How Asymmetry Helps Load Balancing

Berthold Vöcking*
International Computer Science Institute
Berkeley, CA 94704-1198
voecking@icsi.berkeley.edu

Abstract

This paper deals with balls and bins processes related to randomized load balancing, dynamic resource allocation, and hashing. Suppose n balls have to be assigned to n bins, where each ball has to be placed without knowledge about the distribution of previously placed balls. The goal is to achieve an allocation that is as even as possible so that no bin gets much more balls than the average. A well known and good solution for this problem is to choose d possible locations for each ball at random, to look into each of these bins, and to place the ball into the least full among these bins. This class of algorithms has been investigated intensively in the past, but almost all previous analyses assume that the d locations for each ball are chosen uniformly and independently at random from the set of all bins.

We investigate whether a non-uniform and possibly dependent choice of the d locations for a ball can improve the load balancing. Three types of selections are distinguished: 1) uniform and independent 2) non-uniform and independent 3) non-uniform and dependent. Our first result shows that choosing the locations in a non-uniform way (type 2) results in a better load balancing than choosing the locations uniformly (type 1). Surprisingly, this smooth load balancing is obtained by an algorithm called "Always-Go-Left" which creates an asymmetric assignment of the balls to the bins. Our second result is a lower bound on the smallest possible maximum load that can be achieved by any allocation algorithm of type 1, 2, or 3. Our upper and lower bounds are tight up to a small additive constant, showing that the Always-Go-Left scheme achieves almost the optimal load balancing.

Furthermore, we show that our upper bound can be generalized to infinite processes in which balls are inserted and deleted by an adversary.

*Supported by a grant of the "Gemeinsames Hochschulsonderprogramm III von Bund und Ländern" through the DAAD. This research was conducted in part while the author was staying at the Heinz Nixdorf Institute, University of Paderborn, with support provided by the "DFG-Sonderforschungsbereich 376".

1. Introduction

Suppose that we place n balls into n bins by putting each ball into a bin chosen randomly and uniformly at random. One of the classical results in probability theory is that this process terminates with an expected number of $(1 + o(1)) \cdot \ln n / \ln\ln n$ balls in the fullest bin.

Karp, Luby, and Meyer auf der Heide [5] introduced a variant of this process. They suggest to allow two possible locations for each ball, chosen independently and uniformly at random from the set of bins. A simple, parallel algorithm decides in which of the two possible bins the ball is actually placed. The algorithm terminates with $O(\ln\ln n)$ balls in the fullest bin, w.h.p.[1] Thus, providing two choices for each ball results in an exponential decrease of the *maximum load*, i.e., the number of balls in the fullest bin.

More exact results were obtained by Azar, Broder, Karlin, and Upfal [2]. They consider sequential allocation processes in which the balls arrive one by one. Each ball comes with $d \geq 2$ possible locations, chosen independently and uniformly at random from the set of bins, and the ball is placed in the least full among them. They give almost matching upper and lower bounds on the maximum load of this genuine allocation process, that is, they show that the fullest bin gets $\ln\ln n / \ln d \pm \Theta(1)$ balls, w.h.p.

The applications of this simple balls and bins process are numerous, e.g., dynamic resource allocation, hashing, and on-line load balancing. These examples are described exhaustively in [2].

Our contribution. We investigate what happens to the maximum load of the sequential allocation process when the d possible locations for each ball are chosen in a non-uniform and possibly dependent fashion, that is, we still assume that the rules for allocating a ball are the same for all balls but the d possible locations for the same ball may be chosen non-uniformly and dependent from each other.

[1]The term "w.h.p." abbreviates "with high probability", i.e., with probability at least $1 - n^{-\alpha}$, where α is an arbitrary constant.

The goal is to improve on the results known for the uniform process. On the first view, this idea seems strange as a non-uniform choice of the bins may tend to an uneven distribution of the balls, resulting in a higher rather than a lower maximum load. In fact, it is not difficult to prove that, for the allocation using only one possible location for each ball, the expected maximum load gets worse when changing to any non-uniform distribution. We will see, however, that this intuition is deceptive in the case of more than one choice for each ball.

We define a *sequential allocation scheme* to be an algorithm that places one ball after the other and uses the same rules for each ball. We consider sequential allocation schemes that place each ball in one out of d bins chosen according to one of the following types of selection.

1) **uniform and independent:** Each of the d locations of each ball is chosen uniformly and independently at random from $[n]$, where $[n] = \{0 \ldots n-1\}$ denotes the set of bins.

2) **(possibly) non-uniform and independent:** For $1 \leq i \leq d$, the ith location of each ball is chosen independently at random from $[n]$ according to an arbitrary probability distribution $D_i : [n] \to [0,1]$.

3) **(possibly) non-uniform and (possibly) dependent:** The d locations of each ball are chosen at random from the set $[n]^d$ according to an arbitrary probability distribution $D : [n]^d \to [0,1]$.

We assume that, when a ball is placed, the outcome of the random choices for the balls that have not yet been placed is unknown. Regarding the knowledge of already placed balls, we make different assumptions in our lower and upper bounds. For the lower bounds, we will allow algorithms that have full knowledge about the previous assignments of balls to the bins, whereas the algorithms that yield our upper bounds use only knowledge about the number of balls in the d locations of the ball to be placed. As a result of our analyses, we will see that the additional power of having full knowledge about the distribution of the balls does not help very much.

We present an upper bound on the maximum load using a non-uniform but independent allocation scheme (type 2) that clearly improves upon the (lower and upper) bounds known for the uniform allocation process (type 1). Furthermore, we prove an almost matching lower bound on the best possible maximum load that can be achieved by any possibly non-uniform and dependent allocation scheme (type 3). Hence, we prove that non-uniform probability distributions can help to reduce the maximum load whereas dependency among the locations of a ball yields no significant further improvement.

Main Results. Our upper bound is obtained by the following algorithm. We divide the bins into d groups of almost equal size, i.e., each group has size $\Theta(n/d)$. These groups are numbered from 1 to d. For each ball, we choose the ith location ($1 \leq i \leq d$) uniformly and independently at random from the ith group. The ball is placed in one of the least full bins among these locations. If there are several locations with smallest load, the ball is placed in the location of the leftmost group, i.e., the group with the smallest number. Because of this asymmetric tie breaking mechanism our algorithm is called "Always-Go-Left".

The following theorem reflects the core of our upper bound. A more general variant of this theorem, e.g., allowing that the number of balls is larger than the number of bins or considering infinite sequences of insertions and deletions, is given later. The upper bound is given in terms related to the Fibonacci numbers. We define $F_d(k) = 0$ for $k \leq 0$, and $F_d(1) = 1$. For $k \geq 2$ we use a recursive definition, we define $F_d(k) = \sum_{i=1}^{d} F_d(k-i)$. The sequence F_2 corresponds to the standard Fibonacci sequence. We define $\phi_d = \lim_{k\to\infty} \sqrt[k]{F_d(k)}$. The value ϕ_2 corresponds to the golden ratio (see e.g. [6]). For example, $\phi_2 = 1.61\ldots$, $\phi_3 = 1.83\ldots$, and $\phi_4 = 1.92\ldots$ In general, $\phi_2 < \phi_3 < \phi_4 \ldots < 2$, and $\phi_d > 2^{(d-1)/d}$ so that $d \cdot \ln(\phi_d) > (d-1) \cdot \ln 2$.

Theorem 1 *Suppose that n balls are placed sequentially into n bins using the Always-Go-Left algorithm (type 2). Then the number of balls in the fullest bin is $\ln\ln n/(d \cdot \ln \phi_d) + O(1)$, w.h.p.*

Our result generally improves upon the one for the uniform case because $\ln\ln n/(d \cdot \ln \phi_d) < \ln\ln n/\ln d$. In particular, the result shows that the influence of d on the maximum load is linear rather than logarithmic as in the uniform case. But even for the case $d = 2$, we obtain an improvement because the Always-Go-Left algorithm yields maximum load $0.69\ldots \log_2 \ln n + O(1)$ whereas the maximum load in the uniform case is $\log_2 \ln n \pm \Theta(1)$.

We point out that the asymmetry used for tie breaking in our algorithm is important not only for the analysis but for the result itself. Using a fair tie breaking mechanism, i.e., deciding at random which bin gets the ball in case of a tie, results in a higher maximum load, to be precise, in maximum load $\ln\ln n/\ln d \pm \Theta(1)$. This (upper and lower) bound can be shown using a slight variant of the proofs presented by Azar et al. in [2] for the uniform scheme. Also combining the asymmetric tie breaking with a uniform choice of the locations does not help as Azar et al. show that their uniform allocation scheme with an arbitrary tie breaking mechanism is "majorized" by any other uniform allocation scheme. From this we can conclude that the tie breaking mechanism is irrelevant in the uniform case, whereas it is important in the non-uniform case.

Once we have seen that choosing the locations in a non-uniform way combined with an asymmetric tie breaking mechanism reduces the maximum load, it becomes an interesting question whether other kinds of choices for the d locations or other schemes for deciding which of these locations finally gets the ball can improve on this result. The following theorem answers this question negatively.

Theorem 2 *Suppose that n balls are placed sequentially into n bins using an arbitrary sequential allocation scheme choosing d bins for each ball at random according to an arbitrary probability distribution on $[n]^d$ (type 3). Then the number of balls in the fullest bin is $\ln\ln n/(d\cdot\ln\phi_d)-O(1)$, w.h.p.*

Combining Theorem 1 and 2, we conclude that, apart from some additive constants, the Always-Go-Left algorithm achieves the best possible maximum load, i.e., $\ln\ln n/(d\cdot\ln\phi_d)\pm\Theta(1)$.

Generalizations. It is also interesting to study variants of the random allocation process assuming more balls than bins or even an infinite sequence of insertions and deletions. We use a combined approach to model both of these variants. In particular, we assume a possibly infinite sequence $\sigma=\sigma_1\sigma_2\ldots$ of insertions and deletions of balls. Each insertion corresponds to a ball that has to be assigned to one of the bins, each deletion specifies one of the previously inserted balls that is removed from the bin to which it has been assigned. An allocation algorithm has to *serve* all allocations *on-line*, that is, the sequence is presented one by one and a ball has to be placed for each insertion σ_t without knowing future insertions and deletions, i.e., $\sigma_{t+1}\sigma_{t+2}\ldots$. The sequence is assumed to be *oblivious*, i.e., it does not depend on the random choices of the allocation algorithm.

Next we give a generalized version of Theorem 1. *Time* t denotes that point of time at which σ_t is presented but not yet served, and a ball is said to *exist at time t* if it has been inserted but not deleted before t.

Theorem 1 (generalized version) *Suppose that at most $h\cdot n$ balls exist at any point of time. Then the Always-Go-Left algorithm yields maximum load $\ln\ln n/(d\cdot\ln\phi_d)+O(h)$, w.h.p., at any fixed time step t.*

Assuming only $h\cdot n$ insertions and no deletions, our model degenerates to the static problem of allocating $h\cdot n$ balls to n bins. The corresponding best known upper bound for the uniform allocation process is $\ln\ln n/\ln d+O(h)$, w.h.p., given Azar et al. in [2].

Assuming $h=1$ and that, after n insertions, the sequence alternates between deletions of balls that are chosen uniformly at random from the set of existing balls and insertions of new balls, our model corresponds to the dynamic problem considered by Azar et al. in [2]. They show an upper bound on the maximum load of the uniform allocation process of $\ln\ln n/\ln d+O(1)$, w.h.p., for any fixed step $t>n^3$.

Assuming $h=1$ and arbitrary insertions and deletions, our model corresponds to the one considered by Cole et al. in [4, 3]. In [4] they investigate a more complicated routing problem on butterfly networks. Translating their results to the simpler problem of placing balls into bins yields an upper bound of $4\cdot\ln\ln n/\ln 2$, for $d=2$. This translation is described explicitly in [3]. Furthermore, they present an upper bound of $\ln\ln n/\ln d+O(1)$, which, however, holds only for a polynomial number of steps. The bound increases slightly with time in an infinite setting. Applying our proof techniques to the uniform process we obtain the following improvement.

Theorem 3 *Suppose that at most $h\cdot n$ balls exist at any point of time. Then using the uniform allocation algorithm yields maximum load $\ln\ln n/\ln d+O(h)$, w.h.p., at any fixed time step t.*

The rest of the paper is organized as follows. In Section 2, we will prove the upper bounds given in the Theorems 1 and 3. We will first analyze the infinite symmetric scheme (Theorem 3), and then we will show how the analysis changes when considering the asymmetric scheme (Theorem 1). Finally, in Section 3, we will present the proof for the lower bound given in Theorem 2.

2. Proof of the Upper Bounds

In this section, we prove the upper bounds given in the Theorems 1 and 3. We are given an infinite sequence of insertions and deletions of balls, as described in the Introduction. There are n bins, and the number of balls that exist at the same time is at most $h\cdot n$. Each ball is placed in one out of $d\geq 2$ possible locations. We investigate the maximum load produced by the symmetric, uniform and the asymmetric, non-uniform allocation process.

In both the symmetric and the asymmetric case, we use the same kind of analysis, that is, we use witness trees to bound the probability for the bad event that a bin contains too many balls. This witness tree is a tree graph the nodes of which represent balls whose randomly chosen locations are arranged in a bad fashion. In Section 2.1, we will describe the construction of a symmetric witness tree for the symmetric allocation scheme. In Section 2.2, we will show how this construction changes in the asymmetric case. Initially, we make some simplifying assumptions. First, we assume that all events represented by a witness tree are stochastically independent. In Section 2.3, we will remove this simplifying assumption and give a common solution for dealing with dependencies in both the symmetric and the asymmetric case. The second simplifying assumption is that $h = 1$, i.e., at most n balls exist at any time. Finally, in Section 2.4 we will extend our proofs to general h.

The witness tree analysis is not a new idea of ours. It has been introduced by Scheideler, Stemann, and Meyer auf der Heide in the context of PRAM simulations [7]. Their argument has been improved by several authors in several articles, e.g., [1, 3, 4, 8]. Only some of these analyses, however, deal with the sequential allocation of balls to bins, namely [3, 4]. They lose some constant factor in the maximum load whereas our analysis is tight up to an additive constant. Furthermore, we remark that the finite variant of the asymmetric allocation scheme can be analyzed using the techniques introduced by Azar et al. [2] or Mitzenmacher [9, 10], too. We prefer the witness tree analysis, however, as it naturally extends to the case of infinite sequences of adversarial insertions and deletions, which cannot be handled by any of the other methods so far.

2.1. The witness tree for the symmetric scheme

A witness tree is a logical structure that can be constructed in case of a *bad event*, i.e., when the maximum load exceeds some threshold value which we will specify later. Thus an active witness tree is a necessary condition for bad events. In the following, we will show that the existence of an active witness tree is unlikely. Consequently, bad events are unlikely, too.

Definition of a symmetric witness tree. A *symmetric witness tree* of order L is a complete d-ary tree of height L with d^L leaf nodes. Each node v in this tree represents a ball. The assignment of balls to bins has to fulfill the following time constraints. Consider a fixed sequence of insertions and deletions and let t denote the fixed time step given in the theorem. The ball represented by the root node r exists at time t, and each ball represented by a node $v \neq r$ exists at the insertion time of the ball represented by the parent node of v. Note that the same ball may be represented by several nodes in the tree.

Each node of the witness tree represents an event that may occur or not, depending on the random choices for the locations of the balls. We distinguish between edge and leaf events.

- *Edge event:* Consider an edge $e = (u, v)$ with v being the ith child of u. The edge e represents the event that the ith location of u's ball points to the same location as one of the locations of v's ball.
- *Leaf event:* A leaf node v represents the event that each of the d locations of v's ball points to a bin that contains at least three balls at its insertion time.

An edge or a leaf node is *activated* if the random choices for the possible locations of the balls come out in such a way that the corresponding event occurs. The witness tree is *activated* if all of its edges and leaf nodes are activated.

Construction of a symmetric witness tree. We use an active witness tree of order L to represent the bad event that some of the bins contain more than $L + 3$ balls at time t. Assume the allocation process is determined up to time t. Suppose that a bin x holds at least $L + 4$ balls at time t. We show that this implies that there exists an active witness tree of order L. This tree can be constructed as follows. The root gets assigned the *topmost ball* in the bin x, i.e., the ball that was inserted last into x. The symmetric allocation scheme ensures that each of the d locations of that ball must point to a bin that contains at least $L + 3$ balls at the ball's insertion time. The topmost balls in these d bins are assigned to the children of the root. This construction can be continued recursively until we reach the leaf nodes. Each ball assigned to a leaf node lies on top of three other balls, and, hence, each of its d locations points to a bin that contains three other balls at the ball's insertion time.

Probability for the activation of a symmetric witness tree. As a bin including more than $L + 3$ balls implies the existence of an active witness tree of order L, the probability that the maximum load is larger than $L + 3$ at some time t is bounded by the probability that a witness tree of order L is active. In the following, we calculate a bound on the the latter probability. During this calculation we do not make any use of the explicit construction of the witness tree but only of its formal definition. We point out that drawing to the construction instead would result in vast dependencies among the considered events.

The probability that a witness tree of order L is active is bounded above by the number of different witness trees multiplied by the probability that any fixed witness trees becomes active. In this section, we account only for witness trees whose nodes represent distinct balls, that is, we assume that each ball occurs at most once in the witness tree. In Section 2.3 we will show how to deal with witness trees that may include the same ball for several times.

We start by counting the number of different witness trees. The number of possibilities to choose the root's ball is at most n because, by definition, this ball has to exist at the considered time t, and at most n balls exist at any time. The ball represented by the root gives us the time step at which the balls corresponding to the children of the root have to exist. Hence, the number of possibilities to assign a ball to one of the children is n, too. Applying this argument level by level to all nodes in the tree yields that the total number of possibilities to choose a ball for each of the nodes is at most n^m with m denoting the number of nodes in the witness tree.

Next we bound the probability that a fixed witness tree is activated. The probability that the event represented by an edge (u, v) occurs is d/n because, whatever bin the ith location of u points to, the probability that any fixed one of the d locations of v hit that bin is $1/n$. Because of our assumption that all balls are distinct, the events for all edges in the tree are independent, and therefore, the probability that all of the $m-1$ edges are activated is $(d/n)^{m-1}$. The probability for the event represented by a leaf node v is at most 3^{-d} because each of the d locations of $v's$ ball has to point to a bin that contains 3 balls and at any time the number of these bins is bounded above by $n/3$. Note that the latter bound on the number of bins with load ≥ 3 holds deterministically so that we can assume that the occurrence of a leaf event does not give any evidence about the distribution of any other balls than the one represented by the respective leaf node. Therefore, the probability that all of the leaf events occur is at most $3^{-d\cdot q}$ with q denoting the number of leaves in the tree.

Altogether, we can conclude that the probability that there exists an active symmetric witness tree with distinct balls at any fixed time t is at most

$$\begin{aligned} n^m \cdot \left(\frac{d}{n}\right)^{m-1} \cdot 3^{-d\cdot q} &\leq n \cdot d^{2q} \cdot 3^{-d\cdot q} \\ &\leq n \cdot 2^{-q} \\ &= n \cdot 2^{-d^L} \end{aligned}$$

because $m \leq 2q$, $2 \cdot d^2 \leq 3^d$, and $q = d^L$. Consequently, if we choose $L \geq \log_d \log_2 n + \log_d(1+\alpha)$ then the probability for the existence of a witness tree with distinct balls is at most $n^{-\alpha}$.

In Section 1.3 we will extend this argument to witness trees with non-distinct balls. Before, however, we will investigate how the bound above improves in the case of the Always-Go-Left algorithm.

2.2. The witness tree for the asymmetric scheme

The witness trees that we will construct for the asymmetric allocation scheme are asymmetric, too. We will see that the asymmetric trees are much larger than their symmetric counterparts, resulting in a smaller probability for their activation.

Definition of an asymmetric witness tree. An *asymmetric witness tree* is defined similarly to its symmetric counterpart. The only difference is the topology. It is a Fibonacci tree, which is defined recursively as follows. $T_d(1)$ and $T_d(2)$ consist out of a single node, respectively. $T_d(k)$, for $2 \leq k \leq d$, is a rooted tree whose root has $k-1$ children which are the roots of the trees $T_d(k-1), \ldots, T_d(1)$, from left to right. $T_d(k)$, for $k > d$, is a rooted tree whose root has d children which are the roots of the trees $T_d(k-1), \ldots, T_d(k-d)$. The number of leaves in $T_d(k)$ is $F_d(k) \geq \phi_d^{k-2}$. An asymmetric witness tree of order L has the topology of the Fibonacci tree $T_d(d \cdot L + 1)$. Hence, it has $F_d(d \cdot L + 1) \geq \phi_d^{d\cdot L-1}$ leaves.

The activation of an asymmetric witness tree is defined analogously to the symmetric case.

Construction of an asymmetric witness tree. Suppose some of the bins contain more than $L+3$ balls at time t. In this case, we can construct an active, asymmetric witness tree as follows. To simplify the construction, we define labels for the nodes of the witness tree. Each label is a tuple (ℓ, i) of two integers ℓ and i with $0 \leq \ell \leq L$ and $1 \leq i \leq d$. The labels are not unique. The label of the root is $(L, 1)$. For $\ell \geq 1$, the children of a node with label (ℓ, i) have the labels $(\ell, i-1), \ldots, (\ell, 1), (\ell-1, d), \ldots, (\ell-1, i)$, from left to right. The children of a node with label $(0, i)$, for $i \geq 3$, have the labels $(0, i-1), \ldots, (0, 1)$. In the construction of the witness tree we assign balls to the nodes maintaining the following invariant: A ball b that is assigned to a node with label (ℓ, i) was placed into a bin that contained at least $\ell + 3$ other balls at b's insertion time. This bin belongs to group i (except for the root node for which we do not care about the group).

The assignment proceeds iteratively, starting with the root. The root gets assigned the topmost ball in one of the bins including $L+4$ balls. For $1 \leq i \leq d$, the ith location of this ball points to a bin that contains at least $L+3$ balls at time t. The topmost ball of location i is assigned to that child with label $(L-1, i)$. Now assume that we are given a node v with label (ℓ, i) whose ball b is allocated to a bin in group i that contains at least $\ell + 3$ other balls at b's insertion time. For $1 \leq j < i$, the jth location of b points to a bin that contains at least $\ell+3$ balls. This is because the Always-Go-Left scheme prescribes that b would have been placed at the jth location, otherwise. We assign the topmost ball of location j to the child v with label (ℓ, j) (provided that ℓ and i are sufficiently large such that the respective child exists). For $i \leq j \leq d$, the jth location of b contains at least $\ell + 2$ balls at b's insertion time. In this case, we assign the topmost ball of location j to the child with label $(\ell - 1, j)$ (provided that the respective child exists). Obviously, these assignments fulfill the invariant above.

Probability for the activation of an asymmetric witness tree. We derive the same bounds on the number of trees and the probability for the activation of a fixed tree as in the symmetric case. The number of different witness trees is

n^m with m denoting the number of nodes in the tree. The probability that all edge events occur is at most $(d/n)^{m-1}$, and the probability that all leaf events occur is $3^{-d \cdot q}$ with q denoting the number of leaves in the tree.

For the bound on the probability of the edge events, we assume that all groups have exactly the same size. It is not difficult to see, however, that our analysis is robust against changes of constant factors in the group sizes. The bound on the probability for the leaf events may need some further explanations. Analogously to the symmetric case, at most $n/3$ bins contain three or more balls at any time, but we do not know how these bins are distributed among the d groups. For $1 \leq i \leq d$, let β_i denote the fraction of bins in group i that contain three or more bins. Then the probability that all locations of a ball hit one of these bins is $\prod_{i=1}^{d} \beta_i$. Since all groups have size n/d we get $\sum_{i=1}^{d} \beta_i \cdot n/d \leq n/3$ and, hence, $\sum_{i=1}^{d} \beta_i \leq d/3$. As the product of some positive variables that sum up to a fixed value is maximized when the values of all variables are equal, we can conclude that the probability for a leaf event is $\prod_{i=1}^{d} \beta_i \leq 3^{-d}$.

Combining the bounds above yields that the probability that there exists an active asymmetric witness tree with distinct balls at any fixed time t is at most

$$\begin{aligned} n^m \cdot \left(\frac{d}{n}\right)^{m-1} \cdot 3^{-d \cdot q} &\leq n \cdot d^{2q} \cdot 3^{-d \cdot q} \\ &\leq n \cdot 2^{-q} \\ &\leq n \cdot 2^{-\phi_d^{d \cdot L - 1}} \end{aligned}$$

because $m \leq 2q$, $2 \cdot d^2 \leq 3^d$, and $q \geq \phi_d^{d \cdot L-1}$. The only difference to the symmetric case is that the asymmetric tree is larger and, hence, the number of leaves, q, is larger, which results in a smaller probability for the activation of the tree. In particular, if we choose $L \geq (\ln \log_2 n + \ln(1+\alpha))/(d \cdot \ln \phi_d) + 1$, the probability for the existence of a witness tree with distinct balls is at most $n^{-\alpha}$.

2.3. Witness trees with non-distinct balls

Until now we have only estimated the probability for the activation of witness trees with distinct balls. In order to get a bound on the probability for a high maximum load, it remains to consider trees in which balls may occur more than once. In this case the events represented by the edges and leaves are not stochastically independent anymore, which was an important assumption in the calculations above. To remove these dependencies, we prune the witness tree such that only distinct balls remain. Unfortunately, any pruning makes the tree smaller so that the probability for its activation may become larger. We compensate this loss by starting from a larger tree and, additionally, gaining some probability from each pruning event.

Definition of a full witness tree. A *full witness tree* of order L includes κ symmetric or asymmetric witness trees of the same order, respectively, as subtrees, where κ denotes a suitable constant. The root of the full witness tree has κ children each of which has only one child. Hence, the root of the full tree has κ grandchildren. Each of these grandchildren is the root of a symmetric or asymmetric witness tree of order L, respectively. The most important new feature of the full witness tree is that the balls assigned to the children of the root are guaranteed to be pairwise distinct. The root represents the same ball as one of its children, e.g., the ball of the leftmost child. We define that each ball b assigned to a child v of the root has a location that points to the same bin as the root (just as in case of the symmetric or asymmetric witness tree). Let i_b denote the index of this location, and choose j_b from $\{1, \ldots, d\} \setminus \{i_v\}$ arbitrarily. Let u denote v's only child. Then one of the locations of u's ball points to the same bin as the location j_b of b. Below the node u our definitions follow the rules of the symmetric or asymmetric witness tree, respectively. The time constraints among all balls in the full tree and the activation of the full tree are defined analogously to the symmetric and asymmetric trees.

Construction of a full witness tree. The full witness tree is constructed as follows. Suppose some bin x holds holds $L + 4 + \kappa$ balls at time t. Then we assign the κ topmost balls in x to the children of the root, and we assign the ball of the leftmost child to the root, too. Next we assign balls to the grandchildren. Consider a ball b assigned to a child v of the root. One of b's locations points to the bin x. Let this be location i_b. (If there are more than one location pointing to x, fix one arbitrarily.) Select $j_b \neq i_b$ arbitrarily. At b's insertion time, location j_b points to a bin with at least $L+3$ balls. The topmost ball in this bin is assigned to the child of v. Below this child, the construction is continued as done for the symmetric or asymmetric witness tree of order L, respectively.

Pruning the full witness tree. In order to remove dependencies we inspect the non-root nodes of the full witness tree in BFS order. Whenever we inspect a node v that represents the same ball as another node inspected before, we cut the edge e between v and its parent node and cutoff the complete subtree rooted at v. The edge e is called a *cutoff edge*, and we keep it in our witness structure as evidence that the ball of the parent node of v shares a location with a ball represented by another node in the tree. We continue this process until we have obtained either κ cutoff edges or we have inspected all non-pruned nodes. All inspected, non-pruned nodes and the cutoff edges are defined to build the *pruned witness tree*.

Notice that the pruning does not remove any of the children of the root because these nodes represent distinct balls and they are visited first since we traverse the tree in BFS order skipping the root. Therefore, if there fewer than κ cutoff edges, one of the κ symmetric or asymmetric witness trees includes only distinct balls. We have bounded above the probability for the existence of such a witness tree in the previous sections by $n^{-\alpha}$, for any constant α, under the assumption that L is chosen sufficiently large. Therefore, it remains only to estimate the probability for the existence of an active pruned witness tree that includes exactly κ cutoff edges. In the following, we upper-bound this probability under the assumption that each of the witness trees below the grandchildren of the root has at most $2(\alpha+1)\cdot\log_2 n$ nodes. It is easy to check that this number of nodes suffices to obtain probability $n^{-\alpha}$ in the case of a subtree with distinct balls.

Probability for the activation of a pruned witness trees. We have to take into account that a pruned witness tree can have several different topologies. Let $M \leq 2\kappa\cdot(\alpha+1)\cdot\log_2 n$ denote the number of those edges in the full tree that may become a cutoff edge. The number of different topologies for the pruned witness tree is at most M^κ. Now assume the topology is fixed. Let m denote the number of non-root nodes in the pruned witness tree. Then the number of possibilities to assign balls to the tree nodes is n^m. (Recall that the ball of the root is identical to the ball of its leftmost child.) Furthermore, the probability that all edge events are activated is $(d/n)^{m-1}$. (Notice that the edge connecting the root with its leftmost child is active by definition. All other edge events have probability d/n, and these probabilities are independent as we have truncated all redundant balls.) Let q denote the number of *leaves in the pruned tree*, i.e., the nodes that have no children and are not incident to cutoff edges. Then, the probability that all leaves are activated is at most $3^{-d\cdot q}$.

Apparently, we obtained almost the same bounds as for the symmetric and asymmetric witness trees. However, m and q may be much smaller now so that we need to gain some further probability from the cutoff edges. Each of the cutoff edges witnesses the event that a ball b represented by a non-pruned node u incident to the cutoff edge shares some random location with a ball b' of another node u'. The node u' was inspected before u without being truncated and, hence, is guaranteed to be included in the pruned tree, too. The cutoff edge specifies which of the randomly chosen locations of b hits one of the locations of b'. This location of b falls into the same bin as one of the location of b'. In the following we give a bound for the probability of this event. The number of possibilities to select u' and thus b' is bounded above by the number of nodes in the pruned tree minus the node u itself. This number is upper bounded by M. The probability that the location of b hits one of the locations of the fixed ball b' is at most d/n. Thus, $M\cdot d/n$ is an upper bound on the probability for the event represented by a cutoff edge.

Notice that the randomly chosen location represented by the cutoff edge is not considered in any bound on the probability for the activation of the edge and leaf events. Furthermore, events represented by different cutoff edges are independent, too. Therefore, we obtain the following upper bound on the probability for the existence of a pruned witness tree with κ cutoff edges.

$$\begin{aligned} & M^\kappa\cdot n^m\cdot\left(\frac{d}{n}\right)^{m-1}\cdot 3^{-d\cdot q}\cdot\left(\frac{M\cdot d}{n}\right)^\kappa \\ \leq\quad & n\cdot d^{2q}\cdot 3^{-d\cdot q}\left(\frac{M^2\cdot d^3}{n}\right)^\kappa \\ \leq\quad & n\cdot\left(\frac{(2\kappa\cdot(\alpha+1)\cdot\log_2 n)^2\cdot d^3}{n}\right)^\kappa \end{aligned}$$

because $m\leq 2\cdot(q+\kappa)$, $d^2\leq 3^d$, and $M\leq 2\kappa\cdot(\alpha+1)\cdot\log_2 n)$. Since our Theorems are obvious for $d\geq\log n$, we assume $d\leq\log n$. Then the last term in the equation above is bounded by $n^{-\kappa+1+o(1}$. Combining this result with the bounds given in the previous sections, we obtain that the probability that the load exceeds

$$L+3 \quad\leq\quad \frac{\ln\log_2 n+\ln(1+\alpha)}{\ln d}+3+\kappa\ ,$$

in the symmetric case, or

$$L+3 \quad\leq\quad \frac{\ln\log_2 n+\ln(1+\alpha)}{d\cdot\ln\phi_d}+4+\kappa\ ,$$

in the asymmetric case, is at most $n^{-\kappa+1+o(1)}+n^{-\alpha}$, This completes the proof of Theorem 1 and 3, for $h=1$.

2.4. More balls than bins

Finally, we extend the above proof to general h, that is, we assume $h\cdot n$ balls may exist at the same time. The definitions of the symmetric and asymmetric witness trees are changed as follows. Each leaf node in the witness tree represents h balls rather than only one. The locations of these balls point to bins that contain βh balls, for constant β, rather than only 3 balls.

Under these assumptions, the probability for the existence of an active witness trees with distinct balls can be bounded as follows. Let m denote the number of nodes and q the number of leafs in the tree. Then the number of possibilities to assign balls to the nodes is at most

$$\binom{h\cdot n}{h}^q\cdot(h\cdot n)^{m-q}\ .$$

This is because there are $\binom{h\cdot n}{h}^q$ possibilities to choose the balls for each leaf node, and $h \cdot n$ possibilities to choose the ball for each internal node.

The probability for each edge event is d/n. As before, the tree includes $m-1$ edges. The edges to the leaves, however, represent h edge events now. Thus, the probability for the activation of all edge events is $(d/n)^{(h-1)q+m-1}$. The probability that all of the d locations of a ball point to a bin with β balls is β^{-d}. However, we cannot account this probability for all of the balls assigned to the leaves because, when estimating the probability for the edge events, we assumed that all balls assigned to the same leaf node share a random location. Consequently, we have only $d + (h-1)(d-1) = hd - h + 1$ independent locations for the balls at each leaf node. Thus, the probability for the activation of all leaf nodes is at most $\beta^{-q(hd-h+1)}$.

Combining these bounds, the probability for the existence of a witness tree with distinct balls is bounded above by

$$\binom{h\cdot n}{h}^q \cdot (hn)^{m-q} \cdot \left(\frac{d}{n}\right)^{(h-1)q+m-1} \cdot \beta^{-q(hd-h+1)} \ ,$$

which, for β chosen sufficiently large, is bounded above by $n \cdot 2^{-q}$, analogously to the case $h = 1$. The rest of the proof proceeds as before.

3. Proof of the Lower Bound

In this section, we prove the lower bound given in Theorem 2. Suppose n balls are placed sequentially into n bins using an arbitrary sequential allocation scheme choosing d bins for each ball at random according to an arbitrary probability distribution $D : [n]^d \to [0,1]$. We show that the number of balls in the fullest bin is $\ln\ln n/(d \cdot \ln \phi_d) - O(1)$, w.h.p.

The major issue in proving this lower bound is that the possibly dependent random choices for the d locations of a ball are very difficult to handle. As a first step to separate the effects of different locations, we assume that the allocation algorithm uses d disjoint groups each of which consists of n bins numbered from 0 to $n-1$. Obviously, any algorithm A in the original model (with a total number of n bins) can be simulated by an algorithm B in the new model (with $d\cdot n$ bins) so that the maximum load of B is not larger than the maximum load of A. (Recall that we allow full knowledge about the distribution of previously placed balls.) Hence, a lower bound in the model with $d \cdot n$ bins holds for the original model with n bins, too.

Let m and $n_1, \dots, n_d$ denote some positive, real numbers. We define an $(m, n_1, \dots, n_d)$-*allocation* to be an assignment of at least m balls to at most $\sum n_i$ bins satisfying the following properties:

- The bins are divided into d groups $N_1, \dots, N_d$ with $|N_i| \le n_i$, for $1 \le i \le d$.
- Each of the balls is placed into one out of d bins chosen according to an arbitrary, fixed probability distribution $D : N_1 \times \cdots \times N_d \to [0,1]$.
- The balls are assigned one after the other. When a ball is placed, the random choices for the locations of those balls that have not yet been placed are unknown.

The *maximum load* L^* of an $(m, n_1, \dots, n_d)$-allocation is defined to be the number of balls in the fullest bin. The *aggregate load* L is defined by $L = \sum_{i=1}^d L_i$ with L_i denoting the number of balls in the fullest bin from the set N_i. The following lemma gives a lower bound on the aggregate load of an $(n, n, \dots, n)$-allocation scheme.

Lemma 4 *For any $(n, n, \dots, n)$-allocation scheme, $L = \ln\ln n/\ln\phi_d - O(\ln\ln d)$, w.h.p.*

This lemma implies the lower bound given in Theorem 2 because $L^* \ge L/d$. In the rest of this section we deal with the proof of this lemma.

We lower-bound the aggregate load by a recursive function ℓ. Given some positive real numbers $x_0, \dots, x_n$, we define $\ell(x_0, \dots, x_n) = 0$ if $x_0 < \sqrt{n}$, and

$$\ell(x_0, \dots, x_n) = 1 + \min_{1\le i\le d} \ell(x_0', x_1', \dots, x_d')$$

if $x_0 \ge \sqrt{n}$, where

$$x_0' = \frac{(x_0)^2}{(10d)^2 \cdot x_i} \ , \quad x_i' = \frac{x_0}{10d} \ ,$$

and $x_j' = x_j$, for $j \in \{1, \dots, d\} \setminus \{i\}$.

Lemma 5 *Suppose n is sufficiently large. Let $m, n_1, \dots, n_d \le n$ denote some positive numbers with $m \le n_i$, for $i \in [n]$. Then the aggregate load of an $(m, n_1, \dots, n_d)$-allocation is at least $\ell(m, n_1, \dots, n_d)$, with probability $1 - n^{-\alpha}$, for any constant α.*

Proof. We prove the lemma by induction, that is, we assume that the lemma holds for any $(m', \dots)$-allocation with $m' < m$. We allow that the failure probability is slightly increasing during the induction. In particular, we assume that the aggregate load of the $(m', \dots)$-allocation is at least $\ell(m', \dots)$, with probability $1 - n^{-\alpha}/2$ rather than only $1 - n^{-\alpha}$. Notice, however, over all induction steps we lose only a factor of $\ln n$ in the failure probability since we need no more than $\log_2 \ln n$ induction steps.

Let B denote the set of m balls that have to be placed in the bins. Each of the balls comes with d locations chosen according to some probability distribution $D : N_1 \times \cdots \times$

$N_d \to [0,1]$. Let B_1 denote the set of $m/2$ balls that are inserted first, and define $B_2 = B \setminus B_1$. For $j \geq 1$, we define

$$n_i'(j) = \frac{m^j}{(10d)^{j\cdot(j+1)/2} \cdot n_i^{j-1}} \text{ and } m'(j) = \frac{m \cdot n_i'(j)}{(10d)^j \cdot n_i}.$$

The motivation for these two definitions will become clear soon. For the time being, it is sufficient to notice that $n_i > n_i'(1) > n_i'(2) \ldots$ and $m > m'(1) > m'(2) \ldots$

We will show the following two properties

1) There is a set $N_i' \subseteq N_i$ of size $\lfloor n_i'(j) \rfloor$, for some $1 \leq i \leq d$ and $j \geq 1$, such that each bin from N_i' gets assigned at least j balls from B_1.
2) There is a set $B' \subseteq B_2$ with $|B'| \geq m'(j)$ such that the ith location of each ball in B' points to one of the bins in N_i'.

Each of the two properties hold with probability $1 - n^{-\alpha}/4$. The first property requires $m \geq \sqrt{n}$, and the second property requires $m'(j) \geq \sqrt{n}$.

When proving these properties, we give no other evidence about the randomly chosen locations of the balls in B' except that their ith locations fall into the bins of N_i'. Thus, all these balls choose their locations from $N_1 \times \cdots \times N_{i-1} \times N_i' \times N_{i+1} \times \cdots \times N_d$ according to the same probability distribution. Suppose we remove all balls from $B \setminus B'$. Then the allocation of the balls in B' is an $(m'(j), n_1, \ldots, n_i'(j), \ldots, n_d)$ -allocation with some probability distribution $D' : N_1 \times \cdots \times N_i' \times \cdots \times N_d \to [0,1]$. By induction assumption, this allocation has an aggregate load of at least $\ell(m'(j), n_1, \ldots, n_i'(j), \ldots, n_d)$, with probability $1 - n^{-\alpha}/2$.

The allocation of the balls in B', however, does not start with a set of empty bins. Instead it takes place on top of a plateau of height j produced by the balls from B_1 in $N_i'(j)$. Therefore, we can conclude that the aggregate load of the original $(m, n_1, \ldots, n_d)$-allocation is at least

$$j + \ell(m'(j), n_1, \ldots, n_{i-1}, n_i'(j), n_{i+1}, \ldots, n_d) \ , \qquad (1)$$

with probability at least $1 - n^{-\alpha}/2 - 2 \cdot n^{-\alpha}/4 = 1 - n^{-\alpha}$. Notice that the fact that the allocation of the balls in B' takes place on top of the balls in B_1 and is interleaved with the allocation of the balls in $B_2 \setminus B'$ does not affect the lower bound on the aggregate load of the balls in B' because we assume that the allocation algorithm has global knowledge so that no kind of advantage can be taken from the other balls.

For $j = 1$, the lower bound in 1 corresponds exactly to the recursive description of $\ell(m(j), n_1, \ldots, n_d)$. For $j > 1$, the recursion has to be applied repeatedly. In particular, we have defined $n_i'(j)$ and $m'(j)$ in such a way that applying the recursion to the lower bound in 1 repeatedly for j times, whereby we choose the same i in each iteration instead of using the minimum operator, yields exactly the lower bound given in the lemma. Hence, the lower bound in 1 implies the bound in the lemma, and it remains only to prove the two properties stated above.

Proof of property 1: A bin from the set N_i of group i is called *large* if the probability that the ith location from a given ball from B_1 falls into that bin is at least $1/(2dn_i)$. A ball from B_1 is called *interesting* if none of its d locations fall into a small (non-large) bin. The sum of the probabilities assigned to the small bins from N_i is at most $(n_i - 1)/(2d \cdot n_i) < 1/(2d)$. Hence, the probability that a ball is not interesting, i.e., at least one of the d locations from a ball in B_1 points to a small bin, is at most $d/(2d) = 1/2$. Notice that this bound holds regardless of the dependencies between the random choices of the d locations. Thus, the expected number of interesting balls is at least $|B_1|/2 = m/4$. Applying a Chernoff bound yields that at least $m/5$ balls from B_1 are interesting, with probability $1 - n^{-\alpha}/4$, under the assumption that n is sufficiently large, and $m \geq \sqrt{n}$.

Next we calculate how many of the large bins in any one of the d groups contain j or more balls. At least one of the d groups gets a fraction of $1/d$ of the interesting balls, that is, this group gets at least $m/(5d)$ interesting balls. Let i denote this group, and $M \geq m/(5d)$ denote the number of interesting balls placed in this group. We have to assume that the M balls are distributed arbitrarily among the large bins in group i because the assignment of the balls to the groups is done by an unknown mechanism. But in whatever manner the balls are distributed, at least $M/2^j$ large bins contains at least j of the balls, for some $j \geq 1$. (Otherwise, the total number of balls would be smaller than $\sum_{j\geq 1}^{\infty} M/2^j < M$.) We fix the appropriate j. Then the number of large bins from N_i with load j or more is at least

$$\frac{M}{2^j} \geq \frac{m}{5d \cdot 2^j} \geq \frac{m^j}{(10d)^{j\cdot(j+1)/2} \cdot n_i^{j-1}} = n_i'(j) \ ,$$

where the second equation follows from the assumption that $n_i \geq m$. We define N_i' to be a subset of size $\lfloor n_i' \rfloor$ of the set of large bins with load j. Thus property 1 is satisfied.

Proof of property 2: Next we analyze how many balls from B_2 have a location pointing to a bin of N_i'. The probability that the ith location of a given ball points to one of the bins in N_i' is at least $\lfloor n_i'(j) \rfloor/(2d \cdot n_i) \geq n_i'(j)/(4d \cdot n_i)$ because these bins are large and, therefore, each of them has probability at least $1/(2d \cdot n_i)$ to be hit.

As $|B_2| = m/2$ the expected number of balls in B', i.e., the set of the balls with a location in N_i', is at least $m \cdot$

$n'_i(j)/(8d \cdot n_i)$. Applying a Chernoff bound yields that, with probability at most $1 - n^{-\alpha}/4$, the number of balls in B' is at least

$$\frac{m \cdot n'_i(j)}{10d \cdot n_i} \geq \frac{m \cdot n'_i(j)}{(10d)^j \cdot n_i} = m'(j) ,$$

assuming that n is sufficiently large and $m'(j) \geq \sqrt{n}$ so that the Chernoff bound yields the desired result. This equation shows that property 2 is satisfied and, hence, the proof of Lemma 5 is completed. ■

The lower bound on the aggregate load L of an $(n, n, \ldots, n)$-allocation that we can obtain from Lemma 5 depends on how often we can apply the recursion ℓ until the x_0 parameter becomes smaller than $\sqrt{n}$. In the following we investigate how $x_0, \ldots, x_d$ behave when the recursion is applied iteratively. Set $x_0^{(0)} = \cdots = x_d^{(0)} = n$, and let $x_0^{(t)}, \ldots, x_d^{(t)}$ denote the values of $x_0, \ldots, x_d$, respectively, after the recursion ℓ has been applied t times.

Lemma 6 $x_0^{(t)} \geq n \cdot \exp(-c \cdot \phi_d^{\,t} \cdot \ln d)$, *for some suitable constant* $c > 0$.

Proof. We start by introducing another, simpler recursion, which we will relate to the original recursion afterwards. For $t \geq 0$, we define a vector $\delta^{(t)} = (\delta_0^{(t)}, \ldots, \delta_d^{(t)})$ of positive, real values as follows. We define $\delta^{(0)} = (0, \ldots, 0)$. For $t \geq 1$, we define

$$\begin{aligned} \delta_0^{(t)} &= 2 \cdot \delta_0^{(t-1)} - \delta_d^{(t-1)} + 1 \ \text{, and} \\ \delta_j^{(t)} &= \delta_{j-1}^{(t-1)} \ \text{, for } 1 \leq j \leq d. \end{aligned}$$

Notice that $\delta_0^{(t)} = 2 \cdot \delta_0^{(t-1)} - \delta_0^{(t-d-1)} + 1$, for $t \geq d+1$, which shows that δ is related to the Fibonacci numbers F_d since $F_d(t) = 2 \cdot F_d(t-1) - F_d(t-d-1)$, for $t \geq 3$. We can expect that these two recursions behave similarly. In fact, having a closer look to both of these recursion gives us

$$\delta_0^{(t)} = \sum_{i=0}^{t} \sum_{j=0}^{i} F_d(j) \leq \sum_{i=0}^{t} \sum_{j=0}^{i} \phi_d^{\,j-1} = O(\phi_d^{\,t}) ,$$

for any $t \geq 0$. In the following, we will use $\delta_0^{(t)}$ to bound $x_0^{(t)}$, that is, we show

$$x_0^{(t)} \geq n \cdot (10d)^{-2\delta_0^{(t)}} \geq n \cdot \exp(-c \cdot \phi_d^{\,t} \cdot \ln d) ,$$

for some suitable constant $c > 0$, which corresponds to the bound given in the lemma. In order to prove this relationship between δ and x, we need first to show some properties of δ.

Suppose instead of setting $\delta^{(0)} = (0, \ldots, 0)$ we define $\delta^{(0)} = \kappa = (\kappa_0, \ldots, \kappa_d)$, for some numbers $\kappa_0 \geq \kappa_1 \geq \ldots \geq \kappa_d \geq 0$. Then one can show by induction

$$\delta_0^{(t)} = \sum_{i=0}^{t} \sum_{j=0}^{i} F_d(j) + \kappa_0 \sum_{j=0}^{t+1} F_d(j) - \sum_{i=1}^{d} \kappa_i \sum_{j=0}^{t-d+i} F_d(j) ,$$

for any $t \geq 0$. Thus, $\delta_0^{(t)}$ is monotonically increasing in κ_0 and monotonically decreasing in $\kappa_1, \ldots, \kappa_d$. Furthermore, one can observe that exchanging some of the initial values so that $\kappa_i < \kappa_{i+1}$, for some $0 \leq i < d$, does not increase the value of $\delta_0^{(t)}$ for any $t \geq 0$. Another property of δ that can be shown by induction is that

$$\delta_0^{(t)} \geq \delta_1^{(t)} \geq \cdots \geq \delta_d^{(t)} ,$$

for any $t \geq 0$. Hence, any vector δ_t fulfills the property that we assumed for the initial vector κ. Therefore, exchanging some of the vector components $\delta_1^{(t)}, \ldots, \delta_d^{(t)}$ in the tth iteration of the recursion does not increase the value of $\delta_0^{(t')}$, for any $t' \geq t$.

Now consider the vectors $s^{(t)} = (s_0^{(t)}, \ldots, s_d^{(t)})$ defined by

$$s_i^{(t)} = \frac{\log(n) - \log(x_i^{(t)})}{2\log(10d)} ,$$

for any $t \geq 0$ and $0 \leq i \leq d$. Converting the recursion ℓ to the $s^{(t)}$ vectors, we derive the following recursive description of s. The initial vector is $s^{(0)} = (0, \ldots, 0)$, and, for $t \geq 1$,

$$\begin{aligned} s_0^{(t)} &= 2 \cdot s_0^{(t-1)} - s_{i(t)}^{(t-1)} + 1 \ , \\ s_{i(t)}^{(t)} &= s_0^{(t)} + 0.5 \ \text{, and} \\ s_j^{(t)} &= s_j^{(t-1)} \ \text{, for any } j \in \{1, .., d\} \setminus \{i(t)\}. \end{aligned}$$

Here $i(1), i(2), \ldots$ are some integers from $\{1, \ldots, d\}$ which are controlled by the minimum operator in recursion ℓ.

The recursion s differs from the recursion δ in only two aspects. First, the integers $i(1), i(2), \ldots$ determine which of the components $s_1^{(t)}, \ldots, s_d^{(t)}$ is subtracted from $s_0^{(t-1)}$ and then replaced by $s_0^{(t)}$. In comparison to the δ recursion this mechanism corresponds simply to permuting the components $s_1^{(t)}, \ldots, s_d^{(t)}$ in iteration t. We have seen above that this does not increase $s_0^{(t')}$ for any $t' > t$. Second, we add a positive value of 0.5 to one of the components $s_1^{(t)}, \ldots, s_d^{(t)}$ in each iteration. Because of the monotonicity property shown above, this does not increase $s_0^{(t')}$ for any $t' > t$, too.

Consequently, $s_0^{(t)} \leq \delta_0^{(t)}$, for any $t \geq 0$. Thus, we can conclude

$$\begin{aligned} x_0^{(t)} &= n \cdot (10d)^{-2s_0^{(t)}} \\ &\geq n \cdot (10d)^{-2\delta_0^{(t)}} \\ &\geq n \cdot \exp(-c \cdot \phi_d^{\,t} \cdot \ln d) \ , \end{aligned}$$

which completes the proof of Lemma 6. ■

Combining Lemma 5 and Lemma 6 we can derive a lower bound on the aggregate load L as follows. From Lemma 5 we can conclude that the aggregate load L of an $(n, \ldots, n)$-allocation is at least $\ell(n, \cdots, n)$. The value of $\ell(n, \cdots, n)$ is equal to the recursion depth. The recursion ends at the first iteration t with $x_0^{(t)} < \sqrt{n}$. From Lemma 6, we can conclude that $x_0^{(t)} \geq n \cdot \exp(-c \cdot \phi_d^{\,t} \cdot \ln d)$, for some suitable constant c. Hence, L satisfies $n \cdot \exp(-c \cdot \phi_d^{\,L} \cdot \ln d) < \sqrt{n}$. Solving this equation for L yields

$$L \ > \ \log_{\phi_d}\left(\frac{\ln n}{2c \cdot \ln d}\right) \ = \ \frac{\ln\ln n}{\ln \phi_d} - O(\ln\ln d) \ ,$$

which corresponds to the bound given in Lemma 4, and hence completes the proof of Theorem 2.

4. Acknowledgments

I would like to thank Artur Czumaj and Klaus Schröder for helpful discussions and Valerie King for suggesting the name of the algorithm, "Always-Go-Left".

References

[1] M. Adler, P. Berenbrink, and K. Schröder. Analyzing an infinite parallel job allocation process. In *Proc. of the 6th European Symposium on Algorithms (ESA)*, pages 417–428, 1998.

[2] Y. Azar, A. Broder, A. Karlin, and E. Upfal. Balanced allocations. In *Proc. of the 26th ACM Symp. on Theory of Computing (STOC)*, pages 593–602, 1994.

[3] R. Cole, B. M. Maggs, F. Meyer auf der Heide, M. Mitzenmacher, A. W. Richa, K. Schröder, R. K. Sitaraman, and B. Vöcking. Randomized protocols for low-congestion circuit routing in multistage interconnection networks. In *Proc. of the 30th ACM Symp. on Theory of Computing (STOC)*, pages 378–388, 1998.

[4] R. J. Cole, A. Frieze, B. M. Maggs, M. Mitzenmacher, A. W. Richa, R. K. Sitaraman, and E. Upfal. On balls and bins with deletions. In *Proc. of the RANDOM'98*, 1998.

[5] R. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. In *Proc. of the 24th ACM Symp. on Theory of Computing (STOC)*, pages 318–326, 1992.

[6] D. E. Knuth. *The Art of Computer Programming, Volume 3*. Addison–Wesley, 1998.

[7] F. Meyer auf der Heide, C. Scheideler, and V. Stemann. Exploiting storage redundancy to speed up randomized shared memory simulations. In *Proc. of the 12th Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 267–278, 1995.

[8] F. Meyer auf der Heide, K. Schröder, and F. Schwarze. Routing on networks of optical crossbars. In *Proc. of the Euro-Par'96*, pages 299–306, 1996.

[9] M. Mitzenmacher. Load balancing and density dependent jump Markov processes. In *Proc. of the 37th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 213–222, 1996.

[10] M. Mitzenmacher. *The Power of Two Choices in Randomized Load Balancing*. PhD thesis, University of California at Berkeley, 1996.