Closed hashing is computable and optimally randomizable with universal hash functions

Alan Siegel
Department of Computer Science
Courant Institute
251 Mercer Street
N.Y.C., NY 10012

Jeanette P. Schmidt
Department of Computer Science
Polytechnic University
333 Jay Street
Brooklyn, NY 11201

Abstract

Universal hash functions that exhibit $c \log n$-wise independence are shown to give a performance in double hashing, uniform hashing and virtually any reasonable generalization of double hashing that has an expected probe count of $\frac{1}{1-\alpha} + O(\frac{1}{n})$ for the insertion of the αn-th item into a table of size n, for any fixed $\alpha < 1$. This performance is optimal. These results are derived from a novel formulation that overestimates the expected probe count by underestimating the presence of local items already inserted into the hash table, and from a very sharp analysis of the underlying stochastic structures formed by colliding items.

Analogous bounds are attained for the expected r-th moment of the probe count, for any fixed r, and linear probing is also shown to achieve a performance with universal hash functions that is equivalent to the fully random case.

Categories and Subject Descriptors: E.1 [**Data**]: Data Structures—*arrays*; *tables*; E.2 [**Data**]: Data Storage Representations—hash-table representations; F2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*sorting and searching.*

General terms: Algorithms, Theory.

Additional Key Words and Phrases: Clustering, Double hashing, hashing, limited independence, linear probing, open addressing, random probing, uniform hashing, universal hash functions.

The work of the first author was supported in part by ONR grant N00014-85-K-0046, NSF grants CCR-8906949 CCR-8902221, and CCR-9204202.

The work of the second author was supported in part by NSF grant CCR-9110255 and the New York State Science and Technology Foundation Center through its Center for Advanced Technology.

1. Background

Hashing covers a variety of schemes to maintain an associative lookup table $L[0..n-1]$ for a set of keys that belong to some large universe U. In closed hashing, access to a key x is achieved by following a sequence $p(x,1)$, $p(x,2)$, $p(x,3)$, ... of computed probe indices that belong to the range $[0..n-1]$. In particular, a key x is inserted into the hash table L by storing the key in the first vacant table slot among $L[p(x,j)]$, for $j = 1, 2, \cdots$. Retrieval is achieved by probing according to the same sequence of computed indices, until either x is found or a vacant table location is encountered.

Definition 1.

- Let $D = (x_1, x_2, \ldots, x_{\alpha n})$ be a sequence of keys from the universe of integers $U = [0..m-1]$.
- The members of D will be sequentially hashed into table $L[0..n-1]$.
- We say that x_k is hashed at *time* k.
- We say that x_k is *embedded* at location ℓ if the hashing of D stores x_k in $L[\ell]$.
- Suppose T is an increasing subsequence of indices $T \subset (1, 2, \ldots, \alpha n)$. Let D_T denote the subsequence of keys $(x_t)_{t \in T}$.
- For a sequence S and scalar t, S_t denotes the t-th item in S, so that $D_t = x_t$.
- Let the number of probes needed to insert $x_{\alpha n}$ be denoted by $probe_{\alpha n}$.

This paper analyzes $\mathrm{E}[probe_{\alpha n}]$, the expected number of probes needed to insert $x_{\alpha n}$.

Uniform hashing is an idealized model where the probe sequence $p(x,*)$, for each key $x \in U$, is assumed to be a fully independent random function (or permutation).

Traditional double hashing defines $p(x,j) = f(x) - (j-1)d(x) \bmod n$, where the table size n is prime, $f(x)$ is assumed to return an arbitrarily selected integer in $[0..n-1]$, and $d(x)$ is an arbitrarily selected value in $[1,..n-1]$. The random functions $\{(d(x), f(x))\}_{x \in U}$ are assumed to be fully independent and uniformly distributed over their respective ranges. The probe scheme originates in the 1968 Ph.D. thesis of Guy de Balbine [11].

Linear probing uses one independent random function, and the probe sequence is defined by $p(x, j) = (f(x) + 1 - j) \bmod n$.

Tertiary clustering is an idealized model where a random function h is first used to map the data D into the interval $[0, n^2]$, and the image multiset is then hashed via idealized uniform hashing. The point of this formulation is to model circumstances where a pair of distinct hash keys might receive the exact same random probe sequence with probability $\frac{1}{n^2}$, as opposed to the even less feasible model of pure uniform hashing, where the probability is $\frac{1}{n!}$.

Prior work on double hashing includes the results of Guibas and Szemerédi [9], Lueker and Molodowitch [12], and Schmidt and Siegel [16]. Lueker and Molodowitch presented a very elegant proof to show that for random functions f and d and any fixed load factor $\alpha < 1$, the expected number of probes to insert the $(\alpha n + 1)$-st item is $\frac{1}{1-\alpha} + O(\frac{\log^{5/2} n}{\sqrt{n}})$, which is asymptotically equivalent to uniform hashing, and hence, by the result of Yao [21], optimal. Previously, Guibas and Szemerédi, had established a comparable bound for loads $\alpha < \frac{3}{10}$.

Schmidt and Siegel showed that if the probe functions f and g are selected from a set of universal hash functions that exhibit $(c \log n)$-wise independence, then double hashing will have an expected insertion performance that is at most $\frac{1}{1-\alpha} + \epsilon$, for any fixed $\alpha < 1$, $\epsilon > 0$ and a large enough constant c. Consequently, nearly optimal performance can be achieved with functions f and g that are computed by a program that uses a small number of random seeds. The results are established for a class of probe schemes that is considerably larger than the arithmetic progressions of double hashing. Further information on previous work and the implications and formalizations of limited randomness can be found in [16].

We now build upon the results and the frame work of [16] to show that double hashing has an expected probe performance of $\frac{1}{1-\alpha} + O(\frac{1}{n})$, for any fixed $\alpha < 1$. This error bound is new even in the case of full randomness. More importantly, it is shown to hold even when $(c \log n)$-wise independent hash functions are used, where c is a constant that depends on α. In addition, comparable bounds are attained for the expected moments of the probe counts. These results also hold for the generalized double hashing schemes of [16].

The objectives underlying our generalized definition of double hashing are to include double hashing and uniform hashing within the same model, and to characterize the basic probe strategies that enable these schemes to achieve such good performance, both for fully independent hash functions and for those exhibiting limited independence.

Definition 2: The models UH, DH, and DH_ψ.

- In UH, the probe sequence $p(x,*)$ is an independent family of random variables that are uniformly distributed over $[0,n-1]$. Any collection of sequences $p(x_1,*), p(x_2,*), \ldots, p(x_n,*)$ are mutually independent, for distinct x_i.

- DH relaxes the requirement that each individual probe sequence be fully random.

  1. Each individual probe sequence $p(x,*)$ exhibits approximate pairwise independence:

     $$\forall x\ \forall i,j\ i \neq j,\ \forall r,s \in [0,n-1]\ r \neq s: \quad Prob\{p(x,i)=r,\ p(x,j)=s\} = \frac{1}{(n-O(1))^2}.^\dagger$$

  2. Furthermore, the random sequences $\{p(x,*)\}_{x\in D}$ are mutually independent. This condition need only hold for a subset of hash functions $\hat{F} \subset F$, where $\hat{F}$ depends on D, and $\frac{|\hat{F}|}{|F|} \geq 1 - \frac{O(1)}{n^2}$.

  3. In addition, we have the following robustness requirements.

     i) Extremely long probe sequences are quite rare: For a fixed c_0 that depends on α,

     $$\forall x: \sum_{t>c_0 n} Prob\{|\cup_{i=1}^{t}\{p(x,i)\}| < \alpha n + 1\} = \frac{O(1)}{n}.$$

     ii) Probe sequences are unlikely to reprobe locations too frequently.

     $$\forall x\ \forall j<k<h, r\in[0,n-1]: \ Prob\{p(x,j)=p(x,k),\ p(x,h)=r\} = \frac{O(1)}{n^2}.$$

     $$\forall x\ \forall h<i,j<k,\ (h,i)\neq(j,k): \ Prob\{p(x,h)=p(x,i),\ p(x,j)=p(x,k)\} = \frac{O(1)}{n^2}.$$

- In DH_ψ, the statistical probe behavior of an individual probe sequence is subject to the

---

†We use the Big-Oh notation in the following standard way: $f = g + O(h)$ means that $|f-g| = O(|h|)$. Consequently, there is no distinction between $f+O(g)$ and $f-O(g)$. Nevertheless, we shall, upon occasion, use minus signs to suggest that the worst case error is negative. Also, it is quite reasonable to write, say, $\frac{1}{1+O(h)} = 1 - O(h)$, for $h = o(1)$.

same requirements as in DH for the first ψ probes, the global coverage requirement must still hold, and the joint distribution of initial probe sequences, for collections of ψ or fewer probes, is required to be statistically independent, for distinct items. More precisely, we have the following.

1. $\forall x\ \forall i,j \le \psi\ i \ne j,\ \forall r,s \in [0,n-1]\ r \ne s:\quad Prob\{p(x,i)=r,\ p(x,j)=s\} = \frac{1}{(n-O(1))^2}$.

2. Knowing a limited number of the probe values for a small set of keys gives no information about the first few probes for another key. Formally, let Z be a set of keys $\zeta \in U$ with associated probe count bounds j_ζ, where $\sum_{\zeta \in Z} j_\zeta \le \psi$. Let $\{\kappa_{\zeta,1}, \kappa_{\zeta,2}, \ldots, \kappa_{\zeta,j_\zeta}\}_{\zeta \in Z}$ be a multiset of arbitrary probe locations. Then

$$Prob\{\bigwedge_{\zeta \in Z} \bigwedge_{j \le j_\zeta} p(\zeta,j) = \kappa_{\zeta,j}\} = \prod_{\zeta \in Z} Prob\{\bigwedge_{j \le j_\zeta} p(\zeta,j) = \kappa_{\zeta,j}\}.$$

   This condition need only hold for a subset of hash functions $\hat{F} \subset F$, where $\hat{F}$ depends on D, and $\frac{|\hat{F}|}{|F|} \ge 1 - \frac{O(1)}{n^2}$.

3. For some fixed c_0 that depends on α, $\forall x:\ \sum_{t > c_0 n} Prob\{|\cup_{i=1}^{t} \{p(x,i)\}| < \alpha n + 1\} = \frac{O(1)}{n}$.

4. $\forall x\ \forall j < k < h \le \psi,\ r \in [0,n-1]:\ Prob\{p(x,j) = p(x,k),\ p(x,h) = r\} = \frac{O(1)}{n^2}$.

5. $\forall x\ \forall h < i \le \psi, j < k \le \psi,\ (h,i) \ne (j,k)$:

$$Prob\{p(x,h) = p(x,i),\ p(x,j) = p(x,k)\} = \frac{O(1)}{n^2}.$$

According to these definitions, $UH \subset DH \subset DH_\psi$. The exclusionary $r \ne s$ in 1 of the DH and DH_ψ definitions is given to ensure that standard double hashing lies within DH.

In these formal models, a family of hash functions H comprises a finite set of functions. Given the data sequence D, a specific hash function is selected by choosing a member from S at random according to the uniform distribution. The statistical properties defined by UH, DH, and DH_ψ are with respect to H, although 3i for DH and 3 in DH_ψ are also algorithmically dependent.

The key notions used in the analysis of these probe schemes were 1) dependency sets, and

2) multiplicative vacancy overestimators ([16]).

Definition 3.

- The *dependency set* of a hash key x_k is recursively defined to comprise x_k and the dependency set members of the previously inserted keys that reside in the locations unsuccessfully probed on behalf of x_k during its insertion. We denote the dependency set of $x \in D$ (with respect to D) by $dep(x, D)$. The *dependency DAG* for x, $G(x, D)$, has the *root* x; the graph also contains directed edges from x to the items that reside in the occupied locations visited during the insertion of x, and the recursively defined dependency DAGs for each of these items. The DAG is a *tree* if each vertex other than the root has indegree one. A subsequence $S \subset D$ is a *local dependency set* if S hashes by itself into a dependency set. Thus x_k will be the root of a unique dependency DAG when D is hashed, but will root many different different local dependency DAGs, in general. A *partial dependency DAG* $G_r(x, D)$ is the subgraph of $G(x, D)$ restricted to x and paths from x that begin with the one of the first r probes for x.

- Given a set of hash functions (or set of statistical properties satisfied by a set of universal hash functions), a *vacancy estimator* is a function $q(t)$ that overestimates the probability that a slot location l is vacant at time t. The estimate should hold regardless of the value of l and the data in D. The estimator is *multiplicative* if, for any sequence of slots $l_1, l_2, \ldots, l_k$, and corresponding times $t_1, t_2, \ldots, t_k$, the joint probability that slot l_i is vacant at time t_i, for $i = 1, 2, \ldots, k$ is at most $(1 + O(k^2/n)) \prod_i^k q(t_i)$. This multiplicativity need only hold for $k = O(\log n)$.

We will count the expected number of partial dependency DAGs rooted at $x_{\alpha n}$, which means that root $x_{\alpha n}$ may not yet have found a vacant table slot for insertion. Thus the next probe, on behalf of $x_{\alpha n}$, will add another branch to the DAG, if the new slot turns out to be occupied. Let, $x_{\alpha n}$ have r children in the DAG $G(x_{\alpha n}, D)$. Then it will have encountered $r + 1$ DAGs. (The first will have zero children since we do not require the root to be inserted when counting

these structures.) Thus the number of such DAGs actually encountered by $x_{\alpha n}$ is precisely the number of probes needed to insert the key.

A consequence of these definitions is that the expected number of probes to insert the αn-th key $x_{\alpha n}$ is precisely the expected number of local partial dependency DAGs rooted by $x_{\alpha n}$, where the probability that a subsequence hashes to form a local partial dependency DAG is the probability that the colliding probes among the members of the subsequence occur as specified by the DAG structure, and the non-root nodes find their locally determined embedding locations vacant at the times of their insertions within D.

The difficult part of the calculation is to provide an adequate overestimate of the joint vacancy probability for the members of a local dependency set. A good vacancy estimate is essential for reducing the accounting of spurious dependency sets that locally hash into a dependency DAG rooted by $x_{\alpha n}$.

Schmidt and Siegel use this counting approach to attain Theorem 1 in [16], which states that the expected number of probes to insert $x_{\alpha n}$ is at most

$$\mathrm{E}[probe_{\alpha n}] \leq \frac{1}{\sqrt{1-2Q(\alpha n)}} + O(1/n), \tag{1}$$

where $Q(k) = \frac{1}{n}\sum_{i=1}^{k-1} q(i)$, and $q(i)$ is a multiplicative vacancy estimator. This bound, as a function of Q, is established for any hashing procedure that satisfies the statistical properties of DH_ψ, for $\psi \geq c \log n$, for some fixed constant c that is independent of n.

Numerical values for the expected number of probes are attained by quantifying the following *vacancy criterion* which, it turns out, is indeed a multiplicative vacancy overestimator, even in DH_Ψ.

Definition 4: The vacancy criterion $M^{(h)}(T, I)$ and its probability $q^{(h)}(T, I)$.

- Let $M^{(h)}(T, I)$, be the vacancy criterion that says, for each j in $1, 2, \ldots, |I|$, there is no tuple $S \subset (x_1, \ldots, x_{T_j - 1}) - D_T$ of size $|S| \leq h$ that hashes into a dependency tree G rooted at location I_j.

- Let the exact probability of $M^{(h)}(T, I)$ be denoted by $q^{(h)}(T, I)$.

Informally, location I_j is deemed to be occupied by time T_j only if there is some tuple of h or fewer items, among $x_1, \ldots, x_{j-1}$, that hashes by itself into a tree that is rooted at I_j. We choose to ignore non-tree DAGs in this definition.

Calculations in [16] show that with $O(\log n)$-wise independence, these multiplicative vacancy estimators $q^{(h)}$ are, for sufficiently large, fixed h, within any fixed ϵ of the true estimate. Now, the true vacancy estimate $q(i) = 1 - i/n$ gives the bound $\frac{1}{1-\alpha} + O(1/n)$ for the expected insertion cost in (1), which is the actual bound for uniform hashing with fully independent random probes.

Unfortunately, it is necessary to have an asymptotically exact overestimator if the expected probe count is to have an error of $O(\frac{1}{n})$. Otherwise our expectation will be augmented by spurious probe statistics from DAGs other than the actual dependency graph for $x_a \in D$, since there are many items that might initially hash to a given location, though only the first will actually reside there. Yet, the errors resulting from very accurate vacancy formulations appear to be rather difficult to bound satisfactorily.

1.1 Overview

We now use the constructions from [16] plus inclusion-exclusion to establish a sharp approximate isomorphism between double hashing and uniform hashing. The inclusion-exclusion will eliminate the overcount of (spurious) probe collisions between $x_{\alpha n}$ and previous keys that appear to be inserted at locations that satisfy our weak vacancy criterion, but are actually inserted elsewhere, because their apparently vacant insertion slots will actually be already occupied by the time they are hashed from D into L.

The isomorphism shows that double hashing, for example, admits a calculation for the expected probe performance that is the same as that for uniform hashing, apart from a string of principal error terms, plus a few other negligible errors. Some care will be needed to bundle events together in a way that avoids exponential overcounts of these error terms. Then the k-th error term will turn out to be the difference, between uniform hashing and double hashing, in the expected number of look-ahead restricted k-item aggregates (or more precisely, aggregates restricted by vacancy calculations determined from hashing h-item or smaller subtrees from D)

of local dependency DAGs rooted by $x_{\alpha n}$. The resulting error will be dominated by the product of $O(\frac{k^3}{n})$ and the k-th coefficient of a simple generating function g defined from these structures.

The ensuing equation for g depends on the amount of look-ahead that is used but is essentially independent of n. The solution, as the look-ahead parameter† approaches infinity, turns out to have a radius of convergence that exceeds 1, for any fixed load factor $\alpha < 1$. Hence, for some fixed look-ahead that depends on α, the errors sum to $O(\frac{1}{n})$ and decay rapidly in k. Thus our approach throws the question of asymptotic optimality entirely onto the limit properties of a simple family of ordinary differential equations that govern g. While our use of converging differential equations is intended to be self-contained, a comprehensive introduction to the subject can be found in [7].

The resulting isomorphism implies that as $n \to \infty$, there is a limiting probability distribution on the (unembedded) DAG structures encountered by the collision behavior of the αn-th item and its recursive collision descendents. Moreover, this distribution is identical for generalized double hashing and uniform hashing (c.f. Subsection 3.2).

2. Superdependency graphs

The thrust of our asymptotically exact performance proof is to analyze the statistical behavior of double hashing on aggregates of local dependency DAGs. Consequently, we must extend the notion of a dependency set to these larger ensembles of data.

Definition 5.

- The superdependency set of x, $sdep(x, D)$ is the union of all vertex sets belonging to local dependency sets of x: $sdep(x, D) = \cup_{D' \subset D} dep(x, D')$.
- The superdependency graph of x, $G_{sdep}(x, D)$, comprises $sdep(x, D)$ plus the directed edges that occur from the actual collisions when $sdep(x, D)$ is hashed: If y is the r-th member in the adjacency list of z, then the first r probes for z must be to locations already occupied

†Actually, the amount of statistical independence that is needed for our parameter h turns out to be a fixed function of h plus $O(\log n)$, rather than h itself (c.f. Lemmas 6, 7 and Theorem 3 of [16]

by members of $sdep(x, D)$, and y must actually be hashed (from the sequence D) to the r-th probe location of z.

- We also define $\mathcal{G}_{sup}(x)$ to be the set of superdependency graphs that result from any prefix of the actual probe sequence for x.

- The vertex x will be called the *root* of $G_{sdep}(x)$, even though the structure may contain other vertices (that appear earlier in D) with indegree 0. Similarly, we shall call a superdependency graph a *tree* if it is a tree when the edges are viewed as being undirected. The outdegree of the root of a superdependency graph G will be denoted by $degr(G)$.

$G_{sdep}(x, D)$, (and all graphs in $\mathcal{G}_{sup}(x)$), are DAGs, but have a more complex structure than the dependency graph $G(x, D)$, and the partial dependency graphs of x. For example, suppose that y is in x's true dependency set $dep(x, D)$, and suppose that y will be hashed (from D) into a location l. There may be other nodes in local dependency sets $dep(x, D')$ that would reside in location l and would belong to the dependency set of x, were y (and perhaps others) removed from D. We call these vertices dummies. Any vertex (other than root x), which has indegree zero in the DAG is a *dummy*, since no key in the superdependency set is hashed in a way that depends upon its presence. More generally, a dummy is a vertex that belongs to some local dependency set of x, but which will actually reside in a later probe location than that indicated by the local dependency set.

Since limited randomness forces us to examine superdependency structures that are not maximal, we will also use these notions to refer to a local superdependency set, $sdep(x, D')$, which is the union of the local dependency sets in $D' \subset D$.

When analyzing arbitrary superdependency DAGs, we will need to use a canonical traversal process, which extracts a tree-like subset of edges, so that the resulting subgraph is connected and acyclic, when the edges are viewed as being undirected.

Definition 6.

Let $G = (V, E)$ be a superdependency DAG. We call $T = (V, E_t)$ an *omnidirected spanning*

tree if E_t comprises the edges selected as follows. We scan and process the vertices V in order of decreasing index in D. The root is the first vertex processed, and is handled differently from the rest. Its neighbors in G are immediately discovered and are forced to be its children in T. For specificity, a child receiving multiple probes from the root is taken to be connected by the edge that represents the highest probe number. Thereafter, we initiate a standard recursive DFS from the root. The DFS explores the probe edges exiting a vertex in order of decreasing probe number. Edges taken to newly discovered vertices are entered in E_t. When the DFS completes and returns to its origin, the scan is continued to the next vertex that is not yet DFSed. There is one final modification of the DFS. Also in E_t is the very first cross edge encountered during (some recursive level of) each DFS initiated from the scan of a newly discovered vertex, except for the DFS initiated from the root.

The cross edges ensure that the resulting structure is connected, when the edges are viewed as being undirected.

Lemma A in the Appendix (as adapted from Lemma 2, in [16]) shows that a v-vertex e-edge superdependency graph has the same local probability distribution as its dependency counterpart. In particular, let G be a superdependency structure with v vertices and $v-1$ edges. We may deduce that v keys hash into the structure defined by G with probability $\frac{1+O(v^2/n)}{n^{v-1}}$. Moreover, the chances are only $O(v^3 n^{-v})$ that the vertices will hash into a superdependency graph with v vertices and v or more edges, which yields G as its omnidirected spanning tree. The proof of Lemma A is presented in the Appendix, so that the basic argument can be repeatedly applied without elaboration.

This probability distribution suggests that hashing statistics might be quantified from the behavior of superdependency DAGs with $e = v-1$ edges. But before this notion can be exploited, we will have to reduce the (expected) number of local superdependency graphs that can occur.

A way to control this expectation is by extending the definitions of superdependency graphs to include our vacancy criterion. The vacancy criterion requires that any dummy item z in a local superdependency set D', which locally hashes into a location l, must not have h or fewer

items preceding it in D, which hash into a local structure rooted at l. Otherwise we recognize that the local dependency set cannot be global, since l known to be already occupied.

The following definitions incorporate our vacancy criterion into superdependency graphs.

Definition 7.

- The h-superdependency set of x, $sdep^{(h)}(x, D)$ is the union of all local dependency sets of x, in which all elements find their locations empty according to vacancy criterion $M^{(h)}$, which is evaluated with respect to all of D.

$$sdep^{(h)}(x, D) = \cup_{D' \subset D}\{dep^{(h)}(x, D')\},$$

  where a local h-dependency set $S = dep^{(h)}(x, D')$, is defined to be a (possibly empty) subsequence $S \subset D'$, that hashes by itself into a local dependency graph rooted by x, and, at the insertion time of the $s \in S$, selects the apparent location l where the following hold.

  i) Location l is vacant at the time s is hashed from the sequence D'.

  ii) Location l satisfies the vacancy criterion $M^{(h)}$ at the time s is hashed from the sequence D.

- The *h-superdependency graph* of x, $G^{(h)}_{sdep}(x, D)$, comprises $sdep^{(h)}(x, D)$ plus the edges that occur from collisions when $sdep^{(h)}(x, D)$ is hashed by itself.

- We also define $G^{(h)}_{sdep}(x, D')$, the superdependency graph of x, when only D' is hashed. The vacancy criterion $M^{(h)}$ however, is always taken with respect to all of D.

- Finally, we define the set $\mathcal{G}^{(h)}_{sup}(x, D)$, which contains, for each prefix of the probe sequence for x, the resulting h-superdependency graph.

When the full vacancy criterion is used (i.e., $h = n$), of course, x_j will only root $probe_j$ different local superdependency graphs, which are also global. While the number of graphs grows considerably when weaker criteria are used, we still have the following formulation for any h (and actually any vacancy criterion).

$$\begin{aligned} \mathrm{E}[probe_i] &= \sum_{k \geq 0} Prob\{degr(G^{(h)}_{sdep}(x_i, D)) \geq k\} \\ &= \mathrm{E}[|\mathcal{G}^{(h)}_{sup}(x_i, D)|], \end{aligned} \tag{2}$$

where $degr(G^{(h)}_{sdep}(x_i, D))$ is the degree of x_i in the graph.

Our counting of superdependency sets requires a little explanation. It is helpful, for sake of exposition, to limit the discussion to omnidirected spanning trees of superdependency graphs, which have one fewer edges than nodes, although there may be many dummy roots. Imagine, for the moment, that all edges are undirected. Take the root of such a structure to be the vertex with highest index, call it x, and impose a redirection of the edges based on search from x, so that the structure is now an actual tree. A dependency tree R_x can be prescribed by listing, for each vertex v in the graph R_x, the set of descendent vertices reachable from the first probe for v, from the second, ..., up to the last probe that collided with items in the tree. This formulation was used in Theorem 1, of [16], to attain a recursive count of the expected number of dependency graphs rooted by x.

For our redirected superdependency tree, however, such a representation R_x, is ambiguous: the ith edge from v is either to the vertex w with highest index in the ith set, or, if w is a dummy, to $last(w)$, which is the item with highest index in w's last subset, or, if $last(w)$ is also a dummy, to $last^2(w)$, and so forth. In a dependency prescription R_x, there is no such ambiguity; the probe edge must be directed to the vertex with the highest index in the set. But for superdependency trees, it follows that up to negligible terms, $2^\delta \times P(R_x)$ is the probability that a specified set of keys D^x in R_x hash locally into some superdependency structure G_x whose undirected structure matches the tree R_x, where $P(R_x)$ is the probability that D^x hashes as a dependency graph into the structure prescribed by R_x, and δ is the number of nodes that can legally appear as dummies, according to the vacancy criterion as applied to G_x but without regard for the hashing on $D - D^x$.

As a very weak consequence of the vacancy criterion, a node z in a h-superdependency graph $G^{(h)}_{sdep}(x, D)$ cannot be a dummy unless z has more than h descendents in $G^{(h)}_{sdep}(x, D)$.

Definition 8.

- Let this weak consequence be named the *polyexclusion principle.*

We note that a stronger consequence can also be deduced: a hash key z cannot be a dummy that is apparently (successfully) hashed into its l-th probe location if its l-th probe turns out to store the root of any local dependency tree of $j \leq h$ items and whose elements belong to $sdep(z, D)$.

We shall begin by recursively counting tree structures. This counting is more easily done by counting dependency representations R_x, and including a multiplier for each node that can be interpreted as a dummy.

Given a representation R_x, a top-down embedding of the tree has two possible constructions for each node that might be a dummy: one which can occur to a locally vacant location with vacancy probability $q^{(h)}$, and one that can occur to a location contain that already stores the root of a DAG that has at least $h+1$ keys. To get all partial superdependency graphs, we again count structures where the root is not yet embedded.

Theorem 1. Let $\alpha < 1$ and r be fixed constants. Let c, d and h be sufficiently large and fixed. Let $\psi = d \log n$. Let $s(k, \alpha n)$ be the recursively defined overestimate, as quantified below, of the expected number of local h-superdependency *trees* that are rooted by $x_{\alpha n}$, contain k items, and satisfy vacancy criterion $M^{(h)}$, in UH, DH, and DH_ψ. Then

1) The expected number of such trees that have $c \log n$ nodes or less is bounded:

$$\sum_{k \leq c \log n} s(k, \alpha n) = O(1).$$

2) The expected number of such trees with $c \log n$ nodes or less is bounded even when a tree of k nodes is weighted by k^r: $\sum_{k \leq c \log n} k^r s(k, \alpha n) = O(1)$.

3) The expected number of such trees having a node count in the interval $[c \log n, 2c \log n]$ is $O(n^{-2})$ even when a tree of k nodes is weighted by n^r: $\sum_{c \log n \leq k \leq 2c \log n} s(k, \alpha n) = O(n^{-2-r})$.

Proof: Let $s(k, a)$ be the sum, over all of the $(k-1)$-item subsequences in $x_1, \ldots, x_{a-1}$,

of the (over)estimated probability that the tuple plus x_a hashes locally into a superdependency tree whose non-dummy nodes other than x_a find vacant locations, according to $M^{(h)}$. Nodes that can be dummies yield alternative embeddings (superdependency structures), depending upon whether they are interpreted as dummies or not. Dummy interpretations will be given an embedding probability of 1 to avoid multiplying the dependent estimates that a given hash location satisfies the vacancy criterion $M^{(h)}$ at different (locally successful globally unsuccessful) insertion times. We admit a superset of candidate keys as potential dummies by applying the polyexclusion principle as an overestimate of $M^{(h)}$.

Let μ be the constant 1. (We shall later have cause to solve the same system with $\mu = 3$.) By recurring on the subtree rooted at the most recent probe location of x_a, we get:

$$s(1,a) = 1,$$
$$s(k,a) \le \sum_{\substack{h<j\le k-1\\ 0<b\le a-1}} s(j,b)\frac{\mu+q^{(h)}(b)}{n_1}s(k-j,a) + \sum_{\substack{1\le j\le h\\ 0<b\le a-1}} s(j,b)\frac{q^{(h)}(b)}{n_1}s(k-j,a), \quad k>1. \quad (2)$$

We replace the quotient $\frac{\mu+q^{(h)}(b)}{n_1}$ by $\frac{q^{(h)}(b)}{n_1}$, for $j \le h$, because the root of a subtree of size h or less cannot be interpreted to be a local dummy, according to the polyexclusion principle. Here, $\frac{1}{n_1}$ denotes an upper bound for the probability $\frac{1}{n-O(1)}$ of probing a given location. We now suppress the subscript and use the simpler value n, which can only change the resulting a k'th coefficient by a factor of $1+\frac{O(k)}{n}$, which turns out to be negligible in all cases.

We may overestimate s by solving the system

$$t(1,a) = 1,$$
$$t(k,a) = \sum_{\substack{h<j\le k-1\\ 0<b\le a-1}} t(j,b)\frac{\mu+q^{(h)}(b)}{n}t(k-j,a) + \sum_{\substack{1\le j\le \min(h,k-1)\\ 0<b\le a-1}} \frac{t(j,b)q^{(h)}(b)}{n}t(k-j,a), \quad k>1.$$

Set $\tau(k,a) = t(k,a)q^{(h)}(a)$, $\sigma(k,a) = t(k,a)(\mu+q^{(h)}(a))$, and $T(j,a) = \sum_{0<b\le a-1}\frac{t(j,b)}{n}$. The following equivalent system is attained.

$$
\begin{aligned}
\tau(1,a) &= q^{(h)}(a),\\
\mathcal{T}(j,a) &= \frac{1}{n}\sum_{0<b\le a-1}\tau(j,b),\\
\tau(k,a) &= \sum_{1\le j\le k-1}\mathcal{T}(j,a)\tau(k-j,a), \quad k>1,\\
T(j,a) &= \frac{1}{n}\sum_{0<b\le a-1}\frac{\tau(j,b)}{q^{(h)}(b)},\\
\sigma(1,a) &= q^{(h)}(a)+\mu,\\
\mathcal{S}(j,a) &= \frac{1}{n}\sum_{0<b\le a-1}\sigma(j,a),\\
\sigma(k,a) &= \sum_{h<j\le k-1}\mathcal{S}(j,a)\sigma(k-j,a)+\sum_{1\le j\le \min(h,k-1)}\mathcal{T}(j,a)\sigma(k-j,a), \quad k>1.
\end{aligned}
\tag{3}
$$

Let $\tilde{q}$ be a continuous monotone rescaled interpolant of $q^{(h)}$, that is defined on the domain $[0,1]$. In particular, set $\tilde{q}(\beta)=1$, for $\beta<1/n$, and $\tilde{q}(\beta)=q^{(h)}(\beta n-1)$, for $\beta=i/n$. We will use the customary notation $f_\beta=\frac{df}{d\beta}$, for any function f of β.

Then the solutions $\mathcal{T}$, τ, $\mathcal{S}$, σ, and T can be tightly overestimated by $\tilde{\mathcal{T}}$, $\tilde{\mathcal{T}}_\beta$, $\tilde{\mathcal{S}}$, $\tilde{\mathcal{S}}_\beta$, and $\tilde{T}$ in the system:

$$
\begin{aligned}
\tilde{\mathcal{T}}(k,0) &= 0,\\
\tilde{\mathcal{T}}_\beta(1,\beta) &= \tilde{q}^{(h)}(\beta),\\
\tilde{\mathcal{T}}_\beta(k,\beta) &= \sum_{1\le j\le k-1}\tilde{\mathcal{T}}(j,\beta)\tilde{\mathcal{T}}_\beta(k-j,\beta), \quad k>1,
\end{aligned}
\tag{4}
$$

$$
\tilde{T}(k,\beta) = \int_{0\le\alpha\le\beta}\frac{\tilde{\mathcal{T}}_\alpha(k,\alpha)}{\tilde{q}^{(h)}(\alpha)}d\alpha,
\tag{5}
$$

$$
\begin{aligned}
\tilde{\mathcal{S}}(k,0) &= 0,\\
\tilde{\mathcal{S}}_\beta(1,\beta) &= \tilde{q}^{(h)}(\beta)+\mu,\\
\tilde{\mathcal{S}}_\beta(k,\beta) &= \sum_{h<j\le k-1}\tilde{\mathcal{S}}(j,\beta)\tilde{\mathcal{S}}_\beta(k-j,\beta)+\sum_{1\le j\le\min(h,k-1)}\tilde{\mathcal{T}}(j,\beta)\tilde{\mathcal{S}}_\beta(k-j,\beta), \quad k>1,
\end{aligned}
$$

The domains of $\tilde{\mathcal{S}}$, $\tilde{\mathcal{T}}$ and $\tilde{T}$ are $[1,\infty]\times(0,1)$. Here $\tilde{\mathcal{S}}(j,\beta)$, $\tilde{\mathcal{T}}(j,\beta)$ and $\tilde{T}(j,\beta)$, are tight overestimates of $\mathcal{S}(j,n\beta)$, $\mathcal{T}(j,n\beta)$ and $T(j,n\beta)$, respectively. It is not difficult to see that $\mathcal{S}(k,\beta n)<\tilde{\mathcal{S}}(k,\beta)$, and $\sigma(k,\beta n)<\tilde{\mathcal{S}}_\beta(k,\beta)$.

For convenience, we rewrite the system for $\tilde{\mathcal{S}}$ as:

$$
\tilde{\mathcal{S}}(k,0) = 0,
$$

$$\tilde{\mathcal{S}}_\beta(k,\beta) = 0, \quad k \le 0,$$

$$\tilde{\mathcal{S}}_\beta(1,\beta) = \mu + \tilde{q}(\beta),$$

$$\tilde{\mathcal{S}}_\beta(k,\beta) = \sum_{1\le j\le k-1} \tilde{\mathcal{S}}(j,\beta)\tilde{\mathcal{S}}_\beta(k-j,\beta) - \mu \sum_{1\le j\le h} \tilde{T}(j,\beta)\tilde{\mathcal{S}}_\beta(k-j,\beta), \quad k > 1. \tag{6}$$

Let

$$w(x,\beta) = \sum_{0<k} \tilde{\mathcal{S}}(k,\beta)x^{k-1}, \tag{7}$$

and put

$$g(x,\beta) = \sum_{0<k} \tilde{\mathcal{T}}(k,\beta)x^{k-1}. \tag{8}$$

Define the truncated function

$$G^{(h)}(x,\beta) = \sum_{0<k\le h} \tilde{T}(k,\beta)x^{k-1}, \tag{9}$$

and define

$$G(x,\beta) = \sum_{0<k} \tilde{T}(k,\beta)x^{k-1}. \tag{10}$$

Differentiating (7) and expanding via (6) and (9) gives

$$w(x,0) = 0,$$

$$w_\beta(x,\beta) = xw(x,\beta)w_\beta(x,\beta) - \mu x G^{(h)}(x,\beta)w_\beta(x,\beta) + \mu + \tilde{q}(\beta). \tag{11}$$

Manipulating (11) gives,

$$w_\beta(x,\beta) = \frac{\mu + \tilde{q}(\beta)}{1 + \mu x G^{(h)}(x,\beta) - xw(x,\beta)}, \tag{12}$$

where we wish to determine when $w(1,\beta)$ is finite.

Differentiating (8) via (4) gives

$$g(x,0) = 0,$$

$$g_\beta(x,\beta) = xg(x,\beta)g_\beta(x,\beta) + \tilde{q}(\beta). \tag{13}$$

Integrating (13) as is, and applying the quadratic formula gives,

$$g(x,\beta) = \frac{1 - \sqrt{1 - 2x\widetilde{Q}(\beta)}}{x},$$

where

$$\widetilde{Q}(\beta) = \int_0^\beta \tilde{q}(\gamma)d\gamma, \tag{14}$$

and hence

$$g_\beta(x,\beta) = \frac{\tilde{q}(\beta)}{\sqrt{1-2x\widetilde{Q}(\beta)}}. \tag{15}$$

In view of (5) and (8), we see that $G(x,\beta)$ as defined in (10) can also be expressed as:

$$G(x,\beta) = \int_0^\beta \frac{g_b(x,b)}{\tilde{q}(b)} db. \tag{16}$$

Finally, let

$$\zeta(x,\beta) = \int_0^\beta g_b(x,b)\frac{\mu+\tilde{q}(b)}{\tilde{q}(b)} db \tag{17}$$

$$= g(x,\beta) + \mu\int_0^\beta \frac{g_\beta(x,\beta)}{\tilde{q}(\beta)} d\beta$$

$$= g(x,\beta) + \mu G(x,\beta). \tag{18}$$

Then (17) gives

$$\zeta_\beta(x,\beta) = g_\beta(x,\beta)\frac{\mu+\tilde{q}(\beta)}{\tilde{q}(\beta)}, \tag{19}$$

and expanding with (13) gives

$$= xg(x,\beta)g_\beta(x,\beta)\frac{\mu+\tilde{q}(\beta)}{\tilde{q}(\beta)} + \mu + \tilde{q}(\beta),$$

whence applying (19) gives

$$= xg(x,\beta)\zeta_\beta(x,\beta) + \mu + \tilde{q}(\beta),$$

so that (18) can be used to yield

$$= x(\zeta(x,\beta) - \mu G(x,\beta))\zeta_\beta(x,\beta) + \mu + \tilde{q}(\beta),$$

whereupon collecting the ζ_β terms results with

$$\zeta_\beta(x,\beta) = \frac{\mu+\tilde{q}(\beta)}{1+\mu xG(x,\beta) - x\zeta(x,\beta)}. \tag{20}$$

From (19) and (15), we see that ζ_β can also be expressed as:

$$\zeta_\beta(x,\beta) = \frac{(\mu+\tilde{q}(\beta))}{\sqrt{1-2x\widetilde{Q}(\beta)}}. \tag{21}$$

Since our interpolated $\tilde{q}(\beta)$ converges uniformly to $1-\beta$ on $[0,\alpha]$ as h (and n) $\to\infty$, $\widetilde{Q}(\beta)$ as defined in (14) converges uniformly to $\beta-\beta^2/2$, and $\zeta_\beta(x,\beta)$ as expressed in (21) will converge uniformly to $\frac{(\mu+1-\beta)}{\sqrt{1-2x\beta+x\beta^2}}$ on, say, domain $\Gamma = [0,\frac{2+(1-\alpha)^2}{2}]\times[0,\alpha]$.

Now $g_\beta(x,\beta)$ as expressed in (15) will converge uniformly on the same domain Γ, to

$\frac{(1-\beta)}{\sqrt{1-2x\beta+x\beta^2}}$, and $G(x,\beta)$ will converge uniformly and coefficient wise on Γ, (identity (16)). But G^h is the truncated function of G. Hence $|G(x,\beta) - G^h(x,\beta)|$ is uniformly bounded by ϵ_h, where $\epsilon_h \to 0$ as h (and n) $\to \infty$. The point is that G is analytic in x, and its coefficients will be uniformly exponentially decreasing once h and n exceed some fixed constants.

Now consider the differential equations defined by (12) and (20). We see that w_β will converge uniformly with ζ_β to $\frac{(\mu+1-\beta)}{\sqrt{1-2x\beta+x\beta^2}}$ as $h \to \infty$ and $q^{(h)}$ and $G(x,\beta)$ converge (as n also goes to ∞). In particular, for some fixed sufficiently large h, w_β will be bounded on Γ. But then the analyticity of $w_\beta(x,\beta)$, for $|x| \leq 1 + \frac{(1-\alpha)^2}{2}$ guarantees that its coefficients $\tilde{\mathcal{S}}_\beta(k+1,\beta) = \Theta(s(k,\beta n)(\mu+q(\beta n))$ are bounded by the exponentially decaying $O((1+\frac{(1-\alpha)^2}{2})^{-k})$. The size of h and the decay rate are independent of n because the equations are.

In view of the uniform convergence, we conclude that:

1) For large enough constant h, the expected number of local superdependency trees rooted by $x_{\alpha n}$ that have $c \log n$ nodes or less is bounded by $O(w_\beta(1,\beta)\,|_{\beta=\alpha})$.

2) For large enough constant h, the expected number of local superdependency trees rooted by $x_{\alpha n}$ that have $c \log n$ nodes or less when a tree of k nodes is weighted by k^r, is bounded by $O((x\frac{d}{dx})^r w_\beta(x,\beta)\,|_{\beta=\alpha,x=1})$.

3) For large enough constant h, the expected number of local superdependency trees rooted by $x_{\alpha n}$ and having a node count in the interval $[c \log n, 2c \log n]$ is $O(cn^{-3-r} \log n)$ for suitable $c = O(\frac{2(r+3)}{(1-\alpha)^2})$.

The bound on ψ need only be large enough to ensure that the events in a sample of $2c \log n$ items behave as if fully independent, along with (proportionally sized) additional samples, to ensure that the vacancy estimator also has a statistical behavior equivalent to the fully independent case (c.f. Theorem 3 of [16]). ∎

Corollary 1. Let, as in Theorem 1, $\alpha < 1$, and r be fixed. Let c and d be sufficiently large and fixed with $\psi = d \log n$. Then for sufficiently large constant h, the probability, in UH, DH, and DH_ψ, that $x_{\alpha n}$ roots a local superdependency tree that satisfies $M^{(h)}$ and contains $2c \log n$

nodes or more is $O(n^{-r-1})$.

Proof: Such a tree must contain a local superdependency (sub)tree of ℓ nodes for some $\ell \in [c \log n, 2c \log n]$, and root in D. From Theorem 1, the expected number of such trees with specific root x_β, is $O(n^{-r-2})$. Summing this probability over all possible roots x_β gives a bound of $O(n^{-r-1})$. ∎

Theorem 2. Let, as in Theorem 1, $\alpha < 1$, and r be fixed. Let c and d be sufficiently large and fixed with $\psi = d \log n$. Let N_k be the number of local superdependency sets that occur, are rooted by $x_{\alpha n}$, contain k items, satisfy the vacancy criterion $M^{(h)}$, and are not trees. Then for h sufficiently large but fixed, in UH, DH, and DH_ψ, $\mathrm{E}[\sum_{k \le c \log n} k^r N_k] = O(\frac{1}{n})$, and $\sum_{c \log n \le k \le 2c \log n} N_k = O(n^{-r})$.

Proof: We follow the proof schema of Theorem 1. Let $v(k, a)$ be the sum, over all of the subsequences of $k-1$ items in $x_1, \ldots, x_{a-1}$, of the (over)estimated probability that the tuple plus x_a hashes locally into a superdependency DAG whose non-dummy nodes other than x_a find vacant locations, according to $M^{(h)}$. It is convenient to count the number of tree-like structures plus extra-edge DAGs, and subtract the tree counts. We can justify subtracting the tree overcounts by using a recurrence that introduces a tag variable z for DAGs that have extra edges, and take $(g(z) - g(0)) \mid_{z=1}$ as our overestimate for the expected number of DAGs that are not trees.

The counting will again be over the R_x prescriptions of dependency trees with weightings to account for all possible dummies that permit restructuring into different superdependency trees, with additional factors to account for non-tree edges as in Lemma A. Let $v(k, a)$ be an overestimate of the expected number of h-superdependency DAGs rooted at an unembedded x_a, which have k nodes. We again recur on the subtree rooted at the most recent probe location of x_a, and again appeal to the polyexclusion principle to reduce the factor $\frac{(1+q^{(h)}(b))}{n}$ to $\frac{q^{(h)}(b)}{n}$, when applied to embed a $v(j, b)$, where $j \le h$. But there is a catch. If the subDAG rooted by x_b has a DAG probe edge that points outside the subDAG, then the resulting structure may be too large for the vacancy criterion to declare the apparent location of root x_b as being occupied

by time b. Following a rather pessimistic bent, we restore the factor to be $\frac{(1+q^{(h)}(b))}{n}$, in this case. This restoration is done by applying it selectively to the terms of the generating function that carry tag factors of z. This can be formulated as the factor, $\frac{q^{(h)}(b)+z(eval_{z=1}-eval_{z=0})}{n}$, where $eval_{z=x}$ evaluates the factors to its right with x substituted for each appearance of z. This prescription gives the following.

$$v(1,a) = 1,$$

$$v(k,a) = \sum_{\substack{1\le j\le h\\ 0<b\le a-1}} v(k-j,a)\frac{q^{(h)}(b)+z(eval_{z=1}-eval_{z=0})}{n}(1+zO(k^3/n))v(j,b)$$

$$+ \sum_{\substack{h<j\le k-1\\ 0<b\le a-1}} v(k-j,a)\frac{q^{(h)}(b)+1}{n}(1+zO(k^3/n))v(j,b).$$

The probability that root x_b sustains a (tagged) DAG probe to one of the k nodes is, in our models DH and DH_ψ, bounded by $O(k^3/n)$. The expected number of non-tree DAG formations, for our hashing schemes, is derived in Lemma A, which uses DFS traversal to attain a canonical tree from the DAG, and estimates the number of ways additional probes could cause a DAG that is not a tree. The $O(k^3/n)$ factor accounts for the non-tree DAG probing by the root x_b, which can have one extra probe to a member of its local superdependency set, or more, in which case x_b might not be placed in a random location, due to the limited independence for the probe sequence of an individual hash key. The tag variable z records the fact that we are counting structures that are not trees.

The system can be crudely bounded as follows. Let w solve

$$w(1,a) = 1,$$

$$w(k,a) = \sum_{\substack{1\le j\le h\\ 0<b\le a-1}} w(k-j,a)\frac{q^{(h)}(b)}{n}w(j,b) + \sum_{\substack{h<j\le k-1\\ 0<b\le a-1}} w(k-j,a)\frac{q^{(h)}(b)+1}{n}w(j,b).$$

Then it is not difficult to see that

$$v(\ell,a) < \begin{cases} w(\ell,a)(1+zO(\frac{\ell^3}{n}))^{\ell-1}(1+\frac{z}{n}O((\frac{2}{q(a)})^{\ell-1})), & \text{for } \ell\le h,\\ w(\ell,a)(1+zO(\frac{\ell^3}{n}))^{\ell-1}(1+\frac{z}{n}O((\frac{2}{q(a)})^{h-1}))^{\ell-1}, & \text{for } \ell > h.\end{cases}$$

To establish the bound, it suffices to multiply the recurrence for $w(\ell,a)$ by

$$\begin{cases} (1+zO(\frac{\ell^3}{n}))^{\ell-1}(1+\frac{z}{n}O((\frac{2}{q(a)})^{\ell-1})), & \text{for } \ell\le h,\\ (1+zO(\frac{\ell^3}{n}))^{\ell-1}(1+\frac{z}{n}O((\frac{2}{q(a)})^{h-1}))^{\ell-1}, & \text{for } \ell > h,\end{cases}$$

and appeal to induction. We leave the intermediate minutiae to the reader.

Thus the tagged contribution $v(k,a)\mid_{z=1} - v(k,a)\mid_{z=0}$ is bounded by $O(\frac{k(\frac{2}{q(a)})^h + k^4}{n})s(k,a)$, where $s(k,a)$ is as in Theorem 1. The conclusions follow. ∎

Corollary 2. Let, as in Theorem 1, $\alpha < 1$, and r be fixed. Let c, d and h be sufficiently large and fixed with $\psi = d\log n$. Let T be a subsequence of the indices $(1,2,\ldots,\alpha n)$, with $|T| \leq c\log n$, and $T_{|T|} = \alpha n$, so that D_T is a subsequence in $D = (x_1, x_2, \ldots x_{\alpha n})$, and $(D_T)_{|T|} = x_{\alpha n}$. Let G be a local superdependency tree structure with $|T|$ nodes, and $e_{T,G}$ be the event that D_T hashes locally into G, and the hashing satisfies the vacancy criterion $M^{(h)}$. Let $\mathcal{X}_{T,G}$ be the indicator function for $e_{T,G}$, so that $\mathcal{X}$ is zero if the event does not occur, and one if it does.

1) Let N_k be the expectation of the product of $\mathcal{X}_{T,G}$ and the number of local h-superdependency sets S, where $S \subset D - D_T$, $|S| \leq c\log n$, S hashes into a local superdependency DAG that satisfies $M^{(h)}$, that is rooted by some item in $D - D_T$, and is not a tree structure, S hashes locally into an embedded structure that intersects the local hashed embedding of D_T, and no proper subset of S exhibits these properties, so that each S is minimal. Then $\sum_{k\leq c\log n} N_k k^r = O(\frac{|T|^2}{n})\mathrm{Prob}\{e_{T,G}\}$.

2) Let Z_k be the expectation of the product of $\mathcal{X}_{T,G}$ and the number of local superdependency DAGs that comprise $k \leq c\log n$ items solely from $D - D_T$, satisfy $M^{(h)}$, are rooted by any item in $D - D_T$, and locally hash into a structure that intersects the locally hashed embedding of D_T in at least two probes, and do not have a smaller subset of keys that hashes into a local h-superdependency DAG that intersects D_T in two or more probes. Then $\sum_{k\leq c\log n} Z_k k^r = O(\frac{|T|^2}{n})\mathrm{Prob}\{e_{T,G}\}$.

3) Let Z be the expectation of the product of $\mathcal{X}_{T,G}$ and the number of local superdependency DAGs that comprise between $c\log n$ and $2c\log n$ items in $D - D_T$, satisfy $M^{(h)}$, and are rooted by any item in $D - D_T$. Then for sufficiently large constant h and c, $Z \leq n^{-r}\mathrm{Prob}\{e_{T,G}\}$.

Proof: These statements follow immediately from Theorems 1 and 2, where the analog of these claims without a T component are established, from the multiplicativity of the vacancy

estimator and from Lemma A, which quantifies the hashing statistics of subsets of ψ items or less, for DH_ψ. For example, if $D_{\widetilde{T}}$ is a candidate subsequence of k items in $D - D_T$, it can intersect the local hashing of D_T, in DH_ψ with a probability bounded below $O(|T|^2k^3)/n^2$, for case 2. We count minimally sized structures that intersect D_T to avoid the degenerate cases where a loss of randomness occurs, and so many intersections occur that the vacancy criterion is affected, since it is not multiplicative when evaluated at one location for two different times. Similarly, the intersection and non-tree DAG requirements introduce a comparable factor for Case 1. These factors multiply the count of local k-item h-superdependency DAGs that occur. This count has an extra factor of $O(n)$ when compared to the expected number of such structures rooted by $x_{\alpha n}$, since the root is not restricted to be a specific key.

The size bound for ψ should be doubled, since the statistics are subsets of data that are twice as large. ∎

Corollary 2 helps confirm that superdependency trees account for most of the hashing statistics. This observation can be formalized as follows.

Corollary 3. For $|T| = O(\log n)$, large enough constant h, and $\psi = O(\log n)$, the following two events are asymptotically equivalent: (D_T hashes into a local h-superdependency tree rooted by $x_{\alpha n}$), and the event (D_T hashes into a local h-superdependency tree rooted by $x_{\alpha n}$, and the h-superdependency graph of $x_{\alpha n}$ is a tree).

Proof: If D_T hashes into a local h-superdependency tree and $sdep^{(h)}((D_T)_{|T|}, D)$ is not a tree, then one of the three cases in Corollary 2 must be applicable, whence the conclusion follows. ∎

We now need an inclusion-exclusion formula for trees. It will be convenient to change our counting method for superdependency trees. Our recurrence equations counted them as true trees, with possible dummy nodes that actually bundled different superdependency structures together with a single tree representative, and with multiplicative factors designed to give a weighting that overcounted the expected number of such bundled structures.

The errors that result from our inclusion-exclusion formula will be especially important to

bound.

Corollary 4. Let r fixed, and let c, d, and h be sufficiently large and fixed. Let $\mathcal{X}_t^\rho$ be the indicator function that is 1 when x_t roots a local h-superdependency tree with a node count that is in the interval $[c \log n, \rho \log n]$ and the globally defined superdependency DAG rooted by x_t is a tree. Let N_t be the number of local h-superdependency trees with root x_t that have $c \log n$ nodes or less. Let $\psi = d \log n$. Then in UH, DH, and DH_ψ, $\mathrm{E}[N_{\alpha n} \times \mathcal{X}_{\alpha n}^\infty] = O(n^{-r})$.

Proof: We again appeal to the recursive formulation of Theorem 1, and the earlier representation of bundled superdependency DAGs with dummies. Let $s(k, a)$ be the weighted sum, over all of the $(k-1)$-item subsequences in $x_1, \ldots, x_{a-1}$, of the (over)estimated probability that the tuple plus x_a hashes locally into an h-superdependency tree. The weighting, for each h-superdependency tree, is the number of local h-superdependency subtrees (with root x_a) contained by the tuple.

Let G be a local h-superdependency tree, and suppose that x is a dummy node in G. Then we may use x to construct an h-superdependency subtree where x is mistakenly treated as residing in an unsuccessful probe location of its probing "parent" and eliminate earlier contenders for the location, or we may eliminate x and its subtree, and use some other dummy or the true member of the dependency set as the presumed resident of the probe location, or view x as a genuine non-dummy, which gives yet another embedding structure where x's parent probes a (possibly) actual embedding location for x, rather than x's (locally determined) last unsuccessful probe location, or construct the h-superdependency tree that recognizes x as a dummy. These choices give four possibilities, which are overcounted by setting $\mu = 3$ in the recurrence below. The overcount is severe, since only one of the possibilities is multiplied by the vacancy criterion $q(b)$, and eliminating x eliminates a subtree with its additional interpretations (which this counting procedure fails to do.)

By recurring on the subtree rooted at the most recent probe location of x_a, we get:

$$
\begin{aligned}
&s(1,a) = 1,\\
&s(k,a) \le \sum_{\substack{h<j\le k-1\\ 0<b\le a-1}} s(j,b)\frac{\mu + q^{(h)}(b)}{n_1}s(k-j,a) + \sum_{\substack{1\le j\le h\\ 0<b\le a-1}} s(j,b)\frac{q^{(h)}(b)}{n_1}s(k-j,a), \quad k>1,
\end{aligned}
$$

where $\mu = 3$. The analysis of this system is the same as in Theorem 1. The fixed values for h and d will be larger because the larger value of μ will slow down the convergence. The convergence, of course, is still to the system with $\mu = 0$.

We now fix $\rho = 3c$. For this range, we see that in the limit ($h = \infty$)

$$
\begin{aligned}
\mathrm{E}[N_{\alpha n}\mathcal{X}_{\alpha n}^{3c}] &\le \sum_{c\log n\le k\le 3c\log n} s(k,\alpha n),\\
&= O(n^{-r}),
\end{aligned}
\tag{22}
$$

which follows from an analysis comparable to that used in Corollary 1. Again the radius of convergence guarantees that the exponential decay and the resulting boundedness will occur for sufficiently large constant h and c.

For larger ρ, we reason as follows. Let e_{T_1,T_2} be the event that D_{T_1} and D_{T_2} hash into local h-superdependency trees. There is no restriction on the identity of the roots. Suppose that $x_{\alpha n}$ roots an h-superdependency tree T of $3c\log$ keys or more, and let T_1 be an h-superdependency tree of $c\log n$ keys or less that is rooted at $x_{\alpha n}$. We may extract from T an h-superdependency tree T_2 with a vertex count between $c\log n$ and $2c\log n$, but which might not be rooted at $x_{\alpha n}$. There are two cases: the trees are disjoint, or they intersect. If they intersect, the resulting union is a tree than can occur with a probability comparable to that of an h-superdependency tree of the same shape. There is the minor difference that the root of T_2 need not be successfully embedded. We account for this statistical variation by including an extra factor of n.

In view of this, we see that

$$\begin{aligned}
\mathrm{E}[N_{\alpha n} \times \mathcal{X}_{\alpha n}^{\infty}] &\le \sum_{\substack{T_1,T_2 \subset [1..\alpha n] \\ |T_1| \le c\log n,\ (T_1)_{|T_1|} = x_{\alpha n} \\ c\log n \le |T_2| \le 2c\log n \\ T_1 \cap T_2 = \emptyset}} \mathrm{Prob}\{e_{T_1,T_2}\} + n\mathrm{E}[N_{\alpha n} \times \mathcal{X}_{\alpha n}^{3c}] \\
&\le \sum_{\substack{T_1,T_2 \subset [1..\alpha n] \\ |T_1| \le c\log n,\ (T_1)_{|T_1|} = x_{\alpha n} \\ c\log n \le |T_2| \le 2c\log n \\ T_1 \cap T_2 = \emptyset}} \mathrm{Prob}\{e_{T_1,T_2}\} + O(n^{-r+1}) \quad \text{according to (22)}, \\
&\le \mathrm{E}[N_{\alpha n}] \sum_{\substack{T \subset [1..\alpha n] \\ c\log n \le |T| \le 2c\log n}} \mathrm{Prob}\{e_{T,T}\} + O(n^{-r}), \\
&= O(n^{-r+1}), \quad \text{since } \mathrm{E}[N_t] = O(1), \text{ and because of the reasoning in Corollary 1.}
\end{aligned}$$

As r is arbitrary, the result follows. ▮

We now use a different representation, which gives a one-to-one mapping between superdependency trees and their representations. While the following representation is tailored to match the baroque features of superdependency sets, the combinatorial lemmas apply equally well to more general structures.

Definition 9.

- Let $D = (x_1, x_2, \ldots, x_{\alpha n})$ be a sequence of atoms. We say that T is an AND_OR tree over D if T is a singleton AND node, or T is finite and the natural parse tree for

$$T = \bigwedge_{j=1}^{k} \bigvee_{i=1}^{k_j} T_{i,j},$$

  where each $T_{i,j}$ is an AND_OR tree over D. Each AND node of T stores an atom from D, but the OR's do not, and no atom appears in more than one AND node. The descendent atoms of an atom $y \in T$ precede y in D. The adjacency list of each AND node is ordered. The OR node children only bear the implicit ordering induced by the contents of their AND roots.

It is easy to see that this description gives a faithful representation for superdependency trees: an AND node has edges representing the unsuccessful probes by its resident atom. The OR nodes point to children that all collide at a common location. All but the earliest child of an

OR node are dummies. Some AND_OR trees will represent superdependency trees that violate the vacancy criterion; this will cause no difficulty, since we shall not use them.

Definition 10.

- Let T be an AND_OR tree. Then S is an *implicant subtree* of T if S is an AND_OR tree, has the same root as T, its edges and vertices are contained in T, and each AND node in S has the same outdegree as in T. It is easy to see that such an S is a local superdependency tree, and vice versa.
- Let $S \subset T$ denote the property that S is an implicant subtree of T.

By definition, $T \subset T$. It is also worth remarking that if T is an AND_OR tree that satisfies our vacancy criterion, then each implicant subtree $S \subset T$ also satisfies the requirement, since true vacancy is defined with respect to the atoms in D.

Definition 11.

- Let T be an AND_OR tree. Let R be the set of OR nodes in T. The *degree of freedom of* T $fr(T)$ is defined to be:
$$\sum_{v \in R} (outdegree(v) - 1).$$
- Let the node count of T be the number of atoms from D that reside in T.

Lemma 1. Let T be an AND_OR tree. Let χ_T be the indicator function that equals one if the atoms named in T hash into a structure that satisfies the AND_OR structure stated by T, and in a way that satisfies $M^{(h)}$, and equals zero otherwise. Then

$$\sum_{S \subset T} (-1)^{fr(S)} = 1, \quad \text{and hence} \quad \sum_{S \subset T} (-1)^{fr(S)} \chi_S = eval(T),$$

where $eval(T)$ is the Boolean evaluation of the logic statement expressed by T about the probing behavior of its constituent atoms.

Proof: Consider the first formulation. Its proof is by structural induction over T. The formula holds if T is a single node. If the root AND has at $r \geq 2$ children, then we simply note

that $\sum_{S \subset T} (-1)^{fr(S)} = \prod_{j=1}^{r} \sum_{S \subset T_j} (-1)^{fr(S)}$, where T_j comprises the root of T and the j-th OR child as its only child.

If the AND root has only one OR child, which has r children, then $\sum_{S \subset T} (-1)^{fr(S)} = \sum_{j=1}^{r} \binom{r}{j} (-1)^{j-1} \cdot 1 = -(1-1)^r + 1$, since all (non-empty) combinations of the OR node's subtrees can be used to construct $S \subset T$; here structural induction is used to attain the factor 1 that arises from each such subtree S.

Finally, the latter formulation is just a restatement of the former when restricted to the subtree of T that is logically equivalent to $\bigvee_{\substack{S \subset T \\ x_S = 1}} S$. ∎

Theorem 3. For fixed load $\alpha < 1$ and suitably large fixed c, d and h, with $\psi \geq d \log n$, the expected number of probes to insert $x_{\alpha n}$, in UH, DH, and DH_ψ, is bounded by $\frac{1}{1-\alpha} + O(\frac{1}{n})$.

Proof: This bound is elementary and well known for UH, although (1) gives, in passing, an independent albeit not so elementary proof of the fact. Rather than evaluate an even more complicated expression for the expected number of probes in DH and DH_ψ, we shall formulate an elaborate expression that, in UH, must equal $\frac{1}{1-\alpha} + O(\frac{1}{n})$, and observe that the computational differences, for the analogous expression in DH and DH_ψ comprise an additive $O(\frac{1}{n})$.

Definition 12.

- Let Tr_k^ℓ be the set of all fully structured h-superdependency DAG-trees (where all dummies are unambiguously identified as in the AND_OR representation) that have k vertices, and have outdegree ℓ at the root.

  The tree property just says that the structure is a tree if we view the edges as undirected.

- Given $T \subset [1, \alpha n - 1]$, let D_T^+ denote D_T concatenated with αn.

- Given $\mathcal{T} \in Tr_k^\ell$, Let $(H^{(h)}(D_T^+) \rightarrow \mathcal{T})$ denote the event that the set D_T^+ is locally hashed into the tree structure $\mathcal{T}$, and the actual nodes (apart from the root) are assigned locations that satisfy $M^{(h)}$ at the times of their insertion in D. By definition, we require that $|D_T^+| = k$.

- Given an event E, let $\mathcal{X}(E)$ be the indicator function for E, which is one if E occurs, and zero otherwise.

- Let E_{tree} denote the event: $\langle$ the (globally defined) h-superdependency DAG $G^{(h)}_{sdep}(x_{\alpha n}, D)$ is a tree $\rangle$.

- Let $Tr^{\ell}_{*} = \cup_{k \le c \log n} Tr^{\ell}_{k}$.

- Let

$$P^{(\ell)}_1 = \sum_{\substack{T \subset (1,2,\ldots,\alpha n - 1) \\ |T| < c \log n \\ \tau \in Tr^{\ell}_{*}}} (-1)^{fr(\tau)} \mathcal{X}(H^{(h)}(D^{+}_{T_i}) \rightarrow \tau) \mathcal{X}(E_{tree}).$$

  Lemma 1 guarantees that $P^{(\ell)}_1$ achieves the correct zero-one values when restricted to the portion of the event space where $|sdep^{(h)}(x_{\alpha n}, D)| \le c \log n$.

- Let $P^{(\ell)}_2$ be the number of local h-superdependency AND_OR trees that occur with $c \log n$ or fewer atoms from D, with root $x_{\alpha n}$, AND degree ℓ at the root, when $sdep(x_{\alpha n}, D) > c \log n$, and $G^{(h)}_{sdep}(x_{\alpha n}, D)$ is a tree. Let $P^{(\ell)}_2$ be zero when $sdep(x_{\alpha n}, D) \le c \log n$ or $G^{(h)}_{sdep}(x_{\alpha n}, D)$ is not a tree.

By definition, $P^{(\ell)}_1 + P^{(\ell)}_2$ is never negative, and $P^{(\ell)}_1 - P^{(\ell)}_2$ is never positive when $sdep(x_{\alpha n}, D) > c \log n$.

Because of limited independence, and the use of probe axioms that permit a fair amount of deviant probe behavior, there are additional error terms. When the h-superdependency set gets too large, our limited independence will fail to quantify its behavior very well.

- Let

$$P_3 = c_0 n \mathrm{Prob}\{|sdep^{(h)}(x_{\alpha n}, D)| > c \log n\},$$

  which charges a cost of $c_0 n$ probes to the event.

- Even longer probe excursions might occur, which are quantified by setting

$$P_4 = \sum_{t \ge c_0 n} Prob\{probe_{\alpha n} > t\}.$$

We now analyze the expected number of probes contributed by non-tree DAGs. The error P_4 captures some of these events. Consider the h-superdependency DAGs G that are not trees and yield $\mathcal{T} \in Tr^{\ell}$ as their omnidirected spanning tree, where $|\mathcal{T}| = k$.

- Let $\frac{P_5^{(\ell)}}{\ell}$ be the expected number of h-superdependency DAGs of $c \log n$ or fewer nodes that are not trees and occur with root $x_{\alpha n}$, and where $x_{\alpha n}$ has no non-tree edges.

The following two terms account for the extra probes by non-tree edges of a DAG G.

- Let, for $k < c \log n$, $B(k)$ be the indicator function for the event $(|sdep^{(h)}(x_{\alpha n}, D)| = k)$, and $degr(G) > degr(\mathcal{T})$. Set

$$P_6 = \sum_{k \leq c \log n} 2k \mathrm{E}[B(k)],$$

so that we take a penalty of $2|sdep^{(h)}(x_{\alpha n}, D)|$ probes when $x_{\alpha n}$ uses at least one non-tree probe.

- Let, for $k < c \log n$, $ER(k)$ be the indicator function for the event: $(|sdep^{(h)}(x_{\alpha n}, D)| = k)$, and $degr(G) > degr(\mathcal{T}) + 1$. Let

$$P_7 = c_0 n \times \sum_{k \leq c \log n} \mathrm{E}[ER(k)],$$

so that we take a penalty of $c_0 n$ probes when $x_{\alpha n}$ uses two or more non-tree probes.

Finally, we must account for the errors that occur due to the differences in the models and instances of UH, DH, and DH_{ψ}. Of the expressions listed, only P_1 has a contribution that is not already asymptotically counted.

- Let

$$P_8 = \sum_{1 \leq \ell \leq c \log n} \sum_{\substack{T \subset (1,2,\ldots,\alpha n - 1) \\ |T| < c \log n \\ \mathcal{T} \in Tr_*^{\ell}}} |\mathrm{Prob}_{DH_{\psi}} \mathcal{X}(H^{(h)}(D_{T_i}^{+}) \to \mathcal{T}) - \mathrm{Prob}_{UH} \mathcal{X}(H^{(h)}(D_{T_i}^{+}) \to \mathcal{T})|.$$

We conclude that

$$\mathrm{E}[probe_{\alpha n}] = \sum_{0 \leq \ell \leq c \log n} \mathrm{E}[P_1^{(\ell)}] + O(\sum_{0 \leq \ell \leq c \log n} \mathrm{E}[P_2^{(\ell)}] + P_3 + P_4 + P_5 + P_6 + P_7 + P_8).$$

In our models, the probability that a tuple of k items hashes into a local tree can vary by a factor of only $(1 + O(\frac{k^2}{n}))$ (Lemma 2 of [16]). Similarly, the probability that a set of k locations satisfy our multiplicative vacancy overestimator, at specific insertion times, is, up to a factor of $(1 + O(\frac{k^2}{n}))$, independent of the specific family of hash function in DH_ψ (Lemma 6 of [16]). Consequently,

$$P_8 = \sum_{1<k<c\log n} s(k, \alpha n) O(\frac{k^2}{n}) = O(\frac{1}{n})$$

by Theorem 1.

Furthermore,

$\sum_{0\le\ell\le c\log n} \mathrm{E}[P_2^{(\ell)}] = O(\frac{1}{n})$ by Corollary 4;

$P_3 = O(\frac{1}{n})$ by Theorem 1, Corollary 1, Theorem 2, and Corollary 2;

$P_4 = O(\frac{1}{n})$ by the definitions of DH and DH_ψ;

$P_5 + P_6 = O(\frac{1}{n})$ by Theorem 2; and finally,

$P_7 = O(\frac{1}{n})$ by Theorems 1 and 2, Lemma A, the definitions of DH and $DH\psi$, and the fact that the two extra constraints on the root's probing introduce an additional factor of $O(\frac{k^3}{n^2})$ into the accounting. ∎

3. Extensions

We now apply the techniques from Section 2 to see what else can be deduced for variants of double hashing with full and limited randomness, and linear probing and uniform hashing in the case of limited randomness.

3.1 Higher moments

Our generic error bounds also apply to such performance measures as the expected r-th moment of the probe count. The basic reason for this fact is a convenient representation of moments in our dependency set algebra. We may write:

$$\mathrm{E}[probe_i^r] = \sum_{\ell\le c\log n} (\ell^r - (\ell-1)^r) P_1^{(\ell)} + O(\frac{1}{n}).$$

The cutoff $c \log n$ needs only a nominal adjustment, due to the radius of convergence for the generating function with coefficients $s(k, \alpha n)$, and the requirements for DH and DH_Ψ must be

modified to guarantee that for some fixed c_0 that depends on α,

$$\sum_{t>c_0 n} t^{r-1} Prob\{|\cup_{i=1}^{t}\{p(x,i)\}| < \alpha n + 1\} < 1/n.$$

Corollary 5. Let r be fixed, and load $\alpha < 1$ be fixed. Then for sufficiently large fixed c, depending on α and r, and $\psi > c \log n$, where

$$\mathrm{E}_{DH_\psi}[probe^r_{\alpha n}] = \mathrm{E}_{UH}[probe^r_{\alpha n}] + O(\tfrac{1}{n}),$$

provided $\psi \geq c \log n$. ∎

3.2 Tree statistics

An immediate outcome of our limit studies is that there is a probability distribution on dependency DAGs, and this distribution is asymptotically the same in UH, DH and DH_ψ, for sufficiently large $\psi = O(\log n)$. Recall that a hash tree structure is defined to impose, on the vertices, a total order that is consistent with the partial order dictated by the tree structure itself. A reasonably representative distribution theorem is the following.

Theorem 4. Let T be a fixed hash tree structure of k vertices. Then the probability, in UH, DH, and DH_ψ, for sufficiently large $\psi = O(\log n)$, that the dependency DAG $G(x_{\alpha n}, D)$ is isomorphic to T is $\frac{1}{(k-1)!}(1-\alpha)(\alpha - \frac{\alpha^2}{2})^{k-1}(1 + \frac{O(k^2)}{n})$. ∎The error factor comes from Appendix A or from [16]. The principal term can be derived from the proof of Theorem 1 in [16], or direct calculation in UH.

3.3 The criterion $M^{(2)}$

As indicated earlier, the probability distribution for our vacancy estimator $M^{(h)}$ converges much faster than the weak calculations we provide. It is even interesting to observe just how far the vacancy criterion $M^{(2)}$ can be used to attain optimal probe performance for DH_ψ. Surprisingly, perhaps, it turns out that the full use of the $M^{(2)}$ criterion gives a constant for the expected number of 2-superdependency trees for loads $\alpha \leq .669$. The details are omitted.

The basic facts are that we may compute $q^{(2)}$ explicitly to get $q^{(2)}(\alpha n) = e^{1-2\alpha-e^{-\alpha}}(1+1/n)$, and $Q^{(2)}(\alpha n) = (1+e^{-\alpha})e^{1-e^{-\alpha}} - 2$. The value .669 is obtained from a numerical overestimate to

the rescaled differential equation associated with equation (2) and $q^{(2)}$ as stated with $\mu = 1$. A change to $\mu = 3$, for example, would yield a conservative cutoff for optimal performance bounds under $M^{(2)}$.

3.4 Tertiary clustering

In tertiary clustering (which is also called 2-ary clustering), the probes sequence is defined, for $j > 2$, by $p(x,j) = f(p(x,1), p(x,2), j)$ where the probes $p(x,1)$ and $p(x,1)$ are assumed to be fully random, and f is a fully **random** uniformly distributed hashing function mapping $[0, n^2 - 1] \times$ probe index $\mapsto [0, n-1]$. See, for example, [3].

Our performance bounds show that any (reasonable) **deterministic** f, will achieve the same performance as tertiary clustering; the randomness provided by the first two probes is sufficient, as long as the requirements for DH_ψ are satisfied.

3.5 Uniform hashing

Uniform hashing is generally viewed as an idealized model because of its computational requirements. Each probe requires the evaluation of a random probe (or worst yet, each element must be mapped into a permutation). We can use ψ-wise independence to define a reasonably uniform scheme that is robust and has optimal performance, up to negligible errors, for any load factor bounded by 1.

For size $|D| = \alpha n$ data sets, the construction in Section 1.1 of [16] gives a universal set of linear hash functions that, with high probability, maps D without any collisions into, say, $[0, p-1]$, where $p \approx n^4$. We can then hash $[0, p-1]$ into $[0, n-1]$ by defining a ψ-wise independent F_ψ with domain $[0, p-1] \times [1, \psi]$, so that $p(x,j) = f(\psi \times x + j) \bmod n$, where $f \in F_\psi$.

As stated, there is a slight technical flaw. These hash functions will include a small number of defective functions that fail to meet the robust coverage requirements. Even the presence of one defective hash function will spoil the expected performance, if it fails to hash D, no matter how many probes are used. Accordingly, a scheme that uses UH_ψ should either switch to linear probing, or rehash the entire data set, once some large number of probes (such as n, or n^ϵ) have been expended in an unsuccessful attempt to insert a given datum. Alternatively, a more robust

formulation of $p(x,j)$, for our example, might be $p(x,j) = f(\psi \times x + (j \bmod \psi)) + j \bmod n$, where $f \in F_\psi$, and ψ is relatively prime to n.

In any case, we have the following Corollary, that is subject to a robustness requirement such as that stated in the definition of DH.

Corollary 6. Let r be fixed, and load $\alpha < 1$ be fixed. Then for sufficiently large fixed c, depending on α and r, and $\psi > c \log n$, where

$$\mathrm{E}_{UH_\psi}[probe^r_{\alpha n}] = \mathrm{E}_{UH}[probe^r_{\alpha n}] + O(\tfrac{1}{n}),$$

provided $\psi \geq c \log n$, and the hashing includes some form of robustness guarantee.

4. Linear Probing

Linear probing is a hashing scheme that trades some of the efficiency of double hashing for the computational efficiency of having only one non-trivial evaluation per key reference. It originated at a time when computation was more expensive, and search was somewhat local and sequential, which may be still be the case, for some storage devices. In this scheme, $p(x,k) = f(x) - k + 1 \bmod n$. Knuth [11] showed that the expected insertion cost for $x_{\alpha n+1}$ is $\frac{1}{2} + \frac{1}{2}\sum_{k\geq 0}(k+1)!\binom{\alpha n - 1}{k} n^{-k}$, which is less than $(1 + 1/(1-\alpha)^2)/2$.

Let LP denote the model of linear probing with fully independent uniformly distributed hash functions. Let LP_ψ denote a model of linear probing with, for simplicity, uniformly distributed and fully ψ-wise independent hash functions. As before, we construct a formulation for the expected number of probes that will expose an approximate isometry between LP and LP_ψ.

Let $X(I)$ bound the probability that a table of n locations will have more than I consecutive table locations occupied after $\alpha n - 1$ items are inserted in LP (or LP_ψ). It is convenient to use $X(I)$ to analyze a slightly different problem, which involves hashing k items into $[1, 2I]$ without wraparound: if a bottom segment $[1, I_0]$ becomes full, then items subsequently hashing into the segment are discarded.

Let $W(k)$ be the expected number of probes to insert a key x item onto $[1, 2I]$, when x hashes with its first probe is to location $I + 1$, and k keys have already been inserted among the locations $[1, 2I]$ according to linear probing. With probability $1 - X(I)$, the keys inserted

outside of this interval cannot affect the hashing of x. We will require ψ to be so large that $W(k)$ is the same in LP and LP_ψ, for relevant k. Then by inclusion-exclusion,

$$\begin{aligned}\mathrm{E}[probe_{\alpha n}] = &\sum_{\substack{k<2I \\ 0\le j\le 4\lceil eI\rceil - k}} W(k)\binom{\alpha n-1}{k+j}\left(\frac{2I}{n}\right)^{k+j}\binom{k+j}{k}(-1)^j \\ &+ O\left(nX(I) + \sum_{k<2I} W(k)\binom{\alpha n-1}{4\lceil eI\rceil}\left(\frac{2I}{n}\right)^{4\lceil eI\rceil}\binom{4\lceil eI\rceil}{k}\right),\end{aligned} \tag{23}$$

where both $\mathrm{E}[probe_{\alpha n}]$ and $X(I)$ are computed in the same LP or LP_ψ. We charge a penalty of n probes if the table has more than I consecutive locations filled, for some interval, at insertion time $x_{\alpha n}$. The sum

$$\sum_{0\le j\le 4\lceil eI\rceil - k} \binom{\alpha n-1}{k+j}\left(\frac{2I}{n}\right)^{k+j}\binom{k+j}{k}(-1)^j,$$

uses inclusion-exclusion to overestimate the probability that exactly k items will have a first probe hashing into the interval $[1, 2I]$, among locations $[0, n-1]$. The inclusion-exclusion factor $\binom{k+j}{k}$ is due to the fact that the summation begins over k-tuples. The bound on j is selected to be large enough to give a satisfactory error, which is just the last term, and to stay within the freedom $\psi = 4eI$.

The value for W is the same in LP and LP_ψ, since k is suitably bounded, as is the expected number of $(k+j)$-tuples that initially probe $[1, 2I]$. The only model dependent issue is how large I has to be in LP_ψ, so that $X(I)$ and the inclusion-exclusion errors will be small.

The simplest way guarantee that $X(I)$ is small is to use Chernoff-Hoeffding bounds for limited independence.

Evidently, $X(I) < n\mathrm{Prob}\{\text{at least } I \text{ items in } D \text{ have their first probe in } [1, I]\}$, since the factor of n accounts for all shifts of $[1, I]$. But this probability describes a large deviation for a sum of αn ψ-wise independent Bernoulli trials, each having probability $\frac{I}{n}$ of success.

From a weak application Theorem 5 of [17], we attain that if $\psi \ge I(1-\alpha)$, then

$$\mathrm{Prob}\{\text{at least } I \text{ items in } D \text{ have their first probe in } [1, I]\} \le e^{\frac{-(1-\alpha)^2 I}{3}}.$$

It follows that $X(I)$ is nominal in LP and LP_ψ when $\psi = c(\log n)$, for suitably large fixed c.

The remaining errors in the summation (23) are bounded by the inclusion-exclusion terms where $j = 4\lceil eI \rceil - k$, which are bounded by

$$\begin{aligned}
\sum_{k<2I} W(k)\binom{\alpha n-1}{4\lceil eI\rceil}\left(\frac{2I}{n}\right)^{4\lceil eI\rceil}\binom{4\lceil eI\rceil}{k}) &\le 2n\binom{\alpha n-1}{4eI}\left(\frac{2I}{n}\right)^{4eI}\binom{4eI}{2I} \\
&\le 2n(\alpha)^{4eI}\frac{(2I)^{4eI}}{(2I)!((4e-2)I)!} \\
&\le 2n(\alpha)^{4eI}\frac{2^{4eI}e^{4eI}}{2^{2I}(3e)^{(4e-2)I}} \\
&\le 2n(\alpha)^{4eI}\frac{(2e)^{2I}}{2^{2I}(3/2)^{(4e-2)I}} \\
&\le 2n(\alpha)^{4eI}\frac{(e)^{2I}}{(3/2)^{(4e-2)I}} \\
&\le 2n\frac{(\alpha)^{4eI}}{(3/2)^{(4e-8)I}}.
\end{aligned}$$

Since all errors are exponentially small in I, we have the following.

Theorem 5.

For any constant w and any fixed load $\alpha < 1$, there is a constant c such that linear probing with a hash function chosen from a set of $c \log n$-wise independent hash functions results in an expected insertion cost that exceeds that of LP by at most n^{-w}. ∎

We deduce that LP and LP_ψ have, apart from a polynomially small error, the same expected r-th moment of the probe count, for any fixed r.

5. Conclusions

We have shown that in double hashing, a universal family of hash functions, that with high probability provides $c \log n$-wise independence, will give optimal performance for any fixed load bounded below 1. The notion of double hashing is generalized to include almost any reasonable probe scheme. Similarly, linear probing incurs no loss of performance when such hash functions are used. These optimality results apply to the expected r-th moment of the probe count, for any fixed r.

As noted in [16], a pool of $O(\log \log m + \log^2 n)$ random bits are sufficient to achieve suitably random universal families. The hash functions of [18] show that it is indeed possible to program

hash functions that exhibit, with high probability, $c \log n$-wise independence on the subset of keys D, execute only a constant number of arithmetic operations to hash a key, and use $O(\log \log m + \log {}^2 n)$ random bits.

These results comprise a significant step toward understanding why extremely simple functions seem to perform so well when used to double hash arbitrary values into a partially filled table. Nevertheless, there is still a gap between the use of $\log n$-wise independent hash functions, and those that are typically used; yet one may well wonder if real data, when hashed by standard hash functions, might, in fact, exhibit the statistics of $\log n$-wise independence.

Our proof technique analyzed local and global hashing interactions separately, and used analytic tools to measure complicated but weakly correlated events in terms of simpler independent processes. Bundling and thinning techniques were used to eliminate (spurious) combinatorial explosions from more naive counting formulations. Surely these methods can be applied to other probabilistic processes that exhibit weak correlations or that might only be supported by a source of limited randomness.

Appendix A

Lemma A. Adapted from [16]. Let G be a superdependency tree of k vertices, and let $S = (S_1, S_2 \ldots S_k)$ be a sequence of k distinct elements in U. Then for $k = O(n^{1/3})$,

1) The probability that the superdependency DAG $G_{sdep}(S) = G$ under DH_ψ, for $\psi \geq 3k$, is $\frac{1+O(k^2/n)}{n^{k-1}}$.

2) The probability that S hashed into a superdependency DAG, that properly contains the structure G as its omnidirected spanning tree, under DH_ψ, for $\psi \geq 3k$, is bounded by $O(k^3/n^k)$.

Proof: Let $G = (V, E)$, where $|V| = k$. Let $Tr = (V, E_{Tr})$ be the omnidirected spanning tree discovered by the search. Let the sequence $\widehat{S}$ be $(\widehat{S_1}, \widehat{S_2}, \ldots, \widehat{S_k})$, list the vertices in the order of discovery by the DFS, with the root first. We embed the vertices in this processing order.

1) Suppose that G is a tree, whence $E_{Tr} = E$. The root $\widehat{S_1}$ can be embed anywhere; there are no constraints. Consider, now, the embedding of $\widehat{S_j}$, for some $j > 1$. Let $\widehat{S_j}$ have the parent $\widehat{S_i}$ in G, and suppose that vertex $\widehat{S_j}$ has r children in G, has indegree 1 and is the h-th probe of $\widehat{S_i}$. Then the probability that the h-th probe of $\widehat{S_i}$ is to a vacant location (with respect to $\widehat{S_1}, \widehat{S_2}, \ldots, \widehat{S_{j-1}}$) is between 1 and $1 - O(\frac{j-1}{n})$, and the probability that $p(\widehat{S_j}, r+1) = z$ given that $p(\widehat{S_i}, h) = z$ is $\frac{1}{n-O(1)}$.

If $\widehat{S_j}$ is a dummy root with r children, then the probability that its $(r+1)$-st probe does not collide is between 1 and $1 - O(\frac{j-1}{n})$. However, each dummy root will be accompanied by a cross edge to a previously embedded vertex. This cross edge will comprise a probe to a previously embedded vertex, and will have a probability of $\frac{1}{n-O(1)}$ of occuring as specified. The vertex issuing this probe will thus have two constraints (or one if it is a dummy root): one for the unsuccessful cross edge probe, and one for a successful insertion into an unsuccessful probe location of its parent (if present).

We appeal to the independence of individual probe sequences to multiply all k factors to get a value between $(\frac{1}{n})^{k-1}(1 + O(k/n))$ and $(1 - \frac{O(k^2)}{n-1})(\frac{1}{n})^{k-1}$, which proves 1).

2) If $G(S)$ is not a tree then $E \neq E_{Tr}$ and the nodes of Tr have different embeddings since collisions occurred. The tree construction is similar, but some nodes $x \in Tr$, will have gaps in their probe sequences $p(x,1), p(x,2), \ldots$ to their tree children, since edges to nodes that are already embedded or that have embedding locations already specified will be omitted. Now, the initial probe sequences for any k items are mutually independent, as long as the total number of probes is bounded by ψ. Consequently, the probability that V hashes into a DAG that yields E_{Tr} as its search tree, under traversal by the DFS, is at most $\prod_{j=1}^{k} pr_j$, where pr_j *overestimates* the probability that the j-th vertex is hashed to have the correct non-tree probes to previously determined locations.

Let $\widehat{S_j}$ have h_j tree edges. To upper bound pr_j, we distinguish among three cases: $\widehat{S_j}$ has no non-tree edges, $\widehat{S_j}$ has fewer than $h_j + 2$ non-tree edges and at least one, and $\widehat{S_j}$ has at least $h_j + 2$ non-tree edges. Note that if no two locations can be probed twice in a

probe sequence – as is the case in double hashing – then cases two and three combine into the case $\widehat{S_j}$ has at least one and at most $k-1$ non-tree edges.

The first case is as the overestimate in 1), and contributes a probability of at most 1 to pr_1, and at most $\frac{1}{n}(1+O(1/n))$ to pr_j, for $j>1$.

In the second case, there are different DAG structures, depending on which probe count within $(h_j+2,\ldots,2h_j+2)$ is the last and actually embeds $\widehat{S_j}$. Summing over all possible last probe counts, over the possible probe counts that correspond to the first non-tree edge, which is among the first h_j+1 probes, and the set of possible destinations for this first non-tree edge, (which must be to a location already probed by $\widehat{S_j}$ or some other item in S), we get $\frac{O((h_j+1)(h_j+1)k)}{n^2}$ as an overestimate for the probability contributed to pr_j by case 2.

In the third case, there must be two consecutive non-tree edges among the first $2h_j+2$ probes of $\widehat{S_j}$. These edges may go to previously embedded items or collide with earlier probes of $\widehat{S_j}$. To estimate this contribution to pr_j, we ignore the requirement that $\widehat{S_j}$ must be placed successfully and focus on the expected number of ways a first pair of such probes could occur, which is bounded by $(2h_j+1)\frac{O(k^2)}{n^2}$.

Combining like terms from the three cases into factors and multiplying gives

$$\prod pr_j = \prod_{1<j\le k}(\frac{1}{n}+O(\frac{(h_j+1)k^2}{n^2})) = (\frac{1}{n})^{k-1}(1+O(k^3/n)),$$

and hence the probability that G results from the traversal of a non-tree DAG is at most $(\frac{1}{n})^{k-1}(1+O(k^3/n)) - (1-\frac{O(k^2)}{n-1})(\frac{1}{n})^{k-1} = O(k^3/n^k)$ ∎

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] M. Ajtai, J. Komlós, E. Szemerédi. There Is No Fast Single Hashing Algorithm, *IPL*, 7,6, 1978, pp. 270–273.

[3] B. Bollobás, A.Z. Broder, and I. Simon. The cost distribution of clustering in random probing, to appear, *JACM*.

[4] R. Brent. Reducing the Retrieval Time of Scatter Storage Techniques, *CACM*, **16**(2), 1973, pp. 105–109.

[5] H. Chernoff. A measure of asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations, *Ann. Math. Statist.*, 23 (1952), pp. 493-507.

[6] J.L. Carter and M.N. Wegman. Universal Classes of Hash Functions, *JCSS*, 18, 1979, pp. 143–154.

[7] E.A. Coddington and N. Levinson. *Theory of ordinary differential equations*, McGraw Hill, 1955.

[8] D.H. Greene and D.E. Knuth. *Mathematics for the analysis of algorithms*, Birkhauser, 1990.

[9] L. Guibas and E. Szemerédi. The Analysis of Double Hashing, *JCSS*, 16, 1978, pp. 226–274.

[10] W. Hoeffding. On the Distribution of the Number of Successes in Independent Trials, *Ann. Math. Statist.*, 27 (1956), pp. 713-721.

[11] D.E. Knuth. *The Art of Computer Programming, Vol. 3: Sorting and Searching*, Addison-Wesley, Reading, Mass. 1973.

[12] G. Lueker and M. Molodowitch. More Analysis of Double Hashing, *20*th *STOC*, May, 1988, pp. 354–359.

[13] K. Mehlhorn. On the Program size of Perfect and Universal Hash functions, *Proc. 23rd Ann. Symp. on Foundations of Computer Science*, 1982, pp. 170–175.

[14] K. Mehlhorn. *Data Structures and Algorithms 1: Sorting and Searching*, Springer-Verlag, Berlin Heidelberg, 1984.

[15] J.P. Schmidt and A. Siegel. On aspects of universality and performance for closed hashing, *21*st *STOC*, May, 1989, pp. 355–366.

[16] J.P. Schmidt and A. Siegel. Double hashing is computable and randomizable with universal hash functions, submitted.

[17] J.P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding Bounds for Applications with Limited Independence. *Proc. 4th Ann. ACM-SIAM Symp. on Discrete Algorithms*, 1993, 331–340.

[18] A. Siegel. On universal classes of fast hash functions, their time-space tradeoff, and their applications, *Proc. 30th Ann. Symp. on Foundations of Computer Science*, Oct., 1989, pp. 20–25.

[19] J.D. Ullman. A Note on the Efficiency of Hash Functions, *JACM*, Vol 19, 1972, pp. 569–575.

[20] M.N. Wegman and J.L. Carter. New Hash Functions and Their Use in Authentication and Set Equality, Journal of Comp. Syst. Sci. 22, 1981, pp. 265–279.

[21] A.C. Yao. Uniform Hashing Is Optimal, *JACM*, Vol 32, No. 3, July, 1985, pp. 687–693.