Simplified Witness Tree Arguments

Thomas Schickinger* and Angelika Steger

Institut für Informatik, Technische Universität München, 80290 München, Germany,
http://www14.in.tum.de/

Abstract. In this paper we survey some results concerning balls-into-bins-games and the power of two choices. We present a unified and rather elementary analysis for models in the parallel as well as in the sequential setting which is based on witness trees.

1 Introduction

For the balanced allocation of resources in a distributed environment randomized strategies often turn out to achieve good results. Among such strategies so-called *balls-into-bins-games* play a major role.

The common characteristics of balls-into-bins-games can be summarized as follows: A set of *jobs* is to be allocated to a set of *processing units* (short: units) in such a way that the maximum load, also called *congestion* at a single unit is minimized. The general strategy for doing this is to first choose a set of d *candidate units* for each job in a random manner. An appropriate protocol then allocates each job to exactly one of its candidate units. Such a protocol depends heavily on the model under consideration, but it should be intuitively obvious that a smaller d reduces the communication overhead within the protocol, but increases the resulting maximum load.

Surprisingly, it turns out that already the cases $d = 1$ and $d \geq 2$ are fundamentally different. For $d \geq 2$ it is often possible to achieve much better results. Consider for example the allocation of n jobs at n units where the jobs arrive sequentially and choose d units uniformly at random. The job is then allocated at a candidate unit with minimum load breaking ties arbitrarily. It is well known that the maximum load for $d = 1$ is about $\ln n / \ln \ln n$ (see e.g. [Gon81] or [RS98]). It came as quite a surprise when it was shown in [ABKU94] that for $d = 2$ the maximum load is exponentially smaller, namely about $\ln \ln n / \ln d$. This phenomenon is often referred to as the *power of two choices* [Mit96b].

There is a rather broad literature on this phenomenon. An early application of the power of two choices can be found in PRAM simulations on Distributed Memory Machines (see e.g. [KLMadH92], [DMadH93]). Until now many different models for balls-into-bins-games and related problems have been presented and various aspects have been analysed.

Two main techniques for the analysis of balls-into-bins games have emerged: layered induction and witness trees. In this paper we will concentrate on witness

* Supported by DFG-grant Ste 464/3-1.

tree proofs since they are applicable to a larger variety of scenarios and often yield results which are more robust against modifications of the model.

Organization of the Paper This paper is organized as follows. In Sect. 2 we present the basic models for parallel and sequential allocation problems. In Sect. 3 we introduce our generalized proof technique which is based on witness trees. In Sect. 4 and Sect. 5 this technique is applied to parallel and sequential balls-into-bins-games.

2 Two Basic Models

Among the wide variety of different balls-into-bins-games that have been analysed in the literature two fundamentally different models can be distinguished: Parallel and sequential arrival of the jobs. In the sequel we introduce these two variants using classic examples which will also serve as basis for a unified analysis. For simplicity's sake we concentrate on the case when there are exactly n jobs and n units.

2.1 Parallel Arrival

In this model the jobs arrive in parallel and may communicate with the units before they choose their final destination. The communication proceeds in synchronous rounds and the objective is to achieve low congestion using only a small number of communication rounds.

This model and similar variants have achieved much attention in the literature (see e.g. [KLMadH92], [DMadH93], [GMR94], [CMadHS95], [MPR94], [MadHSS95], [ACMR95], [Ste96]). We will consider the model and the algorithm from [Ste96].

A distributed protocol, the so-called *collision protocol* is used to balance the load among the units: Every job chooses $d \geq 2$ candidate units uniformly at random. Then the following steps are repeated until no active, i.e., unassigned jobs remain:

- Every unassigned job j sends a *request* for allocation to its candidate units. (Due to this one-to-one correspondence between candidate units and requests we will use these terms interchangeably.)
- If the number of jobs which want to be allocated at a certain unit exceeds a fixed threshold c then the congestion at this unit is too high and the requests cannot be satisfied. Otherwise the unit sends an acknowledgment to the pending requests. If a jobs receives one or more acknowledgments it is allocated at one of the candidate units that sent them (making an arbitrary choice in case there is more than one possibility) and becomes inactive.

Note that the number of communication rounds for this protocol is not bounded and that, in principle, the protocol may not terminate either. However, it can be shown that for appropriate values of c and t with high probability all jobs are allocated after t rounds. In other word, the protocol finds an assignment with maximum load at most c in at most t time steps.

2.2 Sequential Arrival

In this model the jobs arrive sequentially and each job has to be assigned to a processing unit immediately after its arrival. The arriving jobs may communicate with the units before they choose their final destination. However, the amount of communication should be kept small. Here we focus on the following simple and natural strategy: each job chooses $d \geq 2$ candidate units randomly and checks their current load. Then the job is allocated at a unit with lowest load. This model was introduced in [ABKU94], slight variants and improvements can e.g. be found in [ABKU94], [ACMR95], [CFM$^+$98], [Vöc99], [BCSV00]. [Mit96a], [Mit97], [CS97] consider some related models.

3 A Generalized Approach Using Witness Trees

In this section we introduce the technique that we will later use for the analysis of allocation processes.

3.1 Allocation Graphs

The allocation of the jobs at the units can naturally be modelled by an *allocation graph* $G = (J \cup U, E)$, where J is the set of jobs and U is the set of units. G is bipartite, i.e., the edges in E only connect jobs to units. An edge $e = (j, u)$ corresponds to a job j with candidate unit u. We will also assume that e is labelled with the number $r(e) \in \{1, \ldots, d\}$ of the request which is modelled by e.

Existence of Witness Subgraphs If high congestion arises in the final allocation then a treelike *witness subgraph* can be found in G. Consider for example the collision protocol for the parallel arrival. Assume that the protocol does not terminate within t rounds. Then there must be at least one job j which survives the tth round. This can only be the case if *all* of its requests are not accepted. In other words, for any candidate unit u of the d candidate units chosen by j there must be c other jobs (which are active in the tth round) which all issue a request that conflicts with job j at unit u. The fact that these $d \cdot c$ jobs are still active in the tth round implies that they must have survived round $t-1$. We can thus repeat the above argument for each of these $d \cdot c$ jobs[1] yielding $(d \cdot c)^2$ many jobs which must have survived round $t-2$ and so on.

The basic idea behind proofs using witness subgraphs is the following: First it is shown, as we sketched before, that high congestion implies the existence of a (large) witness subgraph in the allocation graph. Then such large witness subgraphs are shown to arise with very small probability.

[1] In fact we have to modify the argument a bit because the $d \cdot c$ neighbors of the first job are only adjacent to $(d-1)$ units which are not yet part of the witness subgraph. But we will neglect that for the moment.

If the witness subgraph were indeed a tree then the analysis would be rather simple. The difficulty of this proof strategy stems from the fact that some of these jobs might actually be identical. That is, instead of $(d \cdot c)$-ary *witness trees* of depth t we get treelike witness graphs where some branches occur multiple times or where cross-edges introduce awkward dependencies. In previous approaches this usually has been taken care of by extracting a witness tree from the witness subgraph using a kind of breadth-first traversal which stops when it runs into cross edges.

Proof Strategy In this note we propose a different approach. We directly analyse the *allocation graph*, which is a *random* graph where the randomness comes from the jobs choosing their candidate units randomly. Our method for the analysis of witness trees consists of the following steps:

- First we show that high congestion implies the existence of a large witness subgraph in the allocation graph.
- Then we analyse the structure of the allocation graph and show that it is locally "treelike" with high probability. More precisely, we show that all cycles in a "small" radius can be destroyed by deleting just a few edges.
- In a last step we prove that we can still find a large witness tree after the destruction of the cycles. Such witness trees are then shown to occur with very small probability.

The bounds on the probability will be deduced using the well-known *First Moment Method* which is formulated for our purposes in the following lemma.

Lemma 1. (First Moment Method) *Consider a random graph G defined on an arbitrary probability space. Let N denote the number of subgraphs which satisfy a certain property Q. If $\mathbb{E}[N] = \mathcal{O}(n^{-\alpha})$ for $\alpha > 0$ then*

$$\Pr[\exists H \subseteq G \ : \ H \textit{ satisfies } Q] = \Pr[N \neq 0] = \mathcal{O}(n^{-\alpha}).$$

Proof. Follows directly from Markov's inequality: $\Pr[N \geq 1] \leq \mathbb{E}[N]$.

3.2 Multicycles

In this section we introduce a notion which captures the idea that the allocation graph looks "almost like a tree" in a small radius.

Definition 1. *A k-multicycle at vertex v of depth at most t is a tree with root v and exactly k cross edges such that the following holds:*

- *The depth of the tree is at most t.*
- *All leafs are incident to cross edges.*

Obviously, the following simple facts hold for a k-multicycle of depth at most t: The degree of the root is bounded from above by $2k$. The number of vertices is at most $2kt$ and $n - m = 1 - k$.

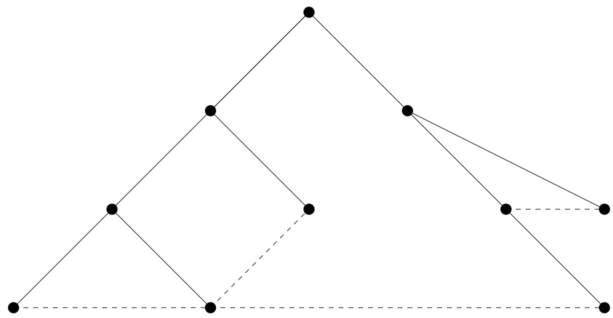

Fig. 1. A 4-multicycle of depth 3

Counting Multicycles If the edges in the multicycle occur independently with probability p, the expected number $\mathbb{E}[M_{k,t}]$ of k-multicycles of depth at most t can be estimated quite easily. Consider first the multicycles with exactly s vertices, where $s \leq 2kt$. To bound the number of possible multicycles we first choose the vertices in the multicycle (at most n^s possibilities). Then we fix the edges of the multicycle (at most 2^{s^2} possibilities) and we label the edges with the number of the corresponding request (at most d^{s-1+k} possibilities). As the edges were assumed to occur independently with probability p, each such multicycle occurs with probability exactly p^{s-1+k}. All in all we get

$$\mathbb{E}[M_{k,t}] \leq \sum_{s=1}^{2kt} n^s \cdot 2^{s^2} \cdot d^{s-1+k} \cdot p^{s-1+k}.$$

For our applications we will usually have that $t = \mathcal{O}(\log\log n)$, $d = \mathcal{O}(1)$ and $p = \mathcal{O}(1/n)$. This suffices to show that k-multicycles of depth at most t do not occur with probability $1 - n^{-\alpha}$ for $k \geq \alpha + 2$ due to the First Moment Method.

Turning Graphs Into Trees The next lemma shows that subgraphs of graphs without large multicycles can be turned into trees by deleting just a small number of edges at the root.

Lemma 2. *Assume that a graph $G = (V, E)$ contains no multicycles of depth at most t with more than k cross edges. Furthermore, consider a subgraph $H = (V', E')$ of G with a root vertex $v \in V'$ where every vertex $w \in V'$ has distance at most t from v in G. Then this subgraph can be turned into a forest by deleting at most $2k$ edges at the root.*

Proof. Traverse H with a BFS starting at v and mark all cross edges C in H. The edges in C together with the paths inside the BFS-tree from the end points of the edges to the root v define a multicycle (which also has root v). Hence, it follows that $|C| \leq k$. The root of the multicycle has degree at most $2k$ and we get a forest $H' \subseteq H$ by deleting all $\leq 2k$ edges at v that belong to the multicycle.

3.3 Putting Everything Together

Now we are in a position to give a detailed description of the steps which are used in our analysis method: For an arbitrary load balancing problem we intend to show that the following claims hold for suitable values of l and α.

1. *Finding Witness Trees:* If the maximum load caused by the allocation algorithm is at least l then we can find a witness subgraph T_l. The depth of T_l is bounded from above by a suitable function $t = t(n)$.
2. *Turning Graphs Into Trees:* After the deletion of $2k$ edges at the root of a witness subgraph T_l we can still find a graph $T_{l-k'}$ which is really a tree for suitable constants k and k'.
3. *Counting Multicycles:* Let $\mathcal{M}$ denote the event that the allocation graph contains a k-multicycle of depth at most t. Then $\Pr[\mathcal{M}] \leq \frac{1}{2}n^{-\alpha}$.
4. *Counting Witness Trees:* Let $\mathcal{T}$ denote that event that the allocation graph contains a witness tree $T_{l-k'}$. Then $\Pr[\mathcal{T}] \leq \frac{1}{2}n^{-\alpha}$.

From that we can conclude that the maximum load caused by the allocation algorithm is less than l with probability $n^{-\alpha}$ by the following reasoning: It holds that $\Pr[\mathcal{M} \cup \mathcal{T}] \leq n^{-\alpha}$ by claim 3 and claim 4. Assume that neither $\mathcal{M}$ nor $\mathcal{T}$ occur and that the maximum load is $l' \geq l$. Then we can find a witness subgraph T_l by claim 1. This witness subgraph can be turned into a witness tree $T_{l-k'}$ by claim 2. This yields a contradiction, since we assumed that $\mathcal{T}$ did not occur.

4 Parallel Allocation

Now we turn to the first application of our proof strategy.

Model and Algorithm We consider the same model as in [Ste96]: n jobs arrive in parallel and must be allocated at n units. The jobs may communicate with the units in synchronous rounds before their allocation. We try to minimize the maximum congestion executing only few communication rounds. This is achieved using the collision protocol, which we have already described in Sect. 2.1.

We intend to prove the following theorem which is very similar to the result in [Ste96]:

Theorem 1. *Let $\alpha > 0$, $\beta := \alpha + 4.5$ and $2 \leq t \leq \frac{1}{\beta}\ln\ln n$. Assume that n is sufficiently large that $\frac{c}{4e^2} \leq n^{0.1}$. Then the c-collision protocol with $d = 2$ terminates after at most t rounds for the threshold*

$$c = \max\left\{ \left(\frac{\beta t \ln n}{\ln\ln n}\right)^{1/(t-1)}, 5 + 2\alpha, 4e^2 + 1 \right\}$$

with probability at least $1 - n^{-\alpha}$.

Finding Witness Subgraphs For the moment we will concentrate on the structure of the witness subgraph and neglect duplication of vertices.

We call a unit *active in round t'* if it still takes part in the collision protocol in round t'. Assume that the collision game does not terminate after t rounds and consider a unit y which is active in round $t+1$. At time t there are at least $c+1$ jobs incident to y. These jobs have not been allocated until round t and, thus, the other $d-1$ units where they are connected to must still be active in round t. Hence, the witness tree exhibits a regular recursive structure (see Fig. 2), i.e., the witness tree T_t for t rounds is composed of $c(d-1)$ witness trees T_{t-1}. The leafs of the witness tree consist of a single unit since all units are active in round 1. Note that the resulting tree is c-ary but not $(c+1)$-ary because only the unit at the root has $c+1$ children. We ignore one of its children in order to get a simple regular structure.

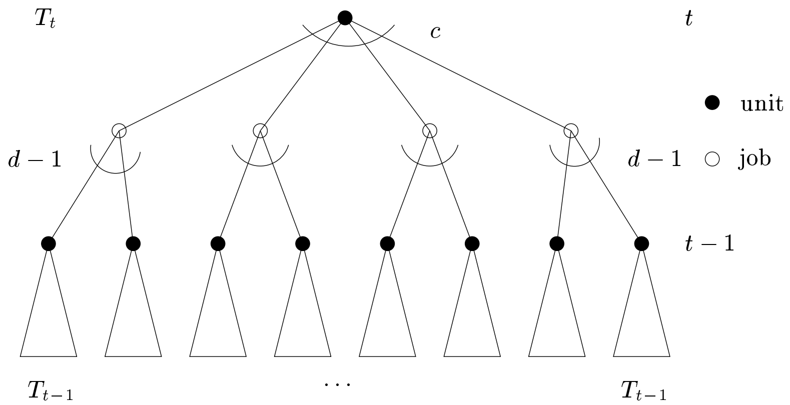

Fig. 2. Witness tree for collision games

Let j_t denote the number of jobs and u_t the number of units in a witness tree T_t. It follows that

$$\begin{array}{ll} j_t = c + c(d-1) \cdot j_{t-1} & ; \quad j_1 = 0 \\ u_t = 1 + c(d-1) \cdot u_{t-1} & ; \quad u_1 = 1 \end{array}.$$

One easily checks that

$$j_t = \frac{c^t(d-1)^{t-1} - c}{c(d-1) - 1} \quad ; \quad u_t = \frac{c^t(d-1)^t - 1}{c(d-1) - 1}.$$

We deduce for the number of edges r_t in T_t that $r_t = d \cdot j_t$ since every job has degree d. Furthermore, T_t contains j_t/c units as inner vertices as each such vertex is incident to c jobs.

Turning Graphs Into Trees Let T_t denote a witness subgraph which certifies that the unit at its root is still active after t rounds of the collision game. Since the root of the witness subgraph has degree c (see Fig. 2) we conclude that we can still find a witness subgraph T_{t-1} after the deletion of $2k$ edges at the root if $c > 2k$. Hence, if there are no k-multicycles and the collision game does not terminate after t rounds then we can find a witness tree T_{t-1}.

Counting Multicycles Counting k-multicycles as shown in Sect. 3.2 we obtain

$$\mathbb{E}[M_{k,t}] \le \sum_{s=1}^{2kt} n^s \cdot 2^{s^2} \cdot d^{s-1+k} \cdot p^{s-1+k}.$$

Using the assumptions on d and t and the fact that $p = 1/n$ it follows easily that there are no k-multicycles with probability $1 - \frac{1}{2}n^{-\alpha}$ for $k \ge 2 + \alpha$.

Counting Witness Trees Using the values of j_t, u_t and r_t we will now calculate the expected number of witness trees T_t. For simplicity's sake we drop the subscripts of j_t, u_t and r_t.

There are at most n^{j+u} possibilities to choose the vertices. After we have fixed the number of the requests (at most d^r possibilities) every edge occurs with probability $1/n$. Furthermore, we take into account $(1/c!)^{j/c} \cdot (1/(d-1)!)^j$ tree automorphisms. Finally, we deduce that

$$\begin{aligned}\mathbb{E}[T_t] &= n^{j+u} \cdot \left(\frac{d}{n}\right)^r \cdot \left(\frac{1}{c!}\right)^{j/c} \cdot \left(\frac{1}{(d-1)!}\right)^j \\ &\le n^{j+u-r} \cdot \left[\frac{e}{c} \cdot \left(\frac{e}{d-1}\right)^{d-1} \cdot d^d\right]^j \\ &= n \cdot \left[\frac{e}{c} \cdot \left(\frac{e}{d-1}\right)^{d-1} \cdot d^d\right]^{\frac{c^t(d-1)^{t-1}-c}{c(d-1)-1}} .\end{aligned}$$

To see this note that $j + u - r = 1$ because $j + u$ and r correspond to the number of vertices resp. edges in the witness tree.

For $d = 2$ this expression is similar to the results in [Ste96]. It holds that

$$\mathbb{E}[T_t] \le n \cdot \left(\frac{e}{c} \cdot 4e\right)^{\frac{c^t-c}{c-1}} \le n \cdot \left(\frac{4e^2}{c}\right)^{c^{t-1}-1} .$$

Similar calculations as in [Ste96] show that $\mathbb{E}[T_t] \le \frac{1}{2}n^{-\alpha}$. Theorem 1 then follows by the arguments given in Sect. 3.3.

Our analysis also yields a result for the case $d \ge 3$ without much additional effort. Note that

$$\gamma := \frac{c^t(d-1)^{t-1} - c}{c(d-1)-1} \ge \frac{(c(d-1))^{t-1} - 1}{d} \ge \frac{(c(d-1))^{t-1}}{2d}.$$

Furthermore, it is easy to check that $(d-1)^2/d \geq 1$ and, thus,

$$\gamma \geq \frac{1}{2}c^{t-1}(d-1)^{t-3} \geq (c(d-1))^{t-3}.$$

Hence, we obtain

$$\begin{aligned}
\mathbb{E}[T_t] &\leq n \cdot \left[\frac{e}{c}\left(\frac{e}{d-1}\right)^{d-1} \cdot d^d\right]^{(c(d-1))^{t-3}} \leq n \cdot \left[\frac{de^d}{c}\left(\frac{d}{d-1}\right)^{d-1}\right]^{(c(d-1))^{t-3}} \\
&\leq n \cdot \left[\frac{d \cdot e^{d+1}}{c}\right]^{(c(d-1))^{t-3}}.
\end{aligned}$$

This enables us to show the following theorem.

Theorem 2. *Let $\alpha > 0$, $\beta := \alpha + 2\ln d + d + 2$ and $2 \leq t \leq \frac{1}{\beta}\ln\ln n$. Then the c-collision protocol with $d \geq 3$ terminates after at most t rounds for the threshold*

$$c = \max\left\{\frac{1}{d-1} \cdot \left(\frac{\beta t \ln n}{\ln\ln n}\right)^{1/(t-3)}, 5 + 2\alpha, de^{d+1} + 1\right\}$$

with probability at least $1 - n^{-\alpha}$.

Using the definition of c and the bound on t we deduce that

$$\begin{aligned}
\mathbb{E}[T_t] &\leq n \cdot \left[d^2 e^{d+1} \cdot \left(\frac{\ln\ln n}{\beta t \ln n}\right)^{\frac{1}{t-3}}\right]^{\frac{\beta t \ln n}{\ln\ln n}} \\
&\leq n \cdot \left[(d^2 e^{d+1})^t \cdot \frac{\ln\ln n}{\beta t \ln n}\right]^{\frac{\beta \ln n}{\ln\ln n}} \leq n \cdot \left[\frac{(d^2 e^{d+1})^{\frac{1}{\beta}\ln\ln n}}{\ln n}\right]^{\frac{\beta \ln n}{\ln\ln n}} \\
&= n \cdot \frac{(d^2 e^{d+1})^{\ln n}}{n^\beta} = n^{2\ln d + d + 2 - \beta} = n^{-\alpha}.
\end{aligned}$$

5 Sequential Allocation

5.1 The Classic $d = 2$ Strategy

Model and Algorithm We consider the same model as in [ABKU94]: n jobs are sequentially allocated at n units. Every job chooses d units independently and uniformly at random and is allocated at a unit with minimum load. Ties are broken arbitrarily.

Theorem 3. *[ABKU94] For the sequential allocation process with d independent choices it holds that the maximum load is at most $\frac{\ln\ln n}{\ln d} + \mathcal{O}(1)$ with probability $1 - n^{-\alpha}$.*

Finding Witness Subgraphs For the moment we ignore cross edges and duplication of vertices. Let l denote the load after the allocation of all jobs. We say that a job j is *on level* l' if $l'-1$ other jobs have been allocated at the same unit as j before j's arrival.

Now we show how a witness tree T_l for a unit x with load l after the allocation of the last job can be constructed. Consider the job j on level l allocated at x. This job has issued requests to $d-1$ units $y_1, \ldots, y_{d-1}$ different from x and all these units must have had load at least $l-1$ at the time when j was allocated. Hence, we can find a witness tree T_{l-1} at $y_1, \ldots, y_{d-1}$. Furthermore, we can also find a witness tree T_{l-1} at x certifying that x had load $l-1$ before j was allocated (see Fig. 3). Finally, we define that T_1 consists of single unit and a single job joined by an edge.

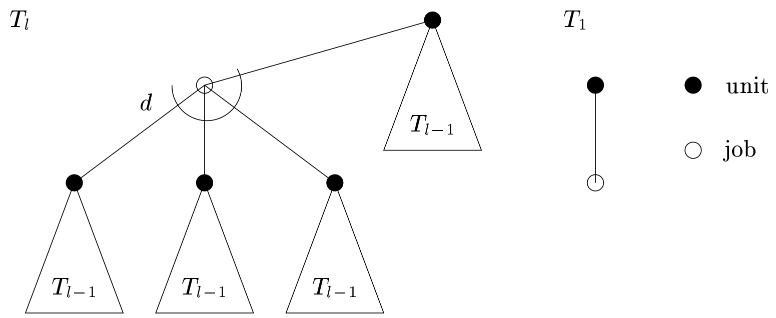

Fig. 3. Witness tree for sequential allocation

We use the variables j_l, u_l and r_l for the witness tree T_l as in the previous section and obtain the following recurrences.

$$\begin{array}{lcl} j_l = 1 + d \cdot j_{l-1} & ; & j_1 = 1 \\ u_l = d \cdot u_{l-1} & ; & u_1 = 1 \end{array}.$$

We easily deduce that

$$j_l = \frac{d^l - 1}{d-1} \quad ; \quad u_l = d^{l-1}.$$

Jobs which belong to a T_1 are called *leaf jobs* and we denote their number by h_l. Since a leaf job is allocated on level 1 and each unit contains exactly one such job it holds that $h_l = u_l = d^{l-1}$.

Turning Graphs Into Trees Consider the witness subgraph T_l of a unit u with load l. u contains jobs $j_1, \ldots, j_l$ where j_i is allocated on level i. For $i \geq 2$ it holds that j_i has $d-1$ incident units different from u with load at least $i-1$

before the insertion of j_i. Hence, it is connected to $d-1$ witness subgraphs T_{i-1} for $i = 2, \ldots, l$. Fig. 4 shows this view on the recursive structure of the witness subgraph, where only one of the $d-1$ witness subgraphs T_{i-1} is drawn for each job j_i.

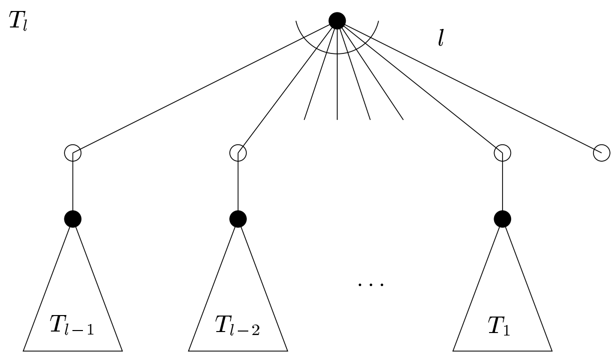

Fig. 4. Different view on witness tree for sequential allocation

From this structure it follows that after the deletion of $2k$ edges at the we can still find a witness tree T_{l-2k}.

Counting Multicycles We count multicycles as in Section 3.2:

$$\mathbb{E}[M_{k,l}] \leq \sum_{s=1}^{2kl} n^s \cdot 2^{s^2} \cdot d^{s-1+k} \cdot p^{s-1+k}.$$

As $p = 1/n$ it again follows easily that for $l = \mathcal{O}(\frac{\ln \ln n}{\ln d})$ there are no k-multicycles with probability $1 - \frac{1}{2}n^{-\alpha}$ for $k \geq 2 + \alpha$.

Counting Witness Trees In order to show that large witness trees occur with small probability we apply a technique from [Vöc99]: We construct the witness tree ignoring jobs on level 1, 2 and 3. If the maximum load after the allocation process amounts to $l+3$ we can find a witness tree T_l which contains only jobs on level 4 and above.

A witness tree is called *active* if its leaf jobs indeed reside on level 4 and above. At any time during the allocation process there are at most $n/3$ units containing at least three jobs. Henceforth, we call these units *heavy.* A witness tree can only be active if all its leaf jobs choose the $d-1$ units which don't belong to the witness tree among the heavy units. This happens with probability $3^{-(d-1)h_l}$. Note that for every job this bound holds deterministically regardless of the random choices of the other jobs. Hence, this probability is independent from the choices of the requests inside the witness tree.

We renounce taking the tree automorphisms into consideration and, thus, we may bound the number of choices for the jobs and units by $n^j \cdot n^u$ and assume that the labelling of the requests for non-leaf jobs is implicit in the order of the chosen vertices. For the labelling of the requests belonging to leaf jobs we have to adjoin the factor d^{h_l}. Each request in the tree occurs independently with probability $1/n$.

This leads to

$$\mathbb{E}[T_l] \le n^{j_l} \cdot n^{u_l} \cdot \left(\frac{1}{n}\right)^{r_l} \cdot d^{h_l} \cdot 3^{-(d-1)h_l} \le n \cdot 2^{-h_l} = n \cdot 2^{-d^{l-1}}.$$

Choosing $l = \log_d \log_2 n^{\alpha+1} + 2 = \frac{\ln \ln n}{\ln d} + \mathcal{O}(1)$ completes the proof of the theorem.

5.2 Always-Go-Left

Model and Algorithm In this section we will consider the sequential allocation strategy from [Vöc99] which surprisingly still improves on the result from [ABKU94] by introducing a slight asymmetry. At first sight this may seem rather unintuitive, but the analysis will show that asymmetry helps in making the witness trees larger and, thus, their occurence already becomes improbable for smaller maximum load.

The n units are divided into d groups of almost equal size, i.e., with $\Theta(n/d)$ units per group. For simplicity's sake we will henceforth assume that each group comprises exactly n/d units.

The unit for the ith request of a job is chosen from the ith group. Then the ball is, as usual, allocated at a unit with minimum load. If there is a draw the unit belonging to the group with the smallest number, i.e., the "left-most" group is selected. In [Vöc99] this algorithm is thus called "Always-Go-Left".

Theorem 4. *For $\alpha > 0$ the Always-Go-Left algorithm achieves maximum load at most $\ln \ln n/(d \ln \Phi_d) + \mathcal{O}(1)$ with probability $1 - n^{-\alpha}$.*

The constant Φ_d is defined with the help of generalized Fibonacci numbers. Define $F_d(k) = 0$ for $k \le 0$, $F_d(1) = 1$, and $F_d(k) = \sum_{i=1}^{d} F_d(k-i)$ for $k \ge 2$. Let $\phi_d = \lim_{k\to\infty} \sqrt[k]{F_d(k)}$, so that $F_d(k) = \Theta(\phi_d^k)$. Then $1.61.. = \phi_2 < \phi_3 < \cdots < 2$.

Finding Witness Trees The recurrences for the witness trees from Sect. 5.1 must slightly be modified as we have to consider to which of the d groups the units belong. In this section we deviate somewhat from the previous notation and number the groups by $0, \ldots, d-1$ instead of $1, \ldots, d$. The reasons for that will become clear shortly. Let $T_{l,i}$ denote a witness tree for a job on level l which is placed in a unit of group i. The adjacent witness trees are $T_{l,0}, \ldots, T_{l,i-1}, T_{l-1,i}, \ldots, T_{l-1,d-1}$. This recurrence also holds for $l = 1$ if we assume that the trees $T_{0,1}, \ldots, T_{0,d-1}$ are empty. $T_{1,0}$ is defined to consist of a single job and a single unit which are joined by an edge (see Fig. 6 for the general

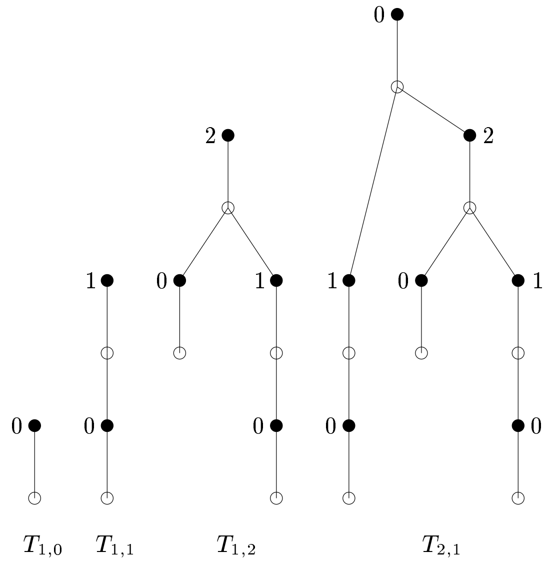

Fig. 5. Small witness trees for always-go-left allocation with $d = 3$

recursive structure and Fig. 5 for the small cases; the numbers correspond to the groups of the units).

Observe that this construction implies that all leaves are connected to a unit in group 0, i.e., their incident edge is labelled with 0.

In order to simplify the recurrence we let $T_x := T_{\lfloor (x+d)/d \rfloor, x \bmod d}$ for $x = 0, 1, \dots$ and may rewrite the adjacent witness trees of T_x as $T_{x-1}, \dots, T_{x-d}$.

Turning Graphs Into Trees By the recursive structure of the witness tree (as presented in Sect. 5.1) it follows that after the deletion of $2k$ edges at the root of a witness tree $T_{l,i}$ we can still find a witness tree $T_{l-2k,i'}$ (with $i' \neq i$) which is at least as large as a $T_{l-2k,0}$ (see Fig. 6 and Fig. 4).

Counting Multicycles Multicycles are counted as in Sect. 5.1 for the classic $d = 2$ case.

$$\mathbb{E}[M_{k,l}] \leq \sum_{s=1}^{2kl} n^s \cdot 2^{s^2} \cdot d^{s-1+k} \cdot p^{s-1+k}.$$

Note that due to the fact that we choose exactly one unit in each group the probability for the edges changes from $1/n$ to d/n. However, one easily checks that the additional factor d^{s-1+k} is asymptotically negligible and we still obtain that for $l = \mathcal{O}(\frac{\ln \ln n}{\ln d})$ there are no k-multicycles with probability $1 - n^{-\alpha}$ for $k \geq 2 + \alpha$.

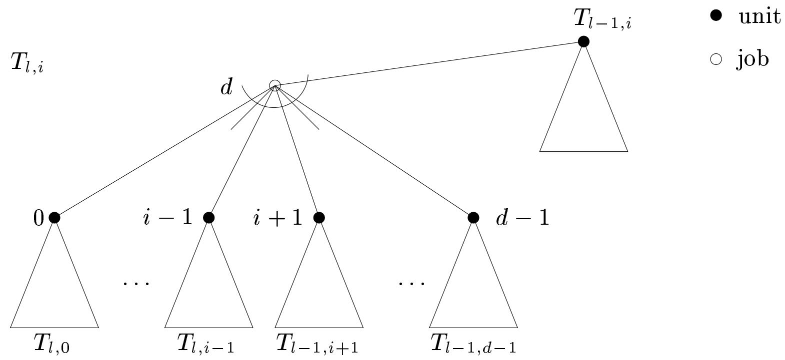

Fig. 6. Witness tree for sequential always-go-left allocation

Counting Witness Trees We get the following recurrences for the witness tree T_x using the same definition for $T_{1,0}$ as for T_1 in Sect. 5.1 (a single unit and a single job joined by an edge).

$$\begin{array}{lll} j_x = 1 + \sum_{i=1}^{d} j_{x-i} & ; & j_{-(d-1)} = \ldots = j_{-1} = 0,\ j_0 = 1 \\ u_x = \sum_{i=1}^{d} u_{x-i} & ; & u_{-(d-1)} = \ldots = u_{-1} = 0,\ u_0 = 1 \end{array}.$$

Let h_x denote the number of leaf jobs in T_x. For h_x we deduce the recurrence

$$h_x = \sum_{i=1}^{d} h_{x-i} \quad ; \quad h_{-(d-1)} = \ldots = h_{-1} = 0,\ h_0 = 1$$

Recall that $F_d(k) = 0$ for $k \leq 0$, $F_d(1) = 1$ and $F_d(k) = \sum_{i=1}^{d} F_d(k-i)$ for $k \geq 2$. Following the argumentation of [Vöc99], it holds that $F_d(k) \geq \Phi_d^{k-2}$. Thus, we conclude that $h_x = F_d(x) \geq \Phi_d^{k-2}$.

The next step is to estimate the number of jobs of j_x in comparison to the number of leaves h_x. By induction we prove that $h_x \geq \frac{1}{4} j_x + \frac{1}{4}$.

The basis of the induction is shown as follows: T_0 only contains one job which is a leaf job and, thus, we have $(j - h) = 0$ and $h = 1$. For the trees $T_1, \ldots, T_{d-1}$ we deduce by induction that $j - h = h$: When we compose a tree T_i for $i = 1, \ldots, d-1$ the non-leaf job that connects the trees $T_0, \ldots, T_{i-1}$ is compensated by the tree T_0 which contains one leaf job but no non-leaf job. Hence, for $0 \leq x < d$ it holds that $h_x \geq \frac{1}{2} j_x \geq \frac{1}{4} j_x + \frac{1}{4}$.

Now we prove the induction step. A tree T_x with $x \geq d$ is composed of subtrees $T_{x-d}, \ldots, T_{x-1}$ and contains one additional non-leaf job. Hence, it holds that

$$h_x = \sum_{i=1}^{d} h_{x-i} \geq \frac{1}{4} \sum_{i=1}^{d} j_{x-i} + \frac{1}{4} d \geq \frac{1}{4}(j_x - 1) + \frac{1}{2} = \frac{1}{4} j_x + \frac{1}{4},$$

completing the proof of the induction step.

The expected number of witness trees T_x is estimated as before. We have at most $n^j \cdot (n/d)^u$ possibilities to choose the jobs and the units in the tree. Note that the order of the choices fixes the group where a certain unit belongs to. Hence, we have only n/d instead of n choices per unit. Every edge occurs independently with probability d/n.

As in Sect. 5.1 we call a witness tree active, if for each leaf job the $d-1$ units that do not belong to the witness tree (that is the units in groups $1, \ldots, d-1$) are heavy. For technical reasons we have to strengthen the definition of heavyness and call a unit heavy if it has load at least 80. We claim that the probability that the $d-1$ units are all heavy is bounded from above by $20^{-(d-1-\lfloor d/4 \rfloor)}$. The proof proceeds as follows. Consider an arbitrary distribution of n balls. Let ξ denote the number of groups in which at least $\frac{1}{20}\frac{n}{d}$ units contain at least 80 balls. From $\xi \cdot \frac{1}{20}\frac{n}{d} \cdot 80 \leq n$ we deduce $\xi \leq \lfloor d/4 \rfloor$. That is, at least $d-1-\lfloor d/4 \rfloor$ units are choosen in groups with at most $\frac{1}{20}\frac{n}{d}$ heavy units. Hence,

$$\begin{aligned}
\mathbb{E}[T_x] &\leq n^j \cdot \left(\frac{n}{d}\right)^u \cdot \left(\frac{d}{n}\right)^r \cdot 20^{-(d-1-\lfloor d/4 \rfloor)h} \leq n \cdot d^{j-1} \cdot 20^{-(d-1-\lfloor d/4 \rfloor)h} \\
&\leq n \cdot d^{4h-1} \cdot 20^{-(d-1-\lfloor d/4 \rfloor)h} \leq n \cdot 2^{(4 \log_2 d - (d-1-\lfloor d/4 \rfloor) \log_2 20)h} \\
&\leq n \cdot 2^{-\frac{1}{4}h} \leq n \cdot 2^{-\frac{1}{4}\Phi_d^{x-2}} .
\end{aligned}$$

A tree $T_{l,i}$ for $i = 0, \ldots, d-1$ is at least as large as a tree $T_{l,0} \equiv T_{(l-1)d}$ and we obtain that

$$\mathbb{E}[T_{ld}] \leq 2^{-\frac{1}{4}\Phi_d^{ld-2}} .$$

This estimate is similar to the bound in [Vöc99]. Setting $l := \log_{\Phi_d}(4 \log_2 n^\alpha)/d + 2 = \ln\ln n/(d \ln \Phi_d) + \mathcal{O}(1)$ yields the desired result. (Note that the constants in this section can be significantly improved. In particular, the inequality $h_x \geq \frac{1}{4}j_x + \frac{1}{4}$ is rather weak and can be improved by more careful argumentations.)

6 Conclusion

We have presented a unified view on witness tree proofs for various balls-into-bins games. We feel that the uniform approach for the different models faciliates the understanding of the results and also provides intuitive insight.

References

[ABKU94] Yossi Azar, Andrej Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC-94)*, pages 593–602, 1994.

[ACMR95] Micah Adler, Soumen Chakrabarti, Michael Mitzenmacher, and Lars Rasmussen. Parallel randomized load balancing. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC-95)*, pages 238–247. ACM Press, 1995.

[BCSV00] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocation: the heavily loaded case. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC-00)*, pages 745–754, 2000.

[CFM+98] Richard Cole, Alan Frieze, Bruce M. Maggs, Michael Mitzenmacher, Andréa W. Richa, Ramesh K. Sitaraman, and Eli Upfal. On balls and bins with deletions. In *Proceedings of the 2nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 1998.

[CMadHS95] Artur Czumaj, Friedhelm Meyer auf der Heide, and Volker Stemann. Improved optimal shared memory simulations, and the power of reconfiguration. In *Proceedings of the 3rd Israel Symposium on Theory of Computing*, pages 11–19, 1995.

[CMM+98] Richard Cole, Bruce M. Maggs, Friedhelm Meyer auf der Heide, Michael Mitzenmacher, Andrea W. Richa, Klaus Schröder, Ramesh K. Sitaraman, and Berthold Vöcking. Randomized protocols for low-congestion circuit routing in multistage interconnection networks. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC-98)*, pages 378–388, 1998.

[CS97] Artur Czumaj and Volker Stemann. Randomized allocation processes. In *Proceedings of the 38th IEEE Symposium on the Foundations of Computer Science (FOCS-97)*, pages 194–203, 1997.

[DMadH93] Martin Dietzfelbinger and Friedhelm Meyer auf der Heide. Simple, efficient shared memory simulations (extended abstract). In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA-93)*, pages 110–119, 1993.

[GMR94] Leslie Ann Goldberg, Yossi Matias, and Satish Rao. An optical simulation of shared memory. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA-94)*, pages 257–267, New York, 1994. ACM Press.

[Gon81] Gaston H. Gonnet. Expected length of the longest probe sequence in hash code searching. *Journal of the ACM*, 28(2):289–304, 1981.

[KLMadH92] Richard M. Karp, Michael Luby, and Friedhelm Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC-92)*, pages 318–326. ACM Press, 1992.

[MadHSS95] Friedhelm Meyer auf der Heide, Christian Scheideler, and Volker Stemann. Exploiting storage redundancy to speed up randomized shared memory simulations. In *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS-95)*, volume LNCS 900, pages 267–278. Springer-Verlag, 1995.

[Mit96a] Michael Mitzenmacher. Density dependent jump markov processes and applications to load balancing. In *Proceedings of the 37th IEEE Symposium on Foundations (FOCS-96)*, pages 213–223, 1996.

[Mit96b] Michael Mitzenmacher. *On the Power of Two Choices in Randomized Load Balancing*. PhD thesis, 1996.

[Mit97] Michael Mitzenmacher. On the analysis of randomized load balancing schemes. In *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Arrays (SPAA-97)*, pages 292–301, 1997.

[MPR94] Philip. D. MacKenzie, C. Greg Plaxton, and Rajmohan Rajaraman. On contention resolution protocols and associated probabilistic phenomena. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC-94)*, pages 153–162, 1994.

[RS98] Martin Raab and Angelika Steger. Balls into bins - a simple and tight analysis. In *Proceedings of the 2nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM-98)*, volume LNCS 1518, pages 159–170, 1998.

[Ste96] Volker Stemann. Parallel balanced allocations. In *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA-96)*, pages 261–269, 1996.

[Vöc99] Berthold Vöcking. How asymmetry helps load balancing. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS-99)*, pages 131–140, 1999.