How Asymmetry Helps Load Balancing

BERTHOLD VÖCKING

Universität Dortmund, Dortmund, Germany

Abstract. This article deals with randomized allocation processes placing sequentially n balls into n bins. We consider multiple-choice algorithms that choose d locations (bins) for each ball at random, inspect the content of these locations, and then place the ball into one of them, for example, in a location with minimum number of balls. The goal is to achieve a good load balancing. This objective is measured in terms of the maximum load, that is, the maximum number of balls in the same bin.

Multiple-choice algorithms have been studied extensively in the past. Previous analyses typically assume that the d locations for each ball are drawn uniformly and independently from the set of all bins. We investigate whether a nonuniform or dependent selection of the d locations of a ball may lead to a better load balancing. Three types of selection, resulting in three classes of algorithms, are distinguished: (1) uniform and independent, (2) nonuniform and independent, and (3) nonuniform and dependent.

Our first result shows that the well-studied uniform greedy algorithm (class 1) does not obtain the smallest possible maximum load. In particular, we introduce a nonuniform algorithm (class 2) that obtains a better load balancing. Surprisingly, this algorithm uses an unfair tie-breaking mechanism, called Always-Go-Left, resulting in an asymmetric assignment of the balls to the bins. Our second result is a lower bound showing that a dependent allocation (class 3) cannot yield significant further improvement.

Our upper and lower bounds on the maximum load are tight up to additive constants, proving that the Always-Go-Left algorithm achieves an almost optimal load balancing among all sequential multiple-choice algorithm. Furthermore, we show that the results for the Always-Go-Left algorithm can be generalized to allocation processes with more balls than bins and even to infinite processes in which balls are inserted and deleted by an oblivious adversary.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; G.3 [**Probability and Statistics**]: *probabilistic algorithms (including Monte Carlo)*

General Terms: Algorithms

Additional Key Words and Phrases: Randomized algorithms, probabilistic analysis, balls and bins processes

---

A preliminary version of this article appeared as VÖCKING, B. 1999. How asymmetry helps load balancing. In *Proceedings of 40th IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, Los Alamitos, Calif., pp. 131–140.

Author's address: Department of Computer Science, Universität Dortmund, Baroper Str. 301, 44221 Dortmund, Germany, e-mail: voecking@markov.cs.uni-dortmund.de.

1. *Introduction*

We investigate balls-and-bins processes related to load balancing, resource allocation, and hashing. Suppose n balls should be placed one after the other into n bins in such a way that the allocation is as even as possible. In particular, we aim at minimizing the *maximum load*, that is, the maximum number of balls that are placed into the same bin. Clearly, if the balls are numbered from 1 to n, then a round-robin allocation yields the optimal allocation with maximum load 1. For several applications, however, such an allocation is practically not possible, for example, when balls are indistinguishable and global knowledge about previously assigned balls is not available. A simple, efficient way to obtain a good load balancing in such cases is to allocate the balls at random. For example, each ball is placed into a bin chosen independently and uniformly at random from the set of all bins. One of the classical results in probability theory is that this simple randomized allocation process terminates with an expected maximum load of $(1 + o(1))\frac{\ln n}{\ln \ln n}$.

Azar et al. [1994, 1999] suggest the following variation of the simple randomized allocation. Their algorithm, which we call *uniform greedy algorithm*, works as follows. For each ball, choose $d \geq 2$ *locations* (i.e., bins) independently and uniformly at random from the set of bins and place the ball into a bin with fewest balls among them, breaking ties arbitrarily. Azar et al. [1994, 1999] show that this algorithm produces a maximum load of only $\frac{\ln \ln n}{\ln d} \pm \Theta(1)$, w.h.p.[1] Thus, having just two random choices yields a great improvement over one choice, while having three choices or more is only a small factor better than two choices. This simple idea of using more than one random choice has been studied extensively in several variations, for example, parallel allocation [Adler et al. 1995; Stemann 1996], infinite sequences of insertions and deletions [Azar et al. 1999; Cole et al. 1998a, 1998b], multiple-choice queueing systems [Mitzenmacher 1996a, 1996b; Vvedenskaya et al. 1996], or balls of different weight [Berenbrink et al. 1999]; and the applicability of this fundamental concept is shown in different studies dealing, for example, with PRAM simulations [Czumaj et al. 2000; Dietzfelbinger and Meyer auf der Heide 1993; Karp et al. 1992; Meyer auf der Heide et al. 1996a] or routing in interconnection networks [Cole et al. 1998a; Luczak and Upfal 1999; Meyer auf der Heide et al. 1996b]. An extensive survey of techniques and results is given in Mitzenmacher et al. [2001].

In this article, we consider variations of the multiple-choice scheme that choose the locations in a different way than the uniform greedy algorithm. In particular, we investigate what happens to the maximum load when the d locations for the same ball can be chosen in a nonuniform and possibly dependent fashion. For example, let us assume $d = 2$ and n is even. A nonuniform algorithm may choose the first location from the bins 0 to $\frac{n}{2} - 1$ and the second location from the bins $\frac{n}{2}$ to $n - 1$. Furthermore, the second choice may depend on the first choice, for example, if the first location is i then the second location is $\frac{n}{2} + \left(i^2 \bmod \frac{n}{2}\right)$. Our goal is to improve on the results known for the uniform greedy process. On the first view, this idea may seem strange as the uniform probability distribution seems to be the natural choice to obtain a good load balancing. In fact, for the single-choice

[1] The term "w.h.p." abbreviates "with high probability", that is, for every fixed $\alpha > 0$, the specified bound holds with probability at least $1 - n^{-\alpha}$.

algorithm (i.e., $d = 1$), this intuition is right. Here any nonuniform selection of the bins leads to an uneven distribution of the balls, resulting in a higher rather than a lower maximum load. We will see, however, that this intuition is deceptive when considering multiple-choice algorithms.

1.1. THREE CLASSES OF ALGORITHMS. A *sequential multiple-choice algorithm* assigns one ball after the other into one out of $d \geq 2$ possible locations that are drawn at random according to the following rules. We assume that the balls are indistinguishable so that the sample of d locations is drawn according to the same probability distribution for each ball and this sample is chosen independently from the locations of other balls. Let $[n] = \{0 \dots n-1\}$ denote the set of bins. Then we obtain the following three classes of algorithms, depending on how the sample of locations is chosen from the probability space $[n]^d$.

—*Class* 1*: Uniform and independent.* Each of the d locations of a ball is chosen uniformly and independently at random from $[n]$.

—*Class* 2*: (Possibly) nonuniform and independent.* For $1 \leq i \leq d$, the ith location of a ball is chosen independently at random from $[n]$ as defined by a probability distribution $D_i : [n] \to [0, 1]$.

—*Class* 3*: (Possibly) nonuniform and (possibly) dependent.* The d locations of a ball are chosen at random from the set $[n]^d$ as defined by a probability distribution $D : [n]^d \to [0, 1]$.

In general, 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 previously placed balls, we make different assumptions for upper and lower bounds. Our algorithms place a ball depending only on the number of balls found in the d locations of that ball. For our lower bound, however, we allow algorithms exploiting knowledge about all previous assignments. In other words, the algorithms can place each ball in one of its d locations using full knowledge about the history of previous allocations. As one result of our analyses, we will see that this additional knowledge does not help very much.

1.2. OUR CONTRIBUTION. We present a multiple-choice algorithm of class 2 that produces a significantly smaller maximum load than the uniform greedy process (class 1). Our algorithm partitions the bins into d groups of almost equal size, that is, each group has size $\Theta(\frac{n}{d})$. For each ball, the algorithm chooses one location from each group. The ith location ($1 \leq i \leq d$) of each ball is chosen uniformly and independently at random from the ith group, and the ball is inserted into a bin with minimum load among these d locations. In particular, if there are several locations with same minimum load then the ball is inserted into the *leftmost* among them, that is, in the bin that belongs to the group with smallest index. Because of this asymmetric tie breaking mechanism our algorithm is called *Always-Go-Left*.

The following theorem contains the core of our analysis. The maximum load of Always-Go-Left is described in terms related to the Fibonacci numbers. We use d-ary Fibonacci numbers that are defined as follows. For $k \leq 0$, $F_d(k) = 0$, $F_d(1) = 1$, and, for $k \geq 1$,

$$F_d(k) = \sum_{i=1}^{d} F_d(k-i).$$

Observe that the sequence F_2 corresponds to the standard Fibonacci sequence. Define $\phi_d = \lim_{k\to\infty} \sqrt[k]{F_d(k)}$. Then, ϕ_2 corresponds to the so-called golden ratio (see, e.g., Knuth [1998]) and ϕ_d is a generalization. For example, $\phi_2 = 1.61\ldots$, $\phi_3 = 1.83\ldots$, and $\phi_4 = 1.92\ldots$. In general, $\phi_2 < \phi_3 < \phi_4 < \cdots < 2$ and $\lim_{d\to\infty} \phi_d = 2$.

THEOREM 1. *Suppose that n balls are placed sequentially into n bins using the Always-Go-Left algorithm* (*class* 2)*. Then the number of balls in the fullest bin is*

$$\frac{\ln\ln n}{d \ln \phi_d} + O(1),$$

w.h.p.

Notice that the influence of the number of random choices, d, on the maximum load is linear rather than only logarithmic as in the case of the uniform greedy algorithm. In fact, our result generally (also for small d) improves upon the one for the uniform greedy algorithm because $\phi_d > 2^{(d-1)/d}$ so that

$$\frac{\ln\ln n}{d \ln \phi_d} \;<\; \frac{\ln\ln n}{(d-1)\ln 2} \;<\; \frac{\ln\ln n}{\ln d}.$$

Even for the case $d = 2$, we obtain a significant improvement. Here the Always-Go-Left algorithm yields maximum load of $0.69\ldots \log_2 \ln n + O(1)$ instead of $\log_2 \ln n \pm \Theta(1)$.

Let us compare these results with a lower bound that Azar et al. [1994, 1999] have shown for the uniform greedy scheme. They prove that the uniform greedy scheme is "majorized" by all other uniform schemes, i.e., the uniform greedy algorithm achieves the best load balancing among all algorithms of class 1. Interestingly, this results holds regardless of the used tie-breaking mechanism, showing that the tie-breaking mechanism is irrelevant in the uniform case. It is not difficult to extend the techniques of Azar et al. [1994, 1999] to show that partitioning the bins and using a fair tie breaking does not reduce the number of balls in the fullest bin below $\frac{\ln\ln n}{\ln d}$. Thus, the combination of partitioning and unfair tie breaking is crucial for our result.

Once we have seen that choosing the locations in a nonuniform way combined with an asymmetric tie breaking mechanism reduces the maximum load, it is an interesting question whether other kinds of choices for the d locations or other schemes for deciding which of these locations receives the ball can improve on this result even further. 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 that chooses d bins for each ball at random according to an arbitrary probability distribution on* $[n]^d$ (*class* 3)*. Then the number of balls in the fullest bin is*

$$\frac{\ln\ln n}{d \ln \phi_d} - O(1),$$

w.h.p.

Combining Theorems 1 and 2, we conclude that, apart from some additive constants, the Always-Go-Left algorithm achieves the best possible maximum load

among all sequential multiple-choice algorithms, namely

$$\frac{\ln \ln n}{d \ln \phi_d} \pm \Theta(1).$$

Thus partioning the bins into groups and using an unfair tie breaking mechanism improves the load balancing considerably but other tricks cannot lead to further significant improvements.

1.3. 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 apects. In particular, we assume the following on-line allocation model.

An oblivious adversary specifies a possibly infinite sequence $\sigma = \sigma_1\sigma_2 \cdots$ of *requests* representing insertions and deletions of balls. Each ball comes with d randomly drawn locations into one of which it has to be inserted by an allocation algorithm. The term *oblivious* means that the specified sequence is specified in advance, that is, it is independent from the random choices of the locations and other decisions made by the allocation algorithm. For example, we can imagine an adversary that inserts keys into a hash table and later removes some of the keys. For each key there are d random hash functions and the allocation algorithm has to assign the key to one of the d table positions specified by the hash functions. Having this application in mind, the adversary can insert the same ball (key) several times and the ball always comes with the same set of randomly drawn locations into one of which it has to be inserted by the allocation algorithm, but not necessarily always into the same location. We assume that insertions and deletions alternate so that at any point of time each ball exists at most once in the table.

The allocation algorithm has to *serve* all requests *on-line*, that is, the sequence of insertions and deletions of balls is presented one by one and, for each request σ_t, the corresponding ball has to be inserted into one of its locations or deleted, respectively, without knowing *future requests*, that is, $\sigma_{t+1}\sigma_{t+2} \ldots$. Let *time t* denote the point of time at which request σ_t is presented but not yet served. A ball is said to *exist at time t* if it is stored in one of the bins at this time. For this on-line model, we can prove the following generalized version of Theorem 1.

THEOREM 3. *Consider any sequence of insertions and deletions such that at most hn balls exist at any point of time. Suppose the balls are assigned to the bins by algorithm Always-Go-Left with parameter d. At any fixed time t, the maximum load is at most* $\frac{\ln \ln n}{d \ln \phi_d} + O(h)$, *w.h.p.*

Assuming only hn insertions and no deletions, our model degenerates to the static problem of allocating hn balls into n bins. The corresponding best previously known upper bound for the uniform allocation process is $\log_d \ln n + O(h)$, w.h.p., given by Azar et al. [1999].

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. [1999]. They show an upper bound on the maximum load of the uniform greedy process of $\log_d \ln n + O(1)$, w.h.p., for any fixed time $t > n^3$. Applying our proof techniques to the uniform greedy process, we obtain the following results for possibly infinite sequences of insertions and deletions.

THEOREM 4. *Consider any sequence of insertions and deletions such that at most hn balls exist at any point of time. Suppose the balls are assigned to the bins by the uniform greedy algorithm with parameter d. At any fixed time t, the maximum load is at most* $\log_d \ln n + O(h)$, *w.h.p.*

1.4. RECENT IMPROVEMENTS. Comparing the above results for the Always-Go-Left scheme with those of the uniform greedy strategy, one can observe that the Always-Go-Left scheme achieves a smaller maximum load if n is sufficiently large. This leaves open the questions of how these two algorithms compare for relatively small n and how they compare with respect to other measures of load balance. In a recent work, with Berenbrink et al. [2000] we compare the load vector obtained by the Always-Go-Left scheme with the vector of the uniform greedy algorithm. In this comparative study, it is shown that Always-Go-Left is "majorized" by the uniform greedy algorithm for any fixed parameter d. In other words, the Always-Go-Left schemes achieves the better load balancing with respect to all prefixes of the load vector rather than only the maximum load and also with respect to all stochastic moments. This result holds for any number of balls and any number of bins that is a multiple of d.

Furthermore, Berenbrink et al. [2000], give tighter bounds on the maximum load for the case when there are more balls than bins. Suppose hn balls are placed into n bins choosing d alternative locations for each ball. Then the uniform greedy process produces a maximum load of $\log_d \ln n + h \pm \Theta(1)$ and the Always-Go-Left algorithm terminates with $\frac{\ln \ln n}{d \ln \phi_d} + h \pm \Theta(1)$ balls in the fullest bin, w.h.p. These results show that the multiple-choice processes are fundamentally different from the single-choice variant because the gap between maximum and average load produced by the multiple-choice processes does not increase with the number of balls but is bounded above by functions depending only on n and d.

1.5. OUTLINE. The rest of the article is organized as follows. In Section 2, we prove the upper bounds given in Theorems 1, 3 and 4. We start by analyzing the infinite symmetric scheme, and then we show how the analysis changes when considering the asymmetric scheme. In both cases, we assume initially that the number of balls that can exist at any point of time is upper-bounded by n. This simplifying assumption is removed at the end of the analysis.

In Section 3, we give the proof for the lower bound given in Theorem 2. In Section 4, we present some experimental results comparing the maximum load obtained by the Always-Go-Left algorithm with the one produced by the uniform greedy process. Finally, Section 5 provides a short discussion of our results.

2. *Proof of the Upper Bounds*

In this section, we prove the upper bounds given in the Theorems 1, 3, and 4. We assume an infinite sequence of insertions and deletions of balls specified by an oblivious adversary. An insertion corresponds to a ball that is assigned to one out of $d \geq 2$ bins. A deletion specifies one of the previously inserted balls that is removed from its bin. There are n bins, and the number of balls that exist at the same time is at most $h \cdot n$. We investigate the maximum load produced by the symmetric, uniform and the asymmetric, nonuniform allocation process.

In both the symmetric and the asymmetric case, we use the same kind of analysis, that is, we use a witness tree to upper-bound the probability for the bad event that a bin contains too many balls. This witness tree is a rooted tree the nodes of which represent balls whose randomly chosen locations are arranged in a bad fashion. This proof technique was introduced by Meyer auf der Heide et al. [1996a]. Our construction, however, is slightly different. In contrast to previous work using this technique, for example, Cole et al. [1998a, 1998b] and Meyer auf der Heide et al. [1996a], our analysis does not lose a constant factor in the maximum load but is exact up to a constant additive term. In Section 2.1, we first describe our construction of a witness tree for the symmetric allocation scheme. In Section 2.2, we then show how this construction changes in the asymmetric case.

Initially, we make some simplifying assumptions. The first assumption is that all events represented by a witness tree are stochastically independent. In Section 2.3, we will remove this simplifying assumption and give a joint solution dealing with dependencies in both the symmetric and the asymmetric case. The second initial assumption is that at most n balls exist at any time, that is, $h = 1$. Finally, in Section 2.4 we will extend our proofs to general h.

2.1. THE WITNESS TREE FOR THE SYMMETRIC SCHEME. An "activated witness tree" is a graph structure that can be constructed when a *bad event* occurs, that is, when the maximum load exceeds some threshold value that will be specified later. In other words, a bad event implies the "activation of a witness tree" so that the probability for the existence of an activated witness tree upper-bounds the probability that a bad event occurs. In the following, we show that the activation of a witness tree is unlikely. Consequently, the bad event that is witnessed by this structure is unlikely as well.

2.1.1. *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 same ball may be represented by several nodes in the tree, but not every assignment of balls to nodes gives a witness tree. The assignment has to fulfill the following time constraints. Consider a fixed sequence of insertions and deletions and let t denote the time step specified in the theorem. The ball represented by the root node r has to exist at time t and, furthermore, each ball represented by a node $v \neq r$ with parent node u has to exist at the insertion time of u's ball. In this way, each node of the witness tree describes an event that may occur or not, depending on the outcome of 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 describes 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 describes the event that each of the d locations of v's ball points to a bin that, at the ball's insertion time, contains at least three *additional balls*, that is, balls that are not represented by the nodes of the tree.

Observe, that edge events are defined in terms of the alternative locations of balls instead of their final resting places. This is a crucial trick to avoid dependencies in the following analysis. The definition for leaf events is slightly different in this respect as it refers to bins containing some number of balls at some specified time.

Here, we will avoid dependencies by making worst-case assumpions about the assignments of the balls to the bins.

The "activation of a witness tree" is defined as follows. We say that *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 edge or leaf event, respectively, occurs, and we say that *the witness tree is activated* if all of its edges and all its leaf nodes are activated.

2.1.2. *Construction of a Symmetric Witness Tree.* We want to use an activated witness tree of order L to witness the bad event that some of the bins contain more than $L + 3$ balls at time t. Therefore, we have to show that the existence of a bin with more than $L + 3$ balls implies the existence of an activated witness tree of order L.

Assume the allocation process is determined up to time t, that is, consider a snapshot of the allocation of balls to bins at time t. Suppose that a bin x holds at least $L + 4$ balls at this time. Then, an activated witness tree of order L can be constructed as follows: The root gets assigned the *topmost ball* in bin x, that is, the ball that was inserted last into this bin. 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. Now consider these child nodes and continue the assignment recursively until we reach the leaf nodes. This construction ensures that all edge events are activated. Observe that the root's ball is placed on top of at least $L + 3$ balls, the balls assigned to the children of the root are placed on at least $L + 2$ balls, and so on. Thus, 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. Thus, all leaf nodes are activated as well so that the existence of an activated witness tree is guaranteed.

2.1.3. *Probability for the Activation of a Symmetric Witness Tree.* The probability that the maximum load is larger than $L+3$ at some given time t is upper-bounded by the probability that a witness tree of order L is activated. In the following, we calculate a bound on the the latter probability. It is important to note that, during this calculation, we do not refer to the construction of the witness tree but only to the structural properties described in the formal definition of the witness tree. We want to 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 activated is bounded from above by the number of different witness trees multiplied by an upper bound on the probability that any particular witness tree is activated. Initially, we account only for those witness trees whose nodes represent distinct balls, that is, we assume that each ball occurs at most once in the witness tree. Of course, this is a major simplification, and the existence of redundant balls has to be taken into account in a full analysis. This simplification, however, will allow us to explain the differences between the symmetric and the asymmetric process more clearly. Later, in Section 2.3, we will show how to deal with witness trees that may draw to the same ball several times.

We start by counting the number of different witness trees, that is, the number of ways to assign balls to the nodes of the tree. The number of possibilities to choose the root's ball is at most n because, by definition, this ball has to exist at time t and the number of existing balls at any time is at most n. Furthermore, the ball

represented by the root gives us the point of time at which the balls corresponding to the children of the root exist. Hence, the number of possibilities to assign a ball to a child is upper-bounded by n, too. Applying this argument level by level to all nodes in the tree yields that the total number of possibilities to assign balls to all nodes in the tree 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 for the activation of an edge (u, v) with v being the ith child of u is at most d/n because the probability that the ith location of u's ball points to any particular one of the d locations of v's ball is at most $\frac{1}{n}$. Because of our assumption that all balls are distinct, the events for all edges in the tree are independent, and hence the probability that all of the $m-1$ edges are activated simultaneously is at most $(\frac{d}{n})^{m-1}$. Furthermore, the probability for a leaf event is at most 3^{-d} because each of the d locations of the ball corresponding to the respective leaf node has to point to a bin that contains three additional balls and at any time the number of these bins is upper-bounded by $n/3$. In particular, 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.

Putting all together, we can conclude that the probability that there exists an activated symmetric witness tree with distinct balls at any fixed time t is at most

$$n^m \cdot \left(\frac{d}{n}\right)^{m-1} \cdot 3^{-d \cdot q}.$$

Applying $m \leq 2q$, $2d^2 \leq 3^d$ and $q = d^L$, this upper bound becomes

$$n \cdot d^{2q} \cdot 3^{-d \cdot q} \;\leq\; n \cdot 2^{-q} \;=\; n \cdot 2^{-d^L}.$$

Consequently, if we assume

$$L \geq \log_d \log_2 n + \log_d(1+\alpha),$$

the probability for the existence of a witness tree with distinct balls is at most $n^{-\alpha}$, for any constant $\alpha > 0$.

In Section 2.3, we extend this argument to witness trees with nondistinct 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 tree that we construct for the Always-Go-Left algorithm is asymmetric. We see that this asymmetric tree is much larger than its symmetric counterpart, resulting in a smaller probability for its activation.

2.2.1. *Definition of an Asymmetric Witness Tree.* An *asymmetric witness tree* is defined similarly to its symmetric counterpart. The only difference is in the topology. An asymmetric witness tree has the topology of a Fibonacci tree, which is defined recursively as follows. $T_d(1)$ as well as $T_d(2)$ consist only of a single node. $T_d(k)$, for $3 \leq k \leq d$, 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.

2.2.2. *Construction of an Asymmetric Witness Tree.* Suppose there is a bin containing $L + 4$ balls at time t. In this case, we can construct an activated, 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. A node with label (ℓ, i) is the root of a Fibonacci tree $T_d(d\ell + i)$. In particular, the label of the root of the witness tree 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. A node with label $(0, i)$, for $i \geq 3$, has only $i-1$ children having the labels $(0, i-1), \ldots, (0, 1)$. Nodes with labels $(0, 1)$ and $(0, 2)$ are leaf nodes and hence do not have children. In this way, the topolgy of the witness tree corresponds to $T_d(dL + 1)$. In the construction of this tree, we assign balls to the nodes maintaining the following invariant.

INVARIANT 5. *A ball b represented by a node with label (ℓ, i) found its resting place in a bin containing 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 the bin with $L + 4$ 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 (in group i) is assigned to that child of the root 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 in the jth instead of the ith 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 Invariant 5.

2.2.3. *Probability for the Activation of an Asymmetric Witness Tree.* As in the symmetric case, the number of different asymmetric witness trees is n^m, the probability that all edge events occur is at most $(\frac{d}{n})^{m-1}$, and the probability that all leaf events occur is $3^{-d \cdot q}$ with m denoting the number of nodes and q denoting the number of leaves in the tree. The first two bounds follow analogously to the symmetric case.[2] The bound on the probability for the leaf events may need some further explanations. As in the symmetric case, at most $\frac{n}{3}$ bins contain three or more balls at any time. We do not know, however, 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

[2] 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 even of constant factors in the group sizes.

bins is $\prod_{i=1}^{d} \beta_i$. Observe that the β_i variables fulfill the constraint $\sum_{i=1}^{d} \beta_i \leq \frac{d}{3}$ because all groups have size $\frac{n}{d}$ so that $\sum_{i=1}^{d} \beta_i \frac{n}{d} \leq \frac{n}{3}$. Now, as the product of a set of positive variables with fixed sum is maximized when the values of all variables are equal, the probability for a leaf event is $\prod_{i=1}^{d} \beta_i \leq 3^{-d}$.

Combining the bounds above yields that the probability for the activation of an asymmetric witness tree is at most

$$n^m \cdot \left(\frac{d}{n}\right)^{m-1} \cdot 3^{-d \cdot q} \quad \leq \quad n \cdot 2^{-q}.$$

This bound is completely analogous to the symmetric case. The difference, however, is that the number of leaf nodes, q, is much larger now. Applying $q \geq \phi_d^{d \cdot L - 1}$, we obtain that the probability for an activated asymmetric witness tree with distinct balls is at most

$$n \cdot 2^{-\phi_d^{d \cdot L - 1}}.$$

If we choose

$$L \;=\; \left\lceil \frac{\ln \log_2 n + \ln(1+\alpha)}{d \cdot \ln \phi_d} \right\rceil + 1 \;=\; \frac{\ln \ln n}{d \ln \phi_d} + O(1),$$

this probability is at most $n^{-\alpha}$, for any constant $\alpha > 0$.

2.3. WITNESS TREES WITH NONDISTINCT BALLS. Until now, we estimated only the probability for the activation of witness trees with distinct balls. In order to get a proper upper bound on the probability for a too large maximum load, we have 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 in such a way 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 showing that the occurence of many pruning events is unlikely.

2.3.1. *Definition of a Full Witness Tree.* A *full witness tree* of order L includes κ symmetric or asymmetric witness trees, respectively, with $\kappa \geq 2$ denoting a suitable constant. Each of these κ witness trees is a subtree of the full witness tree and has order L. 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 one of the κ witness trees of order L mentioned above. 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 time constraints and the activation of a full witness tree are defined similar to the definitions for the symmetric and asymmetric witness trees and become clear during the following description implicitly.

2.3.2. *Construction of a Full Witness Tree.* The full witness tree is constructed as follows: Suppose some bin x holds $L + 4 + \kappa$ balls at time t. Then, we assign the κ topmost balls in x to the children of the root. The root itself does not get assigned a ball. 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. If there are more than

one location pointing to x, fix one arbitrarily. Let i denote the index of this location. Now consider location $i + 1 \bmod d$. At b's insertion time, this location points to a bin with at least $L + 3$ balls. The topmost ball in this bin is assigned to the child of v, which is a grandchild of the root. Below this grandchild the construction is continued following the rules for the symmetric or asymmetric witness tree of order L, respectively.

2.3.3. *Pruning the Full Witness Tree.* In order to remove dependencies we inspect the nonroot 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 cut off the complete subtree rooted at v. 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. 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 nonpruned nodes. Only the visited, nonpruned nodes and the cutoff edges are defined to build the *pruned witness tree*. In this way, our witness structure does not contain redundant balls but it might have become quite small.

2.3.4. *Probability for the Activation of a Pruned Witness Tree.* We distinguish two cases. In the first case, we assume that there fewer than κ cutoff edges. Clearly, this implies that all balls in one of the κ symmetric or asymmetric witness trees contained in the full tree must be distinct. We have bounded above the probability for the existence of such a witness tree in the previous sections. It is at most $n^{-\alpha}$, for any constant α, provided that L is chosen sufficiently large.

In the second case, it remains to estimate the probability for the existence of an activated pruned witness tree with κ cutoff edges. In the following, we upper-bound this probability under the assumption that the number of balls represented by the full witness tree (which corresponds to the number of non-root nodes) is at most $M = 2\kappa(\alpha + 1) \cdot \log_2 n$, which is the number of balls that is needed for the calculations in Case 1. Observe that the number of different shapes for the pruned witness tree is at most M^κ, that is, there are at most M^κ different ways to convert a full witness tree into a pruned witness trees.

Now fix the shape of the tree. Let m denote the number of balls represented by the pruned tree, and let q denote the number of leaf nodes. Then the number of possibilities to choose the balls represented by the tree is at most n^m. Let us assume that the root of the witness tree represents the bin in which the balls represented by the root's children met, and the edge connecting the root with a child represent the event that a location of the child's ball point to this bin. Then there are n ways to choose the bin represented by the root, and once this bin is fixed, the probability that all m edge events are activated is $(\frac{d}{n})^m$. Furthermore, the probability that all q leaf events 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 nonpruned 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, it is guaranteed to

be part of the pruned tree. The cutoff edge specifies which of the randomly chosen locations of b hits one of the locations of b'. The number of possibilities to select u' and thus b' is bounded from above by $m \leq M$. The probability that the specified location of b points to the same bin as one of the locations of the ball b' is at most $\frac{d}{n}$. Thus, the probability that there are κ cutoff events is at most $(M\frac{d}{n})^{\kappa}$.

Notice that all edge, leaf, or cutoff events are essentially independent since the pruning has removed redundant balls. In particular, the upper bound on the probability for any one of these events holds with respect to one particular randomly selected location that is not considered for any other event represented by the pruned tree. Because of this independence, we can conclude that the probability for the existence of a pruned witness tree with κ cutoff edges is at most

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

because $m \leq 2q$, $d^2 \leq 3^d$, and $M \leq 2\kappa\cdot(\alpha+1)\cdot\log_2 n)$. This completes the proof of the Theorems 1, 3, and 4, for $h = 1$.

2.4. MORE BALLS THAN BINS. Finally, let us remove the simplifying assumption that at most n balls can exist at the same time. In the following analysis we investigate what happens if up to hn balls, for general $h > 0$, exist at the same time. The definitions for the symmetric and asymmetric witness trees are changed as follows: Each leaf node now represents h balls (instead of only one) and, in addition, it is associated with one of the bins. Each of the h balls has a location pointing to this bin. During the construction of the witness tree, we find these balls placed in the specified bin. Furthermore, we demand that the other locations of these balls point to bins that contain at least βh balls, for suitably chosen constant $\beta > 1$. In this way, an activated witness tree of order L witnesses the bad event that there exists a bin with $L + \Theta(h)$ balls at some given time t.

Taking into account these modifications, the probability for the existence of an activated witness tree with distinct balls can be bounded as follows. Let m denote the number of nodes and q the number of leaves in the tree. Then, there are at most n^q possibilities to choose the q bins associated with the leaf nodes, $\binom{hn}{h}^q$ possibilities to choose the q times h balls corresponding to these bins, and at most $(hn)^{m-q}$ possibilities to choose the $m-q$ balls associated with the internal nodes. Thus, the number of different witness trees is at most

$$n^q \binom{hn}{h}^q (hn)^{m-q} \ \leq\ \left((en)^h n\right)^q (hn)^{m-q}$$

because $\binom{a}{b} \leq (\frac{ae}{b})^b$. Furthermore, the probability for activating any particular witness tree is

$$\left(\frac{d}{n}\right)^{m-1}\left(\frac{d}{n}\,\beta^{-(d-1)}\right)^{hq} \;\leq\; \left(\frac{d}{n}\right)^{m+hq-1}\beta^{-(d-1)hq}.$$

This can be seen as follows: There are $m-1$ edge events and the probability for the activation of any particular edge event is bounded above by $\frac{d}{n}$. Observe that these events do not influence the outcome of any location of a ball associated with the leaf nodes. The probability that one of the locations of a ball associated with a leaf node falls into the bin corresponding to that leaf node is at most $\frac{d}{n}$. Furthermore, the probability that the other $d-1$ locations fall into bins with βn balls is bounded above by $\beta^{-(d-1)}$. Taking into account that we have h balls associated with each of the q leaf nodes, we obtain the bound above. Consequently, the probability for the existence of a witness tree without redundant balls is at most

$$((\mathrm{e}n)^h n)^q (hn)^{m-q}\cdot\left(\frac{d}{n}\right)^{m+hq-1}\beta^{-(d-1)hq} \;\leq\; n\cdot\left(\mathrm{e}^h h d^{h+2}\beta^{-(d-1)h}\right)^q,$$

applying $m-q \leq 2q-q \leq q$. This term, however, is bounded above by $n\cdot 2^{-q}$, for some sufficiently large, constant β. Now observe that this is exactly the bound that we obtained in the analysis for the symmetric and the asymmetric witness trees with $h=1$, respectively, and hence the rest of the proof can proceed 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 $\frac{\ln\ln n}{d\ln\phi_d} - O(1)$, w.h.p.

The major challenge in proving this lower bound is to control the possibly dependent random choices for the d locations of a ball. As a first step to separate the effects of different locations, let us assume that the allocation algorithm has available d disjoint groups of n bins each instead of only n bins in total. The ith location, however, has to be chosen from the ith group. We claim that any algorithm $\mathcal{A}$ in the original model (with a total number of n bins) can be simulated by an algorithm $\mathcal{B}$ in the new model (with dn bins) in such a way that the maximum load of $\mathcal{B}$ is not larger than the maximum load of $\mathcal{A}$. Observe that the algorithms investigated in the previous section, use only local knowledge when inserting a ball into one of its locations. In fact, the decision where to place the ball depends only on the number of balls found in the d locations chosen for this ball. In such a distributed setting, algorithm $\mathcal{B}$ could have a disadvantage against $\mathcal{A}$ as it cannot see all the balls seen by $\mathcal{A}$. For this reason, we need to investigate the algorithms in a more general model assuming full knowledge about the distribution of previously inserted balls. Under this assumption, the simulation of $\mathcal{A}$ by $\mathcal{B}$ is straightforward. Thus, any lower bound on the number of balls in the fullest bin in the model with dn bins holds for the original model with n bins as well.

Let m and $n_1,\ldots,n_d$ denote some positive numbers. We define an $(m,n_1,\ldots,n_d)$-*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 disjoint groups $N_1, \ldots, N_d$ with $|N_i| \leq n_i$, for $1 \leq i \leq d$.

—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 resting places of previously inserted balls might be known, however.

The *maximum load* L^* of an $(m, n_1, \ldots, n_d)$-allocation is defined to be the number of balls in the fullest bin. The *aggregated 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 aggregated load of an allocation with (at least) n balls to (at most) dn bins partitioned into d groups of size (at most) n bins each.

LEMMA 1. *For any* $(n, n, \ldots, n)$*-allocation scheme,* $L = \dfrac{\ln \ln n}{\ln \phi_d} - O(\ln \ln d)$, *w.h.p.*

The lemma implies the lower bound given in Theorem 2 because $L^* \geq \frac{L}{d}$ so that $L^* = \frac{\ln \ln n}{d \ln \phi_d} - O(1)$, w.h.p. In the rest of this section, we deal with the proof of this lemma.

During the following analysis, the variables $m, n_1, \ldots, n_d$ represent positive numbers. The variable m will typically represent a lower bound on the number of balls, and the variables $n_1, \ldots, n_d$ will represent upper bounds on sizes of groups of bins. To avoid rounding issues, these numbers need not to be integers but there domains are positive reals. We assume $m \leq n_1, \ldots, n_d \leq n$, where n corresponds to the total number of bins in the system that we want to analyze originally. Without loss of generality, we assume that n is a sufficiently large number.

We lower-bound the aggregated load by a function ℓ that is defined recursively as follows. Let $i_1, i_2, \ldots$ be a sequence of integers with domain $\{1, \ldots, d\}$. Each of these variables is an indicator variable representing the decisions made by the considered algorithm in a certain phase of the allocation process. These phases will be defined implicitely during the following analysis.[3] We are interested in the minimum aggregated load over all possible ways to specify these indicator variables. Given some positive real numbers $m, n_1, \ldots, n_d$, we define

$$\begin{aligned}&\ell_{i_1,i_2,\ldots}(m, n_1, \ldots, n_d)\\ &\quad = 1 + \ell_{i_2,i_3,\ldots}\left(\frac{m^2}{(10d)^2 n_{i_1}}, n_1, \ldots, n_{i_1-1}, \frac{m}{10d}, n_{i_1+1}, \ldots, n_d\right),\end{aligned}$$

if $m \geq \sqrt{n}$, and $\ell_{i_1,i_2,\ldots}(m, n_1, \ldots, n_d) = 0$, otherwise. Finally, we define

$$\ell(m, n_1, \ldots, n_d) = \min_{i_1,i_2,\ldots} \ell_{i_1,i_2,\ldots}(m, n_1, \ldots, n_d).$$

We prove a lower bound on the aggregated load in terms of this function and, afterwards, we estimate this function by an appropriate term in closed form.

[3] To simplify the notation, we use an infinite sequence of indicator variables although the number of phases is finite. Only the leading variables are significant.

LEMMA 7. *Let α denote an arbitrary positive constant. The aggregated load of an $(m, n_1, \ldots, n_d)$-allocation is at least $\ell(m, n_1, \ldots, n_d)$, with probability $1 - n^{-\alpha}$.*

PROOF. We prove the lemma by induction, that is, we assume that the lemma holds for any $(m', \ldots)$-allocation with $m' < m$. We allow that the failure probability increases slightly from induction step to induction step, that is, we assume that the aggregated load of the $(m', \ldots)$-allocation is at least $\ell(m', \ldots)$, with probability $1 - \frac{n^{-\beta}}{2}$, for some given $\beta > 0$, and we prove that the stated bound for m holds with probability at least $1 - n^{-\beta}$. Our proof holds for any given β provided that n is sufficiently large depending on β. The induction takes only $\log_2 \ln n$ steps in each of which the bound on the failure probability is increased by a factor 2. Thus, over all induction steps, the failure probability increases at most by a factor $\ln n$. By setting β appropriately, one can achieve probability $1 - n^{-\alpha}$, for any α that one likes.

Let B denote the set of m balls that should be placed into 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 those $\lfloor \frac{m}{2} \rfloor$ balls that are inserted first, and define $B_2 = B \setminus B_1$. For $j \geq 0$, we define

$$n_i(j) \;=\; \frac{m^j}{(10d)^{j(j+1)/2} n_i^{j-1}},$$

and

$$m(j) \;=\; \frac{m\, n_i(j)}{(10d)^j n_i} \;=\; \frac{m^{j+1}}{(10d)^{j(j+3)/2} n_i^j}.$$

The motivation for these two definitions will become clear soon. For the time being, it is sufficient to notice that $m(j) \leq n_i(j)$ (because $m \leq n_i$ by our general assumptions) and $n_i = n_i(0) > n_i(1) > n_i(2) \cdots$ as well as $m = m(0) > m(1) > m(2) \cdots$

We show the following two properties.

(1) There exists $1 \leq i \leq d$ and $j \geq 1$ such that $N_i' \subseteq N_i$ with $|N_i'| = \lfloor n_i(j) \rfloor$ and each bin from N_i' receives at least j balls from B_1.
(2) Fix i and j fulfilling Property 1. 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 $1 - \frac{n^{-\beta}}{4}$, provided that n is sufficiently large. For the first property, we additionally assume $m \geq \sqrt{n}$, and for the second property, we assume $m(j) \geq \sqrt{n}$. If these assumptions do not hold, then we have reached the end of the recursion.

When proving these properties, we give no other evidence about the randomly chosen locations of the balls in B' except that the ith location of each of them points to one of the bins from N_i'. Thus, all these balls choose their locations according to some probability distribution over $N_1 \times \cdots \times N_{i-1} \times N_i' \times N_{i+1} \times \cdots \times N_d$. In particular, the choices for different balls from B' are independent and all these balls use the same distribution. Thus, we can treat these ball recursively as follows. 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 the induction assumption, this allocation

has an aggregated load of at least $\ell(m(j), n_1, \ldots, n_i(j), \ldots, n_d)$, with probability $1 - \frac{n^{-\beta}}{2}$. 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 aggregated load of the original $(m, n_1, \ldots, n_d)$-allocation is at least

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

with probability at least $1 - \frac{n^{-\beta}}{2} - 2\frac{n^{-\beta}}{4} = 1 - n^{-\beta}$.

Observe, 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, that is, we have defined $n_i(j)$ and $m(j)$ in such a way that applying the recursion to the term in (1) repeatedly for j times (choosing the same i in each iteration instead of using the minimum operator) yields exactly the lower bound given in Lemma 7. In other words, the lower bound in (1) implies the lower bound in the lemma. Thus, it remains only to prove the properties stated above.

PROOF OF PROPERTY (1). A bin from the set N_i is called *large* if the probability that the ith location from a given ball of B_1 falls into that bin is at least $\frac{1}{2dn_i}$. Furthermore, a ball from B_1 is called *interesting* if none of its d locations fall into a small bin, or the other way round, a ball is not interesting if at least one of the d locations from a ball in B_1 points to a small bin. Observe that the probability that the ith location of a ball falls into a small (nonlarge) bin is at most $n_i \frac{1}{2dn_i} = \frac{1}{2d}$. Hence, the probability that a ball is interesting is at least $1 - \frac{d}{2d} = \frac{1}{2}$. Notice that this bound holds regardless of possible dependencies between the random choices of the d locations of a ball. Consequently, the expected number of interesting balls is at least $\frac{|B_1|}{2} = \frac{1}{2}\lfloor\frac{m}{2}\rfloor \geq \frac{m}{4} - \frac{1}{2}$. Now applying a Chernoff bound yields that at least $\frac{m}{5}$ balls from B_1 are interesting, with probability $1 - \frac{n^{-\beta}}{4}$, assuming that $m \geq \sqrt{n}$ and n is sufficiently large.

Next we investigate how many of the large bins in any one of the d groups receive j or more balls. At least one of the d groups gets a fraction of $\frac{1}{d}$ of the interesting balls, that is, this group gets at least $\frac{m}{5d}$ interesting balls. Let i denote this group, and $M \geq \frac{m}{5d}$ 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 $\frac{M}{2^j}$ of the large bins receive at least j of the balls, for some $j \geq 1$. (Otherwise, the total number of balls would be smaller than $\sum_{j=1}^{\infty} \frac{M}{2^j} = M$.) Hence, we can fix an appropriate j so that the number of large bins from N_i with load j or more is at least

$$\frac{M}{2^j} \;\geq\; \frac{m}{5d2^j} \;\geq\; \frac{m}{(10d)^{j(j+1)/2}} \;\geq\; \frac{m^j}{(10d)^{j(j+1)/2} n_i^{j-1}} \;=\; n_i(j)$$

because $M \geq \frac{m}{5d}$, $5d2^j \leq (10d)^j \leq (10d)^{j(j+1)/2}$, and $n_i \geq m$. Thus, we can define N_i' to be a subset of size $\lfloor n_i(j) \rfloor$ of the set of large bins from group i with load at least j. In this way, 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

$$\frac{\lfloor n_i(j) \rfloor}{2dn_i} \;\geq\; \frac{n_i(j)}{4dn_i}$$

because these bins are large so that each of them has probability at least $\frac{1}{2dn_i}$ to be selected. Now, as $|B_2| \geq \frac{m}{2}$, the expected number of balls in B' (i.e., the set of the balls with a location in N_i') is at least

$$\frac{m}{2} \cdot \frac{n_i(j)}{4dn_i} \leq \frac{mn_i(j)}{8dn_i}.$$

Finally, applying a Chernoff bound yields that the number of balls in B' is at least

$$\frac{mn_i(j)}{10dn_i} \;\geq\; \frac{mn_i(j)}{(10d)^j n_i} \;=\; m(j),$$

with probability at most $1 - \frac{n^{-\beta}}{4}$, provided $m(j) \geq \sqrt{n}$. Consequently, Property (2) is satisfied and, hence, the proof of Lemma 7 is completed. □

The lower bound on the aggregated load L of an $(n, n, \ldots, n)$-allocation that we can obtain by applying Lemma 7 corresponds to the number of executions of recursion ℓ until the parameter m becomes smaller than $\sqrt{n}$, starting with $m = n$. In the following, we investigate how the parameters $m, n_1, \ldots, n_d$ behave when the recursion is applied several times. For $t \geq 0$, let $m^{(t)}, n_d^{(t)} \ldots, n_d^{(t)}$ denote the values of $m, n_1, \ldots, n_d$, respectively, after the recursion ℓ has been executed t times, starting with

$$m^{(0)} = n_1^{(0)} = \cdots = n_d^{(0)} = n.$$

LEMMA 8. *There is a constant $c > 0$ such that $m^{(t)} \geq n \cdot exp(-c\,\phi_d^t\, ln\, d)$, for every $t \geq 0$.*

PROOF. Under a logarithmic scale, the recursion ℓ is similar to the Fibonacci recursion. This can be seen as follows: Let us define the following logarithmic variants of the variables $m^{(t)}, n_1^{(t)}, \ldots, n_d^{(t)}$. Define

$$s_0^{(t)} \;=\; \frac{\log(n) - \log\left(m^{(t)}\right)}{2\log(10d)},$$

and analogously, for $1 \leq i \leq d$,

$$s_i^{(t)} \;=\; \frac{\log(n) - \log\left(n_i^{(t)}\right)}{2\log(10d)}.$$

In this way,

$$m^{(t)} \;=\; n \cdot (10d)^{-2s_0^{(t)}} \;\geq\; n \cdot \exp(-5s_0^{(t)} \ln d),$$

so that we have to show $s_0^{(t)} = O(\phi_d^t)$.

Consider the vectors $s^{(t)} = (s_0^{(t)}, \ldots, s_d^{(t)})$ as defined by recursion ℓ. The initial vector is $s^{(0)} = (0, \ldots, 0)$, and, for $t \geq 1$, we obtain the following recursive

description

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

The major problem in analyzing the recursion s given by the Eq. (2) is that s is controlled by a vector of integers $i_1, i_2, \ldots$ that are specified by an adversary. Intuitively, this adversary determines which one of d memorized values is chosen in the next iteration so that the final value of s_0 is maximized. The tricky part behind this selection is that the chosen value is replaced by the old value of s_0 such that it is not obvious which of the memorized values is chosen by the adversary.

In order to simplify the analysis, let us introduce another, simpler recursion δ in which the adversary always picks the index with smallest value. After analyzing δ, we show that this is, in fact, the choice that maximimizes the final recursion value. The recursion is defined as follows. We set $\delta^{(0)} = (\kappa_0, \kappa_1, \ldots, \kappa_d)$ for some integers $\kappa_0 \geq \kappa_1 \geq \cdots \geq \kappa_d \geq 0$. For $t \geq 1$, we define

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

Notice that

$$\delta_0^{(t)} = 2\delta_0^{(t-1)} - \delta_0^{(t-d-1)} + 1, \tag{3}$$

for $t \geq d + 1$. This equation shows that the recursion δ basically corresponds to the Fibonacci recursion as the tth d-ary Fibonacci number satisfies

$$\begin{aligned} F_d(t) &= \sum_{i=1}^{d} F_d(t-i) \\ &= F_d(t-1) + \sum_{i=2}^{d+1} F_d(t-i) - F_d(t-d-1) \\ &= 2F_d(t-1) - F_d(t-d-1), \end{aligned}$$

for $t \geq 3$. In fact, an induction on t using equation 3 yields

$$\delta_0^{(t)} = \sum_{i=0}^{t} \sum_{j=0}^{i} F_d(j) + \kappa_0 \sum_{j=0}^{t+1} F_d(j) - \sum_{i=1}^{d} \kappa_i \sum_{j=0}^{t-d+i} F_d(j), \tag{4}$$

for any $t \geq 0$. We come to the following observations holding for any $t \geq 0$.

—$\delta_0^{(t)}$ is monotonically increasing in κ_0 and monotonically decreasing in $\kappa_1, \ldots, \kappa_d$.

—Permuting some of the values of $\kappa = (\kappa_0, \kappa_1, \ldots, \kappa_d)$ does not increase the value of $\delta_0^{(t)}$ as the vector κ is assumed to be nonincreasing.

—An easy induction shows that the vector $\delta^{(\tau)}$, for $0 \leq \tau \leq t$, is nonincreasing as well. Thus, the two observations above can be generalized from κ to any intermediate vector $\delta^{(\tau)} = (\delta_0^{(\tau)}, \delta_1^{(\tau)}, \ldots, \delta_d^{(\tau)})$.

From now on, let us assume $\kappa = (0, \ldots, 0)$. Then the recursions s and δ differ in only two aspects. First, the control integers $i_1, i_2, \ldots$ used in s are replaced by a selection mechanism that always chooses the index with smallest value among the memorized numbers. Our observations above show that this choice maximizes $\delta_0^{(t)}$. Second, in each iteration of s, one of the memorized values with index from 1 to d is increased by a value of 0.5 in each iteration. Our observations show that doing the same for recursion δ cannot increase the value of $\delta_0^{(t)}$. Thus, $\delta_0^{(t)}$ is an upper bound on $s_0^{(t)}$. Consequently, it remains only to show $\delta_i^{(t)} = O(\phi_d^{\,t})$.

The asymptotic behavior of δ_0^t can be estimated as follows. Applying Eq. (4) with $\kappa = (0, \ldots, 0)$ yields

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

This completes the proof of Lemma 8. □

Finally, we combine Lemma 7 and Lemma 8 in order to derive a lower bound on the aggregated load L. Lemma 7 shows that the aggregated load L of an $(n, \ldots, n)$-allocation is at least $\ell(n, \ldots, n)$. The value of $\ell(n, \ldots, n)$ is equal to the recursion depth, where the recursion end is reached when m drops below $\sqrt{n}$. In other words, $\ell(n, \ldots, n) = \tau$ where τ is the smallest integer satisfying $m^{(\tau)} < \sqrt{n}$. Now applying Lemma 8 yields $m^{(t)} \ge n \cdot \exp(-c\,\phi_d^{\,t} \ln d)$ so that we obtain

$$L \;\ge\; \tau \;>\; \log_{\phi_d}\left(\frac{\ln n}{10 \ln d}\right) \;=\; \frac{\ln \ln n}{\ln \phi_d} - O(\ln \ln d),$$

which corresponds to the lower bound on the aggregated load claimed in Lemma 1. Thus, this lemma and, hence, Theorem 2 is shown.

4. *Experimental Results*

Our theoretical results show that the Always-Go-Left process yields a smaller maximum load than the uniform greedy process. The theoretical analysis, however, leaves open the exact constants hidden by the $\pm\Theta(1)$ terms in the Theorems 1 and 2. Therefore, it is interesting to study the maximum load for practical values of n experimentally.

Our experimental comparisons between the uniform greedy and the Always-Go-Left algorithms show a small but obvious advantage for the Always-Go-Left algorithm. In Table I, we extend the the experimental results of Azar et al. [1999, Table 7.1] by new measurements for the Always-Go-Left algorithm. The table lists the maximum load obtained by the different algorithms, for $256 \le n \le 16, 777, 216$. For each configuration, we ran 100 experiments.

5. *Discussion*

We introduced a nonuniform algorithm using an asymmetric tie breaking mechanism that clearly improves upon the bounds known for the uniform greedy allocation. On the other hand, we presented an almost matching lower bound on the best possible maximum load for any sequential multiple-choice algorithm, including those algorithms that exploit full knowledge about previous assignments. Hence,

TABLE I. EXPERIMENTAL RESULTS FOR THE MAXIMUM LOAD FOR n BALLS AND n BINS BASED ON 100 EXPERIMENTS FOR EACH CONFIGURATION

	$d = 1$	$d = 2$		$d = 4$	
n	single-choice	uniform	go-left	uniform	go-left
2^8	**3** . . . 1% **4** . . . 40% **5** . . . 41% **6** . . . 15% **7** . . . 3%	**2** . . . 10% **3** . . . 90%	**2** . . . 29% **3** . . . 71%	**2** . . . 99% **3** . . . 1%	**2** . . . 100%
2^{12}	**5** . . . 12% **6** . . . 66% **7** . . . 17% **8** . . . 4% **9** . . . 1%	**3** . . . 99% **4** . . . 1%	**3** . . . 100%	**2** . . . 91% **3** . . . 9%	**2** . . . 100%
2^{16}	**7** . . . 48% **8** . . . 43% **9** . . . 9%	**3** . . . 64% **4** . . . 36%	**3** . . . 100%	**2** . . . 23% **3** . . . 77%	**2** . . . 100%
2^{20}	**8** . . . 28% **9** . . . 61% **10** . . . 10% **13** . . . 1%	**4** . . . 100%	**3** . . . 96% **4** . . . 4%	**3** . . . 100%	**2** . . . 100%
2^{24}	**9** . . . 12% **10** . . . 73% **11** . . . 13% **12** . . . 2%	**4** . . . 100%	**3** . . . 44% **4** . . . 56%	**3** . . . 100%	**2** . . . 100%

we have shown that asymmetry in combination with a nonuniform choice of the locations for each ball leads to a significant improvement in the load balancing; further tricks, however, can improve only on the additive constants in the bounds for the maximum load.

We want to point out that both the asymmetry and the partitioning of the set of bins are crucial. In fact, Azar et al. [1999] show that the uniform greedy scheme with an arbitrary tie breaking mechanism is "majorized" by any other uniform allocation scheme. In other words, under the uniform algorithm, all tie breaking mechanisms lead to exactly the same stochastic distribution of balls so that using an asymmetric tie breaking mechanism without partitioning the set of bins does not help. Also partitioning the set of bins and using a fair tie breaking mechanism does provably not help as it basically leads to the same maximum load as the uniform greedy scheme. We can summarize that asymmetry helps load balancing but only in combination with a nonuniform choice of the locations for each ball.

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

REFERENCES

ADLER, M., CHAKRABARTI, S., MITZENMACHER, M., AND RASMUSSEN, L. 1995. Parallel randomized load balancing. In *Proceedings of the 27th ACM Symposium on Theory of Computing (STOC)*. ACM, New York, pp. 238–247.

AZAR, Y., BRODER, A., KARLIN, A., AND UPFAL, E. 1994. Balanced allocations. In *Proceedings of the 26th ACM Symposium on Theory of Computing (STOC)*. ACM, New York, pp. 593–602.

AZAR, Y., BRODER, A., KARLIN, A., AND UPFAL, E. 1999. Balanced allocations. *SIAM J. Comput. 29*, 1, 180–200.

BERENBRINK, P., CZUMAJ, A., STEGER, A., AND VÖCKING, B. 2000. Balanced allocations: The heavily loaded case. In *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC)*. ACM, New York, pp. 745–754.

BERENBRINK, P., MEYER AUF DER HEIDE, F., AND SCHRÖDER, K. 1999. Allocating weighted jobs in parallel. *ACM Trans. Comput. Syst. 32*, 3, 1703–1739.

COLE, R. J., MAGGS, B. M., MEYER AUF DER HEIDE, F., MITZENMACHER, M., RICHA, A. W., SCHRÖDER, K., SITARAMAN, R. K., AND VÖCKING, B. 1998a. Randomized protocols for low-congestion circuit routing in multistage interconnection networks. In *Proceedings of the 30th ACM Symposium on Theory of Computing (STOC)*. ACM, New York, pp. 378–388.

COLE, R. J., FRIEZE, A., MAGGS, B. M., MITZENMACHER, M., RICHA, A. W., SITARAMAN, R. K., AND UPFAL, E. 1998b. On balls and bins with deletions. In *Proceedings of the RANDOM'98*.

CZUMAJ, A., MEYER AUF DER HEIDE, F., AND STEMANN, V. 2000. Contention resolution in hashing based shared memory simulations. *SIAM J. Comput. 29*, 5, 1703–1739.

DIETZFELBINGER, M. AND MEYER AUF DER HEIDE, F. 1993. Simple, efficient shared memory simulations. In *Proceedings of the 5th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. ACM, New York, pp. 110–119.

KARP, R., LUBY, M., AND MEYER AUF DER HEIDE, F. 1992. Efficient PRAM simulation on a distributed memory machine. In *Proceedings of the 24th ACM Symposium on Theory of Computing (STOC)*. ACM, New York, pp. 318–326.

KNUTH, D. E. 1998. *The Art of Computer Programming*, Vol. 3. Addison–Wesley, Reading, Mass.

LUCZAK, J., AND UPFAL, E. 1999. Reducing network congestion and blocking probability through balanced allocation. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, Los Alamitos, Calif., pp. 587–595.

MEYER AUF DER HEIDE, F., SCHEIDELER, C., AND STEMANN, V. 1996a. Exploiting storage redundancy to speed up randomized shared memory simulations. *Theoret. Comput. Sci. 162*, 245–281.

MEYER AUF DER HEIDE, F., SCHRÖDER, K., AND SCHWARZE, F. 1996b. Routing on networks of optical crossbars. In *Proceedings of the Euro-Par'96*. pp. 299–306.

MITZENMACHER, M. 1996a. Load balancing and density dependent jump Markov processes. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, Los Alamitos, Calif., pp. 213–222.

MITZENMACHER, M. 1996b. The power of two choices in randomized load balancing. Ph.D. dissertation, Univ. California at Berkeley, Berkeley, Calif.

MITZENMACHER, M., RICHA, A., AND SITARAMAN, R. 2001. *Handbook of Randomized Computing*. "The power of two random choices: A survey of the techniques and results." Kluwer Press.

STEMANN, V. 1996. Parallel balanced allocations. In *Proceedings of the 8th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. ACM, New York, pp. 261–269.

VVEDENSKAYA, N. D., DOBRUSHIN, R. L., AND KARPELEVICH, F. I. 1996. Queueing system with selection of the shortest of two queues: An assymptotic approach. *Prob. Inf. Trans. 32*, 1, 15–27.

RECEIVED AUGUST 2000; REVISED MARCH 2003; ACCEPTED APRIL 2003