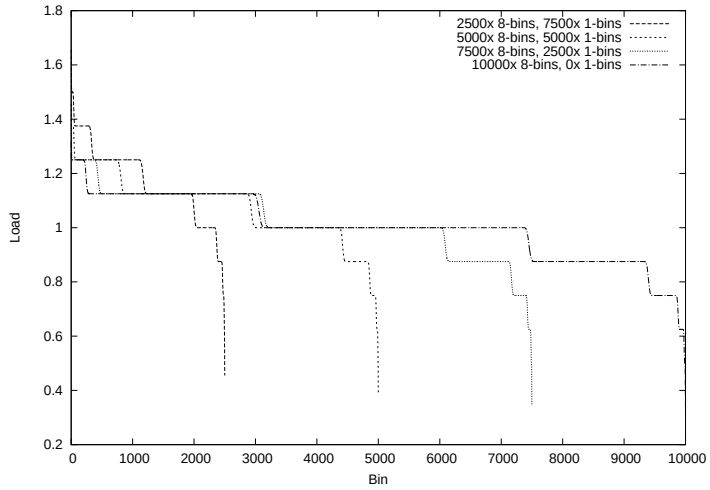

Figure 3.7: Two distinct capacities 8 and 1, but only the bins of size 8 are plotted.

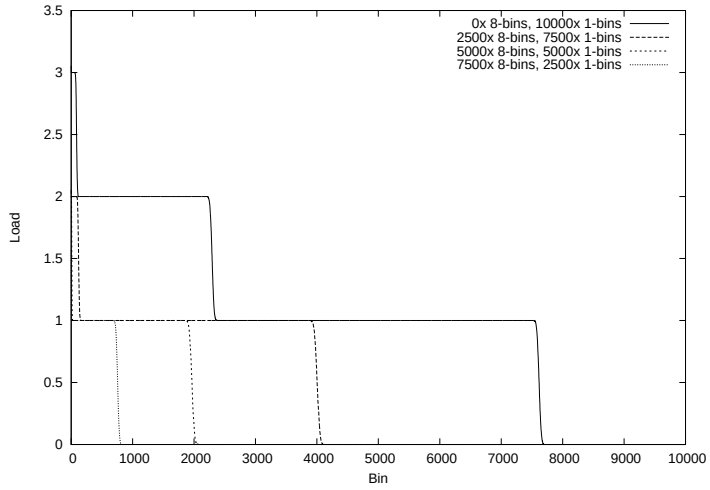

Figure 3.8: Two distinct capacities 8 and 1, but only the bins of size 1 are plotted.

distribution in the experiment concerning the correctness of our theoretical results. The difference between the logarithmic and constant bounds is simply too small. Assuming whether a value such as 2 is $\Theta(1)$, $\Theta(\ln\ln(n))$ or $\Theta(\ln(n))$ would be daring. More significant results in this respect would be possible by increasing n. The selected values in this section are a trade-off between the accuracy and running time of the simulation programs.

We have already seen in Figure 3.7 and 3.8 that the bins with higher loads are likely to be small bins. The experiment depicted in Figure 3.9 provides further indication. Again we have two capacities, 1 and 10, and we consider different ratios on them. We want to see when the maximum load is likely to be in a small or large bin. The total number of bins is $n = 1,000$, and the fraction of large bins varies from 0% to 100%. The plot shows, for each point on the

curve, the fraction of $1,000$ independent runs in which a small bin of capacity 1 was among the maximally loaded.

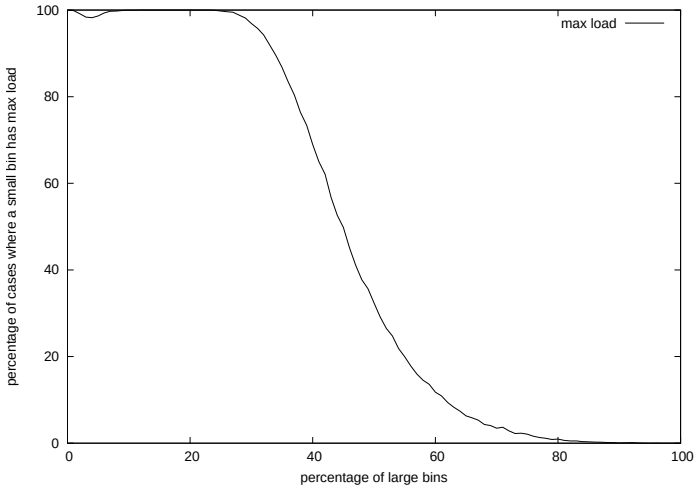

Figure 3.9: Two distinct capacities, location of maximum load.

The maximum load is more likely to be in one of the small bins as long as the pull of the big bins is not too strong. With about 45% large bins the fraction of small bins containing the maximum load drops below 50%. Then the probability to choose a big bin is already $\frac{4500}{4500+550} > 0.89$.

3.4.3 The Heavily Loaded Case

In Figure 3.2 on page 51 we have already seen an example for the heavily loaded case ($m \gg C$) when we simulated the uniform game in which all bins have the same capacity. We observed that, in accordance with Observation 3.3.7, the difference between the maximum load and the average load $\frac{m}{n}$ is independent of the number of balls m. In this section we find indication that the same may hold if the bins have random capacities.

In the experiment that is depicted in Figure 3.10 on page 56 we fix $n = 10,000$ as well as a total capacity C, a multiple of n. We then generate individual bin capacities such that the (expected) total capacity is equal to the prescribed capacity C, using an approach similar to that in Section 3.4.2, Figure 3.4. For each fixed value of C, we throw $100 \cdot C$ many balls into the systems and at certain points throughout this process plot the current deviation of the maximum load from the average load as a function of the number of balls currently in the system (that is, we measure this quantity after the $(i \cdot C)$-th ball for $i = 1, 2, \ldots, 100$). The plot shows one such curve for a variety of values of C. What we see is essentially a bundle of parallel lines, indicating that indeed the deviation of the maximum load from the average does not grow with

the number of balls thrown, apparently regardless of the underlying total capacity. The positions of the lines also match our intuition and predictions as the lines get closer to zero as the total capacity increases, meaning the maximum load approaches the average load for large capacities. Notice that the curves slightly jiggle up and down. One might not expect such behaviour when tracing a term depending on the maximum load (which ought to be monotonic). However, we plot the deviation of the maximum load from the average and this quantity may well decrease (somewhat).

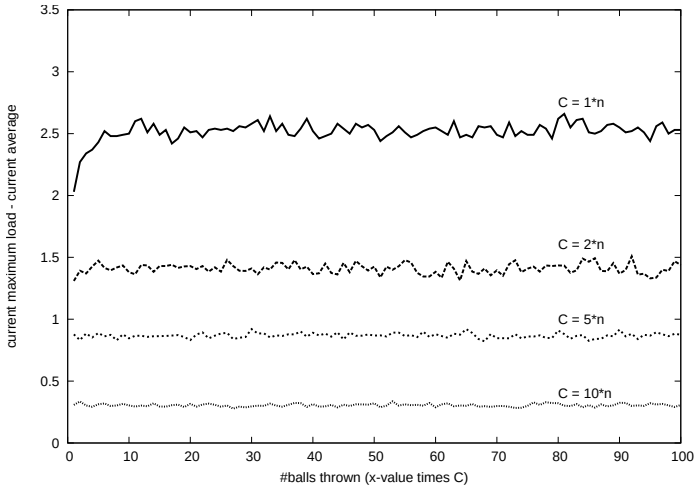

Figure 3.10: Heavily loaded case: Deviation of maximum from average.

3.4.4 Optimal Probability Distribution

So far the probabilities were chosen to be proportional to the capacities. This is a natural approach and works well if the differences between the capacities are small. However, if this is not the case, it might be beneficial to use another strategy and alter the probabilities. Theorem 3.3.9 shows, for instance, that in certain cases in which a constant fraction of all capacities is of order $\ln \ln(n)$, a constant maximum load can be achieved by simply ignoring the low-capacity bins.

Let us consider the following setting: The number of bins is $n = 100$, half of them have capacity $c_i = 1$ and the other half (integer) capacity $c_i = x$, $2 \leq x \leq 14$. The number of balls is $m = C = \sum_{i=1}^{n} c_i$, and the probability of a bin that has capacity c is set to $\frac{c^t}{C(t)}$ where $C(t) = \sum_{i=1}^{n} c_i^t$.

Note that the probabilities sum up to 1 and that bins with the same capacity have the same probability. Since we have only two different capacities, all probabilities are fixed as soon as the

probability for one bin is set. For this probability, however, we can choose any value in the open interval $\left(0, \frac{2}{n}\right)$[1].

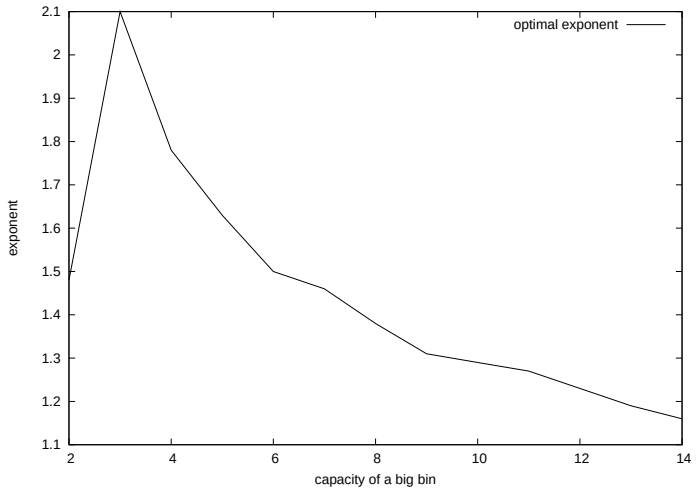

Figure 3.11: Optimal probability distribution: Optimal exponent in case of 50 1-bins and 50 x-bins.

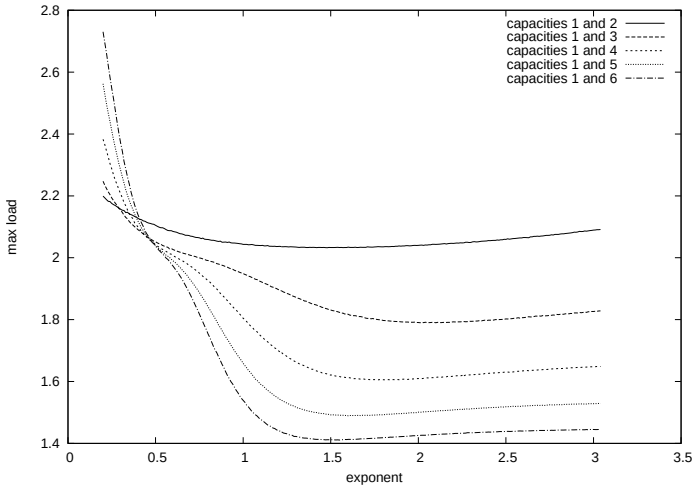

Figure 3.12: Maximum load as a function of the exponent for different capacities.

The question is, given x, what is the optimal exponent t? The curve in Figure 3.11 represents our experimental results. In our experiments we simulate the random allocation according to the altered probability distribution. For every capacity $c \in \{2, 3, ..., 14\}$ and every exponent $t \in \{1, 1.005, ..., 3\}$, the maximum load is averaged over $1,000,000$ repetitions, and the best values for t are used in the plot. It shows that the optimal exponent can differ considerably

[1]Let p denote the probability for a bin with capacity $c \in \{1, x\}$ where $x \in \{2, 3, ...\}$. Since $\frac{c^t}{1+x^t}$ can take any value in $(0, 1)$, it follows:

$$p = \frac{c^t}{C(t)} = \frac{c^t}{\frac{n}{2} \cdot 1^t + \frac{n}{2} \cdot x^t} = \frac{2}{n} \cdot \frac{c^t}{1 + x^t} \in \left(0, \frac{2}{n}\right)$$

from 1. For the array in which 50 bins have capacity 1 and 50 bins capacity 3, the optimal exponent is about 2.1.

Figure 3.12 shows the resulting maximum loads as a function of the exponent for different capacities x. Comparing the values for $t = 1$ and for the optimal t, the difference in the maximum loads is up to 0.2.

3.5 Conclusions and Open Problems

We have analysed the multiple-choice game with unit-sized balls and heterogeneous bins assuming that a bin's load is determined by the number of balls it contains divided by its capacity.

First we assumed that the probabilities of the bins are proportional to their capacities and that the number of balls equals the total capacity of the bins. For the maximum load of the 2-choice game we obtained a bound that is not worse than the one in the standard game [4] which is a special case of our model (all capacities set to one). The generalisation to all $d \geq 2$ is still open. The missing link is Observation 3.3.2 which was only shown for $d = 2$.

In case of uniform bins, that is, all bins have the same capacity, we also considered the heavily-loaded case ($m \gg C$). Based on [8], we found that the deviation from the average load does not grow with the number of balls. Simulations suggest that this might generally be the case for arbitrary capacity distributions. Future work could address this problem analytically.

Other experiments indicate that the constants in the asymptotic bounds are small so that our strategy can also be employed in applications with a small number of bins. Furthermore, analytical and experimental results show that it can be beneficial to choose differing probability distributions over the heterogeneous bins. It would be interesting to continue the work and obtain more general results, but it seems difficult to analyse these games mathematically.

4 Balls into Bins with Related Random Choices

We consider a multiple-choice balls-into-bins game in which the random choices are neither uniform nor independent. We use Godfrey's hypergraph-based model [34] and improve and extend his results. The findings presented in this chapter have appeared in preliminary form in the following paper:

[6] Petra Berenbrink, André Brinkmann, Tom Friedetzky, and Lars Nagel. Balls into bins with related random choices. In *Proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures*, SPAA '10, pages 100–105, New York, NY, USA, 2010. ACM

The propositions and proofs have been enhanced and new results have been added.

4.1 Introduction

In this chapter a variation of the *balls-into-bins* game is considered which has been introduced by Godfrey [34]. In his model $m < n$ balls have to be allocated into n bins with a maximum load of one. In contrast to the standard multiple-choice game from [3], he assumes that the choices of the bins are *not* uniform and independent at random. Every ball comes with a set of *clusters*, where each cluster is simply a set of bins. The ball will randomly pick a cluster and then commit to one of the least loaded bins within that cluster. For every ball the probability that a fixed bin is in the chosen cluster has to be roughly the same. Hence, the assignment of the bins to the clusters can be arbitrary or even regular as long as every bin is in roughly the same amount of clusters.

We look at a generalisation of Godfrey's model. Again, we assume that each ball comes with a set of clusters and that it randomly picks a cluster and a least loaded bin within it. In contrast to Godfrey, we only require that on average (the average is taken over the choices of all balls) any bin will occur in not too many chosen clusters. This model is captured by what we will

introduce in Definition 4.2.2 and Definition 4.2.3 as one-sided probabilistic balancedness and averaged balancedness, respectively. A further generalisation of the original model assumes that the same cluster can be chosen in multiple successive steps.

4.1.1 Related Work

A detailed overview of the related work can be found in Chapter 2. Here we restrict our attention to previous work that is relevant to the results presented in this chapter. That is, we concentrate on protocols that achieve a constant maximum load and on settings where the balls' choices of bins are not (necessarily) independent or uniform.

The model used in this paper may be regarded as a generalisation of the d-choice model in that a cluster represents the d choices of a ball. Applying the bounds for the multiple-choice game stated in Section 2.3, we obtain that the GREEDY$[d]$ protocol of [3] yields constant load per bin for $d = \ln(n)$ and the GOLEFT protocol of [90] for $d = \ln\ln(n)$.

Byers *et al.* [17, 18] consider a model in which the probability distribution over the bins is not uniform. The motivation for this model comes from the properties of P2P networks like Chord [84] which apply consistent hashing [43] to allocate items / requests (balls) to peers (bins). In the basic scenario, the deviation from the average probability $\frac{1}{n}$ becomes $\Omega(\ln(n))$ *w.h.p.* Byers *et al.* show that this imbalance only leads to a small shift in the maximum load [18]. Wieder proves in [94] that the same holds true in the heavily loaded case ($m \gg n$) only if the number of choices d is allowed to grow (slightly) with the imbalance. (For a more detailed description see Section 2.5.)

4.1.1.1 Godfrey's Model and Results

Most relevant to our results are the graph-based models of Kenthapadi and Panigrahy [47] and Godfrey [34] in which the balls' choices are not uniform and independent. The former consider the 2-choice game and identify the bins with vertices in an underlying graph G. Each ball can only choose pairs of bins that are connected by an edge in G. The surprising result is that, compared to the standard 2-choice game in [3], the bound on the maximum load is basically the same if the graph is (almost) n^ϵ-regular and if ϵ is not too small. For $\epsilon > \frac{8 \cdot \ln\ln(n)}{\ln(n)}$, the maximum load is $\ln\ln(n) + \mathcal{O}(\epsilon^{-1}) + \mathcal{O}(1)$ *w.h.p.* Generally, for Δ-regular graphs they show an upper bound of $\ln\ln(n) + \mathcal{O}\left(\frac{\ln(n)}{\ln(\Delta/\ln^4(n))}\right) + \mathcal{O}(1)$ which is nearly matched by their lower bound of $\ln\ln(n) + \frac{\ln(n)}{\ln(\Delta)+\ln\ln(n)}$.

Godfrey [34] extends this model to d-uniform multi-hypergraphs in which each hyperedge

connects d of the n vertices representing the bins. Each ball comes with a possibly different multi-hypergraph (called cluster set) and chooses one hyperedge (called cluster) *i.u.r.*. In this cluster it then allocates itself to a random bin among the least loaded. Note that the clusters are allowed to overlap and that the number of clusters is not bounded. Yet, similar to [17, 18, 94], Godfrey demands that the probabilities of the bins must not deviate too much or, more precisely, that the bins are β-balanced:

Definition 4.1.1 (Definition 2.1 in [34])**.** *A random set of bins B is β-balanced if, for all bins j,*

$$\frac{1}{\beta \cdot n} \leq \mathbf{Pr}\left[\, j \in B \,\right] \cdot \mathbb{E}\left[\frac{1}{|B|} \mid j \in B\right] \leq \frac{\beta}{n}.$$

The main contribution in his paper is the following theorem which upper bounds the number of balls m such that, *w.h.p.*, the protocol (Algorithm 2) succeeds in finding an allocation with maximum load equal to one:

Theorem 4.1.2 (Theorem 2.1 in [34])**.** *Let $\epsilon > 0$, $\delta \in (0,1)$, and suppose that for each ball i, B_i is β-balanced and $|B_i| \geq 26 \cdot \frac{(1+\epsilon)^2}{\epsilon^2 \cdot \delta} \cdot \ln(n)$. Let $\beta' = (1 + \epsilon + o(1)) \cdot \beta$ and $\alpha = (1-\delta)/\lceil 1 - \frac{\ln(\beta')}{\ln(1-(\beta'-1)/(\beta'^2-1))} \rceil$. Then with probability $1 - \mathcal{O}(n^{-1})$, the maximum load is one after placing $m = \alpha \cdot n$ balls, and the maximum load is $\lceil \frac{1}{\alpha} \rceil$ after placing $m = n$ balls.*

The upper bound on the maximum load in the case $m = n$ follows from the first bound by trivially running the original algorithm $\lceil \frac{1}{\alpha} \rceil$ times.

A simplified version of Theorem 4.1.2 is:

Corollary 4.1.3. *Suppose that for each ball i, B_i is β-balanced and $|B_i| \geq c \cdot \ln(n)$ where c is a sufficiently large constant. Then with probability $1 - \mathcal{O}(n^{-1})$, the maximum load is one after placing $m \leq \frac{n}{f_c(\beta)}$ balls, where $f_c(\beta) \in \Theta(\beta \cdot \ln(\beta))$ depends solely on β and c.*

Proof. In order to prove the bounds on $f_c(\beta) = \frac{1}{\alpha}$, we take a closer look at the expression $\frac{1-\delta}{\alpha}$ (from Theorem 4.1.2). For this we use that, for $x > 0$, $\frac{1}{x+1} < \ln(x+1) - \ln(x) < \frac{1}{x}$ (Lemma 1.1.5).

Upper bound:

$$\begin{aligned}
\frac{1-\delta}{\alpha} &= \left\lceil 1 - \frac{\ln(\beta')}{\ln\left(1 - \frac{\beta'-1}{\beta'^2-1}\right)} \right\rceil = \left\lceil 1 - \frac{\ln(\beta')}{\ln\left(1 - \frac{\beta'-1}{(\beta'-1)\cdot(\beta'+1)}\right)} \right\rceil \\
&= \left\lceil 1 - \frac{\ln(\beta')}{\ln\left(\frac{\beta'}{\beta'+1}\right)} \right\rceil = \left\lceil 1 - \frac{\ln(\beta')}{\ln(\beta') - \ln(\beta'+1)} \right\rceil \\
&= \left\lceil \frac{\ln(\beta'+1)}{\ln(\beta'+1) - \ln(\beta')} \right\rceil \overset{(L.\ 1.1.5)}{<} \lceil (\beta'+1) \cdot \ln(\beta'+1) \rceil
\end{aligned}$$

Lower bound:

$$\frac{1-\delta}{\alpha} = \left\lceil \frac{\ln(\beta'+1)}{\ln(\beta'+1)-\ln(\beta')} \right\rceil \overset{(L.\ 1.1.5)}{>} \lceil \beta' \cdot \ln(\beta'+1) \rceil$$

Therefore,

$$\frac{\lceil \beta' \cdot \ln(\beta'+1) \rceil}{1-\delta} < f_c(\beta) = \frac{1}{\alpha} < \frac{\lceil (\beta'+1) \cdot \ln(\beta'+1) \rceil}{1-\delta}.$$

Since δ and ϵ are constants and since $\beta' = (1+\epsilon+o(1)) \cdot \beta$, it follows that

$$f_c(\beta) \in \Theta(\beta \cdot \ln(\beta)).$$

□

4.1.2 Contributions

Our contributions are the following:

- We improve Godfrey's upper bound on the number of balls m from $\frac{n}{\Theta(\beta \cdot \ln \beta)}$ to $\frac{n}{\Theta(\beta)}$ (Theorem 4.3.3). This is asymptotically optimal (β need not be constant) as we show in Observation 4.3.6.
- We enhance the original model by the concept of *runs*. In this new model, each ball i tosses a (biased) coin: With constant probability p, $0 < p < 1$, it runs the protocol as described above, but with the remaining probability it copies the previous ball's choice B_{i-1}, that is, it reuses the previous cluster of bins. In Theorem 4.4.1 we prove the same asymptotic upper bound on m.
- We introduce relaxed definitions of balancedness (one-sided balancedness and averaged balancedness) and show that our results hold for these models.
- Aside from the lower bound on the number of balls m, we also show an asymptotically matching lower bound for the cluster size d in Observation 4.3.5.
- We considerably simplify Godfrey's original proof. Our proofs are essentially applications of Chernoff bounds, where [34] employs a relatively complicated coupling argument.

While the concept of balancedness allows one to investigate the *balls-into-bins* model in the presence of bounded dependencies, the runs, in addition to the obvious effect of saving on randomness, are also of practical relevance (*e.g.* for cloud computing). We are not aware that this

particular model introducing runs (with or without the concept of balancedness) has previously been studied.

4.1.3 Motivation

With regards to the dependencies as mentioned above, consider, for example, the case where the balls and the bins are distributed as points in $\mathbb{R}^2$. This is actually a relevant model, since when designing and analysing *e.g.* peer-to-peer or cloud-based systems, it is frequently supposed that the participants (peers in the former case, users and data servers in the latter) are embedded into some geometric space. B_i will then consist of the bins closest to ball i *w.r.t.* the embedding and a given distance metric. In this case, the distribution of bins in the set B_i is not chosen *i.u.r.*, and the geographical distance between two bins may determine their probability to be in a joint set B_i.

As already hinted at, the model can be considered as being motivated by demands arising in cloud and grid computing. If a cloud or grid provider accepts to run a job, it should place this job as near as possible to the data being accessed by this job. Otherwise, access latencies may substantially reduce the performance for this job. Nevertheless, the provider increases the number of choices by either replicating frequently accessed data to different computing centres, or by allowing to distribute the data over multiple data centres [71, 87]. Translating back to our scenario, none of these approaches will generally result in a perfectly uniform and independent choice of bins for our balls. This model will be analysed in Section 4.3.

The cloud scenario includes an additional extension to the standard *balls-into-bins* games, where the selection of bins may or may not depend on the choices of previous balls. It may be presumed that there is a given probability that the peer accessing the cloud in step i will also be the peer accessing the network in step $i + 1$, and therefore, in our setting, that $B_{i+1} = B_i$. The underlying process is that a new peer in a cloud environment typically moves multiple objects, like big databases, into the cloud after entering it for the first time. After this initialisation step, the allocated storage capacity typically stays relatively invariant. These *runs* are analysed in Section 4.4.

4.2 Models and Definitions

In this section we introduce notation used in the remaining technical sections. Balls are numbered $1, \ldots, m$ and bins are denoted $b_1, \ldots, b_n$. Ball i comes with a set of s_i many clusters of bins $\mathcal{B}_i = \{B_1, \ldots, B_{s_i}\}$. Each such cluster contains $c \cdot \ln(n)$ many bins where $c = c(n)$ is chosen

so that $c \cdot \ln(n)$ is an integer. (In all results c is lower-bounded by a constant and can be chosen smaller than a constant[1].) Each $\mathcal{B}_i$ contains arbitrarily many such clusters, subject to it being β-balanced (see Definitions 4.2.2 and 4.2.3). In a model with *runs* we assume that every ball i has the choice between the cluster that was used by ball $i-1$ and a newly chosen one from $B_i \in \mathcal{B}_i$. For details see Algorithm 3.

The load vector is defined as before. The load vector L_C of a cluster C is the load vector restricted to the bins in C.

4.2.1 Goodness of Balls

Let $\textsc{Empty}(B)$ denote the number of empty bins in any given cluster B.

Definition 4.2.1 (Good and bad balls)**.** *For $i = 1, \ldots, m$, we call the i-th ball* good *if it finds strictly more than half of its chosen cluster empty, that is,* $\textsc{Empty}(B_i) > \frac{|B_i|}{2} = \frac{c}{2} \cdot \ln(n)$. *Otherwise we label it as* bad.

The factor of $\frac{1}{2}$ has been chosen for convenience; in principle any constant would do just as well. (We do generally not attempt to optimise any constants.) Occasionally the *goodness* of balls $1, \ldots, i$ is referred to as the induction hypothesis for ball $i+1$. Note that the induction base is trivial.

4.2.2 β-balancedness

Our definition of *one-sided β-balancedness* is based on Definition 4.1.1, quoted from [34]. Since we will assume that all clusters have the same size $c \cdot \ln(n)$, we simplify the definition slightly. More importantly, we are able to drop the lower bound $\frac{1}{\beta \cdot n}$.

Definition 4.2.2 (One-sided β-balancedness)**.** *For $\beta \geq 1$, a set of clusters $\mathcal{B}_i$ is β-balanced if for all bins j and* i.u.r. *chosen $B_i \in \mathcal{B}_i$,*

$$\mathbf{Pr}\left[\, j \in B_i \,\right] \leq \frac{\beta \cdot c \cdot \ln(n)}{n}.$$

For the proofs in this chapter it is even sufficient if $\mathbf{Pr}\left[\, j \in B_i \,\right] \leq \frac{\beta \cdot c \cdot \ln(n)}{n}$ *on average*, where the average is taken over all balls. We call this generalisation of one-sided β-balancedness *averaged β-balancedness*:

[1] If the only restriction on c is $c > \hat{c}$ for some constant $\hat{c} \geq 1$, then we can choose c to be $c := \hat{c} \cdot \frac{\lceil \hat{c} \cdot \ln(n) \rceil}{\hat{c} \cdot \ln(n)} < 2 \cdot \hat{c}$ (provided that $n \geq 2$).

Definition 4.2.3 (Averaged β-balancedness)**.** *For $\beta \geq 1$, a sequence of clusters $\mathcal{B}_1, \ldots, \mathcal{B}_m$ of sets is averaged β-balanced if for all bins j and* i.u.r. *chosen $B_1 \in \mathcal{B}_1, \ldots, B_m \in \mathcal{B}_m$,*

$$\sum_{i=1}^{m} \mathbf{Pr}\left[\, j \in B_i \,\right] \leq m \cdot \frac{\beta \cdot c \cdot \ln(n)}{n}.$$

The elimination of the lower bound and the relaxation of the upper bound make the model much more practical. In particular, it is now feasible that every ball's cluster set consists of only one cluster. Then the bins' probabilities in the cluster sum up to 1, whereas the remaining $n - d$ bins have probability 0. In Godfrey's model on the other hand, every bin must have a probability greater than 0 which implies that it must be in at least one cluster of every cluster set. Additionally, its probability is upper-bounded by $\frac{\beta \cdot d}{n}$ for each ball.

4.2.3 Protocols / Models

Basic model. Similar to Godfrey's paper [34], each ball $i = 1, \ldots, m$ runs the protocol presented in Algorithm 2. It chooses a cluster *i.u.r.* and allocates itself to a randomly chosen bin among the bins of lowest load within the cluster.

Algorithm 2 The simple protocol for ball $i \in \{1, \ldots, m\}$

1: *i.u.r.* choose a cluster of bins $B_i \in \mathcal{B}_i$
2: *i.u.r.* choose a bin $b \in B_i$ of lowest load
3: allocate the ball to bin b

Compared to Godfrey's results, the main difference is that our results hold for a larger number of balls m and for averaged β-balanced sequences of clusters (see Theorem 4.3.3 and 4.4.1).

Model with runs. In the extended model, allowing for runs, we consider the algorithm as described in Algorithm 3. In contrast to Algorithm 2, each ball i reuses the previous cluster B_{i-1} with constant probability $p \in (0, 1)$. We assume that there is a randomly preselected set of bins B_0.

Algorithm 3 The extended protocol for ball $i \in \{1, \ldots, m\}$

1: with constant probability p reuse cluster B_{i-1}, with the remaining probability $1 - p$ choose a new cluster $B_i \in \mathcal{B}_i$ *i.u.r.*; either way, let B denote the chosen cluster
2: *i.u.r.* choose a bin $b \in B$ of lowest load
3: allocate the ball to bin b

4.3 Analysis of the Basic Model

4.3.1 Upper Bounds

To ease the analysis of the algorithms we will consider the following variant: For all balls $j \in [m]$ we place a *token* with label j into *each* bin b of cluster B_j. Then, for all balls $j \in [m]$ we allocate the ball into a least loaded bin *i.u.r.* chosen from the bins containing a token labelled with j. We say a token with label j is *redeemed* if the corresponding bin receives ball j.

Lemma 4.3.1. *Assume that we run the simple protocol (Algorithm 2) and that the bin choices are one-sided averaged β-balanced. Let g be a positive constant. If $m \leq \frac{n}{g \cdot \beta}$ and $c \geq 3 \cdot g \cdot (k+1)$, every bin will receive at most $\frac{2 \cdot c \cdot \ln(n)}{g}$ tokens with probability at least $1 - n^{-k}$.*

Proof. Fix any bin b. Consider Bernoulli random variables $X_1, \ldots, X_m$ with $X_q = 1$ if b contains a token with label q, and $X_q = 0$ otherwise. Let $X = X_1 + \ldots + X_m$ count the number of tokens. Since the system is averaged β-balanced, we get the expected value

$$\mathbb{E}[X] = \sum_{q=1}^{m} \mathbb{E}[X_q] = \sum_{q=1}^{m} \mathbf{Pr}\left[\, X_q = 1 \,\right] = \sum_{q=1}^{m} \mathbf{Pr}\left[\, b \in B_q \,\right] \leq m \cdot \frac{\beta \cdot c \cdot \ln(n)}{n}.$$

Define $\mu = \frac{m \cdot \beta \cdot c \cdot \ln(n)}{n}$. Since $m \leq \frac{n}{g \cdot \beta}$, we have $\mu \leq \frac{c \cdot \ln(n)}{g}$. Using Lemma 1.2.1, we obtain

$$\mathbf{Pr}\left[X \geq \frac{2 \cdot c \cdot \ln(n)}{g} \right] \leq e^{-\frac{c \cdot \ln(n)}{3 \cdot g}} = n^{-\frac{c}{3 \cdot g}} \leq n^{-(k+1)}$$

for $c \geq 3 \cdot g \cdot (k+1)$. Hence, with probability $1 - n^{-k}$ no bin receives more than $\frac{2 \cdot c \cdot \ln(n)}{g}$ tokens. □

The following lemma relates the number of tokens per bin to the goodness of a ball.

Lemma 4.3.2. *Assume that the simple protocol (Algorithm 2) is run and consider any ball i, $i \in [m]$. Let $h > 6$ be a constant. If all previous balls are good, if $c \geq 6 \cdot h \cdot k$ and if every bin in B_i has at most $\frac{c \cdot \ln(n)}{h}$ many tokens, then ball i will be good with probability $1 - n^{-k}$.*

Proof. In order to prove that ball i is good, we need to show that more than half of the bins in B_i are empty when we throw ball i.

Since every bin in B_i has at most $\frac{c \cdot \ln(n)}{h}$ tokens, the total number of tokens in bins from B_i is at most $\frac{(c \cdot \ln(n))^2}{h}$. This immediately gives us a bound on the number of times that bins from B_i can appear in previously selected sets B_j, $1 \leq j < i$:

$$\sum_{j=1}^{i-1} |B_i \cap B_j| \leq \frac{(c \cdot \ln(n))^2}{h}. \tag{4.1}$$

Define random variables Y_j with $Y_j = 1$ if a token in $B_i \cap B_j$ is redeemed in step j and $Y_j = 0$ otherwise. The probability that a particular empty bin in B_j receives the ball (meaning the token is redeemed) is at most $\frac{2}{c \cdot \ln(n)}$ due to the induction hypothesis that all previous balls have been good. Hence,

$$\mathbf{Pr}\left[\, Y_j = 1 \,\right] \leq \frac{|\textsc{Empty}(B_i \cap B_j)| \cdot 2}{c \cdot \ln(n)} \leq \frac{|B_i \cap B_j| \cdot 2}{c \cdot \ln(n)}. \tag{4.2}$$

Let $Y := \sum_{j=1}^{i-1} Y_j$ count the balls in B_i. Using (4.1) and (4.2), we can bound the expected value by

$$\begin{aligned} \mathbb{E}[Y] &= \sum_{j=1}^{i-1} \mathbb{E}[Y_j] = \sum_{j=1}^{i-1} \mathbf{Pr}\left[\, Y_j = 1 \,\right] \leq \frac{2}{c \cdot \ln(n)} \cdot \sum_{j=1}^{i-1} |B_i \cap B_j| \\ &\leq \frac{2}{c \cdot \ln(n)} \cdot \frac{(c \cdot \ln(n))^2}{h} = \frac{2 \cdot c \cdot \ln(n)}{h} \end{aligned}$$

Notice that the Y_j are not independent, but negatively correlated and, thus, satisfy the conditions of Lemma 1.2.3. We apply said lemma with $\mu = \frac{2 \cdot c \cdot \ln(n)}{h}$ and obtain

$$\mathbf{Pr}\left[Y \geq \left(1 + \frac{1}{2}\right) \cdot \mu = \frac{3 \cdot c \cdot \ln(n)}{h} \right] \leq e^{-\frac{2 \cdot c \cdot \ln(n)}{4 \cdot h \cdot 3}} = n^{-\frac{c}{6 \cdot h}} \leq n^{-k}$$

which holds for every $c \geq 6 \cdot h \cdot k$. This implies that with a probability of $1 - n^{-k}$ there are at most

$$\frac{3 \cdot c \cdot \ln(n)}{h} < \frac{1}{2} \cdot c \cdot \ln(n)$$

many redeemed tokens in B_i and that the i-th ball is good. □

Theorem 4.3.3. *Let $m \leq \frac{n}{24 \cdot \beta}$ and $c \geq 72 \cdot (k+2)$ where $\beta \geq 1$ and k is some positive constant. Assume that the bin choices are one-sided (averaged) β-balanced. After running Algorithm 2 for each ball, the maximum load is 1 with probability at least $1 - n^{-k}$.*

Proof. Since the one-sided β-balanced system is a special case, it is sufficient to prove the statement for an averaged β-balanced system. For $i \in [m]$, let $\mathcal{E}_i$ denote the event that balls $1, \ldots, i$ are good. For $i \in [m]$, let $\mathcal{F}_i$ denote the event that ball i is the first bad ball, that is, $\mathbf{Pr}\left[\, \mathcal{F}_i \,\right] = \mathbf{Pr}\left[\, \text{ball } i \text{ bad} \mid \mathcal{E}_{i-1} \,\right]$. Let $\mathcal{F}_0$ denote the event that no ball is bad, *i.e.*, $\mathcal{F}_0 = \mathcal{E}_m$. If we consider the probability space of all possible combinations of good and bad balls then

$\{\mathcal{F}_i\}_{i=0,\dots,m}$ defines a partition. Therefore,

$$\begin{aligned}\mathbf{Pr}\left[\neg\mathcal{E}_m\right] &= \sum_{i=0}^{m}\mathbf{Pr}\left[\neg\mathcal{E}_m \mid \mathcal{F}_i\right]\cdot\mathbf{Pr}\left[\mathcal{F}_i\right] \\ &= \mathbf{Pr}\left[\neg\mathcal{E}_m \mid \mathcal{F}_0\right]\cdot\mathbf{Pr}\left[\mathcal{F}_0\right] + \sum_{i=1}^{m}\mathbf{Pr}\left[\neg\mathcal{E}_m \mid \mathcal{F}_i\right]\cdot\mathbf{Pr}\left[\mathcal{F}_i\right] \\ &= \sum_{i=1}^{m}\mathbf{Pr}\left[\neg\mathcal{E}_m \mid \mathcal{F}_i\right]\cdot\mathbf{Pr}\left[\mathcal{F}_i\right] = \sum_{i=1}^{m}\mathbf{Pr}\left[\mathcal{F}_i\right] = \sum_{i=1}^{m}\mathbf{Pr}\left[\text{ball } i \text{ bad} \mid \mathcal{E}_{i-1}\right].\end{aligned}$$

Fix any $i \in [m]$. In the following we will upper-bound $\mathbf{Pr}\left[\mathcal{F}_i\right] = \mathbf{Pr}\left[\text{ball } i \text{ bad} \mid \mathcal{E}_{i-1}\right]$. For this consider the variant of Algorithm 2, described at the beginning of this section, that places tokens into the bins.

The number of tokens any bin receives can be bounded as follows: Define $g := 24$. Since $m \leq \frac{n}{24\cdot\beta} = \frac{n}{g\cdot\beta}$ and $c \geq 72\cdot(k+2) = 3\cdot g\cdot(k+2)$, we can apply Lemma 4.3.1 and get that, with probability $1 - n^{-(k+1)}$, no bin receives more than $\frac{2\cdot c\cdot\ln(n)}{g}$ tokens.

Now we can use this bound to show that, *w.v.h.p.*, ball i is good. Define $h := 12$. Since $c \geq 72\cdot(k+2) = 6\cdot h\cdot(k+2)$ and since every bin in B_i has at most $\frac{c\cdot\ln(n)}{12} = \frac{c\cdot\ln(n)}{h}$ tokens, we can apply Lemma 4.3.2 and get that ball i is good with probability at least $1 - n^{-(k+2)}$.

Since this holds for all balls, the probability for (at least) one of the m balls being bad is

$$\mathbf{Pr}\left[\neg\mathcal{E}_m\right] = \sum_{i=1}^{m}\mathbf{Pr}\left[\mathcal{F}_i\right] \leq \sum_{i=1}^{m} n^{-(k+2)} \leq \frac{n}{24\cdot\beta}\cdot n^{-(k+2)} < n^{-(k+1)}.$$

Both probabilities together imply that the probability for any bin receiving more than one ball is bounded by

$$n^{-(k+1)} + n^{-(k+1)} < n^{-k}.$$

□

4.3.2 Lower Bounds

We will provide lower bounds for the cluster size d as well as for number of balls m. In order to prove the former we will use the following observation. Let $\mathcal{G}_d$ denote an arbitrary game in which the n bins are divided into the same $\frac{n}{d}$ disjoint clusters of size d for every ball i. Thus, all $\mathcal{B}_i$, $i \in [m]$, are identical and 1-balanced.

Observation 4.3.4. *Given the number of bins n, fix any integer $d > 0$ that divides n. Then, for all $d' \in [d-1]$, there exists a game $\mathcal{H}_{d'}$ on n bins in which all clusters have size d', in which all cluster sets are 1-balanced, and whose load vector majorises the load vector of game $\mathcal{G}_d$.*

Proof. Assume that the bins in G_d and $\mathcal{H}_{d'}$ are numbered from 1 to n. We call a bin j, $j \in [n]$, in one game the *corresponding* bin of bin j in the other game.

We define $\mathcal{H}_{d'}$ as follows: For every cluster C_i in $\mathcal{G}_d$ we add d clusters $C_{i,j}$, $j \in [d]$, of size d' to $\mathcal{H}_{d'}$. Let the bins in C_i be numbered from 1 to d, then each cluster $C_{i,j}$ contains the (corresponding) d' bins from j to $j + d' - 1 \pmod d$. (Note that the cluster set is 1-balanced since every bin is contained in exactly d' clusters.)

We use a simple coupling that exploits the fact that the probability for choosing cluster C_i in game $\mathcal{G}_d$ equals the probability that any of the d clusters $C_{i,j}$ in game $\mathcal{H}_{d'}$ is selected. Thus, we can map the random choice of a cluster C_i in game $\mathcal{G}_d$ to the random choices of the d clusters $C_{i,j}$ in game $\mathcal{H}_{d'}$. Now, whenever cluster C_i and any cluster $C_{i,j}$ are chosen, the ball in game $\mathcal{G}_d$ can make the optimal choice among the d bins because it sees all of them. In the game $\mathcal{H}_{d'}$, however, the ball can only select one of the d' bins in $C_{i,j}$ which possibly leads to a worse choice. Since the number of balls is identical in C_i and $\bigcup_{j\in[d]} C_{i,j}$ at all times, the load vector of cluster C is always majorised by the load vector of $\bigcup_{j\in[d]} C_{i,j}$, and since this holds for all clusters in $\mathcal{G}_d$, the load vector of game $\mathcal{G}_d$ is always majorised by the load vector of game $\mathcal{H}_{d'}$. □

The following observation provides an asymptotically matching lower bound for the cluster size. It improves on Godfrey's Theorem 3.1 in [34] and, thus, answers his open question about a better lower bound.

Observation 4.3.5. *Fix the number of bins $n \geq 1,000$ and $c = c(n) \in [0.8, 1]$ in such a way that $c \cdot \ln(n)$ is an integer that divides n. If the number of balls is $m \geq 0.4 \cdot n$, then, for any $d \in \{1, 2, ..., c \cdot \ln(n)\}$, there exists a 1-balanced distribution of clusters B_i with $|B_i| = d$ such that,* w.h.p.*, Algorithm 2 results in a maximum load greater than* 1.

Proof. Because of Observation 4.3.4 we only have to prove the case $d = c \cdot \ln(n)$. For this we consider the game $\mathcal{G}_d$.

Similar to Godfrey's proof of Theorem 3.1 in [34], we can regard the selection of the clusters as a single-choice balls-into-bins game with $q := \frac{n}{d}$ bins and $m = k \cdot n$ balls. We show that at least one cluster receives strictly more than d balls if $k \geq 0.4$ which implies $\ell_{max} > 1$.

In the analysis we assume that k is a constant, but since the maximum load grows with the number of balls, it follows that the observation holds for any $k \geq 0.4$.

Let $r := k \cdot \frac{c \cdot \ln(q \cdot d)}{\ln(q)}$ and note that $r < 2 \cdot k = \mathcal{O}(1)$. The number of balls is

$$m = k \cdot n = k \cdot q \cdot d = k \cdot q \cdot c \cdot \ln(n) = k \cdot q \cdot c \cdot \ln(q \cdot d) = r \cdot q \cdot \ln(q)$$

Theorem 2.2.2 states that then, with probability $1 - o(1)$, the maximum load ℓ_{max} in any cluster is

$$\ell_{max} > (d_r - 1 + \alpha) \cdot \ln(q) \tag{4.3}$$

where α can be any constant in $(0, 1)$ and d_r is the unique solution of

$$f_r(x) := 1 + x \cdot (\ln(r) - \ln(x) + 1) - r = 0.$$

for which $d_r > r$. Let $0.99 < \alpha < 1$ and assume that $d_r > 1.4$. Then, due to $n \geq 1,000$, we obtain

$$\frac{c \cdot \ln(n)}{\ln(q)} + 1 - \alpha < \frac{c \cdot \ln(n)}{\ln(n) - \ln(c \cdot \ln(n))} + 0.01 < 1.4 < d_r$$

and finally

$$\ell_{max} > (d_r - 1 + \alpha) \cdot \ln(q) > \ln(n).$$

So, it only remains to show that $d_r > 1.4$ if $k \geq 0.4$. For the analysis we vary k, but fix n and c. Note that then r grows with k and $m = k \cdot n$. d_r should also grow with m because it governs the lower bound on ℓ_{max} in (4.3). This can be more formally shown by analysing the function $f_r(x)$ which, of course, can also be understood as a function of r:

$$g_x(r) := 1 + x \cdot (\ln(r) - \ln(x) + 1) - r.$$

Assuming $x > r > 0$, it follows from the derivatives

$$f'_r(x) = (\ln(r) - \ln(x) + 1) + x \cdot \left(-\frac{1}{x}\right) = \ln(r) - \ln(x) < 0$$

and

$$g'_x(r) = x \cdot \frac{1}{r} - 1 = \frac{x}{r} - 1 > 0$$

that, in order to fulfil $f_r(x) = 0$, an increase in r has to go with an increase in x.

Now we can finish the proof by showing that $r \geq c \cdot k \geq 0.32$ implies $d_r > 1.4$. For $r = 0.32$ this follows from

$$f_{0.32}(1.4) = 1 + 1.4 \cdot (\ln(0.32) - \ln(1.4) + 1) - 0.32 > 0.013 > 0$$

because only a greater x can reduce the function value. And as d_r grows with r, $d_r > 1.4$ will also hold for any $r > 0.32$. □

The next observation provides a matching lower bound for the number of balls.

Observation 4.3.6. *The result of Theorem 4.3.3 is asymptotically tight in terms of the number of balls m.*

Proof. Consider the variation of game $\mathcal{G}_d$ in which the first cluster B_1 has probability $\frac{\beta \cdot d}{n}$ to be chosen. (Since we cannot simply set the probabilities, we increase the probability by adding so many copies of the first cluster B_1 until the probability for choosing one of the copies is $\frac{\beta \cdot d}{n}$.) If $m > \frac{n}{\beta}$, then the expected number of balls allocated to the bins of B_1 will be larger than d. □

4.4 Analysis of the Model with Runs

In this section we consider the model in which a cluster is reused by the next ball with constant probability p (see Algorithm 3).

Theorem 4.4.1. *Let $k > 0$, $0 < p < 1$ and $c \geq 216 \cdot \frac{k+2}{1-p}$. Assume that the clusters are one-sided (averaged) β-balanced and that $m \leq \frac{1-p}{72 \cdot \beta} \cdot n$. After running Algorithm 3 for each ball, the maximum load will be 1 with probability at least $1 - n^{-k}$.*

Proof. In order to show that all balls are good, we pursue the same approach as in the proof of Theorem 4.3.3 and consider the variant of Algorithm 3 that uses tokens. That is, each ball $\ell \in [m]$ chooses a cluster of bins B_ℓ (according to the protocol) and places a *token* with label ℓ into *every* bin of B_ℓ. Fix any ball $i \in [m]$. In order to upper bound $\mathbf{Pr}\left[\text{ ball } i \text{ bad } \middle|\text{ balls } 1, \ldots, i-1 \text{ good }\right]$, we first upper bound the number of tokens that a bin $b \in B_i$ receives.

In each step j, Algorithm 3 decides by a (biased) coin toss whether to choose a fresh $B_j \in \mathcal{B}_j$ or to reuse B_{j-1}. In the following we denote by a *run* a maximal sequence of steps $k, k+1, \ldots, k'$ where the algorithm chooses a fresh $B_k \in \mathcal{B}_k$, and then uses B_k throughout steps $k, k+1, \ldots, k'$ but not $k'+1$. We assume now that we have two different types of token. If the algorithm chooses a new set (first step of a run) a *blue* token is used, otherwise (remaining steps of a run) a *red* token is used. Then, for all balls $j \in [i]$, we allocate the ball into an *i.u.r.* chosen bin among the least loaded bins containing a (red or blue) token labelled j.

Define $g := \frac{72}{1-p}$. Since $m \leq \frac{(1-p) \cdot n}{72 \cdot \beta} = \frac{n}{g \cdot \beta}$ and $c \geq \frac{216}{1-p} \cdot (k+2) = 3 \cdot g \cdot (k+2)$, we can apply Lemma 4.3.1 and get that, with probability $1 - n^{-(k+1)}$, no bin receives more than

$$t_{max} := \frac{2 \cdot c \cdot \ln(n)}{g} = \frac{(1-p) \cdot c \cdot \ln(n)}{36}$$

blue tokens.

It remains to bound the number of red tokens. Fix any bin b and assume that b receives $t \leq t_{max}$ many blue tokens. Since, for every blue token, we will bound the number of red tokens independently, a bound for the combined length of t_{max} runs is certainly also a bound for t runs. In the following we can therefore use t_{max} instead of t.

For $1 \leq r \leq t_{max}$, let Y_r denote the number of all (red *and* blue) tokens of the run that starts with the r-th blue token. Then the Y_r, $1 \leq r \leq t_{max}$, are geometrically distributed random variables with $\mathbf{Pr}[\, Y_r = z+1 \,] = p^z \cdot (1-p)$ and $\mathbb{E}[Y_r] = \frac{1}{1-p}$. Let $Y := Y_1 + \cdots + Y_{t_{max}}$ be the random variable that counts the red and blue tokens in bin b. Its expected value is $\mathbb{E}[Y] = \frac{t_{max}}{1-p}$. Applying Lemma 1.2.4 with $\delta = 1$, we obtain

$$\begin{aligned}\mathbf{Pr}[\, Y > (1+\delta)\cdot\mathbb{E}[Y]\,] &\leq \exp\left(-\frac{t_{max}}{2}\cdot\frac{\delta^2}{1+\delta}\right) = \exp\left(-\frac{t_{max}}{4}\right) \\ &= \exp\left(-\frac{(1-p)\cdot c\cdot\ln(n)}{4\cdot 36}\right) \leq n^{-\frac{(1-p)\cdot c}{144}} < n^{-(k+2)}.\end{aligned}$$

Hence, with a probability of at least $1 - n^{-(k+1)} - n \cdot n^{-(k+2)}$ no bin receives more than

$$2\cdot\mathbb{E}[Y] = \frac{2\cdot t_{max}}{1-p} = \frac{2\cdot c\cdot\ln(n)}{36} = \frac{c\cdot\ln(n)}{18}$$

(red or blue) tokens.

Define $h := 18$. Since $c \geq 216 \cdot \frac{k+2}{1-p} > 6\cdot h\cdot(k+2)$ and every bin in B_i has at most $\frac{c\cdot\ln(n)}{18} = \frac{c\cdot\ln(n)}{h}$ tokens, we can apply Lemma 4.3.2 and get that ball i is good with probability at least $1 - n^{-(k+2)}$. Since this holds for all balls, the probability that at least one of the m balls is bad is

$$m\cdot n^{-(k+2)} \leq \frac{n\cdot(1-p)}{72\cdot\beta}\cdot n^{-(k+2)} < n^{-(k+1)}.$$

The probability for the maximum load to exceed 1 is bounded by

$$n^{-(k+1)} + n^{-(k+1)} + n^{-(k+1)} < n^{-k}.$$

□

4.5 Conclusions

In this chapter a variation of the standard *balls-into-bins* games has been considered in which the bin choices are non-uniform and dependent. Each ball $i = 1, \ldots, m$, in turn, runs the following

protocol: (1) it *i.u.r.* chooses a cluster of bins $B_i \in \mathcal{B}_i$, and (2) it *i.u.r.* chooses one of the least loaded bins in B_i and allocates itself to it.

For this protocol we have shown that the *maximum load is one* with high probability provided that (i) the cluster size d is roughly $\Omega(\ln(n))$, (ii) the bin choices are averaged β-balanced (*i.e.*, every bin should expectedly show up in no more than $\frac{\beta \cdot m \cdot d}{n}$ many of the m many chosen clusters), and (iii) the number of balls m is upper-bounded by $\frac{n}{24 \cdot \beta}$.

Finally, we extended our results to a generalised model in which a cluster of bins is reused by the next ball with a certain probability p. Again, if (i) and (ii) hold and if the number of balls m is bounded by $\frac{(1-p) \cdot n}{72 \cdot \beta}$, the *maximum load is one* with high probability.

Since we restricted our analysis to $\ell_{max} = 1$ and $m \leq \frac{n}{\Theta(\beta)}$, it remains an open problem to show tight bounds on the maximum load for $m \gg \frac{n}{\beta}$.

5 Load Balancing in a Parallel Environment

In this chapter we investigate how GREEDY$[d]$ performs in a parallel environment. In our model the balls arrive in batches of size n and the loads of the bins are only updated between batches. We show that the gap between maximum and average load is $\mathcal{O}(\ln(n))$ and therefore independent of the number of balls thrown.

5.1 Introduction

The major disadvantage of the multiple-choice strategy is that it unfolds its potential only in sequential settings while many load balancing applications are parallel. To prove the bounds for the standard multiple-choice game in [3, 8], for example, it is assumed that the m balls are allocated one after the other and that the bin loads are immediately updated after each ball. Different strategies have been developed for parallel environments to deal with concurrent requests [2, 82, 61, 12, 1]. They base their decisions on the number of new requests, allow extra rounds of communication and in some cases let balls rechoose.

In this chapter we investigate how the bare GREEDY$[2]$ protocol performs in a parallel environment in which $m \geq n$ balls are allocated into n bins. Thus, concurrent requests to the same bin are answered with the same current load and no additional information like the number of new requests. We model this by updating the bins only after every n-th ball and prove that the gap between maximum and average load is still independent of the number of balls[1]. With high probability, the gap is $\mathcal{O}(\ln(n))$. We show this upper bound for $m \leq poly(n)$. The generalisation to arbitrary m is outstanding.

In a parallel environment the bare GREEDY$[2]$ protocol naturally performs worse than the best adapted strategies, but in some situations its simplicity and the avoidance of extra communication may compensate for the increase in the maximum load.

[1] In the sequential setting the gap is known to be $\Theta(\ln \ln(n))$ *w.h.p.* (Theorem 2.3.2).

5.1.1 Related Work

Balls-into-bins games are described in the introductory chapter. Results related to the multiple-choice paradigm are summarised in Chapter 2, parallel strategies in particular in Section 2.4.

Here we give a brief overview of publications that analyse the heavily-loaded case in which the number of balls m significantly exceeds the number of bins n. The first paper by Berenbrink *et al.* [8] explains and applies the main approach which forms the basis of the analyses in [8, 94, 85]. The approach is two-part in that the respective bounds on the maximum load are proven for a polynomial number of balls first before the result is extended to arbitrary m.

The first part is shown by induction on the number of balls or rather batches, where a batch consists of n balls. For every batch t, bounds on the distribution are proven based on the hypothesis that such bounds hold for all previous batches $\tau < t$. In each step of this *outer* induction, an *inner* induction is used to derive the bounds on the distribution. This inner induction is a *layered induction* which, though a bit more involved, follows the concept described in Section 1.2.3.

The second part applies a *short memory lemma* which states that two balls-into-bins processes that start with the same number of balls and with a balanced and unbalanced load vector, respectively, will be stochastically indistinguishable after a certain *recovery time*[2]. In the case of the standard game the recovery time is $\Delta \cdot poly(n)$, where Δ denotes the maximal difference between any two loads. The lemma is used to reduce the problem for general m to the case $m = poly(n)$.

Following this approach, Berenbrink *et al.* analyse the standard d-choice game for arbitrary $m > n$ and bound the gap between maximum and average load by $\mathcal{O}(\ln \ln(n))$. Wieder shows in [94] that this bound also holds for non-uniform probabilities if the imbalance in the probability distribution is compensated by a slight increase in the number of choices d. Talwar *et al.* apply a short memory lemma to prove that even in case of weighted balls the gap is independent of m, provided that the weight distribution fulfils "mild assumptions" [85].

5.1.2 Contributions

We consider a balls-into-bins game in which m balls are thrown into n bins. The process essentially follows GREEDY$[d]$, but we introduce explicit batches of size n and assume that all balls within one batch are allocated concurrently. We restrict our analysis to the case $d = 2$, and describe a somewhat surprising experimental observation regarding values $d > 2$ in Section 5.5: It appears that the larger d, the higher the maximum load.

[2] In [8, 94, 85] the recovery or *mixing time* follows from the fact that the underlying Markov process is *rapidly mixing*, that is, it rapidly converges to its stationary distribution – regardless of its initial state distribution (see definitions in [68, 66]). This is shown by a coupling argument.

It should be noted that [9] uses the concept of batches as well, but that there it simply is a technique used in the proofs, and that the actual process is unaware of the concept. Here, on the other hand, the allocation *itself* uses batches, and each batch is allocated concurrently; that is, load vectors are updated only every n many balls (an approach that would clearly reduce the problem to a simple single-choice game if we were to look at only one batch, starting from an empty system).

Similar to [9], we wish to show that after throwing m balls into n bins the deviation of the maximum load from the average load, $\frac{m}{n}$, is $\mathcal{O}(\ln(n))$. However, we prove this upper bound only for $m \leq poly(n)$. The generalisation to arbitrary m is still outstanding.

5.1.3 Motivation

Due to the nature of load balancing and the number of participants involved, it is natural to assume that most applications run in parallel environments. For this reason it is not surprising that the first generalisations of the d-choice game aimed at parallelising the algorithm. The strategies allow for additional communication between the balls and their chosen bins, where the messages contain acknowledgments, rejections or just information about the current load and the number of new requests. The balls base their decision on the returned messages and can also choose new bins in some protocols.

Even though the number of communication rounds and the adaptations to the algorithm can be kept quite low, we think it is interesting to see how the unchanged GREEDY$[d]$ protocol performs in a parallel setting. Naturally it cannot achieve as good a maximum load as the best parallel strategies, but it may be sufficient in some cases or even superior if extra communication is inadmissible.

5.2 Result and Outline

We analyse GREEDY$[d]$ in the case where the loads of the bins are only updated every n many balls. As usual, m balls are placed into n bins, and we assume that the bins are initially empty. The allocation at *time* t is given by the load vector directly after the t-th batch. The number of balls is polynomial bounded in the number of bins, that is, $m \leq n^{\delta}$ with δ being an arbitrary positive constant.

The main result is Theorem 5.2.1. It implies Corollary 5.2.2 which states that the gap between maximum and average load is independent of the number of balls m.

Theorem 5.2.1. *Let $\delta \geq 1$ be an arbitrary constant. Suppose we allocate $m \leq n^\delta$ balls (in $\frac{m}{n}$ batches) to n bins using* GREEDY[2]. *Then the number of bins with load at least $\frac{m}{n} + i + \gamma$ is upper bounded by $n \cdot \exp(-i)$,* w.v.h.p.*, where γ denotes a suitable constant.*

The theorem states that there exists a constant $\gamma \geq 0$ such that the number of bins with load at least $\frac{m}{n} + i + \gamma$ is at most $n \cdot \exp(-i)$ *w.v.h.p.* If $n < n_0$ for some constant n_0, then the theorem is immediately satisfied with $\gamma = n_0^\delta$. This trivial observation will be used in various places throughout the chapter, usually whenever certain inequalities hold only in the case $n \geq n_0$ (for some suitably chosen n_0).

We will prove the theorem by induction on t showing that in each step the process maintains various invariants with probability of at least $1-n^{-\kappa}$. Since there are no more than $n^{\delta-1}$ batches, with a probability of at least $1 - n^{-\kappa+\delta-1}$ the invariants hold throughout the process.

Corollary 5.2.2. *If $m \leq n^\delta$, the maximum load is $\frac{m}{n} + \mathcal{O}(\ln(n))$* w.v.h.p.

The analysis closely follows the paper *Balanced Allocations: The Heavily Loaded Case* by Berenbrink, Czumaj, Steger and Vöcking [9].

5.2.1 Definitions and Invariants

The average number of balls per bin at time t is $\frac{m}{n} = \frac{n \cdot t}{n} = t$. We call bins with fewer than t balls *underloaded* and bins with more than t balls *overloaded*. Frequently we will refer to *holes* in the distribution. For a given bin, the number of holes is defined to be the number of balls it is short of the average load at that point of time.

We will prove that *w.v.h.p.* the following invariants hold:

- $L(t)$: At time t, there are at most $0.7 \cdot n$ holes.
- $H(t)$: At time t, there are at most $0.47 \cdot n$ balls of height at least $t + 5$.

In the proof we will assume that $L(0), ..., L(t-1)$ and $H(0), ..., H(t-1)$ hold. (Contrary to [9], we do not need $L(t)$ to show $H(t)$.) We will analyse the underloaded and overloaded bins separately, the corresponding analyses communicating only through the two invariants from above. We will finally use invariant H to derive Theorem 5.2.1.

Throughout the analysis, we use the following notation:

Definition 5.2.3. *For $i, t \geq 0$, $\alpha_i^{(t)}$ denotes the fraction of bins with load at most $t - i$ at time t and $\beta_i^{(t)}$ the fraction of bins with load at least $t + i$.*

5.2.2 Outline

The underloaded bins are analysed in Section 5.3, the overloaded bins in Section 5.4. Since our analysis follows the outline of the proof of [9], we will refer to the corresponding results whenever we state a lemma that is similar to its counterpart in [9]. We sometimes have to use different parameters, and where that substantially changes the proof, the corresponding statements are reproven. In the cases where that is not necessary, the proofs are omitted.

We should point out that our approach to bounding the overloaded bins is quite different from that in [9]. In order to make that proof more accessible, several longish and mostly technical lemmas are placed in separate sections (5.4.3 and 5.4.4).

Finally, in Section 5.5 we present a simulation with a surprising outcome: As expected, the 2-choice algorithm performs better than the single-choice algorithm, but any further increase in d results in a higher maximum load.

5.3 Analysis of the Underloaded Bins

In this section we analyse the distribution of the holes. We prove the following two invariants for time $t \geq 0$. Let c_1 and c_2 denote suitable constants with $c_1 \leq c_2$.

- $L_1(t)$: For $1 \leq i \leq c_1 \cdot \ln(n)$, $\alpha_i^{(t)} \leq 1.6 \cdot 0.3^i$.
- $L_2(t)$: For $i \geq c_2 \cdot \ln(n)$, $\alpha_i^{(t)} = 0$.

Invariants $L_1(t)$ and $L_2(t)$ imply $L(t)$ as the number of holes at time t is at most

$$\sum_{i=1}^{\lfloor c_2 \cdot \ln(n) \rfloor} 1.6 \cdot 0.3^{\min(i, \lceil c_1 \cdot \ln(n) \rceil)} \cdot n \leq 0.7 \cdot n$$

for n large enough. We shall now prove that $L_1(t)$ and $L_2(t)$ hold *w.v.h.p.* if $d = 2$.

Lemma 5.3.1 (Lemma 2.1 in [9])**.** *Let ℓ be an arbitrary integer and assume that after batch $t-1$ there exist (at most) $a_\ell \cdot n$ bins with at most ℓ balls and (at most) $a_{\ell-1} \cdot n$ bins with at most $\ell - 1$ balls. Suppose that b is a bin with load exactly ℓ after batch $t - 1$. Then for each ball of batch t the probability to be placed into bin b by* GREEDY$[2]$ *is (at least)*

$$\frac{2 - a_\ell - a_{\ell-1}}{n}.$$

The proof is similar to the one of Lemma 2.1 in [9].

Observation 5.3.2 (Observation 2.2 in [9]). *The probability that a ball from batch t goes to any fixed bin with load at most $t-4$ is at least $\frac{1.9}{n}$, unless invariant $L_1(t-1)$ fails.*

Proof (copied and adapted from [9], page 1357). Applying invariant $L_1(t-1)$, there are at most $\alpha_i^{(t-1)} \cdot n \leq 1.6 \cdot 0.3^i \cdot n$ bins with load at most $(t-1)-i$ after batch $t-1$ for every $1 \leq i \leq c_1 \cdot \ln(n)$. Since the load information is only updated between the batches, this upper bound holds for all balls from batch t. Now, applying Lemma 5.3.1 yields that the probability that a ball from batch t is assigned to a bin with load at most $(t-1)-i$ is at least

$$\frac{2-\alpha_i^{(t-1)}-\alpha_{i+1}^{(t-1)}}{n} \geq \frac{2-1.6\cdot 0.3^i - 1.6\cdot 0.3^{i+1}}{n}$$

For $i \geq 3$, this probability is larger than $\frac{1.9}{n}$. □

This observation is used in the proof of the next lemma which upper bounds the size of the deepest holes.

Lemma 5.3.3 (Lemma 2.3 in [9]). *Let $t \geq 0$. Suppose the probability that one of the invariants $L_1(0), ..., L_1(t-1)$ fails is at most $n^{-\kappa'}$ for $\kappa' \geq 1$. For any fixed $\kappa > 0$, there exist constants c_0, c_1, c_2, c_3 (solely depending on κ) such that*

- *there are at most $n \cdot 1.6 \cdot 0.3^i$ bins containing at most $t-i$ balls, for $c_0 < i \leq c_1 \cdot \ln(n)$, and*
- *every bin contains at least $t - c_2 \cdot \ln(n)$ balls,*

with probability at least $1 - n^{-\kappa} - n^{-\kappa'}$, provided $n \geq c_3$.

We omit the proof because it is almost identical to the proof of Lemma 2.3 in [9]. The only detail changed is the bound for the number of bins with at most $t-i$ balls. It was $0.18 \cdot 3^{-i+2}$ and is now $1.6 \cdot 0.3^i$. Since the result of Observation 5.3.2 coincides with Observation 2.2 in [9], the major part of the proof is the same. Only in the last part, where we replace the old bound with the new bound, we need to adapt c_0 and set it to $c_0 = 390$ which, however, does not change the statement of the Lemma.

$L_2(t)$ is already proven by the second part of Lemma 5.3.3. The first part shows $L_1(t)$, but only for $i > c_0$. For the remaining cases $1 \leq i \leq c_0$, we will use the recursive formula from the next lemma to prove $\alpha_i^{(t)} \leq 1.6 \cdot 0.3^i$. The lemma is similar to Lemma 2.4 in [9], but statement and proof are easier because here the batches are not divided into sub-batches.

Lemma 5.3.4 (Lemma 2.4 in [9]). *Let $\epsilon > 0$ and let ℓ be a positive integer. Suppose for $i = 0, ..., 4$ there are at most $a_i \cdot n$ bins with load at most $\ell - i$ at time $t-1$. Then, at time t, the*

number of bins with load at most ℓ is at most $g(0) \cdot n$, w.v.h.p.*, where the function g is recursively defined by*

$$g(i) = \begin{cases} a_i & \text{if } i = 4 \\ (1+\epsilon) \cdot (g(i+1) + (a_i - g(i+1)) \cdot E) & \text{otherwise,} \end{cases}$$

where

$$E = \exp(-(2 - g(i+1) - a_i)).$$

We omit the proof because it is only a simplified version of the one in [9].

The recursive formula $g(i)$ enables us to prove $L_1(t)$ for every $i \in \{1, \ldots, c_0 - 1\}$. First assume $i \in \{2, ..., c_0 - 1\}$ and set $(a_0, ..., a_4) := (\alpha_{i-1}^{(t-1)}, ..., \alpha_{i+3}^{(t-1)})$ where $\alpha_{i-1}^{(t-1)}, ..., \alpha_{i+3}^{(t-1)}$ are the bounds of invariant $L_1(t-1)$. Choosing $\epsilon = 0.001$ and running a C program that implements $g(i)$ and gets $(a_0, ..., a_4)$ as input, we obtain

$$\alpha_i^{(t)} \leq g(0) \leq 1.6 \cdot 0.3^i$$

for all $i \in \{1, \ldots, c_0 - 1\}$. Thus, invariant $L(t)$ is shown for $i \geq 2$.

In the case $i = 1$ we cannot apply this approach because a_0 would correspond to $\alpha_0^{(t-1)}$ and would therefore not be covered by invariant $L_1(t-1)$. The next lemma provides an upper bound on $\alpha_0^{(t-1)}$ based on invariant $H(t-1)$.

Lemma 5.3.5 (Lemma 2.5 in [9])**.**
Suppose $H(t-1)$ is fulfilled. Let $(a_0, ..., a_4) := (\alpha_0^{(t-1)}, ..., \alpha_4^{(t-1)})$. Then

$$a_0 \leq 1 - \frac{a_1 + a_2 + a_3 + a_4 - 0.47}{4}.$$

Proof (copied and adapted from Lemma 2.5 in [9]). At any time $\tau \geq 0$, the number of holes at time τ is $A_\tau = \sum_{j \geq 1} \alpha_j^{(\tau)} \cdot n$. Since the number of balls above the average height is equal to the number of holes, we can conclude that A_τ also corresponds to the number of balls with height at least $\tau + 1$ at time τ. Now, suppose invariant $H(\tau)$ holds. Then, there are at most $B_\tau = 0.47 \cdot n$ balls of height at least $\tau + 5$ at time τ. Combining these two bounds, the number of balls with height $\tau + 1$, $\tau + 2$, $\tau + 3$ or $\tau + 4$ is lower-bounded by $A_\tau - B_\tau$. This implies that at least $(A_\tau - B_\tau)/4$ bins contain more than τ balls at time τ. As a consequence, the number of bins

containing at most τ balls is upper-bounded by $n - (A_\tau - B_\tau)/4$. Hence,

$$\alpha_0^{(\tau)} \cdot n \leq n - \frac{A_\tau - B_\tau}{4} \leq n \cdot \left(1 - \frac{\sum_{j=1}^{4} \alpha_j^{(\tau)} - 0.47}{4}\right)$$

Finally, setting $\tau = t - 1$ and $\alpha_j^{(\tau)} = a_j$ gives the lemma. □

Lemma 5.3.6. *If $L(0), ..., L(t-1)$ and $H(t-1)$ hold, then $L(t)$ holds* w.v.h.p.

Proof (copied and adapted from [9], page 1361). Since all other cases are already covered by the previous results in this section, it remains to show $L(t)$ for $i = 1$. The proof uses Lemma 5.3.4 and 5.3.5.

Again, using the C program, we verify $g(0) \leq 1.6 \cdot 0.3$ for all choices of $a_{i'} \in [0, 1.6 \cdot 0.3^{i'}]$, $1 \leq i' \leq 4$, and $a_0 \in [0, 1 - \frac{1}{4} \cdot (a_1 + a_2 + a_3 + a_4 - 0.47)]$. For this purpose we need to discretise the domains of the a_i's. For the discretisation, we use the monotonicity of $g(0)$: The term $g(0)$ is monotonically increasing in each of the terms $a_0, ..., a_4$. Therefore, it suffices to check the parameters $a_1, ..., a_4$ in discrete steps of a suitable size $\delta > 0$ while assuming

$$a_0 = 1 - \frac{a_1 + a_2 + a_3 + a_4 - 0.47 - 4 \cdot \delta}{4}.$$

Choosing $\epsilon = 0.001$ and $\delta = 0.002$, the C program confirms $g(0) \leq 1.6 \cdot 0.3$ in all cases. □

5.4 Analysis of the Overloaded Bins

In this section we prove invariant $H(t)$ which concerns the overloaded bins. It states that there are not more than $0.47 \cdot n$ balls with height at least $t + 5$. The proof is based on the induction hypothesis that the invariants $H(\tau)$ and $L(\tau)$ hold for all $\tau < t$, especially $L(t-1)$. Thus, we assume that there are at most $0.7 \cdot n$ holes at time $t - 1$ which implies that there at most $0.7 \cdot n$ balls above level $t - 1$.

Almost all lemmas in this section are new. Yet, the framework of the proof is again similar to [9]: We show two invariants $H_1(t)$ and $H_2(t)$ that imply $H(t)$ and also Theorem 5.2.1. In order to formulate these invariants, we first define two functions h and f:

Definition 5.4.1 (Functions h and f; [9], page 1362)**.** *Define*

$$h(i) := 67 \cdot 0.34^i.$$

Let ℓ denote the smallest integer i such that $h(i) \leq n^{-0.9}$ and let $\sigma \geq 1$ denote a suitable

constant (that will be specified later). For $i \geq 4$, define:

$$f(i) := \begin{cases} h(i) & \text{for } 4 \leq i < \ell \\ max\{h(i), \frac{1}{3} \cdot n^{-0.9}\} & \text{for } i = \ell \\ \sigma \cdot n^{-1} & \text{for } i = \ell + 1 \end{cases}$$

The next observation summarises a few properties of f.

Observation 5.4.2.

(1) $f(\ell) = h(\ell)$

(2) $f(i+1) = 0.34 \cdot f(i)$ *for* $4 \leq i \leq \ell - 1$

(3) $f(\ell + 1) \leq 0.34 \cdot f(\ell) = h(\ell + 1)$ *(for sufficiently large n)*

Proof. Property (1), that is $f(\ell) = max\{h(\ell), \frac{1}{3} \cdot n^{-0.9}\} = h(\ell)$, follows from

$$h(\ell) = 0.34 \cdot h(\ell - 1) > 0.34 \cdot n^{-0.9} > \frac{1}{3} \cdot n^{-0.9}.$$

Property (2): Due to (1) and the definition of f, $f(i) = h(i)$ as well as $f(i+1) = h(i+1)$ hold for all $i \in \{4, ..., \ell - 1\}$. Therefore,

$$f(i+1) = 67 \cdot 0.34^{i+1} = 67 \cdot 0.34^i \cdot 0.34 = 0.34 \cdot f(i)$$

Finally, (3) holds (for sufficiently large n) because

$$\frac{f(\ell)}{f(\ell+1)} \geq \frac{n^{-0.9}}{3 \cdot \sigma \cdot n^{-1}} = \frac{n^{0.1}}{3 \cdot \sigma} \geq 0.34^{-1}.$$

□

The invariants $H_1(t)$ and $H_2(t)$ are defined as follows:

- $H_1(t)$: $\beta_i^{(t)} \leq f(i)$ for $5 \leq i \leq \ell$,
- $H_2(t)$: $\sum_{i>\ell} \beta_i^{(t)} \leq \sigma \cdot n^{-1}$.

Invariant $H_1(t)$ implies that the number of balls decreases exponentially with each level, dropping below $n^{-0.9}$ on level $t + \ell$. Invariant $H_2(t)$ addresses the balls above level $t + \ell$ and claims that their number is bounded by a constant σ.

Observation 5.4.3. *$H_1(t)$ and $H_2(t)$ imply $H(t)$.*

Proof. The number of balls on level $t+5$ and higher is bounded by

$$\sum_{i=5}^{\ell+1} f(i) \cdot n \overset{(O.5.4.2)}{\leq} \sum_{i=5}^{\ell+1} h(i) \cdot n = n \cdot 67 \cdot 0.34^5 \cdot \sum_{i=0}^{\ell-4} 0.34^i$$
$$\overset{(L.1.1.3)}{<} n \cdot 67 \cdot 0.34^5 \cdot \frac{1}{1-0.34} < 0.47 \cdot n$$

for sufficiently large n. □

Observation 5.4.4. *If $L(t)$, $H_1(t)$ and $H_2(t)$ hold* w.v.h.p. *for all t, then Theorem 5.2.1 is fulfilled.*

Proof. First we show that the number of bins with load at least $\frac{m}{n} + i + 5$ is upper bounded by $n \cdot e^{-i}$: From Definition 5.4.1 and Observation 5.4.2 it follows that, for $i \geq 5$, the fraction β_i of balls on level i is upper-bounded by $h(i)$. Thus, it suffices to show that $e^{-k} \geq h(k+4)$ for $k \geq 1$:

$$1.08^k \geq 0.9 \;\Rightarrow\; e^{-k} \cdot 0.34^{-k} \geq 67 \cdot 0.34^4 \;\Leftrightarrow\; e^{-k} \geq h(k+4) = 67 \cdot 0.34^{k+4}$$

It remains to prove that this upper bound holds *w.v.h.p.* for all $t \leq \frac{n^\delta}{n} = n^{\delta-1}$. This follows directly from the statement that $L(t)$, $H_1(t)$ and $H_2(t)$ hold *w.v.h.p.* for all t. □

We will prove that $H_1(t)$ and $H_2(t)$ hold *w.v.h.p.* if $H_1(0), ..., H_1(t-1)$, $H_2(t-1)$ and $L(t-1)$ are fulfilled. The invariants for $t-1$ in particular provide us with properties of the load distribution which all balls of batch t base their decisions on. For convenience, the upper bounds derived from $L(t-1)$, $H_1(t-1)$ and $H_2(t-1)$ are bundled in the function $\hat{f}$ which is introduced in Definition 5.4.5. The subsequent Observation 5.4.6 proves that $\hat{f}(i)$ indeed upper-bounds the fraction of balls on level $t+i$, $0 \leq i \leq \ell$, before batch t is thrown.

Definition 5.4.5.

$$\hat{f}(i) = \begin{cases} 1 & \textit{for } i = 0 \\ \frac{0.7}{i+1} & \textit{for } 1 \leq i \leq 4 \\ f(i+1) & \textit{for } 5 \leq i \leq \ell \end{cases}$$

Observation 5.4.6. *Assume that the induction hypothesis holds. Then, at time $t-1$, the fraction of bins on level $t+i$, $0 \leq i \leq \ell$, is bounded by $\hat{f}(i)$.*

Proof. For $i = 0$, the fraction of bins that have a ball on level $t+0$ is trivially bounded by 1; *i.e.* all bins could have a ball there.

$L(t-1)$ implies that there are at most $0.7 \cdot n$ balls above level $t-1$ at time $t-1$. From this it follows that, for $i > 0$, there are at most $\frac{0.7}{i+1} \cdot n$ balls of height at least $t+i$. In the definition of $\hat{f}$ we use this bound for $i = 1, ..., 4$.

Finally, for $i = 5, ..., \ell$, we simply use invariant $H_1(t-1)$. □

$\hat{f}$ has the following properties:

Observation 5.4.7.

(1) $2 \cdot \hat{f}(i-k) \leq k$ *for* $k \in \{1, ..., i\}$ *and* $i \in \{2, ..., \ell\}$

(2) $\hat{f}(i) \leq h(i+1)$ *for all* $i \in \{0, ..., \ell\}$

Proof. (1) If $k \geq 2$, this statement is obviously true. Assume $k = 1$. If $i \geq 6$, then

$$2 \cdot \hat{f}(i-1) \leq 2 \cdot \hat{f}(6-1) = 2 \cdot f(6) = 2 \cdot 67 \cdot 0.34^6 < 0.21 < 1$$

and if $2 \leq i \leq 5$, then

$$2 \cdot \hat{f}(i-1) = 2 \cdot \frac{0.7}{i} \leq \frac{1.4}{2} = 0.7 < 1.$$

(2) We consider all cases:

$$\begin{aligned}
\hat{f}(0) &= 1 < 67 \cdot 0.34 \\
\hat{f}(i) &= \frac{0.7}{i+1} < 67 \cdot 0.34^{i+1} = h(i+1) \qquad \text{for } i \in \{1,2,3,4\} \\
\hat{f}(i) &= f(i+1) = h(i+1) \qquad \text{for } i \in \{5, ..., \ell-2\} \\
\hat{f}(\ell-1) &= f(\ell) \overset{(O.5.4.2)}{=} h(\ell) \\
\hat{f}(\ell) &= f(\ell+1) \leq \frac{\sigma}{n} < n^{-0.9} \cdot 0.34^2 \leq h(\ell+1) \qquad \text{for } n \text{ large enough}
\end{aligned}$$

□

Throughout the proofs of $H_1(t)$ and $H_2(t)$ the following notation will be used.

Definition 5.4.8 ($B_{i,k}$, $Z_{b,i,k}$)**.** *For all* $k \in \{1, ..., t+i\}$*, we denote the set of bins that have exactly* $t+i-k$ *balls at time* $t-1$ *with* $B_{i,k}$*.* $B_{i,0}$ *is defined as the set of all bins with at least* $t+i$ *balls.*

For all $k \in \{0, ..., t+i\}$*, we define* $Z_{b,i,k}$ *to be the random variable that is* 1 *if bin* $b \in B_{i,k}$ *receives at least* k *balls with batch* t *and* 0 *otherwise.*

5.4.1 Proof of $H_1(t)$

In order to prove $H_1(t)$ we have to show $\beta_i^{(t)} \leq f(i)$ for $i \in \{5, ..., \ell\}$. Fix any such i. Let Q denote the number of bins that contain at least $t+i$ balls at time $t-1$, and R the number of bins that reach level $t+i$ with batch t for the first time. Observe that $\beta_i^{(t)} \cdot n \leq Q + R$. Thus, it suffices to show $Q + R \leq f(i) \cdot n$.

Observation 5.4.9. *$Q \leq 0.34 \cdot f(i) \cdot n$ for $5 \leq i \leq \ell$.*

Proof. Applying invariant $H_1(t-1)$, we obtain

$$Q \leq \beta_{i+1}^{(t-1)} \cdot n \leq f(i+1) \cdot n \overset{(O.5.4.2)}{\leq} 0.34 \cdot f(i) \cdot n.$$

□

The more difficult task is to bound R. This will be done in Lemma 5.4.11. In its proof we will use the functions defined next.

Definition 5.4.10. *We define*

$$\phi(i,k) := \frac{\hat{f}(i-k)}{f(i)} \cdot \binom{n}{k} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n - 2 \cdot \hat{f}(i-k)}\right)^k \cdot \frac{e^{-2 \cdot \hat{f}(i-k)}}{1 - \frac{2 \cdot \hat{f}(i-k)}{k+1}}$$

and

$$\Phi(i) := \sum_{k=1}^{i} \phi(i,k).$$

Moreover, we define

$$\Phi_{k_1}^{k_2}(i) := \sum_{k=k_1}^{k_2} \phi(i,k)$$

so that we will be able to partition $\Phi(i)$.

The following lemma bounds R *w.v.h.p.* The lemmas needed to prove the bound will be provided in Section 5.4.3.

Lemma 5.4.11. *Let $i \in \{5, ..., \ell\}$ and let κ be any positive constant. Then $R < 0.66 \cdot f(i) \cdot n$ holds with probability $1 - n^{-\kappa}$, provided that n is sufficiently large.*

Proof. The number R of bins that have at least $t+i$ balls after batch t but not before can be written as the sum

$$R = \sum_{k=1}^{t+i} \sum_{b \in B_{i,k}} Z_{b,i,k}.$$

Due to the linearity of expectation we obtain

$$\begin{aligned}
\mathbb{E}[R] &= \sum_{k=1}^{t+i} \sum_{b \in B_{i,k}} \mathbb{E}[Z_{b,i,k}] = \sum_{k=1}^{t+i} \sum_{b \in B_{i,k}} \mathbf{Pr}\left[\, Z_{b,i,k} = 1 \,\right] \\
&\overset{(L.5.4.21)}{\leq} \sum_{k=1}^{i} n \cdot \hat{f}(i-k) \cdot \mathbf{Pr}\left[B\left(\frac{2}{n} \cdot \hat{f}(i-k), n\right) \geq k \right] \\
&= n \cdot f(i) \cdot \sum_{k=1}^{i} \frac{\hat{f}(i-k)}{f(i)} \cdot \mathbf{Pr}\left[B\left(\frac{2}{n} \cdot \hat{f}(i-k), n\right) \geq k \right] \\
&= n \cdot f(i) \cdot \sum_{k=1}^{i} \frac{\hat{f}(i-k)}{f(i)} \cdot \sum_{j=k}^{n} \binom{n}{j} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n}\right)^{j} \cdot \left(1 - \frac{2 \cdot \hat{f}(i-k)}{n}\right)^{n-j} \\
&\overset{(L.1.1.4)}{\leq} n \cdot f(i) \cdot \sum_{k=1}^{i} \frac{\hat{f}(i-k)}{f(i)} \cdot \sum_{j=k}^{n} \binom{n}{j} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n - 2 \cdot \hat{f}(i-k)}\right)^{j} \cdot e^{-2 \cdot \hat{f}(i-k)} \\
&\overset{(L.5.4.17)}{<} n \cdot f(i) \cdot \sum_{k=1}^{i} \frac{\hat{f}(i-k)}{f(i)} \cdot \binom{n}{k} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n - 2 \cdot \hat{f}(i-k)}\right)^{k} \cdot \frac{e^{-2 \cdot \hat{f}(i-k)}}{1 - \frac{2 \cdot \hat{f}(i-k)}{k+1}} \\
&\overset{(D.5.4.10)}{=} n \cdot f(i) \cdot \Phi(i) \\
&\overset{(L.5.4.18)}{\leq} n \cdot f(i) \cdot \max\{\Phi(5), \Phi(6)\} \\
&\overset{(D.5.4.10)}{=} n \cdot f(i) \cdot \max\{\Phi_1^4(5) + \Phi_5^5(5), \Phi_1^1(6) + \Phi_2^5(6) + \Phi_6^6(6)\} \\
&\overset{(L.5.4.19)}{<} n \cdot f(i) \cdot \max\{0.1583 + 0.1779, 0.1897 + 0.0592 + 0.1628\} \\
&= n \cdot f(i) \cdot \max\{0.3362, 0.4117\} = n \cdot f(i) \cdot 0.4117
\end{aligned}$$

Since the $Z_{b,i,k}$ are negatively correlated, Chernoff bounds (Lemma 1.2.3) can be applied. For every $\epsilon \in (0, 1]$,

$$\begin{aligned}
\mathbf{Pr}\left[\, R \geq (1+\epsilon) \cdot 0.4117 \cdot f(i) \cdot n \,\right] &\leq e^{-0.4117 \cdot f(i) \cdot n \cdot \epsilon^2/3} \leq e^{-0.4117 \cdot f(\ell) \cdot n \cdot \epsilon^2/3} \\
&\leq e^{-0.4117 \cdot n^{-0.9} \cdot n \cdot \epsilon^2/9} < e^{-0.0457 \cdot n^{0.1} \cdot \epsilon^2} \\
&\leq n^{-\kappa}
\end{aligned}$$

where the last inequality holds for any given κ and $\epsilon > 0$, provided that n is sufficiently large. We choose $\epsilon = 0.6$ and get $\mathbf{Pr}\left[\, R \geq 0.66 \cdot f(i) \cdot n \,\right] \leq n^{-\kappa}$. □

Corollary 5.4.12. *Invariant $H_1(t)$ holds* w.v.h.p.

Proof. From Observation 5.4.9 and Lemma 5.4.11 it follows for $i \in \{5, ..., \ell\}$ that

$$\beta_i^{(t)} \cdot n \leq Q + R \leq (0.34 + 0.66) \cdot f(i) \cdot n = f(i) \cdot n$$

with the probability given in Lemma 5.4.11. □

5.4.2 Proof of $H_2(t)$

In order to bound the number of balls above level $t + \ell$, we will show that, *w.v.h.p.*, no ball of batch t will have a height above this threshold. However, should this nevertheless happen, then it is likely that the according bins will not receive many such balls.

Definition 5.4.13. *Define the random variables*

- X_t *which is one if at least one ball of batch* t *is allocated to a level greater than* $t + \ell$*, and zero otherwise;*
- Y_t *which counts the number of balls from batch* t *that have height greater than* $t + \ell$*;*
- $X_{t,\ell,\lambda}$ *(for any positive integer* λ*) which is one if at least one bin with at most* $t + \ell - \lambda$ *balls at time* $t - 1$ *reaches level* $t + \ell + 1$ *or higher with the* t*-th batch;*
- $Y_{t,\ell,\lambda}$ *which counts the number of balls from batch* t *that fall into a bin with at least* $t+\ell-\lambda$ *balls at time* $t - 1$*.*

In the next two lemmas we will bound the probabilities for the events $Y_{t,\ell,\lambda} \geq \gamma$ (for constant γ), $X_t = 1$ and $X_{t,\ell,\lambda} = 1$. They will be used to prove $H_2(t)$ in Lemma 5.4.16.

Lemma 5.4.14. *Let* $\lambda \geq 1$ *be an integer constant (*$\lambda \ll \ell$*) and let* $Y_{t,\ell,\lambda}$ *be the number of balls that fall into bins with load at least* $t + \ell - \lambda$ *at time* $t - 1$*. Then for any constant* γ*:*

$$\mathbf{Pr}\left[\, Y_{t,\ell,\lambda} \geq \gamma \,\right] \leq n^{-0.7\cdot\gamma}$$

Proof. The probability that any ball of batch t falls into a bin of load at least $t+\ell-\lambda$ is bounded by $\hat{f}(\ell - \lambda)^2$. The probability that at least γ balls of the batch fall into such bins is therefore bounded by

$$\begin{aligned}\mathbf{Pr}\left[\, Y_{t,\ell,\lambda} \geq \gamma \,\right] &\overset{(L.1.1.2)}{\leq} \left(\frac{e \cdot n \cdot f(\ell+1-\lambda)^2}{\gamma}\right)^{\gamma} \overset{(D.5.4.1)}{\leq} \left(\frac{e \cdot n}{\gamma} \cdot \left(\frac{n^{-0.9}}{0.34^{\lambda-1}}\right)^2\right)^{\gamma} \\ &= \left(\frac{e \cdot n^{-0.8}}{\gamma \cdot 0.34^{2\cdot(\lambda-1)}}\right)^{\gamma} = \left(\frac{e}{\gamma \cdot 0.34^{2\cdot(\lambda-1)}}\right)^{\gamma} \cdot n^{-0.8\cdot\gamma} < n^{-0.7\cdot\gamma}.\end{aligned}$$

□

Lemma 5.4.15.

$$\mathbf{Pr}\,[\,X_t = 1\,] \leq n^{-0.9}$$

For $\lambda \geq 2$,

$$\mathbf{Pr}\,[\,X_{t,\ell,\lambda} = 1\,] \leq n^{-0.8\cdot\lambda+1}.$$

Proof. The probability for $X_t = 1$, that any ball of batch t will have height at least $t + \ell + 1$, is bounded by the sum of the bins' probabilities to receive a ball on this level.

$$\mathbf{Pr}\,[\,X_t = 1\,] \leq \sum_{k=0}^{t+\ell+1} \sum_{b\in B_{\ell+1,k}} \mathbf{Pr}\,[\,Z_{b,\ell+1,k} = 1\,]$$

(Recall that $B_{\ell+1,0}$ contains all bins on level $t + \ell + 1$ and higher at time $t - 1$.)

The probability for $X_{t,\ell,\lambda} = 1$, that a bin from level $\ell - \lambda$ or lower receives enough balls with batch t to reach level $\ell + 1$, is bounded by the sum of the probabilities for the particular bins to reach this level.

$$\mathbf{Pr}\,[\,X_{t,\ell,\lambda} = 1\,] \leq \sum_{k=\lambda}^{t+\ell+1} \sum_{b\in B_{\ell+1,k}} \mathbf{Pr}\,[\,Z_{b,\ell+1,k} = 1\,]$$

In both cases, applying Lemma 5.4.21 yields the results. □

Lemma 5.4.16. *Invariant H_2 holds* w.v.h.p. *over all batches.*

Proof (copied and adapted from [9], page 1364). Lemma 5.4.15 states that

$$\mathbf{Pr}\,[\,X_t = 1\,] \leq n^{-0.9}.$$

Let $\lambda \geq 2$ be an integer constant. Lemma 5.4.15 states that the probability that any bins from level $\ell - \lambda$ or lower reach level $\ell + 1$ is at most $n^{-0.8\cdot\lambda+1}$, and Lemma 5.4.14 states that the probability that γ balls fall into bins above level $\ell - \lambda$ is at most $n^{-0.7\cdot\gamma}$. It follows that

$$\mathbf{Pr}\,[\,Y_t \geq \gamma\,] \leq n^{-0.8\cdot\lambda+1} + n^{-0.7\cdot\gamma}$$

and therefore $Y_t = \mathcal{O}(1)$, *w.v.h.p.*

Thus, we can assume that there exists a suitable constant j so that $Y_t \leq j$. A violation of $H_2(t)$ implies that the bins with load at least $t+\ell+1$ contain more than σ balls of height at least $t + \ell + 1$. Observe that these balls must have been placed during the last σ rounds or one of the invariants $H_2(1), ..., H_2(t-1)$ is violated. That is, if $H_2(1), ..., H_2(t-1)$ hold, then a violation

of $H_2(t)$ implies that $j \cdot \sum_{\tau=t-\sigma}^{t} X_\tau \geq \sigma$. The probability for this event is at most

$$\mathbf{Pr}\left[\sum_{\tau=t-\sigma}^{t} X_\tau \geq \frac{\sigma}{j}\right] \leq \binom{\sigma}{\sigma/j} \cdot \left(\frac{1}{n^{0.9}}\right)^{\sigma/j} \leq \left(\frac{e \cdot j}{n^{0.9}}\right)^{\sigma/j} \leq n^{-\tilde{\kappa}}$$

for any constant $\tilde{\kappa}$, provided that n is sufficiently large. Consequently, invariant H_2 holds, *w.v.h.p.*, over all batches. □

This completes the proof of Theorem 5.2.1. It follows from Observation 5.4.4 because $L(t)$, $H_1(t)$ and $H_2(t)$ hold *w.v.h.p.* (Lemma 5.3.6, Corollary 5.4.12 and Lemma 5.4.16).

5.4.3 Lemmas Used for Proving $H_1(t)$

Lemma 5.4.17. *Let $i \in \{5, ..., \ell\}$ and $k \in \{1, ..., i\}$. Then:*

$$\sum_{j=k}^{n} \binom{n}{j} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n - 2 \cdot \hat{f}(i-k)}\right)^j < \binom{n}{k} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n - 2 \cdot \hat{f}(i-k)}\right)^k \cdot \frac{1}{1 - \frac{2 \cdot \hat{f}(i-k)}{k+1}}$$

Proof. Consider the quotient of two successive summands:

$$\begin{aligned}
\frac{\binom{n}{j+1} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n-2 \cdot \hat{f}(i-k)}\right)^{j+1}}{\binom{n}{j} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n-2 \cdot \hat{f}(i-k)}\right)^{j}} &= \frac{(n-j) \cdot 2 \cdot \hat{f}(i-k)}{(j+1) \cdot (n - 2 \cdot \hat{f}(i-k))} \\
&\leq \frac{(n-k) \cdot 2 \cdot \hat{f}(i-k)}{(k+1) \cdot (n - 2 \cdot \hat{f}(i-k))} \\
&\overset{(O.5.4.7)}{\leq} \frac{2 \cdot \hat{f}(i-k)}{k+1} \overset{(O.5.4.7)}{<} 1
\end{aligned}$$

From this we can derive the statement:

$$\begin{aligned}
& \sum_{j=k}^{n} \binom{n}{j} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n - 2 \cdot \hat{f}(i-k)}\right)^j \\
\leq\ & \binom{n}{k} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n - 2 \cdot \hat{f}(i-k)}\right)^k \cdot \sum_{\mu=0}^{n-k} \left(\frac{2 \cdot \hat{f}(i-k)}{k+1}\right)^\mu \\
<\ & \binom{n}{k} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n - 2 \cdot \hat{f}(i-k)}\right)^k \cdot \sum_{\mu=0}^{\infty} \left(\frac{2 \cdot \hat{f}(i-k)}{k+1}\right)^\mu \\
\overset{(L.1.1.3)}{=}\ & \binom{n}{k} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n - 2 \cdot \hat{f}(i-k)}\right)^k \cdot \frac{1}{1 - \frac{2 \cdot \hat{f}(i-k)}{k+1}}
\end{aligned}$$

□

Lemma 5.4.18. *Let $6 \leq i \leq \ell$. The terms*

$$\begin{aligned}
\Phi_1^{i-5}(i) &= \sum_{k=1}^{i-5} \frac{\hat{f}(i-k)}{f(i)} \cdot \binom{n}{k} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n-2 \cdot \hat{f}(i-k)}\right)^k \cdot \frac{e^{-2 \cdot \hat{f}(i-k)}}{1-\frac{2 \cdot \hat{f}(i-k)}{k+1}}, \\
\Phi_{i-4}^{i-1}(i) &= \sum_{k=i-4}^{i-1} \frac{\hat{f}(i-k)}{f(i)} \cdot \binom{n}{k} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n-2 \cdot \hat{f}(i-k)}\right)^k \cdot \frac{e^{-2 \cdot \hat{f}(i-k)}}{1-\frac{2 \cdot \hat{f}(i-k)}{k+1}} \\
\Phi_i^i(i) &= \frac{\hat{f}(i-i)}{f(i)} \cdot \binom{n}{i} \cdot \left(\frac{2 \cdot \hat{f}(i-i)}{n-2 \cdot \hat{f}(i-i)}\right)^i \cdot \frac{e^{-2 \cdot \hat{f}(i-i)}}{1-\frac{2 \cdot \hat{f}(i-i)}{i+1}} \\
&= \frac{1}{f(i)} \cdot \binom{n}{i} \cdot \left(\frac{2}{n-2}\right)^i \cdot \frac{e^{-2}}{1-\frac{2}{i+1}}
\end{aligned}$$

are maximal for $i = 6$.

Proof. In all cases we will show that $\Phi_{k_1}^{k_2}(i+1) \leq \Phi_{k_1}^{k_2}(i)$ so that $\Phi_{k_1}^{k_2}(6)$ must be indeed the maximum. We will often use that k and i are bounded: $k_1 \leq k \leq k_2$, $6 \leq i \leq \ell$. Additionally, we will make use of:

(1) $\frac{0.34^{-k}}{k+1} = \frac{2.9411...^k}{k+1}$ grows with k.

(2) If $k \in \{i-4, ..., i-1\}$ and $i \geq 6$, then $(k+1) \cdot (i-k+1) = i+1+k \cdot (i-k)$ is minimal for $k = i-1$.

Case $\Phi_1^{i-5}(i)$: Note that $k \leq i-5$ implies $i-k \geq 5$ and therefore

$$\hat{f}(i-k) = f(i-k+1) = 67 \cdot 0.34^{i-k+1}.$$

We use

$$\begin{aligned}
\frac{\phi(i+1,k+1)}{\phi(i,k)} &= \frac{\frac{\hat{f}(i+1-(k+1))}{f(i+1)} \cdot \binom{n}{k+1} \cdot \left(\frac{2 \cdot \hat{f}(i+1-(k+1))}{n-2 \cdot \hat{f}(i+1-(k+1))}\right)^{k+1} \cdot \frac{e^{-2 \cdot \hat{f}(i+1-(k+1))}}{1-\frac{2 \cdot \hat{f}(i+1-(k+1))}{k+1+1}}}{\frac{\hat{f}(i-k)}{f(i)} \cdot \binom{n}{k} \cdot \left(\frac{2 \cdot \hat{f}(i-k)}{n-2 \cdot \hat{f}(i-k)}\right)^k \cdot \frac{e^{-2 \cdot \hat{f}(i-k)}}{1-\frac{2 \cdot \hat{f}(i-k)}{k+1}}} \\
&= \frac{f(i)}{f(i+1)} \cdot \frac{n-k}{k+1} \cdot \frac{2 \cdot \hat{f}(i-k)}{n-2 \cdot \hat{f}(i-k)} \cdot \frac{1-\frac{2 \cdot \hat{f}(i-k)}{k+1}}{1-\frac{2 \cdot \hat{f}(i-k)}{k+2}} \\
&\overset{(O.5.4.7)}{\leq} \frac{1}{0.34} \cdot \frac{1}{k+1} \cdot \frac{2 \cdot \hat{f}(i-k)}{1} \cdot \frac{1-\frac{2 \cdot \hat{f}(i-k)}{k+1}}{1-\frac{2 \cdot \hat{f}(i-k)}{k+2}} \\
&< \frac{2 \cdot 67 \cdot 0.34^{i-k+1}}{0.34 \cdot (k+1)} \overset{(1)}{\leq} \frac{2 \cdot 67 \cdot 0.34^{i-(i-5)+1}}{0.34 \cdot (i-5+1)} \\
&= \frac{2 \cdot 67 \cdot 0.34^5}{i-4} \leq \frac{2 \cdot 67 \cdot 0.34^5}{6-4} = 67 \cdot 0.34^5 < 0.31
\end{aligned}$$

and

$$\begin{aligned}
\frac{\phi(i+1,1)}{\phi(i,1)} &= \frac{\frac{\hat{f}(i+1-1)}{f(i+1)} \cdot \binom{n}{1} \cdot \left(\frac{2\cdot\hat{f}(i+1-1)}{n-2\cdot\hat{f}(i+1-1)}\right)^1 \cdot \frac{e^{-2\cdot\hat{f}(i+1-1)}}{1-\frac{2\cdot\hat{f}(i+1-1)}{1+1}}}{\frac{\hat{f}(i-1)}{f(i)} \cdot \binom{n}{1} \cdot \left(\frac{2\cdot\hat{f}(i-1)}{n-2\cdot\hat{f}(i-1)}\right)^1 \cdot \frac{e^{-2\cdot\hat{f}(i-1)}}{1-\frac{2\cdot\hat{f}(i-1)}{1+1}}} \\
&= \frac{2\cdot\hat{f}(i)}{2\cdot\hat{f}(i-1)} \cdot \frac{n-2\cdot\hat{f}(i-1)}{n-2\cdot\hat{f}(i)} \cdot \frac{e^{-2\cdot\hat{f}(i)}}{e^{-2\cdot\hat{f}(i-1)}} \cdot \frac{1-\frac{2\cdot\hat{f}(i-1)}{2}}{1-\frac{2\cdot\hat{f}(i)}{2}} \\
&< \frac{\hat{f}(i)}{\hat{f}(i-1)} \cdot e^{2\cdot\hat{f}(i-1)-2\cdot\hat{f}(i)} \cdot \frac{1-\hat{f}(i-1)}{1-\hat{f}(i)} \\
&< 0.34 \cdot e^{134\cdot 0.34^i - 134\cdot 0.34^{i+1}} = 0.34 \cdot e^{134\cdot 0.34^i\cdot(1-0.34)} \\
&\leq 0.34 \cdot e^{134\cdot 0.34^6\cdot 0.66} < 0.39
\end{aligned}$$

to obtain:

$$\begin{aligned}
\Phi_1^{i+1-5}(i+1) &= \sum_{k=1}^{i+1-5} \phi(i+1,k) = \sum_{k=0}^{i-5} \phi(i+1,k+1) \\
&= \phi(i+1,1) + \sum_{k=1}^{i-5} \phi(i+1,k+1) \\
&< 0.39\cdot\phi(i,1) + \sum_{k=1}^{i-5} 0.31\cdot\phi(i,k) < \sum_{k=1}^{i-5} \phi(i,k) = \Phi_1^{i-5}(i)
\end{aligned}$$

Case $\Phi_{i-4}^{i-1}(i)$: Note that since $k \in \{i-4,...,i-1\}$, we have $1 \leq i-k \leq 4$ and therefore $\hat{f}(i-k) = \frac{0.7}{i-k+1}$.

We can reuse the first lines from the previous case to prove a useful inequality:

$$\begin{aligned}
\frac{\phi(i+1,k+1)}{\phi(i,k)} &\leq \frac{1}{0.34}\cdot\frac{1}{k+1}\cdot\frac{2\cdot\hat{f}(i-k)}{1} < \frac{2\cdot 0.7}{0.34\cdot(k+1)\cdot(i-k+1)} \\
&\overset{(2)}{\leq} \frac{2\cdot 0.7}{0.34\cdot(i-1+1)\cdot(i-(i-1)+1)} = \frac{1.4}{0.34\cdot i\cdot 2} \\
&\leq \frac{1.4}{0.34\cdot 6\cdot 2} < 0.35
\end{aligned}$$

We obtain:

$$\begin{aligned}
\Phi_{i+1-4}^{i+1-1}(i+1) &= \sum_{k=i+1-4}^{i+1-1} \phi(i+1,k) = \sum_{k=i-4}^{i-1} \phi(i+1,k+1) \\
&< \sum_{k=i-4}^{i-1} 0.35\cdot\phi(i,k) < \sum_{k=i-4}^{i-1} \phi(i,k) = \Phi_{i-4}^{i-1}(i)
\end{aligned}$$

Case $\Phi_i^i(i)$: From

$$\begin{aligned}\frac{\Phi_{i+1}^{i+1}(i+1)}{\Phi_i^i(i)} &= \frac{\frac{1}{f(i+1)} \cdot \binom{n}{i+1} \cdot \left(\frac{2}{n-2}\right)^{i+1} \cdot \frac{e^{-2}}{1-\frac{2}{i+1+1}}}{\frac{1}{f(i)} \cdot \binom{n}{i} \cdot \left(\frac{2}{n-2}\right)^{i} \cdot \frac{e^{-2}}{1-\frac{2}{i+1}}} \\ &= \frac{f(i)}{f(i+1)} \cdot \frac{n-i}{i+1} \cdot \frac{2}{n-2} \cdot \frac{1-\frac{2}{i+1}}{1-\frac{2}{i+2}} \\ &< \frac{1}{0.34} \cdot \frac{2}{i+1} \cdot \frac{i+1-2}{i+2-2} \cdot \frac{i+2}{i+1} \leq \frac{1}{0.34} \cdot \frac{2}{6+1} \cdot \frac{6-1}{6} \cdot \frac{6+2}{6+1} \\ &= \frac{1}{0.34} \cdot \frac{80}{294} < 0.81\end{aligned}$$

follows that $\Phi_{i+1}^{i+1}(i+1) < 0.81 \cdot \Phi_i^i(i)$. □

Lemma 5.4.19. *If $n \geq 29$, then:*

$$\begin{aligned}\Phi_1^4(5) &< 0.1583 \\ \Phi_5^5(5) &< 0.1779 \\ \Phi_1^1(6) &< 0.1897 \\ \Phi_2^5(6) &< 0.0592 \\ \Phi_6^6(6) &< 0.1628\end{aligned}$$

Proof. Define δ_k to be 0.01 if $k = 1$ and $\delta_k = 0$ otherwise. Then for $n \geq 29$,

(1) $\frac{n \cdot (n-1) \cdot \ldots \cdot (n-k+1)}{(n-2 \cdot \hat{f}(5-k))^k} < 1 + \delta_k$ for $k \in \{1, 2, 3, 4\}$

because, for $k = 1$,

$$n < n + 0.29 - 1.01 \cdot 0.28 = n + 0.01 \cdot 29 - 1.01 \cdot \frac{1.4}{4+1} \leq (1 + 0.01) \cdot (n - 2 \cdot \hat{f}(5-1))$$

and, for $k = 2$,

$$n \cdot (n-1) = n^2 - n < n^2 - \frac{2.8}{4} \cdot n + \frac{1.96}{16} = \left(n - \frac{1.4}{4}\right)^2 = \left(n - \frac{2 \cdot 0.7}{5-2+1}\right)^2.$$

For $k = 3, 4$, it holds because the additional factors are $\frac{n-k+1}{n-2 \cdot \hat{f}(5-k)} < 1$.

With similar arguments one can prove

(2) $\frac{n \cdot (n-1) \cdot \ldots \cdot (n-k+1)}{(n-2 \cdot \hat{f}(6-k))^k} < 1$ for $k \in \{2, 3, 4, 5\}$.

For $n \geq 29$,

(3) $\frac{n}{n-2 \cdot f(6)} < 1.01$

because $n < n + 0.29 - 0.21 < n + 0.1 \cdot 29 - 1.01 \cdot 2 \cdot 67 \cdot 0.34^6 \leq 1.01 \cdot (n - 2 \cdot f(6))$.

$$\begin{aligned}
\Phi_1^4(5) &= \sum_{k=1}^{4} \frac{\hat{f}(5-k)}{f(5)} \cdot \binom{n}{k} \cdot \left(\frac{2 \cdot \hat{f}(5-k)}{n - 2 \cdot \hat{f}(5-k)} \right)^k \cdot \frac{e^{-2 \cdot \hat{f}(5-k)}}{1 - \frac{2 \cdot \hat{f}(5-k)}{k+1}} \\
&= \sum_{k=1}^{4} \frac{\hat{f}(5-k)}{f(5)} \cdot \frac{n \cdot (n-1) \cdot \ldots \cdot (n-k+1)}{k!} \cdot \frac{(2 \cdot \hat{f}(5-k))^k}{(n - 2 \cdot \hat{f}(5-k))^k} \cdot \frac{e^{-2 \cdot \hat{f}(5-k)}}{1 - \frac{2 \cdot \hat{f}(5-k)}{k+1}} \\
&\overset{(1)}{<} \sum_{k=1}^{4} \frac{\hat{f}(5-k)}{f(5)} \cdot (1 + \delta_k) \cdot \frac{(2 \cdot \hat{f}(5-k))^k}{k!} \cdot \frac{e^{-2 \cdot \hat{f}(5-k)}}{1 - \frac{2 \cdot \hat{f}(5-k)}{k+1}}
\end{aligned}$$

After applying $f(5) = 67 \cdot 0.34^5$ and $\hat{f}(5-k) = \frac{0.7}{5-k+1}$ for $k \in \{1, 2, 3, 4\}$ and after computing the sum with a computer program, we get:

$$\Phi_1^4(5) < 0.1583$$

$$\begin{aligned}
\Phi_2^5(6) &= \sum_{k=2}^{5} \frac{\hat{f}(6-k)}{f(6)} \cdot \binom{n}{k} \cdot \left(\frac{2 \cdot \hat{f}(6-k)}{n - 2 \cdot \hat{f}(6-k)} \right)^k \cdot \frac{e^{-2 \cdot \hat{f}(6-k)}}{1 - \frac{2 \cdot \hat{f}(6-k)}{k+1}} \\
&= \sum_{k=2}^{5} \frac{\hat{f}(6-k)}{f(6)} \cdot \frac{n \cdot (n-1) \cdot \ldots \cdot (n-k+1)}{k!} \cdot \frac{(2 \cdot \hat{f}(6-k))^k}{(n - 2 \cdot \hat{f}(6-k))^k} \cdot \frac{e^{-2 \cdot \hat{f}(6-k)}}{1 - \frac{2 \cdot \hat{f}(6-k)}{k+1}} \\
&\overset{(2)}{<} \sum_{k=2}^{5} \frac{\hat{f}(6-k)}{f(6)} \cdot \frac{(2 \cdot \hat{f}(6-k))^k}{k!} \cdot \frac{e^{-2 \cdot \hat{f}(6-k)}}{1 - \frac{2 \cdot \hat{f}(6-k)}{k+1}}
\end{aligned}$$

Again, we apply $f(6) = 67 \cdot 0.34^6$ and $\hat{f}(6-k) = \frac{0.7}{6-k+1}$ for $k \in \{2, 3, 4, 5\}$ and compute the sum with a computer program. The result is:

$$\Phi_2^5(6) < 0.0592$$

$$\begin{aligned}
\Phi_5^5(5) &= \frac{\hat{f}(5-5)}{f(5)} \cdot \binom{n}{5} \cdot \left(\frac{2 \cdot \hat{f}(5-5)}{n - 2 \cdot \hat{f}(5-5)} \right)^5 \cdot \frac{e^{-2 \cdot \hat{f}(5-5)}}{1 - \frac{2 \cdot \hat{f}(5-5)}{5+1}} \\
&= \frac{1}{f(5)} \cdot \binom{n}{5} \cdot \left(\frac{2}{n-2} \right)^5 \cdot \frac{e^{-2}}{1 - \frac{2}{6}} < \frac{1}{67 \cdot 0.34^5} \cdot \frac{2^5}{5!} \cdot \frac{3 \cdot e^{-2}}{2} \\
&= \frac{32 \cdot 3 \cdot e^{-2}}{67 \cdot 0.34^5 \cdot 120 \cdot 2} < 0.1779
\end{aligned}$$

$$\begin{aligned}\Phi_6^6(6) &= \frac{\hat{f}(6-6)}{f(6)} \cdot \binom{n}{6} \cdot \left(\frac{2 \cdot \hat{f}(6-6)}{n-2 \cdot \hat{f}(6-6)}\right)^6 \cdot \frac{e^{-2 \cdot \hat{f}(6-6)}}{1-\frac{2 \cdot \hat{f}(6-6)}{6+1}} \\ &= \frac{1}{f(6)} \cdot \binom{n}{6} \cdot \left(\frac{2}{n-2}\right)^6 \cdot \frac{e^{-2}}{1-\frac{2}{7}} < \frac{1}{67 \cdot 0.34^6} \cdot \frac{2^6}{6!} \cdot \frac{7 \cdot e^{-2}}{5} \\ &= \frac{64 \cdot 7 \cdot e^{-2}}{67 \cdot 0.34^6 \cdot 720 \cdot 5} < 0.1628\end{aligned}$$

$$\begin{aligned}\Phi_1^1(6) &= \frac{\hat{f}(6-1)}{f(6)} \cdot \binom{n}{1} \cdot \left(\frac{2 \cdot \hat{f}(6-1)}{n-2 \cdot \hat{f}(6-1)}\right)^1 \cdot \frac{e^{-2 \cdot \hat{f}(6-1)}}{1-\frac{2 \cdot \hat{f}(6-1)}{1+1}} \\ &= \frac{f(6)}{f(6)} \cdot n \cdot \frac{2 \cdot f(6)}{n-2 \cdot f(6)} \cdot \frac{e^{-2 \cdot f(6)}}{1-f(6)} \overset{(3)}{<} 1.01 \cdot 2 \cdot f(6) \cdot \frac{e^{-2 \cdot f(6)}}{1-f(6)} \\ &= 1.01 \cdot 2 \cdot 67 \cdot 0.34^6 \cdot \frac{e^{-2 \cdot 67 \cdot 0.34^6}}{1-67 \cdot 0.34^6} < 0.1897\end{aligned}$$

□

5.4.4 Lemmas Used for Proving $H_2(t)$

Lemma 5.4.20. *Assume $\ell \geq 5$. Fix an integer constant $\lambda \geq 2$. For all $k \in \{\lambda, ..., \ell+1\}$ it holds that*

$$\left(\frac{6 \cdot \hat{f}(\ell+1-k)}{k}\right)^k \leq n^{-0.8 \cdot \lambda}$$

for sufficiently large n.

Proof. Define $c(\ell) := 6 \cdot 67 \cdot 0.34^{\ell+2}$ and recall that $\hat{f}(i) \leq h(i+1) = 67 \cdot 0.34^{i+1}$ for $i \in 0, ..., \ell-2$ (see Definition 5.4.5 and Observation 5.4.7). Instead of the original function we consider

$$w(k) := \left(c(\ell) \cdot \frac{0.34^{-k}}{k}\right)^k = \left(\frac{6 \cdot 67 \cdot 0.34^{\ell+2-k}}{k}\right)^k \geq \left(\frac{6 \cdot \hat{f}(\ell+1-k)}{k}\right)^k$$

and show $w(k) \leq n^{-0.8 \cdot \lambda}$. For this we regard $w(k)$ as a continuous function on the interval $(0, \infty) \subset \mathbb{R}$ and perform a curve sketching. In order to identify the extrema we derive $w(k)$ and

obtain:

$$\begin{aligned} w'(k) &= \left(\frac{c(\ell)^k}{k^k \cdot 0.34^{k^2}}\right)' \\ &= \frac{c(\ell)^k \cdot \ln(c(\ell)) \cdot k^k \cdot 0.34^{k^2} - c(\ell)^k \cdot k^k \cdot 0.34^{k^2} \cdot (\ln(k) + 1 + 2 \cdot k \cdot \ln(0.34))}{(k^k \cdot 0.34^{k^2})^2} \\ &= \frac{c(\ell)^k \cdot \ln(c(\ell)) - c(\ell)^k \cdot (\ln(k) + 1 + 2 \cdot k \cdot \ln(0.34))}{k^k \cdot 0.34^{k^2}} \\ &= \left(\frac{c(\ell)}{k \cdot 0.34^k}\right)^k \cdot (\ln(c(\ell)) - \ln(k) - 1 - 2 \cdot k \cdot \ln(0.34)) \end{aligned}$$

$w'(k) = 0$ is fulfilled if and only if $g(k) := \ln(c(\ell)) - \ln(k) - 1 - 2 \cdot k \cdot \ln(0.34) = 0$. Since $\ln(c(\ell)) - 1 - 2 \cdot k \cdot \ln(0.34)$ is linear in k and $\ln(k)$ is a strictly concave function, there are at most two solutions. Let $\epsilon > 0$, $\epsilon \to 0$ and $\psi \to \infty$, then:

$$\begin{aligned} g(\epsilon) &= \ln(c(\ell)) - \ln(\epsilon) - 1 - 2 \cdot \epsilon \cdot \ln(0.34) \ \to \ \infty \\ g(1) &= \ln(6 \cdot 67 \cdot 0.34^{\ell+2}) - \ln(1) - 1 - 2 \cdot 1 \cdot \ln(0.34) \\ &= \ln(6 \cdot 67) + (\ell + 2) \cdot \ln(0.34) - 1 - 2 \cdot \ln(0.34) \\ &< 7.16 - 1.07 \cdot (\ell + 2) \ \leq \ 7.16 - 1.07 \cdot (5 + 2) \ < \ 0 \\ g(\psi) &= \ln(c(\ell)) - \ln(\psi) - 1 - 2 \cdot \psi \cdot \ln(0.34) \\ &> \ln(c(\ell)) - 1 - \ln(\psi) + 2.14 \cdot \psi \ \to \ \infty \end{aligned}$$

Since $g(k)$ is continuous on $(0, \infty)$, $g(k)$ and, thus, $w'(k)$ must have a root in the interval $(0, 1]$. Because of the gradients at ϵ and 1, $w(k)$ must have a maximum there. The second extremum is a minimum in the interval $(1, \infty)$. It may or may not be in the given domain $[\lambda, \ell + 1]$. In any case, restricted to $[\lambda, \ell + 1]$, $w(k)$ is maximal at one of the boundaries. And since $\ell = \Theta(\ln(n))$ (which can be derived from Definition 5.4.1), it follows that, for all integers $k \in \{\lambda, ..., \ell + 1\}$, some constant c and sufficiently large n,

$$\begin{aligned} w(k) &\leq \max\{w(\lambda), w(\ell + 1)\} \ \leq \ \max\left\{\left(\frac{6 \cdot h(\ell + 2 - \lambda)}{\lambda}\right)^\lambda, \left(\frac{6 \cdot h(1)}{\ell + 1}\right)^{\ell+1}\right\} \\ &\leq \max\left\{\left(\frac{6 \cdot n^{-0.9}}{\lambda \cdot 0.34^{\lambda-2}}\right)^\lambda, \left(\frac{6 \cdot 67 \cdot 0.34}{c \cdot \ln(n)}\right)^{c \cdot \ln(n)}\right\} \\ &\leq \max\left\{\left(\frac{6}{\lambda \cdot 0.34^{\lambda-2}}\right)^\lambda \cdot n^{-0.9 \cdot \lambda}, 165^{0.83 \cdot \ln(n)} \cdot n^{-0.83 \cdot \ln \ln(n)}\right\} \ < \ n^{-0.8 \cdot \lambda} \end{aligned}$$

□

Lemma 5.4.21. *Let $i \in \{5, ..., \ell + 1\}$, then*

$$\sum_{k=1}^{t+i} \sum_{b \in B_{i,k}} \mathbf{Pr}\left[\, Z_{b,i,k} = 1 \,\right] \leq \sum_{k=1}^{i} \hat{f}(i-k) \cdot n \cdot \mathbf{Pr}\left[B\left(\frac{2}{n} \cdot \hat{f}(i-k), n\right) \geq k \right].$$

For $\lambda \geq 2$ and sufficiently large n,

$$\sum_{k=\lambda}^{t+\ell+1} \sum_{b \in B_{\ell+1,k}} \mathbf{Pr}\left[\, Z_{b,\ell+1,k} = 1 \,\right] \leq n^{-0.8 \cdot \lambda + 1}.$$

The probability that any bin receives a ball with height at least $\ell + 1$ is bounded by

$$\sum_{k=0}^{t+\ell+1} \sum_{b \in B_{\ell+1,k}} \mathbf{Pr}\left[\, Z_{b,\ell+1,k} = 1 \,\right] \leq n^{-0.9}.$$

Proof. First we bound the probability for a bin to get the next ball. Let $b \in B_{i,k}$, $0 < k < i$ (where the $B_{i,k}$ are defined in Definition 5.4.8). The probability for b to be chosen is $\frac{1}{n}$. The probability that any bin on the same level or higher is chosen is bounded by $\frac{\hat{f}(i-k) \cdot n}{n} = \hat{f}(i-k)$ (see Observation 5.4.6). Since every ball has two choices, the probability $p_{i,k}$ for b to get the ball is

$$p_{i,k} \leq \frac{1}{n} \cdot \hat{f}(i-k) + \hat{f}(i-k) \cdot \frac{1}{n} = \frac{2}{n} \cdot \hat{f}(i-k).$$

For all $b \in B_{i,k}$, $k \geq i$, the probability $p_{i,k}$ to get the ball is bounded by

$$p_{i,k} \leq \frac{2}{n} = \frac{2}{n} \cdot \hat{f}(0).$$

For any bin $b \in B_{i,0}$, let $p_{i,0}$ be the maximal probability for b to be chosen. Then

$$p_{i,0} \leq p_{i,1}.$$

Now, for each bin $b \in B_{i,k}$, $0 \leq k \leq t + i$, we can upper-bound the probability for the event $Z_{b,i,k} = 1$ (that b receives at least one ball from batch t with height at least $t + i$):

$$\mathbf{Pr}\left[\, Z_{b,i,k} = 1 \,\right] \leq \begin{cases} \mathbf{Pr}\left[\, B(n, p_{i,1}) \geq 1 \,\right] \leq \mathbf{Pr}\left[B\left(n, \frac{2}{n} \cdot \hat{f}(i-1)\right) \geq 1 \right] & \text{for } k = 0 \\ \mathbf{Pr}\left[\, B(n, p_{i,k}) \geq k \,\right] \leq \mathbf{Pr}\left[B\left(n, \frac{2}{n} \cdot \hat{f}(i-k)\right) \geq k \right] & \text{for } 1 \leq k \leq i \\ \mathbf{Pr}\left[\, B(n, p_{i,k}) \geq k \,\right] \leq \mathbf{Pr}\left[B\left(n, \frac{2}{n} \cdot \hat{f}(i-i)\right) \geq i \right] & \text{for } k > i \end{cases}$$

Define $\tilde{B}_{i,1} = B_{i,1} \cup B_{i,0}$. Then

$$\sum_{k=0}^{1} \sum_{b \in B_{i,k}} \mathbf{Pr}\left[\, Z_{b,i,k} = 1 \,\right] \leq |\tilde{B}_{i,1}| \cdot \mathbf{Pr}\left[B\left(n, \frac{2}{n} \cdot \hat{f}(i-1)\right) \geq 1 \right].$$

Define $\tilde{B}_{i,i} = \cup_{k=i}^{t+i} B_{i,k}$. Then

$$\sum_{k=i}^{t+i} \sum_{b \in B_{i,k}} \mathbf{Pr}\left[\, Z_{b,i,k} = 1 \,\right] \leq |\tilde{B}_{i,i}| \cdot \mathbf{Pr}\left[B\left(n, \frac{2}{n} \cdot \hat{f}(i-i)\right) \geq i \right].$$

For $1 < k < i$, define $\tilde{B}_{i,k} = B_{i,k}$ and observe that $\sum_{k=1}^{i} |\tilde{B}_{i,k}| = n$ and that, for $1 \leq k \leq i$, $|\tilde{B}_{i,k}| \leq \hat{f}(i-k) \cdot n$ (see Definition 5.4.5 and Observation 5.4.6).

The first inequality stated in the lemma follows from:

$$\begin{aligned}
\sum_{k=1}^{t+i} \sum_{b \in B_{i,k}} \mathbf{Pr}\left[\, Z_{b,i,k} = 1 \,\right] &\leq \sum_{k=0}^{t+i} \sum_{b \in B_{i,k}} \mathbf{Pr}\left[\, Z_{b,i,k} = 1 \,\right] \\
&\leq \sum_{k=1}^{i} |\tilde{B}_{i,k}| \cdot \mathbf{Pr}\left[B\left(n, \frac{2}{n} \cdot \hat{f}(i-k)\right) \geq k \right] \\
&\leq \sum_{k=1}^{i} \hat{f}(i-k) \cdot n \cdot \mathbf{Pr}\left[B\left(n, \frac{2}{n} \cdot \hat{f}(i-k)\right) \geq k \right]
\end{aligned}$$

For the proofs of the second and the third statement in the lemma, we can reuse the analysis of the first one. Here, i is set to $\ell + 1$.

$$\begin{aligned}
&\sum_{k=\lambda}^{t+\ell+1} \sum_{b \in B_{\ell+1,k}} \mathbf{Pr}\left[\, Z_{b,\ell+1,k} = 1 \,\right] \leq \sum_{k=\lambda}^{\ell+1} |\tilde{B}_{\ell+1,k}| \cdot \mathbf{Pr}\left[B\left(n, \frac{2}{n} \cdot \hat{f}(\ell+1-k)\right) \geq k \right] \\
&= \sum_{k=\lambda}^{\ell+1} |\tilde{B}_{\ell+1,k}| \cdot \sum_{j=k}^{n} \binom{n}{j} \cdot \left(\frac{2}{n} \cdot \hat{f}(\ell+1-k)\right)^{j} \cdot \left(1 - \frac{2}{n} \cdot \hat{f}(\ell+1-k)\right)^{n-j} \\
&\leq \sum_{k=\lambda}^{\ell+1} |\tilde{B}_{\ell+1,k}| \cdot \binom{n}{k} \cdot \left(\frac{2}{n} \cdot \hat{f}(\ell+1-k)\right)^{k} \\
&\leq \sum_{k=\lambda}^{\ell+1} |\tilde{B}_{\ell+1,k}| \cdot \left(\frac{e \cdot n}{k}\right)^{k} \cdot \left(\frac{2}{n} \cdot \hat{f}(\ell+1-k)\right)^{k} \\
&\leq \sum_{k=\lambda}^{\ell+1} |\tilde{B}_{\ell+1,k}| \cdot \left(\frac{6 \cdot \hat{f}(\ell+1-k)}{k}\right)^{k} \overset{(L.5.4.20)}{\leq} \sum_{k=\lambda}^{\ell+1} |\tilde{B}_{\ell+1,k}| \cdot n^{-0.8 \cdot \lambda} \leq n^{-0.8 \cdot \lambda + 1}
\end{aligned}$$

In particular, for $\lambda = 3$,

$$\sum_{k=3}^{t+\ell+1} \sum_{b \in B_{\ell+1,k}} \mathbf{Pr}\left[\, Z_{b,\ell+1,k} = 1 \,\right] \leq n^{-1.4}.$$

This can be used to show the last inequality:

$$\begin{aligned}
&\sum_{k=0}^{t+\ell+1} \sum_{b \in B_{\ell+1,k}} \mathbf{Pr}\left[\, Z_{b,\ell+1,k} = 1 \,\right] \leq \sum_{k=1}^{\ell+1} |\tilde{B}_{\ell+1,k}| \cdot \mathbf{Pr}\left[B\left(n, \frac{2}{n} \cdot \hat{f}(\ell+1-k)\right) \geq k \right] \\
&\leq \quad n^{-1.4} + \sum_{k=1}^{2} |\tilde{B}_{\ell+1,k}| \cdot \mathbf{Pr}\left[B\left(n, \frac{2}{n} \cdot \hat{f}(\ell+1-k)\right) \geq k \right] \\
&\leq \quad n^{-1.4} + \sum_{k=1}^{2} \hat{f}(\ell+1-k) \cdot n \cdot \left(\frac{6 \cdot \hat{f}(\ell+1-k)}{k}\right)^k \\
&\leq \quad n^{-1.4} + \sum_{k=1}^{2} f(\ell+2-k) \cdot n \cdot \left(\frac{6 \cdot f(\ell+2-k)}{k}\right)^k \\
&\leq \quad n^{-1.4} + f(\ell+1) \cdot n \cdot 6 \cdot f(\ell+1) + f(\ell) \cdot n \cdot \left(\frac{6 \cdot f(\ell)}{2}\right)^2 \\
&\leq \quad n^{-1.4} + 6 \cdot \left(\frac{\sigma}{n}\right)^2 \cdot n + 9 \cdot \left(n^{-0.9}\right)^3 \cdot n \leq n^{-1.4} + 6 \cdot \sigma^2 \cdot n^{-1} + 9 \cdot n^{-1.7} \leq n^{-0.9}
\end{aligned}$$

□

5.5 Larger Values of d

Considering the results for d-choice balls-into-bins games one would assume the batched protocol to be monotone in the same sense that non-batched protocols are: The more choices we have per ball, the better the final load distribution. Surprisingly this does not seem to be the case. The experiment charted in Figure 5.1 shows how the gap between maximum and average load as a function of d. The rapid decrease from $d = 1$ to $d = 2$ is followed by a seemingly logarithmic increase.

The experiments fix $n = 1,000$ bins and allocate $1,000$ batches with $1,000$ balls each (a total of $1,000,000$ balls). Each data point is actually the average of five individual experiments. Notice that the y-axis is logarithmically scaled. The tall peak on the left corresponds to $d = 1$, the single-choice process.

One explanation of this observed phenomenon may be that bins that fall behind will have a relatively high probability to receive many balls from the next batch. For example, let $d = \Theta(n)$. All balls of the first batch find the bins empty, and the allocation is stochastically equivalent to the single-choice game. This implies that expectedly about $\frac{n}{e}$ bins remain empty. With the next batch almost all balls will have at least one of these bins among their d choices and allocate to one of them. The balls of the third batch will then mainly commit to the remaining bins of load 0 and bins of load 1. And so on.

For $n = d = 25$ and $m = 20 \cdot n$, the process is traced in Table 5.1. One can clearly see that

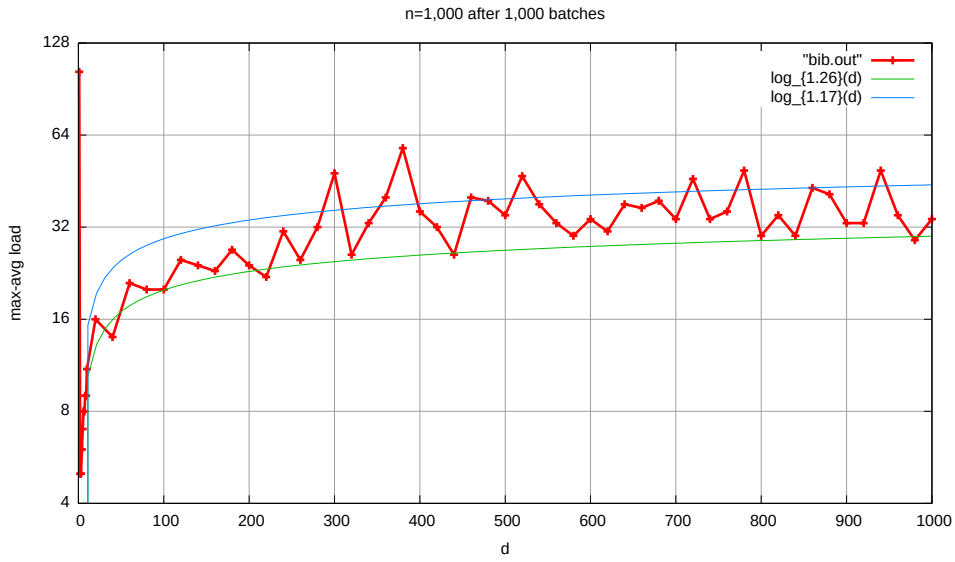

Figure 5.1: Gap between maximum and average load as a function of d.

the least loaded bins often receive so many balls that they top all other bins. The same effect is observable for smaller d, but since a bin's probability to be among the choices of a ball is smaller, the increase is not as extreme.

5.6 Conclusions

In this chapter we have analysed GREEDY$[2]$ in a dynamic parallel environment and showed that the gap between maximum and average load stays $\mathcal{O}(\ln(n))$ *w.v.h.p.* if the batch size equals the number of bins n and if the number of balls m is polynomially bounded in n. Our simulations suggest that this gap grows with the number of choices d. This is surprising because in other models applying GREEDY$[d]$ in the heavily-loaded case [8, 94, 85] the maximum load decreases with d.

The main open problem is the generalisation to arbitrary m. Considering the analyses in [8, 94, 85], proving and applying a *short memory lemma* seems to be a promising approach. Furthermore, it would be interesting to analyse the game for $d > 2$.

Batch	Bins																								
1	0	0	0	1	0	1	1	1	1	0	2	2	2	1	2	1	0	1	5	0	0	2	0	2	0
2	3	2	4	1	2	1	1	1	1	4	2	2	2	1	2	1	3	1	5	1	3	2	0	2	3
3	3	2	4	4	2	3	2	2	3	4	2	2	2	2	2	1	3	1	5	1	3	2	15	2	3
4	3	2	4	4	2	3	2	2	3	4	2	2	2	2	2	5	3	6	5	16	3	3	15	2	3
5	3	7	4	4	3	3	2	3	3	4	4	3	6	4	8	5	3	6	5	16	3	3	15	5	3
6	5	7	4	4	4	3	20	4	4	4	4	3	6	4	8	5	3	6	5	16	4	3	15	5	4
7	5	7	4	4	4	11	20	4	4	4	4	9	6	4	8	5	11	6	5	16	4	6	15	5	4
8	5	7	4	8	6	11	20	4	8	9	5	9	6	6	8	5	11	6	5	16	7	6	15	5	8
9	5	7	13	8	6	11	20	16	8	9	5	9	6	6	8	6	11	6	6	16	7	6	15	7	8
10	18	7	13	8	6	11	20	16	8	9	14	9	7	7	8	6	11	6	6	16	7	7	15	7	8
11	18	7	13	8	15	11	20	16	8	9	14	9	7	7	8	7	11	13	14	16	7	7	15	7	8
12	18	10	13	8	15	11	20	16	8	9	14	9	10	10	8	12	11	13	14	16	9	12	15	11	8
13	18	10	13	16	15	11	20	16	12	9	14	11	10	10	15	12	11	13	14	16	9	12	15	11	12
14	18	12	13	16	15	11	20	16	12	16	14	11	11	11	15	12	12	13	14	16	22	12	15	11	12
15	18	12	13	16	15	17	20	16	12	16	14	14	16	15	15	12	12	13	14	16	22	12	15	18	12
16	18	17	13	16	15	17	20	16	15	16	14	14	16	15	15	15	16	13	14	16	22	17	15	18	17
17	18	17	26	16	15	17	20	16	15	16	14	16	16	15	15	15	16	22	15	16	22	17	15	18	17
18	18	17	26	16	18	17	20	16	17	16	29	16	16	17	15	15	16	22	16	16	22	17	17	18	17
19	18	17	26	16	18	17	20	16	17	17	29	16	17	17	25	28	16	22	16	16	22	17	17	18	17
20	18	17	26	21	18	17	20	21	17	17	29	25	17	17	25	28	16	22	18	20	22	17	17	18	17

Table 5.1: Load distribution. $n = 25$ bins, $d = 25$ choices, 20 batches.

Bibliography

[1] Micah Adler, Petra Berenbrink, and Klaus Schröder. Analyzing an infinite parallel job allocation process. In *Proceedings of the 6th Annual European Symposium on Algorithms*, ESA '98, pages 417–428, London, UK, 1998. Springer-Verlag.

[2] Micah Adler, Soumen Chakrabarti, Michael Mitzenmacher, and Lars Rasmussen. Parallel randomized load balancing. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, STOC '95, pages 238–247, New York, NY, USA, 1995. ACM.

[3] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. In *Proceedings of the 26th ACM Symposium on Theory of Computing (STOC)*, pages 593–602, 1994.

[4] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM J. Comput.*, 29:180–200, September 1999.

[5] C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Dev.*, 17:525–532, November 1973.

[6] Petra Berenbrink, André Brinkmann, Tom Friedetzky, and Lars Nagel. Balls into bins with related random choices. In *Proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures*, SPAA '10, pages 100–105, New York, NY, USA, 2010. ACM.

[7] Petra Berenbrink, André Brinkmann, Tom Friedetzky, and Lars Nagel. Balls into non-uniform bins. In *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium*, IPDPS '10, pages 1–10. IEEE, 2010.

[8] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: the heavily loaded case. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, STOC '00, pages 745–754, New York, NY, USA, 2000. ACM.

[9] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: The heavily loaded case. *SIAM J. Comput.*, 35:1350–1385, June 2006.

[10] Petra Berenbrink, Tom Friedetzky, Zengjian Hu, and Russell Martin. On weighted balls-into-bins games. In Volker Diekert and Bruno Durand, editors, *Proceedings of the 22nd International Symposium on Theoretical Aspects of Computer Science (STACS 2005)*, volume 3404 of *Lecture Notes in Computer Science*, pages 231–243. Springer Berlin / Heidelberg, 2005.

[11] Petra Berenbrink, Tom Friedetzky, Zengjian Hu, and Russell Martin. On weighted balls-into-bins games. *Theor. Comput. Sci.*, 409:511–520, December 2008.

[12] Petra Berenbrink, Friedhelm Meyer auf der Heide, and Klaus Schröder. Allocating weighted jobs in parallel. In *Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, SPAA '97, pages 302–310, New York, NY, USA, 1997. ACM.

[13] Petra Berenbrink, Friedhelm Meyer auf der Heide, and Klaus Schröder. Allocating weighted jobs in parallel. *Theory Comput. Syst.*, 32(3):281–300, 1999.

[14] M. Bienkowski, M. Korzeniowski, and F. Meyer auf der Heide. Dynamic load balancing in distributed hash tables. In *Proceedings of the 4th International workshop on Peer-To-Peer Systems (IPTPS)*, pages 217–225, 2005.

[15] B. Bollobas. *Random Graphs*. Cambridge University Press, 2 edition, 2001.

[16] Tony Bourke. *Server load balancing*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[17] J. Byers, J. Considine, and M. Mitzenmacher. Simple load balancing for distributed hash tables. In *Proceedings of the 2nd International workshop on Peer-To-Peer Systems (IPTPS)*, pages 80–87, February 2003.

[18] John W. Byers, Jeffrey Considine, and Michael Mitzenmacher. Geometric generalizations of the power of two choices. In *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, SPAA '04, pages 54–63, New York, NY, USA, 2004. ACM.

[19] Richard Cole, Alan M. Frieze, Bruce M. Maggs, Michael Mitzenmacher, Andréa W. Richa, Ramesh K. Sitaraman, and Eli Upfal. On balls and bins with deletions. In *Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science*, RANDOM '98, pages 145–158, London, UK, October 1998. Springer-Verlag.

[20] 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 *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 378–388, New York, NY, USA, 1998. ACM.

[21] Artur Czumaj. Recovery time of dynamic allocation processes. In *Proceedings of the tenth annual ACM symposium on Parallel algorithms and architectures*, SPAA '98, pages 202–211, New York, NY, USA, 1998. ACM.

[22] Artur Czumaj, Friedhelm Meyer auf der Heide, and Volker Stemann. Contention resolution in hashing based shared memory simulations. *SIAM Journal on Computing*, 29:1703–1739, March 2000.

[23] Artur Czumaj and Volker Stemann. Randomized allocation processes. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 194–203, Washington, DC, USA, 1997. IEEE Computer Society Press.

[24] A. de Moivre. *Doctrine of Chances*. Woodfall, London, United Kingdom, 1718.

[25] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics)*. Springer, 3 edition, 2005.

[26] Martin Dietzfelbinger and Friedhelm Meyer auf der Heide. Simple, efficient shared memory simulations. In *Proceedings of the fifth annual ACM symposium on Parallel algorithms and architectures*, SPAA '93, pages 110–119, New York, NY, USA, 1993. ACM.

[27] Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[28] Derek L. Eager, Edward D. Lazowska, and John Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Trans. Softw. Eng.*, 12:662–675, May 1986.

[29] Guy Even and Moti Medina. Revisiting randomized parallel load balancing algorithms. In *Structural Information and Communication Complexity, 16th International Colloquium*, pages 209–221, May 2009.

[30] Guy Even and Moti Medina. Parallel randomized load balancing: A lower bound for a more general model. In Jan van Leeuwen, Anca Muscholl, David Peleg, Jaroslav Pokorn, and Bernhard Rumpe, editors, *SOFSEM 2010: Theory and Practice of Computer Science*, volume 5901 of *Lecture Notes in Computer Science*, pages 358–369. Springer Berlin / Heidelberg, January 2010.

[31] Eyal Even-Dar, Alex Kesselman, and Yishay Mansour. Convergence time to nash equilibria. In *Proceedings of the 30th international conference on Automata, languages and programming*, ICALP'03, pages 502–513, Berlin, Heidelberg, 2003. Springer-Verlag.

[32] Philippe Flajolet, Peter J. Grabner, Peter Kirschenhofer, and Helmut Prodinger. On ramanujan's q-function. *Journal of Computational and Applied Mathematics*, 58:103–116, March 1995.

[33] Hans Freudenthal. Models in applied probability. *Synthese*, 12:202–212, 1960.

[34] B. P. Godfrey. Balls and bins with structure: Balanced allocations on hypergraphs. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '08, pages 511–517, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[35] Gaston H. Gonnet. Expected length of the longest probe sequence in hash code searching. *J. ACM*, 28:289–304, April 1981.

[36] Edda Happ. *Analyses of Evolutionary Algorithms*. Phd thesis, University of Saarbrücken, 2009.

[37] Klaus Heubeck. Urnenmodelle und ihre anwendung in der versicherungsmathematik. *Blätter der DGVFM*, 11:371–429, 1974.

[38] C. I. Huygens. *Oeuvres Completes de Christian Huygens*, volume 14, chapter Van Rekeningh in Spelen van Geluk (1665). Nijhoff, The Hague, Netherlands, 1920.

[39] Jacek Izydorczyk and Michal Izydorczyk. Microprocessor scaling: What limits will hold? *Computer*, 43:20–26, August 2010.

[40] N. Johnson and S. Kotz. *Urn Models and Their Application*. John Wiley and Sons, 1977.

[41] Michio Kaku. *Physics of the Future : How Science Will Shape Human Destiny and Our Daily Lives by the Year 2100*. Doubleday, New York, USA, 2011.

[42] T. Kamae, U. Krengel, and G. L. O'Brien. Stochastic inequalities on partially ordered spaces. *The Annals of Probability*, 5:899–912, December 1977.

[43] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pages 654–663, New York, NY, USA, 1997. ACM.

[44] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[45] Richard M. Karp, Michael Luby, and Friedhelm Meyer auf der Heide. Efficient pram simulation on a distributed memory machine. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, STOC '92, pages 318–326, New York, NY, USA, 1992. ACM.

[46] Ananth Rao Karthik, Ananth Rao, Karthik Lakshminarayanan, Sonesh Surana, Richard Karp, and Ion Stoica. Load balancing in structured p2p systems. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, Feb. 2003.

[47] Krishnaram Kenthapadi and Rina Panigrahy. Balanced allocation on graphs. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA '06, pages 434–443, New York, NY, USA, 2006. ACM.

[48] D. E. Knuth. *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.

[49] V. F. Kolchin, B. A. Sevsatyanov, and V. P. Chistyakov. *Random Allocation*. V. H. Winston and Sons, 1978.

[50] K. Königsberger. *Analysis 1*. Springer, Berlin, 3 edition, 1995.

[51] Elias Koutsoupias, Marios Mavronicolas, and Paul Spirakis. Approximate equilibria and ball fusion. *Theory of Computing Systems*, 36:2003, 2002.

[52] R. Landauer. Irreversibility and heat generation in the computing process. *IBM J. Res. Dev.*, 5:183–191, July 1961.

[53] T. Lindvall. *Lectures on the coupling method*. John Wiley and Sons, New York, NY, USA, 1992.

[54] F. Lopez and G. Sanz. Markovian couplings staying in arbitrary subsets of the state space. *Journal of Applied Probability*, 39:197–212, March 2002.

[55] P. D. MacKenzie, C. G. Plaxton, and R. Rajaraman. On contention resolution protocols and associated probabilistic phenomena. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, pages 153–162, New York, NY, USA, 1994. ACM.

[56] P. A. MacMahon. *Combinatory Analysis*, volume 1. Cambridge University Press, 1915.

[57] Gurmeet Singh Manku. Balanced binary trees for id management and load balance in distributed hash tables. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, PODC '04, pages 197–205, New York, NY, USA, 2004. ACM.

[58] A. Marshall and I. Olkin. *Inequalities: Theory of Majorization and Its Applications*. Academic Press, New York, NY, USA, 1979.

[59] M. Matsumoto. Mersenne twister home page. http://www.math.sci.hiroshima-u.ac.jp/∼m-mat/MT/emt.html, 2007. [Online; accessed 03-April-2011].

[60] Friedhelm Meyer auf der Heide, Christian Scheideler, and Volker Stemann. Exploiting storage redundancy to speed up randomized shared memory simulations. *Theor. Comput. Sci.*, 162:245–281, August 1996.

[61] M. Mitzenmacher. Load balancing and density dependent jump markov processes. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 213–222, Washington, DC, USA, 1996. IEEE Computer Society.

[62] M. Mitzenmacher. *The power of two choices in randomized load balancing*. Phd thesis, University of California at Berkeley, Department of Computer Science, Berkeley, CA, USA, 1996.

[63] M. Mitzenmacher, A. Richa, and R. Sitaramani. *Handbook of Randomized Computing: Volume 1*, chapter The power of two random choices: A survey of techniques and results, pages 255–312. Springer, 2001.

[64] Michael Mitzenmacher. On the analysis of randomized load balancing schemes. In *Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, SPAA '97, pages 292–301, New York, NY, USA, 1997. ACM.

[65] Michael Mitzenmacher, Balaji Prabhakar, and Devavrat Shah. Load balancing with memory. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS '02, pages 799–808, Washington, DC, USA, 2002. IEEE Computer Society.

[66] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[67] G. E. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8):114–117, April 1965.

[68] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[69] Moni Naor and Udi Wieder. Novel architectures for p2p applications: The continuous-discrete approach. *ACM Trans. Algorithms*, 3, August 2007.

[70] M. A. Nielsen and I. L. Chang. *Quantum computation and quantum information*. Cambridge University Press, New York, NY, USA, 2000.

[71] Mayur R. Palankar, Adriana Iamnitchi, Matei Ripeanu, and Simson Garfinkel. Amazon s3 for science grids: a viable solution? In *Proceedings of the 2008 international workshop on Data-aware distributed computing*, DADC '08, pages 55–64, New York, NY, USA, 2008. ACM.

[72] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the chernoff–hoeffding bounds. *SIAM J. Comput.*, 26:350–368, April 1997.

[73] Yuval Peres, Kunal Talwar, and Udi Wieder. The $(1 + \beta)$-choice process and weighted balls-into-bins. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1613–1619, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[74] Martin Raab and Angelika Steger. "balls into bins" - a simple and tight analysis. In *Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science*, RANDOM '98, pages 159–170, London, UK, 1998. Springer-Verlag.

[75] S. Ramanujan. Question 294. *J. Indian Math. Soc. 3*, page 128, 1911.

[76] S. Ramanujan. On question 294. *J. Indian Math. Soc. 4*, pages 151–152, 1912.

[77] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, pages 161–172, New York, NY, USA, 2001. ACM.

[78] Kenneth H. Rosen. *Discrete Mathematics and Its Applications*. McGraw-Hill Higher Education, 5th edition, 2002.

[79] P. Sanders. On the competitive analysis of randomized static load balancing. In *Proceedings of the first Workshop on Randomized Parallel Algorithms, RANDOM*, 1996.

[80] Thomas Schickinger and Angelika Steger. *Diskrete Strukturen (Band 2)*. Springer-Verlag, Mar 2001.

[81] C. Shanti. Multicore processors to take over the world. http://www.tgdaily.com/ mobility-features/44987-multicore-processors-to-take-over-the-world, 2009. [Online; accessed 07-April-2011].

[82] Volker Stemann. Parallel balanced allocations. In *Proceedings of the eighth annual ACM symposium on Parallel algorithms and architectures*, SPAA '96, pages 261–269, New York, NY, USA, 1996. ACM.

[83] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.*, 31:149–160, August 2001.

[84] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11:17–32, February 2003.

[85] Kunal Talwar and Udi Wieder. Balanced allocations: the weighted case. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC '07, pages 256–265, New York, NY, USA, 2007. ACM.

[86] H. Thorisson. *Coupling, Stationarity and Regeneration*. Springer-Verlag, New York, 2000.

[87] U. Čibej, B. Slivnik, and B. Robič. The complexity of static data replication in data grids. *Parallel Computing*, 31(8+9):900–912, 2005.

[88] Berthold Vöcking. How asymmetry helps load balancing. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 131–140, Washington, DC, USA, 1999. IEEE Computer Society.

[89] Berthold Vöcking. Symmetric vs. asymmetric multiple choice algorithms. In *Proceedings of the 2nd ARACNE Workshop, Aarhus, Denmark*, pages 7–15, 2001.

[90] Berthold Vöcking. How asymmetry helps load balancing. *Journal of the ACM*, 50:568–589, July 2003.

[91] R. von Mises. Über aufteilungs- und besetzungswahrscheinlichkeiten. *Revue de la Faculté de Sciences de I'Université d'Istanbul*, 4:145–163, 1939.

[92] N. D. Vvedenskaya, R. L. Dobrushin, and F. I. Karpelevich. Queueing system with selection of the shortest of two queues: An asymptotic approach. *Probl. Peredachi Inf.*, 32:20–34, 1996.

[93] W. A. Whitworth. *Choice and Chance*. Cambridge: Deighton, Bell and Co, New York, NY, USA, 4 edition, 1886.

[94] Udi Wieder. Balanced allocations with heterogenous bins. In *Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, SPAA '07, pages 188–193, New York, NY, USA, 2007. ACM.