How Asymmetry helps Load Balancing

Original Paper by Berthold Vocking

Who? Abhimanyu M A
Sumesh T A

From? Indian Institute of Technology, Kanpur

When? November 15, 2011

Classic Ball Bin Problem

What ? Given n balls place them in n bins.

Goal Minimizing the number of balls in the largest bin

Constraints

- Balls are indistinguishable
- Global knowledge of bins are not available
- Assignment has to be *Online*

Classic Ball Bin Problem

What ? Given n balls place them in n bins.

Goal Minimizing the number of balls in the largest bin

Constraints

- Balls are indistinguishable
- Global knowledge of bins are not available
- Assignment has to be *Online*

Classic Ball Bin Problem

What ?	Given n balls place them in n bins.
Goal	Minimizing the number of balls in the largest bin
Constraints	

- Balls are indistinguishable
- Global knowledge of bins are not available
- Assignment has to be *Online*

Classic Randomized Algorithm

As was seen in such problems Randomization works really well.

Algorithm	For each ball chose a bin randomly and allocate.
Expected Load	$(1 + o(1))\frac{ln\ n}{ln\ ln\ n}$ w.h.p

Classic Randomized Algorithm

As was seen in such problems Randomization works really well.

Algorithm

For each ball chose a bin randomly and allocate.

Expected Load

$(1 + o(1))\frac{\ln n}{\ln \ln n}$ w.h.p

Classic Randomized Algorithm

As was seen in such problems Randomization works really well.

Algorithm	For each ball chose a bin randomly and allocate.
Expected Load	$(1 + o(1))\frac{ln\ n}{ln\ ln\ n}$ w.h.p

Can we do better ?

Yes we can.

With a little bit more of information & random bits.

Power of two random choices

Can we do better ?

Yes we can.
With a little bit more of information & random bits.

Power of two random choices

Power of Two Random Choices

Instead of a single bin choose two bins.

Not so simple

1. Requires double the number of random bits.
2. Requires Communication.

Algorithm

1. Choose 2 bins randomly, uniformly from set of n bins.
2. Find bin with lower number of balls.
3. Add ball to that bin.

Expected Load $\frac{\ln \ln n}{\ln 2} \pm \Theta(1)$ w.h.p **Exponential Reduction**

Proof We shall be presenting a "simpler" proof, while proving our results.

Power of Two Random Choices

Instead of a single bin choose two bins.
Not so simple

1. Requires double the number of random bits.
2. Requires Communication.

Algorithm

1. Choose 2 bins randomly, uniformly from set of n bins.
2. Find bin with lower number of balls.
3. Add ball to that bin.

Expected Load $\frac{\ln \ln n}{\ln 2} \pm \Theta(1)$ w.h.p **Exponential Reduction**

Proof We shall be presenting a "simpler" proof, while proving our results.

Power of Two Random Choices

Instead of a single bin choose two bins.
Not so simple

1. Requires double the number of random bits.
2. Requires Communication.

Algorithm

1. Choose 2 bins randomly, uniformly from set of n bins.
2. Find bin with lower number of balls.
3. Add ball to that bin.

Expected Load $\frac{\ln \ln n}{\ln 2} \pm \Theta(1)$ w.h.p **Exponential Reduction**

Proof We shall be presenting a "simpler" proof, while proving our results.

Power of Two Random Choices

Instead of a single bin choose two bins.
Not so simple

1. Requires double the number of random bits.
2. Requires Communication.

Algorithm

1. Choose 2 bins randomly, uniformly from set of n bins.
2. Find bin with lower number of balls.
3. Add ball to that bin.

Expected Load $\frac{\ln \ln n}{\ln 2} \pm \Theta(1)$ w.h.p **Exponential Reduction**

Proof We shall be presenting a "simpler" proof, while proving our results.

Power of Two Random Choices

Instead of a single bin choose two bins.
Not so simple

1. Requires double the number of random bits.
2. Requires Communication.

Algorithm

1. Choose 2 bins randomly, uniformly from set of n bins.
2. Find bin with lower number of balls.
3. Add ball to that bin.

Expected Load $\frac{\ln \ln n}{\ln 2} \pm \Theta(1)$ w.h.p **Exponential Reduction**

Proof We shall be presenting a "simpler" proof, while proving our results.

Power of Two Random Choices

Instead of a single bin choose two bins.
Not so simple

1. Requires double the number of random bits.
2. Requires Communication.

Algorithm

1. Choose 2 bins randomly, uniformly from set of n bins.
2. Find bin with lower number of balls.
3. Add ball to that bin.

Expected Load $\frac{\ln \ln n}{\ln 2} \pm \Theta(1)$ w.h.p **Exponential Reduction**

Proof We shall be presenting a "simpler" proof, while proving our results.

Power of Two Random Choices

Instead of a single bin choose two bins.
Not so simple

1. Requires double the number of random bits.
2. Requires Communication.

Algorithm

1. Choose 2 bins randomly, uniformly from set of n bins.
2. Find bin with lower number of balls.
3. Add ball to that bin.

Expected Load $\frac{\ln \ln n}{\ln 2} \pm \Theta(1)$ w.h.p **Exponential Reduction**

Proof We shall be presenting a "simpler" proof, while proving our results.

Even more choices

What if we had more than 2, say $d \geq 2$ choices for the bins ?

It doesn't help much

Expected Load

$\frac{\ln \ln n}{\ln d} \pm \Theta(1)$ w.h.p **Logarithmic Reduction**

Even more choices

What if we had more than 2, say $d \geq 2$ choices for the bins ?

It doesn't help much

Expected Load

$\frac{\ln \ln n}{\ln d} \pm \Theta(1)$ w.h.p **Logarithmic Reduction**

Even more choices

What if we had more than 2, say $d \geq 2$ choices for the bins ?

It doesn't help much

Expected Load

$\frac{\ln \ln n}{\ln d} \pm \Theta(1)$ w.h.p **Logarithmic Reduction**

Formalizing a few things

We can see that any algorithm has to be sequential and online. Now there are three classes of algorithms. Consider a set $[n] = \{0, 1, \dots n-1\}$ bins

1. *Uniform & Independent* - Each of the d locations of the ball are chosen uniformly and randomly from $[n]$
2. *(Non)Uniform & Independent* - For $1 \leq i \leq d$ the bin for the i^{th} location is chosen from a probability distribution D_i.
3. *(Non)Uniform & (In)Dependent* - The d locations for the balls are chosen at random from the set $[n]^d$ with a probability distribution D.

This includes all possibilities for Sequential and Online Algorithms.

Formalizing a few things

We can see that any algorithm has to be sequential and online. Now there are three classes of algorithms. Consider a set $[n] = \{0, 1, \dots n-1\}$ bins

Three Classes

1. *Uniform & Independent* - Each of the d locations of the ball are chosen uniformly and randomly from $[n]$
2. *(Non)Uniform & Independent* - For $1 \leq i \leq d$ the bin for the i^{th} location is chosen from a probability distribution D_i.
3. *(Non)Uniform & (In)Dependent* - The d locations for the balls are chosen at random from the set $[n]^d$ with a probability distribution D.

This includes all possibilities for Sequential and Online Algorithms.

Always Go Left Greedy Algorithm

We can do better with a non-uniform algorithm.

Surprise !!!

Algorithm

1. Partition the bin into d groups of *almost* equal size. Each group will have size $\Theta(\frac{n}{d})$
2. The i^{th} location of the ball is taken uniformly randomly from the i^{th} set of bins only. (See non-uniformity).
3. Insert ball into the minimum bin among them.
4. If there are more than one bin with same number of minimum balls, choose the leftmost bin.

The algorithm seems highly *biased* and *intuitively* we may think it is less efficient, but NO

Expected Load

$\frac{\ln \ln n}{d \ln \phi_d} \pm \Theta(1)$ w.h.p **Linear Reduction**

Always Go Left Greedy Algorithm

We can do better with a non-uniform algorithm.

Surprise !!!

Algorithm

1. Partition the bin into d groups of *almost* equal size. Each group will have size $\Theta(\frac{n}{d})$
2. The i^{th} location of the ball is taken uniformly randomly from the i^{th} set of bins only. (See non-uniformity).
3. Insert ball into the minimum bin among them.
4. If there are more than one bin with same number of minimum balls, choose the <u>leftmost</u> bin.

The algorithm seems highly *biased* and *intuitively* we may think it is less efficient, but NO

Expected Load

$\frac{\ln \ln n}{d \ln \phi_d} \pm \Theta(1)$ w.h.p **Linear Reduction**

Always Go Left Greedy Algorithm

We can do better with a non-uniform algorithm.

Surprise !!!

Algorithm

1. Partition the bin into d groups of *almost* equal size. Each group will have size $\Theta(\frac{n}{d})$
2. The i^{th} location of the ball is taken uniformly randomly from the i^{th} set of bins only. (See non-uniformity).
3. Insert ball into the minimum bin among them.
4. If there are more than one bin with same number of minimum balls, choose the leftmost bin.

The algorithm seems highly *biased* and *intuitively* we may think it is less efficient, but NO

Expected Load

$\frac{\ln \ln n}{d \ln \phi_d} \pm \Theta(1)$ w.h.p **Linear Reduction**

Always Go Left Greedy Algorithm

We can do better with a non-uniform algorithm.

Surprise !!!

Algorithm

1. Partition the bin into d groups of *almost* equal size. Each group will have size $\Theta(\frac{n}{d})$
2. The i^{th} location of the ball is taken uniformly randomly from the i^{th} set of bins only. (See non-uniformity).
3. Insert ball into the minimum bin among them.
4. If there are more than one bin with same number of minimum balls, choose the leftmost bin.

The algorithm seems highly *biased* and *intuitively* we may think it is less efficient, but NO

Expected Load

$\frac{\ln \ln n}{d \ln \phi_d} \pm \Theta(1)$ w.h.p **Linear Reduction**

Always Go Left Greedy Algorithm

We can do better with a non-uniform algorithm.

Surprise !!!

Algorithm

1. Partition the bin into d groups of *almost* equal size. Each group will have size $\Theta(\frac{n}{d})$
2. The i^{th} location of the ball is taken uniformly randomly from the i^{th} set of bins only. (See non-uniformity).
3. Insert ball into the minimum bin among them.
4. If there are more than one bin with same number of minimum balls, choose the leftmost bin.

The algorithm seems highly *biased* and *intuitively* we may think it is less efficient, but NO

Expected Load

$\frac{\ln \ln n}{d \ln \phi_d} \pm \Theta(1)$ w.h.p **Linear Reduction**

Always Go Left Greedy Algorithm

We can do better with a non-uniform algorithm.

Surprise !!!

Algorithm

1. Partition the bin into d groups of *almost* equal size. Each group will have size $\Theta(\frac{n}{d})$
2. The i^{th} location of the ball is taken uniformly randomly from the i^{th} set of bins only. (See non-uniformity).
3. Insert ball into the minimum bin among them.
4. If there are more than one bin with same number of minimum balls, choose the <u>leftmost</u> bin.

The algorithm seems highly *biased* and *intuitively* we may think it is less efficient, but NO

Expected Load

$\frac{\ln \ln n}{d \ln \phi_d} \pm \Theta(1)$ w.h.p **Linear Reduction**

Always Go Left Greedy Algorithm

We can do better with a non-uniform algorithm.

Surprise !!!

Algorithm

1. Partition the bin into d groups of *almost* equal size. Each group will have size $\Theta(\frac{n}{d})$
2. The i^{th} location of the ball is taken uniformly randomly from the i^{th} set of bins only. (See non-uniformity).
3. Insert ball into the minimum bin among them.
4. If there are more than one bin with same number of minimum balls, choose the <u>leftmost</u> bin.

The algorithm seems highly *biased* and *intuitively* we may think it is less efficient, but NO

Expected Load

$\frac{\ln \ln n}{d \ln \phi_d} \pm \Theta(1)$ w.h.p **Linear Reduction**

Always Go Left Greedy Algorithm

We can do better with a non-uniform algorithm.

Surprise !!!

Algorithm

1. Partition the bin into d groups of *almost* equal size. Each group will have size $\Theta(\frac{n}{d})$
2. The i^{th} location of the ball is taken uniformly randomly from the i^{th} set of bins only. (See non-uniformity).
3. Insert ball into the minimum bin among them.
4. If there are more than one bin with same number of minimum balls, choose the leftmost bin.

The algorithm seems highly *biased* and *intuitively* we may think it is less efficient, but NO

Expected Load

$\frac{\ln \ln n}{d \ln \phi_d} \pm \Theta(1)$ w.h.p **Linear Reduction**

Always Go Left Greedy Algorithm

We can do better with a non-uniform algorithm.

Surprise !!!

Algorithm

1. Partition the bin into d groups of *almost* equal size. Each group will have size $\Theta(\frac{n}{d})$
2. The i^{th} location of the ball is taken uniformly randomly from the i^{th} set of bins only. (See non-uniformity).
3. Insert ball into the minimum bin among them.
4. If there are more than one bin with same number of minimum balls, choose the <u>leftmost</u> bin.

The algorithm seems highly *biased* and *intuitively* we may think it is less efficient, but NO

Expected Load

$\frac{\ln \ln n}{d \ln \phi_d} \pm \Theta(1)$ w.h.p **Linear Reduction**

Equation

Fibonacci Constant (Detour)

We shall take a moment to explain ϕ_d which is nothing but the Fibonacci Constant for a Fibonacci Equation of order d.

$$\begin{aligned} F_i = &\quad F_{i-1} + F_{i-2} + ...F_{i-d} &\quad \text{If } i > d \\ = &\quad 0 &\quad \text{Otherwise} \end{aligned}$$

The constant is nothing but $\phi_d = \lim_{n\to\infty} \frac{F_n}{F_{n-1}}$
The smallest value is $\phi_2 = 1.61803...$ and the largest being $\phi_\infty = 2$
$\phi_2 < \phi_3 < \phi_4 < ... < 2$

Small Note

To prove the bound is indeed better we can see that $\phi_d > 2^{\frac{d-1}{d}}$. Applying it we get

$$\frac{ln\,ln\ d}{d\ ln\,\phi_d} < \frac{ln\,ln\ d}{d-1\ ln\,2} < \frac{ln\,ln\ d}{ln\ d}$$

Theorems

Theorem *Suppose that n balls are placed sequentially into n bins using the Always-Go-Left Algorithm then the number of balls in the fullest bin is*

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

Even more importantly

Theorem *Suppose that n balls are placed sequentially into n bins using any arbitrary sequential allocation scheme that choose d bins for each ball at random according to an arbitrary probability distribution on $[n]^d$ (Class 3). Then then number of balls in the fullest bin is*

$$\frac{\ln \ln n}{d \ \ln \phi_d} - O(1) \ w.h.p$$

Proof of Upper Bound

Witness Trees

What

Think about "Delay Sequences" witness trees can be thought of as the tree version of "delay sequence".
Let ε be some event, given that ε has happened it can lead us to infer that few other events ε_i would have happened (pre-requisites), which in turn would again have other events as pre-requisites hence the tree structure.
Even though not found in literature, we found it easy to visualize it

Example

Suppose ε is i^{th} bin has x balls after the j^{th} step. It says that all the d bins chosen during the j^{th} step should have had atleast $x - 1$ balls.

What we intend to do

1. We intend to use the witness tree to upper bound the probability of a bad event.
2. An "activated witness tree" is a tree that can be constructed for a given event, we shall be looking at the same for the *bad events*
3. Initially we will be doing a HUGE SIMPLIFICATION we will assume that each ball is included only once in the witness tree, this is as expected wrong but we will be later removing this simplification.
4. We shall be doing the simple math of

   Probability of bad event = Number of Witness trees possible $\times$ Probability for the "activation" of a witness tree.

Symmetric Witness Tree

We shall now see how we can use the witness tree argument to come up with a simpler proof for the "uniform" algorithm

Structure Each symmetric witness tree of order L is a complete d-ary tree of height L with d^L leaf nodes.

Node Each node represents a ball, (infact more than one node may represent the same ball). The nodes have to be proper , that is a node can be the ancestor of only another node that was inserted before it. Each node represents an event that may or may not happen.

Symmetric Witness Tree - Contd.

Edge Event: Consider an edge $e = (u, v)$ where v is the i^{th} child of u. Such an edge exists iff the i^{th} location of u's ball points to the same as one of the locations of v's ball.

Leaf Event: A node is a leaf node if at the time of the insertion of the ball , each of the d locations of the ball points to bins that have atleast 3 additional balls (balls not represented in the witness tree)
What we can see is that we can give any assignment to the nodes, and find the probability that it satisfies all edge events.

Construction of a Symmetric Witness Tree

We shall now prove that given that the maximum load is $L + 4$ we shall be able to construct a witness tree of order L

On the black board

Analyzing the Probability

Possibilities

If we notice, we can give any labeling from $1, .., n$ to any of the nodes. Few of them may not make sense but we can easily upper bound the possibilities.
Total number of trees $\leq n^m$ where m is the number of nodes.

Probability of Edge

Probability that v is the i^{th} child of u is at most $\frac{d}{n}$ this is because ball v has d bins to select and each of the bin may be selected with probability $\frac{1}{n}$.
Probability of Edge $\leq \left(\frac{d}{n}\right)^{m-1}$ ($m-1$ edges)

Probability of Leaf

Each leaf node's d choices have to point to a bin that have additional 3 balls, since the maximum number of bins with this is $\frac{n}{3}$, this probability can not be greater than $\frac{1}{3}$. Hence for each leaf node the probability is 3^{-d}
Probability for Leaf Event $\leq 3^{-d \cdot q}$ where q is number of leaves.

Analyzing the Probability

We can now look at the total probability.

$$\Pr[\text{Witness Tree}] \leq n^m \cdot \left(\frac{d}{n}\right)^{m-1} \cdot 3^{-d\cdot q} \tag{1}$$

We shall now apply the following easy to see bounds

$m \leq$	$2q$	Internal nodes is less than number of leaves
$2d^2 \leq$	3^d	Basic Algebra
$q =$	d^L	Property of complete L order d-ary tree

$$\begin{aligned} n^m \cdot \left(\frac{d}{n}\right)^{m-1} \cdot 3^{-d\cdot q} &\leq n \cdot d^{2q} \cdot 3^{-d\cdot q} && (2)\\ &\leq n \cdot 2^{-q} && (3)\\ &\leq n \cdot 2^{-d^L} && (4) \end{aligned}$$

Analyzing the Probability

We now need to come up with a value for L which gives

$$n \cdot 2^{-d^L} \leq n^{-\alpha}$$
$$L \geq log_d\, log_2\, n + log_d(1+\alpha)$$

The given value for L satisfies it.

Asymmetric Witness Tree

There is only a slight change in the structure, what we see is asymmetric tree has a large size and larger number of leaves, hence probability decreases.

Structure

The structure is a “fibonacci tree” of order L and d-ary fibonacci tree. Please note that the number of leaf nodes of fibonacci tree are bounded by $F_d(d \cdot L + 1) \geq \phi_d^{d \cdot L - 1}$

Construction of a Asymmetric Witness Tree

We shall now prove that given that the maximum load is $L+4$ we shall be able to construct a witness tree of order L

On the black board

Analyzing the Probability

Possibilities

Total number of trees $\leq n^m$ where m is the number of nodes.

Probability of Edge

The same probability but argued in a slightly different way, if the i^{th} location of u has to point to v it has to be one that is chosen from the i^{th} partition containing $\frac{n}{d}$ bins is $\frac{d}{n}$.
Probability of Edge $\leq \left(\frac{d}{n}\right)^{m-1}$ ($m-1$ edges)

Probability of Leaf

This again we get the same probability but requires explanation. We now know that there are not more than $\frac{n}{3}$ bins but now we have groups, and we do not know the distribution of the bins, but what we can see is that if we let β_i to denote the fraction , the probability is $\prod_{i=1}^{d} \beta_i$, by basic law of maths since sum is preserved it is maximum for equal values. Probability for Leaf Event $\leq 3^{-d \cdot q}$ where q is number of leaves.

Analyzing the Probability

We can now look at the total probability.

$$\Pr[\text{Witness Tree}] \leq n^m \cdot \left(\frac{d}{n}\right)^{m-1} \cdot 3^{-d\cdot q} \tag{5}$$

We shall apply the bounds we already worked out since they apply here as well,except for number of leaves

$$\begin{aligned} n^m \cdot \left(\frac{d}{n}\right)^{m-1} \cdot 3^{-d\cdot q} &\leq n \cdot d^{2q} \cdot 3^{-d\cdot q} && (6)\\ &\leq n \cdot 2^{-q} && (7)\\ & && (8) \end{aligned}$$

Now by property of d-ary Fibonacci trees

$$q = \phi^{d\cdot L-1} \tag{9}$$

Analyzing the Probability

$$\Pr[\text{Witness Tree}] \leq n \cdot 2^{-\phi^{d \cdot L-1}} \tag{10}$$

Finding out suitable value for L gives.

$$L \geq \frac{ln\ \ log_2\ n + ln\ (1+\alpha)}{d \cdot ln\ \phi_d} + 1 \tag{11}$$

Which gives us

$$L = \frac{ln\ \ log_2\ n}{d \cdot ln\ \phi_d} + O(1) \tag{12}$$

as the required high probability bound.

Something we have been avoiding

Remember we talked about having the same ball multiple number of times, which actually helps us but is WRONG. We will now correct that.
We call this procedure

Pruning

Procedure

We consider $L + 4 + \kappa$ balls in the maximum bin and construct a “Full Witness Tree” and then “Prune” it.

Construction & Pruning on Blackboard

Analysis of Pruned Witness Tree

First note that we will be upper bounding the number of nodes by $M = 2\kappa(\alpha+1)log_2 n$. **Why ?**

Possibilities

Total number of prunings $= M^{\kappa}$

Probability of Edge

$\left(\frac{d}{n}\right)^{m-1}$ ($m-1$ edges)

Probability of Leaf

$3^{-d\cdot q}$ where q is number of leaves.

Probability of Pruning

If a vertex is pruned then it means the similar vertex existed before, so let b be the cut off vertex and b' the ball seen before, if you look at the parent on of the edge has to point to one of the d locations pointed by b, hence probability $\leq \frac{Md}{n}$, M is added to consider all such vertices. This has to happen for all κ cuts.
Probability $\leq \left(\frac{Md}{n}\right)^{k}$

Analysis of Pruned Witness Tree

$$\begin{aligned}
\text{Prob} &\leq M^{\kappa} \cdot n^{m+1} \cdot \left(\frac{d}{n}\right)^{m} \cdot 3^{-d \cdot q} \cdot \left(\frac{Md}{n}\right)^{k} && (13)\\
&\leq n \cdot d^{2q} \cdot 3^{-d.q} \left(\frac{M^2 d}{n}\right)^{k} && (14)\\
&\leq n \left(\frac{M^2 d}{n}\right)^{k} \qquad \text{By } d^2 \leq 3^d && (15)\\
&\leq n \left(\frac{(2\kappa \cdot (\alpha+1) \cdot log_2 n)^2 \cdot d}{n}\right)^{\kappa} && (16)\\
&= n^{-\kappa+1+o(1)} && (17)
\end{aligned}$$

Lower Bound

We shall now show that for any sequential online assignment of balls to bins with maximum of d local queries the maximum load bin will have on expectation $\frac{\ln \ln n}{d \ln \phi_d} - O(1)$ balls w.h.p

Lower Bound Simulation

The major challenge is to control/understand the possibly dependent choices for the d- locations.
So instead what we do is we say that an algorithm A on n bins with dependency relation, it can be simulated by another algorithm B on dn bins where the i^{th} location will be taken from i^{th} set of bins.
We also give algorithm B global knowledge

Problem Definition- of B

We define the allocation as a $(m, n_1, n_2, \ldots, n_d)$ where we assign at least m balls to at most $\sum n_i$ bins with the conditions that

1. The bins are divided into d disjoint groups $N_1, \ldots N_d$ where $|N_i| \leq n_i$ for $1 \leq i \leq d$
2. Each of the balls is placed into one out of the d bins with probability distribution $D : N_1 \times \ldots \times N_d$.
3. The balls are assigned one after the other. We can only look back.

What we shall prove

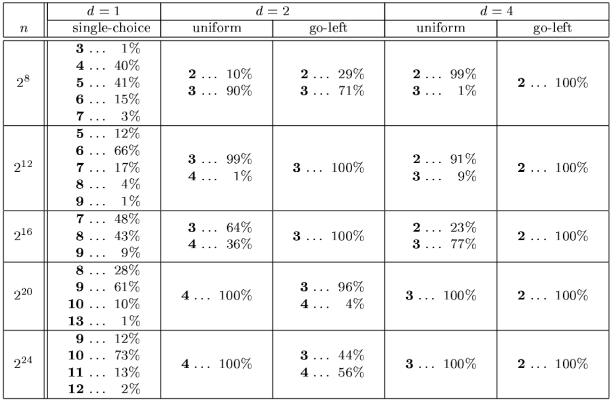
Let L^* denote the total maximum load in all bins, now we define *aggregated load* $L = \sum_{i=1}^{d} L_i$ where L_i is the size of the largest bin in the set N_i.
We shall prove that

Theorem *For any $(n, n, \dots n)$ allocation scheme*
$L = \frac{\ln \ln n}{\ln \phi_d} - O(\ln \ln d)$
This shall prove that.

Theorem *Suppose that n balls are placed sequentially into n bins using any arbitrary sequential allocation scheme that choose d bins for each ball at random according to an arbitrary probability distribution on $[n]^d$ (Class 3). Then then number of balls in the fullest bin is*

$$\frac{\ln \ln n}{d \ \ln \phi_d} - O(1) \ w.h.p$$

Experimental Results



n	$d = 1$	$d = 2$		$d = 4$	
	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%

What we felt

1. The algorithm is highly simple, and as we saw has no disadvantages when compared to uniform algorithm.
2. The upper bound proof was highly elegant and made simple compared to Adler et al.
3. The lower bound proof is also very important since it proves that Always Go Left is optimal for all sequential algorithms.
4. We wonder about the parallel and distributed versions of the algorithm.

Questions