Asymmetric Balanced Allocation with Simple Hash Functions[*]

Philipp Woelfel[†]

Abstract

We show that for the asymmetric sequential allocation scheme of Vöcking (2003) one can use very simple hash functions. The hash functions we use are a straightforward extension of the hash functions introduced by Dietzfelbinger and Woelfel (2003). In order to evaluate a hash function a few arithmetic operations and table lookups suffice. Moreover, we show that the scheme has essentially the same behavior if the same balls are allowed to be inserted multiple times (i.e. they may be deleted and reinserted afterwards).

1 Balls Into Bins

The classical balls into bins process is the following: Suppose that there are n balls which are placed into n bins by sampling a random location for each ball independently. It is well known that once all balls are placed, the most heavily loaded bin contains approximately $\ln n/\ln\ln n$ balls with high probability. However, the maximum load can be improved significantly, if the balls are placed sequentially and each ball has $d \geq 2$ random locations to choose from. As shown by Azar, Broder, Karlin and Upfal (1999), if each ball is placed in the location which has the minimum load among d alternatives, then the maximum load is $\ln\ln n/\ln d+\Theta(1)$ w.h.p. (i.e. with probability at least $1-n^{-\alpha}$ for arbitrary $\alpha > 0$). Hence, already $d = 2$ yields an exponential decrease of the load of the fullest bin.

Many variants of the basic scheme have been analyzed, but one of the most significant improvements was devised by Vöcking (2003). He considered a variant where the n bins are split into d groups of size $r = n/d$ (for reasons of simplicity we assume that n is a multiple of d). It is convenient to give each group a unique number from 1 to d and to assume that the groups are laid out from left to right with increasing number. Now a ball is placed as follows: In each of the d groups, a bin is chosen independently and uniformly at random. Then the ball is placed into the bin which has the least load out of the d randomly chosen ones. If there are multiple such bins (with least load), then the ball is placed into the leftmost bin among them. The rather surprising result of Vöcking was that now the maximum load decreases to $\ln\ln n/(d\cdot\ln\phi_d)+O(1)$ w.h.p., where $1.61 < \phi_d < 2$ is a constant to be determined more precisely later. Moreover, contrary to many earlier results, Vöcking's analysis also covers continuing processes in which in each step one arbitrary ball may be inserted or deleted as long as there are never more than n balls in the system (however, it must be determined in advance which ball will be deleted at which time).

In applications the balls are often keys (imagine the balls to have labels from a given finite universe U) and their locations have to be determined by hash functions. This is obviously the case for hashing schemes (e.g. open hashing) which are among the most important applications of the balls into bins process.

Here all known results suffer from at least one of the following two obstacles: First, in many applications a ball may be removed from a bin and be reinserted again later. If we determine the ball's position by the hash value of its label, then it has exactly the same bin positions to choose from whenever it is reinserted. But most proofs of balls into bin processes do not work if the same balls may be reinserted (after being removed) multiple times. However, Cole, Frieze, Maggs, Mitzenmacher, Richa, Sitaraman and Upfal (1998) have considered a process in which labeled balls may be inserted and removed in such a way that at any time each ball label appears at most once in the system. When a ball is inserted for the first time, it is assigned two random locations and picks a random bin with minimal load. If a ball is later reinserted again, then it has the same two possible locations to choose a bin from. The authors show that if the insertion and deletion sequence is fixed in advance, at any time T the maximum load of a bin is at most $(4\ln\ln n)/\ln 2$ w.h.p. As for all other results it is not known, yet, whether it also works with simple hash functions using $o(n)$ space (see the discussion below). Moreover, it is not known whether the asymmetric scheme of Vöcking (whose maximum load is even for $d = 2$ better than the one mentioned above), works if reinsertions of the same ball are allowed.

The second obstacle is, that one needs true random hash functions which for real applications should be

[*]Research supported by DFG grant Wo 1232/1-1

[†]University of Toronto, Department of Computer Science, 10 King's College Road, Toronto, ON, M5S 3G4. E-mail: pwoelfel@cs.toronto.edu

easy to evaluate (i.e. in constant time). Although it was shown recently how to simulate (with a small error probability) hash functions behaving truly random on m keys (see Östlin and Pagh, 2003, and Dietzfelbinger and Woelfel, 2003), such schemes require at least $c \cdot m$, $c > 1$, additional space as well as $c \cdot m$ random words (e.g. in the construction of Östlin and Pagh $c > 8$). In particular if a continuous (dynamic) balls into bins process is investigated, m may be arbitrary large compared to the number of bins, n. However, it can be seen that for static balls into bins processes $O(\log n)$-wise independent random values suffice and there is a construction of a n^γ-wise independent hash family, $0 < \gamma < 1$, due to Siegel (2004), which uses n^ϵ space, $\epsilon > 0$, and whose hash functions can be evaluated in constant time. This "constant" evaluation time, though, is known to be huge and the construction is extremely impractical. Moreover, e.g. for the asymmetric scheme of Vöcking, it is not obvious, whether even n^γ-wise independence suffices for $\gamma < 1$. Since the scheme is not static, after a long sequence of insertions and deletions the maximum load may be influenced by the random choices of much more than n balls.

That the problem considered here is of practical relevance is indicated by an experimental work of Broder and Mitzenmacher (2001), who used the asymmetric scheme of Vöcking in order to improve IP lookups of IP routers. Due to the lack of efficient hash functions which are known to guarantee a small maximum load in Vöcking's allocation scheme, they used very simple functions based on the multiplication over a finite field. A theoretical analysis of an allocation scheme using such simple hash functions is not known.

One idea to overcome the limitations of efficient hash families in balanced allocation schemes was brought up very recently by Dietzfelbinger and Weidling (2005) – although in a different setting, where all balls may be rearranged whenever a new ball arrives. The authors propose to use the high-performance hash families devised by Dietzfelbinger and Meyer auf der Heide (1992) in order to first split the set of all balls into $n^{1/3}$ groups such that with high probability each group contains at most $(1+\epsilon)n^{2/3}$ balls. Then, each of the groups is assigned a unique set of $n^{2/3}$ bins and the allocation algorithm is applied separately for each of the groups. But for each group the same hash functions (which are chosen at random from high-performance hash families similar to those described by Dietzfelbinger and Meyer auf der Heide, 1992) are used. Then with high probability this scheme behaves in one group as if all (at most $(1+\epsilon)n^{2/3}$) balls in this group are distributed completely at random. Thus, the maximum load in this group can be bounded under the assumption of true randomness. If w.h.p. the maximum load in one group does not exceed a certain threshold, then w.h.p. in none of the groups this threshold is exceeded. It might be possible to use this "splitting technique" also in our setting of the balls into bins problem. However, while the splitting trick seems to work for most static allocation problems, it is not obvious, whether it also works in the dynamic case (note, that the allocation problem considered by Dietzfelbinger and Weidling is also a static one). The reason is, that in the dynamic case, one has to take all keys into account which end up in one group over time. Moreover, this splitting trick does of course not solve the problem of reinsertion of the same balls. Finally, due to the fact that "two levels" of hash functions have to be used (one for splitting, the other for the actual ball placing), and each of the levels requires high-performance hash functions, we assume that the hash families we chose here are more practical. However, it seems worth to elaborate this matter in future investigations.

2 The Result

We reanalyze the allocation scheme of Vöcking and prove that it has essentially the same behavior even if very simple random hash functions are used and if the same balls may be reinserted after being removed. Throughout this text U and $R = [r]$ are finite sets and $|R| \le |U|$. A family H of hash functions $U \to R$ is said to be k-wise independent, if for any k different keys $x_1, \dots, x_k \in U$ and a randomly chosen hash function $h \in H$ the hash function values $h(x_1), \dots, h(x_k)$ are uniformly and independently distributed. A family of hash functions $U \to R$ is said to be *uniform* if it is $|U|$-wise independent. For constants k many constructions of k-wise independent hash families are known whose functions can be evaluated efficiently (sometimes, k-wise independency is only achieved approximately, but this does not change any of our results). Simple examples are families consisting of polynomials of degree $k-1$ over finite fields (Wegman and Carter, 1979) or using integer arithmetics (Dietzfelbinger, 1996). In particular constructions where describe constructions where (approximate) k-wise independence is achieved with very few arithmetic operations and some table lookups are described by Thorup and Zhang (2004) and Dietzfelbinger and Woelfel (2003).

In the following we assume that $\mathcal{H}_r^k$ denotes a k-wise independent hash family of functions $U \to R$. We consider the following balls into bins process. There are n bins which are split into d groups of equal size r, where $r = n/d$ is an integer. The elements in U are called balls. A ball $x \in U$ may be inserted in a bin, be removed later and afterwards reinserted again.

However, it is not allowed to exist in the system twice at the same time and at any time there are at most n balls in the system. We use a hash function vector $\vec{h} = (h_1, \ldots, h_d)$, $h_i : U \to [r]$, in order to determine the balls possible locations. For a ball $x \in U$, the hash function value $h_i(x)$, $1 \leq i \leq d$, describes the choice of its bin in the ith group. A ball is alway placed in a bin which has the least load among the d possibilities. If among the d locations of a ball there are multiple bins with minimum load, then the ball is placed in the leftmost of them. The hash functions remain unchanged during the process, thus if a ball is reinserted it has the same locations to choose from as when it was inserted for the first time. However, the sequence of insertions and deletions must be fixed before the hash function vector is chosen at random.

We choose the hash function vector from a family $\mathcal{R}^k_{\ell,n}(d)$ which we formally define later. We shall discuss its properties now, though. Let $\vec{h} = (h_1, \ldots, h_d) \in \mathcal{R}^k_{\ell,n}(d)$. Then h_i is described by a hash function g (which is the same for all h_i) from a k-wise independent hash family $\mathcal{H}^k_\ell$, a hash function f_i from a k-wise independent hash family $\mathcal{H}^k_r$ and a random vector $(z_{i,1}, \ldots, z_{i,\ell})$. The function h_i is defined by $h_i(x) = (f_i(x) + z_{i,g(x)}) \bmod r$. Hence, the function values $h_1(x), \ldots, h_d(x)$ can be determined by evaluating $d+1$ hash functions from k-wise independent hash families (e.g. polynomials of degree $k-1$), d table lookups and d additions modulo r. The family $\mathcal{R}^k_{\ell,n}(d)$ is the straightforward generalization of the family of hash function pairs $\mathcal{R}^k_{\ell,n}(2)$ introduced by Dietzfelbinger and Woelfel (2003).

In order to state the result we need a generalization of the Fibonacci numbers. Let $F_d(j) = 0$ for $j \leq 0$ and $F_d(1) = 1$. For $j \geq 2$ we define recursively $F_d(j) = \sum_{i=1}^{d} F_d(j-i)$. Now let $\phi_d = \lim_{j\to\infty} F_d(j)^{1/j}$. The value ϕ_2 corresponds to the golden ration. Note that $1.61 < \phi_2 < \phi_3 < \ldots < 2$, and $d \ln(\phi_d) > (d-1) \cdot \ln 2$ (see Vöcking, 2003).

THEOREM 2.1. *Let α, κ and k be arbitrary constants and $2 \leq d = O(\log\log n)$. Further, let $\vec{h}$ be chosen randomly from $\mathcal{R}^{2k}_{\ell,n}(d)$. Suppose that balls are sequentially inserted into and removed from n bins according to the process described above, where the d locations of each ball are determined by $\vec{h}$. Then at any time T, the probability that the number of balls in the fullest bin exceeds*

$$\frac{\ln\log_2 n + \ln(1+\alpha)}{d \cdot \ln \phi_d} + e \cdot d + \kappa + 3$$

is at most $n^{-\alpha} + n^{1+o(1)}(n^{-\kappa} + \ell^{-k})$.

The maximum load we achieve in this theorem is the same as the one Vöcking achieves assuming true randomness, except for the additive term of $e \cdot d$. The difference in the probability bound is the additional term ℓ^{-k}. Choosing $\ell = n^{1-\epsilon}$ and k as a large enough constant yields an arbitrary small polynomial probability for encountering an overloaded bin, while the evaluation time for the hash functions remains constant. As long as $d = O(\sqrt{\log\log n})$, the maximum load of a bin is still bounded by $O\big((\log\log n)/d\big)$ with high probability. If d is a constant we even have the same maximum load as under true randomness assumption up to a constant additive term and it is known that using true randomness one cannot achieve a better maximum load with this scheme. Super-constant terms of d seem to be rather unrealistic for applications, especially if one has to evaluate d hash functions in order to place a ball. Moreover, having a super-constant value of d would require that the application fixes the number of hash functions after it becomes aware of the number of balls and bins.

In order to give an example of the efficiency of our hash functions, let us choose $k = 2$ and $\ell = n^{1-\epsilon/2}$ for some $0 < \epsilon < 1$. Then the evaluation time for the hash functions is dominated by the functions from the 4-wise independent hash families. These functions could simply be polynomials of degree 3. An even more efficient alternative are the functions from Thorup and Zhang (2004), which were shown to map e.g. double words into single words by three table lookups, one addition and one bitwise operation. The space requirements for the hash function vector is $O(d \cdot n^{1-\epsilon/2}) = o(n)$ and with a probability of $n^{-1+\epsilon+o(1)}$ the maximum load does not exceed $(\ln\ln n)/(d \ln \phi_d) + O(1)$ for constant d.

In the following section, we first prove the result under the assumption that the hash functions are chosen from a uniform hash family. In Section 4, we then show how to modify this analysis in order to adapt for the simple hash functions defined there.

3 Analysis under the Uniform Hashing Assumption

Say that at time T we want to determine the maximum load of a bin. Let $S_t \subseteq U$, $1 \leq t \leq T$, be the set of all balls in the system at time t and let $\vec{S} = (S_1, \ldots, S_T)$ be the *vector of active ball sets.* Throughout this section we assume that the d hash functions $h_1, \ldots, h_d$ are chosen independently at random from a uniform hash family, i.e. the distribution of all balls into the bins is completely random. Let $\vec{h} = (h_1, \ldots, h_d)$.

We consider a directed graph $F(\vec{S}, \vec{h})$ which is uniquely determined by $\vec{S}$ and $\vec{h}$. Each node is labeled with a time t and with a ball $x \in S_t$. There is a directed edge labeled i, $1 \le i \le d$, from a node (x,t) to a node (x',t') if and only if $h_i(x) = h_i(x')$. We call the event that there is an edge labeled i from a node (x,t) to a node (x',t') an *edge event*. Obviously, for $x \neq x'$ such an edge event (for fixed i) occurs with a probability of $1/r = d/n$. However, edge events may not be independent if they occur between several nodes labeled with the same ball.

Witness Trees We define a so-called *pruned full witness tree* (short: PFWT), which exists as a subgraph in $F(\vec{S},\vec{h})$ if there is a bin with large load. We start first with a *witness tree* in which the same ball label may appear on multiple nodes. We then merge several witness trees to a *full witness tree*. Finally, we *prune* the full witness tree in such a way that appearances of the same ball in the resulting subtree is restricted in an appropriate way.

The construction of the non-pruned witness tree is similar to that of Vöcking. One difference concerns the leaves of the witness tree and is necessary for the analysis for the simple hash functions. In addition we have to be more careful about the exact times at which certain balls are present in order to be able to make the proof work for reinsertions of the same balls. The pruned witness trees have to be modified more significantly in order to analyze the scheme for reinsertions.

In the following let $\mu = \mu(d)$ be an integer depending on d, to be determined later. Suppose that at time t ball x lies in a bin b at a height of ℓ, i.e. there are $\ell - 1$ balls below x. Let i be the group of b and let t' be the time before t when x was inserted in b (i.e. $x \notin S_{t'-1}$ and $x \in S_{t'} \cap \ldots \cap S_t$). Let $b_1, \ldots, b_d$ be the d possible locations of x, where b_j is in group j. Then at time t bin b $(= b_i)$ has at least $\ell - 1$ balls below it. Moreover at time t' *all* bins had a height of at least $\ell - 1$ because if there were a bin of height less than $\ell - 1$, then x would have been placed in this bin instead of in bin b_i. Finally, at time t' the bins $b_1, \ldots, b_{i-1}$ even had a height of at least ℓ because otherwise one of these bins would have been chosen for x due to the preference of groups further to the left.

We now define the witness tree $W_{t,\ell}(b)$ recursively. In this tree each edge will be directed from a parent to its child. Let $I(t, \ell, b, x)$ be the incident that at time t ball x is located in bin b at height ℓ (i.e. it has $\ell - 1$ balls below it). The following definition ensures that for any witness tree $W_{t,\ell}(b)$ with a root labeled (x,t) incident $I(t, \ell, b, x)$ occurs (in italics we describe the properties of the bins $b_1, \ldots, b_d$ ensuring this). Assume that for some fixed t, ℓ, b, x, incident $I(t, \ell, b, x)$ occurs. The root of $W_{t,\ell}(b)$ is a node u labeled (x,t).

Case 1: $\boldsymbol{\ell \le \mu}$. $W_{t,\ell}(b)$ consists only of the vertex u.

Case 2: $\boldsymbol{\ell = \mu + 1}$ and $\boldsymbol{1 \le i \le 2}$. (*At time t' each bin b_j, $j \neq i$, contains at least μ balls and at time t bin b_i contains at least μ balls below x.*)
The root u of $W_{t,\ell}(b)$ has $\mu \cdot d$ children, namely the roots $u_{j,\ell'}$ of the witness trees $W_{t,\ell'}(b_i)$ and $W_{t',\ell'}(b_j)$ for $1 \le j \le d$, $j \neq i$, and for $1 \le \ell' \le \mu$. Each edge $(u, u_{j,\ell'})$ is labeled j.

Case 3: $\boldsymbol{\ell = \mu + 1}$ and $\boldsymbol{3 \le i \le d}$. (*At time t' each bin b_j, $1 \le j < i$, contains at least $\mu + 1$ balls.*)
The root u of $W_{t,\mu+1}(b)$ has $i - 1$ children $u_1, \ldots, u_{i-1}$. These are the roots of $W_{t',\mu+1}(b_j)$ for $j = 1, \ldots, i-1$ and each edge (u, u_j) is labeled j.

Case 4: $\boldsymbol{\ell > \mu + 1}$. (*At time t' each bin b_j contains at least ℓ balls for $j < i$ and at least $\ell - 1$ balls for $j > i$. At time t the bin b_i contains at least $\ell - 1$ balls below ball x.*)
The root u of $W_{t,\ell}(b)$ has d children $u_1, \ldots, u_d$. The node u_j is the root of $W_{t',\ell}(b_j)$ for $j = 1, \ldots, i-1$ and the root of $W_{t',\ell-1}(b_j)$ for $j = i+1, \ldots, d$. The node u_i is the root of $W_{t,\ell-1}(b_i)$. Each edge (u, u_j) is labeled j.

It is crucial for our proof that for two adjacent nodes labeled (x,t) and (x',t'), if x and x' are in the same group, then $t = t'$.

Properties of a Witness Tree Consider a witness tree $W = W_{T,L+\mu+1}(B)$ for a fixed bin B and a fixed time T. We call L the *order* of the witness tree. Consider an edge (u, u^*) labeled i^* in the tree W, where u is labeled (x,t) and u^* is labeled (x^*, t^*). Assume that ball x is located in a bin in group i at time t. Then by construction the following holds:

(W1) $x \neq x^*$ and $h_{i^*}(x) = h_{i^*}(x^*)$.

(W2) $i = i^* \;\Rightarrow\; t = t^*$.

(W3) $x \in S_{t^*} \cap \cdots \cap S_t$ and $x^* \in S_{t^*-1} \cap S_{t^*}$.

(W4) $i \neq i^* \;\Rightarrow\; x \notin S_{t^*-1}$.

The statement $x^* \in S_{t^*-1}$ of property (W3) may need a justification: According to the construction of the witness tree, if $i \neq i^*$, then the insertion of x is the operation that leads from the set S_{t^*-1} to the set S_{t^*}. I.e., $S_{t^*} = S_{t^*-1} \cup \{x\}$. Since $x^* \in S_{t^*}$ it follows that x^* is also in S_{t^*-1}. If $i = i^*$, then by construction of the witness tree at time t^* ball x^* is below ball x in the same bin in group i. Hence, ball x^* must have already been in the system before time t^*.

Now fix a vector of active ball sets $\vec{S} = (S_1, \ldots, S_T)$ and a hash function vector $\vec{h} = (h_1, \ldots, h_d)$. Let

$W = W_{T,\ell}(B)$ be a witness tree obtained from $\vec{S}$ and $\vec{h}$ for some bin B and a height ℓ at time T. Consider an arbitrary node u in W with ball label x. Then the time label t of u is uniquely determined by its parent or T: It is T for the root. If u is not the root, then it is determined by the label (x_p, t_p) of the parent u_p of u as follows. If the balls x and x_p are placed in the same bin, then the time labels are equal (i.e. $t = t_p$). Otherwise, t is the maximum value satisfying $t \leq t_p$ and $x_p \notin S_{t-1}$.

Moreover, the group in which x is located at time t is uniquely determined by the edge pointing to u: If u has no incoming edge, i.e. u is the root, then it is the group of B. Otherwise it is the jth group, where j is the label of the edge pointing to u. Finally, the bin b in which x is placed is uniquely determined by the edge pointing to u and $\vec{h}$: It is either B in case u is the root and it is $h_j(x)$ otherwise.

Since in a witness tree for any node u labeled (x,t) the bin b where x is located at time t as well as the group i of this bin are uniquely determined, it is justified to say that b is the bin of u and i is the group of u.

We call the nodes which are parents of leaves in the witness tree (i.e. the roots of $W_{t,\mu+1}(b)$, where b is a bin in group 1 or 2) *border nodes.* If we remove all leaves from a witness tree (i.e. the border nodes become leaves) then the resulting tree has the same topology as that of the *asymmetric witness tree* considered by Vöcking (2003). This topology is that of a Fibonacci tree $T_d(k)$ defined recursively as follows: $T_d(1)$ and $T_d(2)$ consist of a single node. $T_d(k)$, $k > 2$, is a rooted tree whose root has as children the roots of $T_d(k-1), \ldots, T_d(\max\{k-d, 1\})$. It is easy to see that if we remove all leaves from a witness tree $W_{T,L+\mu+1}(B)$, then we obtain a tree with the same topology as the Fibonacci Tree $T_d(L \cdot d + i)$, where i is the group of B.

Let q be the number of border nodes, m be the number of inner nodes and ℓ be the number of leaves in a witness tree $W = W_{T,L+\mu+1}(B)$. The following properties of W can be easily derived from the topology of the Fibonacci Trees (see also Vöcking, 2003):

1. $q = F_d(L \cdot d + i) \geq (\phi_d)^{d \cdot L + i - 2} \geq (\phi_d)^{d \cdot L - 1}$.
2. $m \leq 2q$.
3. $\ell = \mu \cdot d \cdot q$.
4. The depth of W is $O(L \cdot d)$.

Distinct Balls We consider first the case that there is a witness tree $W_{T,L+\mu+1}(b)$ whose ball labels are all distinct. This ensures that all edge events are independent. We bound the probability that such a witness tree with a root labeled with a ball in S_T can be embedded in $F(\vec{S}, \vec{h})$. There are n balls in S_T to choose the root from. Now consider a root u of a sub tree $W_{t,\ell}(b)$ with ball label x. Assume first that u is not a border node and let $u_1, \ldots, u_d$ be the d children of u. Recall that for each j, $1 \leq j \leq d$, the time label t_j is uniquely determined by u. For each node u_j with ball label x_j it holds $x_j \in S_{t_j}$ and thus there are at most n possibilities for each ball label of a child u_j. If u is a border node, then it has $\mu \cdot d$ children whose balls are all in $S_{t'}$ and distinct, where t' is the insertion time of x. Hence, there are $\binom{n}{\mu \cdot d}^q$ possibilities to choose the ball labels of all leaves and d^ℓ possibilities to label the edges pointing to the ℓ leaves with values $i \in \{1, \ldots, d\}$. The witness tree has $\ell + m - 1$ edges and each edge exists with a probability of d/n. To conclude, the probability that a witness tree $W_{T,L+\mu+1}(b)$ exists is at most

$$\begin{aligned} n^m \cdot \binom{n}{\mu d}^q \cdot d^\ell \cdot (d/n)^{\ell+m-1} & \\ \leq\ (en/\mu d)^{\mu dq} \cdot d^{2\ell+m-1} \cdot n^{-\ell+1} & \\ \leq\ n \cdot (e/\mu)^{\mu dq} \cdot d^{\mu dq + m - 1}\ \leq\ n \cdot d^{2q} \cdot \left(\frac{ed}{\mu}\right)^{\mu dq}. & \end{aligned}$$

We now choose $\mu = ed + 2$ and obtain a probability bound of

$$\begin{aligned} n \cdot d^{2q} \cdot \left(1 - \frac{2}{\mu}\right)^{\mu dq} &\leq\ n \cdot d^{2q} \cdot (1/e)^{2dq} \\ &<\ n \cdot 2^{-q}\ \leq\ n \cdot 2^{-(\phi_d)^{d \cdot L - 1}} \end{aligned}$$

(using $d \geq 2$ and $e^{-2d} \leq d^{-3}$). This is exactly the same probability bound as Vöcking's. However, our "root-ball" has height $L + \mu + 1$ while Vöcking's has height $L + 4$. Therefore we get the following result analogue to Vöcking's: For $L \geq (\ln \log_2 n + \ln(1+\alpha)/(d \cdot \ln \phi_d) + 1$, the probability for the existence of a witness tree of order L with distinct balls is at most $n^{-\alpha}$.

Pruned Witness Trees Eventually, the witness tree as described above contains multiple nodes labeled with the same ball. In this case, the edge events are not independent and we have to prune some of them. In order to handle this, we first have to merge multiple witness trees together to a full witness tree. Assume that at time T there is a bin b containing at least $L + \mu + \kappa + 2$ balls. We may assume w.l.o.g. that the topmost ball of bin b is inserted at time T. Further we assume w.l.o.g. that b is in the first (leftmost) group, because after inserting the topmost ball in a bin with highest load, one of the bins with highest load is always in the leftmost group.

Consider the $\kappa+1$ topmost balls $x_0, \ldots, x_\kappa$ (ordered top-down) in bin b and let b_i, $1 \leq i \leq \kappa$, be the bin in the second group in which x_i might have been placed. Note that each ball x_i has a height of at least $L+\mu+2$. Then the root of the full witness tree is a node labeled (x_0, T) which has κ children labeled $(x_1, T), \ldots, (x_\kappa, T)$. All edges are labeled 1. In addition each of the nodes labeled (x_i, T) has as a child the root of the witness tree $W_{t_i, L+\mu+1}(b_i)$, where t_i+1 is the insertion time of x_i and b_i is determined by $h_2(x_i)$ in the second group (consequently the corresponding edge is labeled 2). The root of $W_{T,L+\mu+1}(b_i)$ is guaranteed to exist, because by construction at the insertion time of x_i the bin b_i must have had at least $L+\mu+1$ balls (otherwise x_i would have been placed in bin b_i rather than in bin b).

We now have to prune the full witness tree in order to ensure that all edge events are independent. We do this by finding prunable node pairs during a depth first search. (Note that Vöcking uses a breadth first search, but this doesn't yield the desired probability bounds in our case, because we might not encounter enough border nodes.)

Definition 3.1. Let u and u' be different nodes with labels x and x' (possibly $x = x'$). The ordered pair (u, u') is called *prunable*, if there is $1 \leq j \leq d$ such that $h_j(x) = h_j(x')$ and the in-edge of u' is not labeled j.

In order to remove prunable nodes, we do a depth first search through the full witness tree and inspect each node. Whenever we inspect a node u', we check whether there was another node u inspected before such that (u, u') is a prunable node pair. If this is the case, we *prune* the edge (p, u'), where p is the parent of u'. This means that we cutoff the complete subtree rooted at u' and mark the node u' as a *pruning node* (the node u' and the edge (p, u') remain in the tree).

Once we have marked κ pruning nodes, we stop the depth first search early and remove all non-visited nodes and edges incident to them from the remaining witness tree. The resulting tree is the *pruned full witness tree* (short: PFWT) *of order L*. If we do not find κ pruning nodes during the depth first search, then obviously one of the witness trees $W_{t_i, L+\mu+1}(b_i)$, $1 \leq i \leq \kappa$, contains no pruning node. In this case we can bound the probability under the assumption of distinct balls for the witness tree $W_{t_i, L+\mu+1}(b_i)$ (we prove below that the balls of such a witness tree without pruning nodes are distinct).

In the following we distinguish between *vertical* and *diagonal* edges in the witness tree. An edge (p, u') is vertical, if the ball associated with p is located in the same group as the ball associated with u'. Otherwise, the edge is diagonal. Note that if p is not the root, then the edge (p, u') is vertical if and only if the edge pointing to p has the same label as the edge (p, u').

Claim 3.2. *If there is a directed path from a node v_1 to a different node v_2 such that v_1 and v_2 are labeled with the same ball, then this path contains at least one diagonal edge other than the first edge on the path.*

Proof. Let v_1 and v_2 be two nodes labeled (x_1, t_1) and (x_2, t_2), respectively. Assume first that there is a directed path of vertical edges leading from v_1 over $u_1, \ldots, u_s$ to v_2. Hence, the groups of all nodes $u_1, \ldots, u_s$ on this path are equal and thus the groups of v_1 and v_2 are equal. Let the group of all the nodes $v_1, u_1, \ldots, u_s, v_2$ be i. It follows from the properties of the witness tree (W2) that the time labels of all nodes on this path are equal and in particular $t_1 = t_2$. If $y_1, \ldots, y_s$ are the ball labels of $u_1, \ldots, u_s$, then property (W1) ensures that $h_i(x_1) = h_i(y_1) = \cdots = h_i(y_s) = h_i(x_2)$. Hence, x_1 and x_2 exist in the same bin at different heights at the same time t_1. This implies $x_1 \neq x_2$.

Now assume that the first edge (v_1, u_1) on the directed path $(v_1, u_1, \ldots, u_r, v_2)$ is a diagonal edge and there are no other diagonal edges on this path. Then with the same argument as above, the time label of u_1 is the same as that of v_2, that is t_2. Since the edge (v_1, u_1) is diagonal it follows from (W4) that $x_1 \notin S_{t_2-1}$. On the other hand, $x_2 \in S_{t_2-1}$ according to (W3). Hence, x_1 and x_2 are different balls. ■

Claim 3.3. *Assume that there is a directed path $u_1, \ldots, u_k$ of nodes with ball labels $x_1, \ldots, x_k$ such that all edges $(u_1, u_2), \ldots, (u_{k-1}, u_k)$ of this path have the same label i. Then $h_i(x_1) = \cdots = h_i(x_k)$ and all balls $x_1, \ldots, x_k$ are disjoint.*

Proof. Each in-edge of a node u_j determines the group in which the corresponding ball x_j is and according to the witness tree property (W1) an edge (u_j, u_{j+1}) labeled i implies $h_i(x_j) = h_i(x_{j+1})$. Hence, all balls but possibly x_1 are in the ith group. Therefore, all edges but possibly the first one are vertical. According to Claim 3.2 the balls are disjoint. ■

We are now ready to derive a lemma which allows us to conclude that the edge events in a PFWT are independent.

Lemma 3.1. *In the PFWT any two different nodes labeled with the same ball are pruning nodes and have different groups.*

Proof. Let v and v' be two nodes in the PFWT, both labeled with the same ball x. Assume w.l.o.g. that v is inspected before v'. Let i and i' be the groups of v and v', respectively. Further, let $p(v)$ be the topmost node in the tree such that there is a directed path leading from

$p(v)$ to v containing only i-edges. Let $p(v')$ be defined analogously for v' and i'-edges. Finally, let y and y' be the labels of $p(v)$ and $p(v')$, respectively. Since v' is obviously not the root, $p(v') \neq v'$.

Assume that $i = i'$ contrary to the claim of the lemma. If $p(v)$ and $p(v')$ were the same node, then there would be a directed path from $p(v) = p(v')$ to v' passing over v. Hence, there would be a directed path from v to v' consiting entirely of edges labeled i and the ball labels of v and v' would differ according to Claim 3.3. Since this contradicts our assumption, we know that $p(v) \neq p(v')$.

Assume first that v is not the root and thus $p(v) \neq v$. By Claim 3.3 we obtain $h_i(y) = h_i(x) = h_i(y')$. Due to the fact that neither the in-edge of $p(v)$ nor the in-edge of $p(v')$ is labeled i, it follows that $\big(p(v), p(v')\big)$ and $(p(v'), p(v))$ are prunable pairs. Hence, either $p(v)$ or $p(v')$ must be a pruning node (whichever was inspected last) and thus either v or v' does not exist in the PFWT. If v is the root, we still have $h_i(x) = h_i(y')$. In this case $\big(v, p(v')\big)$ is a prunable pair and $p(v')$ is a pruning node (it obviously was inspected after the root v) and v' does not exist in the PFWT. In any case, $i = i'$ contradicts the existence of either v or v' in the PFWT and thus v and v' have different groups.

It remains to show that v and v' are both pruning nodes. We have $y' \neq x$ but $h_{i'}(y') = h_{i'}(x)$ according to Claim 3.3. Since the in-edge of $p(v')$ is by construction not labeled i' and the in-edge of v (if v has an in-edge) is labeled $i \neq i'$, it follows that $\big(v, p(v')\big)$ and $\big(p(v'), v\big)$ are prunable pairs (clearly $v \neq p(v')$ due to $x \neq y'$). Then $p(v')$ is inspected before v because otherwise it would be a pruning node and v' would not exist. Hence, v is a pruning node. It is obvious that v' becomes a pruning node once it is inspected because (v, v') is a prunable pair and $i \neq i'$) and v' is inspected after v. ■

Probability of the existence of a PFWT By construction all pruning nodes in the PFWT are leaves. Consider some (not necessarily distinct) inner nodes $u_1, \ldots, u_r$ with labels $y_1, \ldots, y_r$ such that u_i has as a child a pruning node v_i and all the pruning nodes v_i are distinct but labeled with the same ball x. According to Lemma 3.1 the groups $i_1, \ldots, i_r$ of the nodes $v_1, \ldots, v_r$ are different. Hence, the events $h_{i_1}(y_1) = h_{i_1}(x), \ldots, h_{i_r}(y_r) = h_{i_r}(x)$ are all independent. Now it is easy to see that in the PFWT all edge events are independent.

Recall that the full witness tree of order L consists of the root u_0 labeled (x_0, T), its children $u_1, \ldots, u_\kappa$, labeled $(x_1, T), \ldots, (x_\kappa, T)$, and that the node u_i, $1 \leq i \leq \kappa$, has as a child the root of the witness tree $W_{t_i, L+\mu+1}(b_i)$, where t_i is the insertion time of x_i and b_i is in the second group.

We now bound the probability that a PFWT of order L with exactly κ pruning nodes exists. Let N be the total number of nodes in the full (unpruned) witness tree. The number of topologies for a PFWT with κ nodes marked as pruning nodes is at most N^κ. We fix such a topology. Let M be the number of nodes in the PFWT and let m be the number of inner nodes in the PFWT. The set of nodes is partitioned into four groups: pruning nodes, border nodes, full leaves and normal nodes. A node is a border node, if it has μd non-pruning nodes as children which are themselves leaves. The number of border nodes is denoted q. The children of border nodes are called full leaves and their number is ℓ. Hence, the number of normal nodes is $M - \kappa - q - \ell$.

In order to bound the probability that a PFWT with the fixed topology occurs we first pick a root ball $X \in S_T$ and then do a depth first search through the PFWT picking a ball for each node we visit. At each step we bound the probability that the ball we picked may be a child of its parent (i.e. that the corresponding edge event occurs) by d/n and the probability that it is a pruning node, if according to the topology it is supposed to be.

Whenever we visit the next node u, the label (x, t) of its parent has already been determined. Let i be the label of u's in-edge. Recall that the time label t' of u is uniquely determined by its parent. Consider first the case that according to the topology of the PFWT u is not a pruning node. We choose for the ball label of u a ball $x' \in S_{t'}$ (hence there are n possibilities to choose x') and the probability that the corresponding edge event occurs is d/n. This event has to occur for all normal and all border nodes but not for the root (hence for $M - \kappa - \ell - 1$ nodes). In addition, if u is one of the q border nodes, we choose the balls of all its $\mu \cdot d$ children at once. They are all in $S_{t'}$, where t' is the insertion time of x', and are disjoint (because otherwise by Lemma 3.1 at least one of the nodes would be a pruning node). Hence, there are at most $\binom{n}{\mu d}$ possibilities to choose the balls and at most $d^{\mu d}$ possibilities to label the in-edges of the nodes with values in $\{1, \ldots, d\}$. The probability that all corresponding edge events occur is $(d/n)^{\mu d}$. Hence, using $\mu = ed + 2$, the probability that a border node has μd appropriate children is at most

$$\binom{n}{\mu d} \cdot d^{\mu d} \cdot \left(\frac{d}{n}\right)^{\mu d} \leq \left(\frac{en}{\mu d}\right)^{\mu d} \cdot \frac{d^{2\mu d}}{n^{\mu d}}$$
$$= \left(\frac{ed}{\mu}\right)^{\mu d} \leq (1 - 2/\mu)^{\mu d} \leq e^{-2d}.$$

Now consider the case that u is one of the κ pruning nodes and let i be the label of u's in-edge. We distinguish the case that the label x' of u exists somewhere else in the PFWT from the case that u is the only node with label x'. There are less than κ possibilities to choose the ball x' in such a way that another pruning node with the same label has already been visited before. As with normal nodes, we here just use the probability bound d/n of the corresponding edge event $h_i(x) = h_i(x')$. Now assume that x' is not the label of any other previously visited pruning node. Then according to Lemma 3.1 there also is no other previously visited (non-pruning) node with label x'. Hence, the hash values $h_j(x')$ with $1 \le j \le d$ and $i \ne j$ are random values independent from all other random choices made so far. Therefore, the probability that the corresponding edge event occurs and that x' becomes a pruning node (i.e., that for some other previously chosen ball x'' we have $h_j(x') = h_j(x'')$ for some $j \ne i$) is at most $(d/n) \cdot M(d-1)d/n \le M \cdot d^3/n^2$. We count at most n possible choices for the ball x'.

To conclude, the probability that we can choose all balls in such a way that we obtain a PFWT with one of the N^κ possible topologies is at most

$$\begin{aligned} p \;:=\; & N^\kappa \cdot n \cdot \left(n \cdot \frac{d}{n}\right)^{M-\kappa-\ell-1} \cdot \left(e^{-2d}\right)^q \\ & \cdot \left(\kappa \cdot \frac{d}{n} + \frac{n \cdot M \cdot d^3}{n^2}\right)^\kappa \\ \le\; & n \cdot d^{M-\ell-1} \cdot e^{-2dq} \cdot \left(\frac{\kappa \cdot N + N \cdot M \cdot d^2}{n}\right)^\kappa . \end{aligned}$$

We now restrict ourselves to $d = O(\log\log n)$, $\kappa = O(1)$ and $L = O\big(\log n/(\log\log n)^3\big)$. Recall that $\mu = O(d)$ and that the depth of a witness tree of order L is $O(Ld)$. Hence, the depth of the PFWT is also $O(Ld)$. Since all non-border nodes have at most d children, we have

$$M \le N \le d^{O(Ld)} = (\log\log n)^{O(\log n/(\log\log n)^2)} = n^{o(1)}.$$

It is easy to see that a leaf can only be a non-full leaf for two reasons: Either itself or one of its siblings (i.e. one of its parents other children) is the last node visited during the depth first search or itself or one of its siblings is a pruning node. Therefore, there are at most $(\kappa+1) \cdot \mu \cdot d = O(d^2)$ leaves which are not full leaves. Hence, the number of inner nodes is $m = M - \ell - O(d^2)$. Similarly, there are at most $\kappa + 1 = O(1)$ parents of leaves which are not border nodes. Due to the fact that the inspection was done in a depth first order, there are always at most $O(Ld)$ nodes with a single child (recall that $O(Ld)$ is a bound on the depth of the tree). Hence, $2q \ge m - O(Ld)$. Therefore,

$$\begin{aligned} d^{M-\ell-1} \cdot e^{-2dq} \;&\le\; d^{m+O(d^2)} \cdot e^{-dm+O(Ld^2)} \\ =\; e^{(\ln d - d)m + O((L+\ln d)\cdot d^2)} \;&\le\; e^{O(\log n/\log\log n)} \\ &= n^{o(1)}. \end{aligned}$$

Hence, we obtain for the probability p that there is a PFWT of order L and with exactly κ pruning nodes:

$$p \;=\; n^{1+o(1)} \cdot \left(\frac{n^{o(1)}}{n}\right)^\kappa \;=\; n^{-\kappa+1+o(1)}.$$

Combining this with the bound of $n^{-\alpha}$ for the probability that a witness tree of order L without pruning nodes exists, we obtain the following result.

THEOREM 3.1. *Let α and κ be constants and $2 \le d = O(\log\log n)$. Under the assumption that all hash functions $h_1, \dots, h_d$ are chosen independently at random from a uniform hash family, the probability that at time T the load of the maximum bin exceeds*

$$\frac{\ln\log_2 n + \ln(1+\alpha)}{d \cdot \ln\phi_d} + e \cdot d + \kappa + 3$$

is at most $n^{-\alpha} + n^{-\kappa+1+o(1)}$.

The additive term in the above maximum load is by $e \cdot d$ larger than the corresponding result of Vöcking. However, the difference to the proof of Vöcking's is that our bound is obtained only by analyzing subgraphs of $F(\vec{S}, \vec{h})$. More precisely, our proof is equivalent to considering all possible sets V of pairs (x,t) with $x \in S_t$ and summing up the probabilities that the graph spanned by these balls and the random hash function vector $\vec{h}$ is a witness tree of order L and with a root ball in S_T. This is crucial for the next section where we show that our bound remains valid for simple hash functions.

4 Analysis for Simple hash functions

Recall that $\mathcal{H}_r^k$ is a k-wise independent hash family of functions mapping U to R. We now define our family of hash function vectors.

DEFINITION 4.1. Let $k \ge 2$ and $r, \ell \in \mathbb{N}$. For $f \in \mathcal{H}_r^k$, $g \in \mathcal{H}_\ell^k$ and $z = (z_0, \dots, z_{\ell-1}) \in [r]^\ell$ the hash function $h_{f,g,z} : U \to [r]$ is defined by $x \mapsto \big(f(x) + z_{g(x)}\big) \bmod r$. The family $\mathcal{R}_{\ell,r}^k(d)$ consists of all hash function vectors $\vec{h} = (h_1, \dots, h_d)$ where $h_i = h_{f_i,g,z^{(i)}}$ with $f_i \in \mathcal{H}_r^k$, $g \in \mathcal{H}_\ell^k$ and $z^{(i)} \in [r]^\ell$ for $1 \le i \le d$.

It is important for our proofs, that all hash functions h_i, $1 \le i \le d$ of a hash function vector $\vec{h}$ share the same g-function.

We use the following two claims.

Claim 4.2. *Let $V \subseteq U$ and $\vec{h} = (h_1, \ldots, h_d) \in \mathcal{R}^{2k}_{\ell,n}(d)$. Under the condition that g satisfies $|V| \le \max\{2k,\ |g(V)| + k\}$, the hash function values $h_i(x)$ with $x \in V$ and $1 \le i \le d$ are all uniformly and independently distributed.*

Proof. If $|V| \le 2k$, then the claim is obvious because all hash functions f_i, $1 \le i \le d$, are chosen independently from a $2k$-wise independent hash family. Now assume $2k < |V| \le |g(V)| + k$. Let V' be a maximal subset of V such that g is injective on V'. Now let V_1 be the union of $V - V'$ and the set of elements $x' \in V'$ for which there is $x \in V - V'$ such that $g(x) = g(x')$. Since $|V - V'| \le k$ and g is injective on V', there are at most k elements $x' \in V'$ which are added to V_1. Hence, $|V_1| \le 2k$. It follows that all hash function values $h_i(x)$ with $x \in V_1$ and $1 \le i \le d$ are uniformly distributed because f_i is chosen from a $2k$-wise independent hash family. Now let $V_2 = V - V_1$. Since V_2 is a subset of V', g is injective on V_2. All hash function values $h_i(x)$ with $x \in V_2$ and $1 \le i \le d$ are uniformly and independently distributed, because each of them is determined by a unique offset $z^{(i)}_{g(x)}$. Moreover, $g(V_1) \cap g(V_2) = \emptyset$ and thus these offsets do not influence the hash function values of elements in V_1. Hence, all hash function values for $x \in V$ are distributed independently and uniformly. ∎

Claim 4.3. *Let $V \subseteq U$ and $\vec{h} = (h_1, \ldots, h_d) \in \mathcal{R}^{2k}_{\ell,n}(d)$. The probability that $|V| \ge \max\{2k,\ |g(V)| + k\}$ is at most $O\Big(|V|^{2k} \cdot \ell^{-k}\Big)$.*

Proof. Assume that $|V| \ge \max\{2k,\ |g(V)| + k\}$. We first show that there is a subset $V' \subseteq V$ such that $|V'| = 2k$ and $|g(V')| \le k$.

Obviously, $|V| \ge 2k$. If $|V| = 2k$, then $|g(V)| \le k$ and we may choose $V' = V$. If $|V| > 2k$, then $|g(V)| = |V| - k > |V|/2$. In this case there is a key x which collides with no other key $x' \in V$ (i.e. $g(x) \notin g(V - \{x\})$). Hence, if we remove this key x from V, then for the resulting set $V' = V - \{x\}$ we still have $|g(V')| = |V'| - k$. Therefore, we may iteratively remove keys until we obtain a subset $V' \subseteq V$ with $|V'| = 2k$ and $|g(V')| = |V'| - k = k$.

There are $\binom{|V|}{2k}$ possibilities to choose $V' \subseteq V$. Since g maps V' to $[\ell]$, there are $\binom{\ell}{k}$ possibilities to choose k hash function values in $[\ell]$. The probability that a key $x \in V'$ is mapped to one of these k positions is k/ℓ. Since g is chosen from a $2k$-wise independent hash family, the probability that this is the case for all $2k$ keys in V' is $(k/\ell)^{2k}$. Hence, we obtain the following upper bound for the probability that $|g(V)| \le |V| - k$, where $s = |V| \ge 2k$:

$$\binom{\ell}{k} \cdot \binom{s}{2k} \cdot (k/\ell)^{2k} \le \left(\frac{e \cdot \ell}{k}\right)^k \cdot \left(\frac{e \cdot s}{2k}\right)^{2k} \cdot \left(\frac{k}{\ell}\right)^{2k} \le \frac{s^{2k}}{\ell^k} \cdot \left(\frac{e^3}{4k}\right)^k = O\left(\frac{s^{2k}}{\ell^k}\right). \qquad ∎$$

We can now prove the main theorem. Choose $\vec{h} \in \mathcal{R}^{2k}_{\ell,n}(d)$ at random and distribute the balls according to $\vec{h}$. Let as before $\vec{S} = (S_1, \ldots, S_T)$, where $S_t \subseteq U$ is the set of balls in the system at time t. We bound the probability that $F(\vec{S}, \vec{h})$ has as a subgraph a PFWT of order L with the root ball in S_T.

Assume that W is such a subgraph. We use a depth first search in order to inspect each node of W. Whenever we inspect a new node, we add its ball label to the set V (which is empty at the beginning). We do this until either we have inspected the complete PFWT or until the set V of already visited balls satisfies $|V| = \max\{2k,\ |g(V)| + k\}$. At the beginning we obviously have $|V| = 0 < \max\{2k,\ |g(V)| + k\}$ and in each step $|V|$ increases by 1 and $|g(V)|$ may increase by 1 but may also remain unchanged. Hence, at each step either the inequality $|V| < \max\{2k,\ |g(V)| + k\}$ remains true or we obtain a set which yields equality.

Assume first that the set V of all balls labeling nodes in the witness tree satisfies $|V| < \max\{2k,\ |g(V)| + k\}$. In this case, according to Claim 4.2 the hash function vector $\vec{h}$ distributes all balls in V completely at random. Hence, the probability that the corresponding subgraph of $F(\vec{S}, \vec{h})$ forms a PFWT of order L can be bounded exactly as if $\vec{h}$ was chosen from a uniform hash family.

Now assume that we stop the depth first search with a set V of size $M = \max\{2k,\ |g(V)| + k\}$. Then according to Claim 4.2 the hash function values $h_i(x)$, $x \in V$, $1 \le i \le d$, are again distributed completely at random. Let G be the subtree of the PFWT consisting of the M nodes inspected during the depth first search and let $\lambda < \kappa$ be the number of pruning nodes found during the inspection. The number of possible topologies for G is at most $N^{\lambda+1}$ (we have to multiply the number of topologies for PFWTs with λ pruning nodes with N due to the fact that there are N possibilities to stop the depth first search). It is easy to see that we can bound the probability that G exists as a subgraph exactly as we did in the previous section for the PFWT, replacing κ with λ in the probability bounds and N^κ by $N^{\lambda+1}$ for the number of topologies. (It is easy to accommodate to the fact that G contains one additional leaf.) Due to $N = n^{o(1)}$ we obtain almost the same probability bound, namely

$n^{-\lambda+1+o(1)}$. But since it may be $\lambda = 0$ this does not suffice to obtain a probability smaller than 1. However, according to Claim 4.3 the probability that the set V has the cardinality $\max\{2k, |g(V)| + k\}$ is bounded by $|V|^{2k} \cdot \ell^{-k} = n^{o(1)} \cdot \ell^{-k}$ for $k = O(1)$. Note that even under the condition that this event on V occurs, the keys in V are distributed completely at random. Hence, the probability that either a witness tree of order L or a PFWT of order L with κ pruning nodes of order $L \geq (\ln \log_2 n + \ln(1+\alpha)/(d \cdot \ln \phi_d) + 1$ occurs is bounded by

$$n^{-\alpha} + n^{-\kappa+1+o(1)} + n^{-\lambda+1+o(1)} \cdot \ell^{-k} \leq n^{-\alpha} + n^{1+o(1)} \cdot \left(n^{-\kappa} + \ell^{-k}\right).$$

This completes the proof of the main theorem.

Acknowledgment

The author is grateful to Berthold Vöcking for and Joe Rideout for helpful comments on the text.

References

Y. Azar, A. Broder, A. Karlin and E. Upfal (1999). Balanced allocations. *SIAM Journal on Computing*, volume 29, pp. 180–200.

A. Broder and M. Mitzenmacher (2001). Using multiple hash functions to improve IP lookups. In *INFOCOM*, pp. 1454–1463.

E. R. Cole, A. M. Frieze, B. M. Maggs, M. Mitzenmacher, A. W. Richa, R. K. Sitaraman and E. Upfal (1998). On balls and bins with deletions. In *Proceedings of the 2nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, volume 1518 of *Lecture Notes in Computer Science*, pp. 145–158.

M. Dietzfelbinger (1996). Universal hashing and k-wise independent random variables via integer arithmetic without primes. In *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1046 of *Lecture Notes in Computer Science*, pp. 569–580.

M. Dietzfelbinger and F. Meyer auf der Heide (1992). Dynamic hashing in real time. In J. Buchmann, H. Ganziger and W. J. Paul (eds.), *Informatik-Festschrift zum 60. Geburtstag von Günter Hotz*, pp. 95–119. Teubner.

M. Dietzfelbinger and C. Weidling (2005). Balanced allocation and dictionaries with tightly packed constant size bins. In *Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 3580 of *Lecture Notes in Computer Science*, pp. 166–178.

M. Dietzfelbinger and P. Woelfel (2003). Almost random graphs with simple hash functions. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 629–638.

A. Östlin and R. Pagh (2003). Uniform hashing in constant time and linear space. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 622–628.

A. Siegel (2004). On universal classes of fast high performance hash functions, their time-space tradeoff, and their applications. *SIAM Journal on Computing*, volume 33, pp. 505–543.

M. Thorup and Y. Zhang (2004). Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 615–624.

B. Vöcking (2003). How asymmetry helps load balancing. *Journal of the ACM*, volume 50, pp. 568–589.

M. N. Wegman and J. L. Carter (1979). New classes and applications of hash functions. In *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 175–182.