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
Incyte Pharmaceuticals, Inc.
3174 Porter Drive
Palo Alto, CA 94304

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 result is the first proof of optimal performance for any classical closed hashing scheme in a model of randomized computation.

The error bound also gives a slight improvement to prior analyses that were based on an idealized computational model that uses fully independent hash functions.

These results are derived from a novel formulation that overestimates the expected probe count by underestimating the presence of local keys already inserted into the hash table, and from a refined analysis of the underlying stochastic structures formed by colliding keys.

Analogous bounds are attained for the expected r-th moments of the probe count, for any fixed r. The proof method extends naturally to show that uniform hashing with universal hash functions has a performance that is equivalent to that with fully random functions, and similar conclusions are also established for linear probing.

The work of the first author was supported in part by ONR grant N00014-85-K-0046, NSF grants CCR-8906949 CCR-8902221, CCR-9204202, and CCR-9504175.

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.

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**]: Non-numerical Algorithms and Problems—*sorting and searching.*

General terms: Algorithms, Theory.

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

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, which is also known as open-address hashing, access to a key x is achieved by following a sequence $p(x,1)$, $p(x,2)$, $p(x,3)$, $\ldots$ of computed probe indices that belong to the range $[0..n-1]$. In particular, a key x is inserted into 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, in which case x is known to lie outside of the data set.

Definition 1.

- Let, for $\alpha < 1$, $D = (x_1, x_2, \ldots, x_{\alpha n})$ be a sequence of keys from the universe of integers $U = [0..m-1]$.
- The elements of D will be sequentially hashed into the table $L[0..n-1]$.
- We say that x_k is hashed at *time* k, and x_k is *embedded* at location ℓ if the hashing of D stores x_k in $L[\ell]$.
- $A \subset B$ will have the usual meaning when A and B are sets. When B is a sequence, A will be a subsequence of B.
- For a sequence S and scalar t, S_t denotes the t-th item in S, so that $D_t = x_t$.
- For $T \subset (1, 2, \ldots, \alpha n)$, let D_T denote the subsequence of keys $(x_t)_{t \in T}$.
- 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 $[0, n^2 - 1]$, and the image multiset is then hashed via idealized uniform hashing. (The model has also been called 2-ary clustering in some publications.) 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], Ajtai, Guibas, Komlós and Szemerédi, Lueker and Molodowitch [12], and Schmidt and Siegel [16]. Lueker and Molodowitch presented a remarkably elegant proof to show that for any fixed load factor $\alpha < 1$, the expected number of probes to locate 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 a result of Yao [21], is optimal in terms of the average number of probes to retrieve keys. Lueker and Molodowitch note that Ajtai, Guibas, Komlós and Szemerédi also established a comparable performance bound for double hashing. Previously, Guibas and Szemerédi, had established the bound for loads $\alpha < \frac{3}{10}$. All of these results require the functions $\{(f(x), d(x))\}_{x \in D}$ to be fully independent and uniformly distributed.

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 $\frac{1}{1-\alpha} + \epsilon + O(\frac{1}{n})$, for any fixed $\alpha < 1$, $\epsilon > 0$ and a large enough constant c. The results are established for a class of probe schemes that is considerably larger than the arithmetic progressions of double hashing.

We now build upon the results and the framework 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. These results also hold for generalizations of double hashing schemes as given in [16]. In addition, comparable bounds are attained for the expected moments of the probe counts.

When combined with the highly random hash functions in [18], the results give the first formal proof of optimal performance for closed hashing with programmable functions that can be evaluated in constant time. (However, the evaluation time is a very large constant. Thus, the result just establishes a scalable plausibility check for current practices, as opposed to a provably good implementation scheme.)

The objectives underlying the generalized hashing models 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.

This approach is more than a matter of intellectual interest. By building a mathematical infrastructure to express the performance in terms of statistical properties that are identical for uniform hashing and double hashing, and by establishing the formulation in a way that is insensitive to limited randomness (with $(c\log n)$-wise independence), the matter of establishing the performance for double hashing simplifies to the summation of error estimates. These terms show that the performance characteristics are far less diverse, across these models of hashing, than one might expect. Once they are shown to be statistically equivalent, we can appeal to the trivial analysis of uniform hashing for fully independent hash functions to conclude, for example, that double hashing with $(c\log n)$-wise independent hash functions, for suitable c, requires an expected $\frac{1}{1-\alpha}+O(\frac{1}{n})$ probes to insert the αn-th key into a table of size n. Simply put, this paper is all about establishing a bunch of $O(\frac{1}{n})$ error terms; the rest comes for free.

Definition 2: The models UH, DH, DH_ψ, and DH_∞^ψ.

- In UH, the probe sequence $p(x,*)$ is an infinite family of independent 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 probe sequence $p(x,*)$ exhibits approximate pairwise independence:

     $\forall x\ \forall i,\ \forall r \in [0, n-1]:\quad \mathrm{Prob}\{p(x,i) = r\} = \frac{1}{n-O(1)}$.[†]

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

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

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

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

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

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

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

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

For DH_ψ, three slight variants are worth defining. For the few instances where it is necessary to distinguish among them, we use the notation $DH_\psi^{(a)}$, $DH_\psi^{(b)}$ and $DH_\psi^{(c)}$. Common characteristics will be attributed to DH_ψ.

- In DH_ψ, the statistical probe behavior of an individual probe sequence is subject to the 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 exhibit statistical independence, for distinct keys. More precisely, we have the following.

  1) $\forall x\ \forall i,\ \forall r \in [0,n-1]:\quad \mathrm{Prob}\{p(x,i) = r\} = \frac{1}{n-O(1)}$.

[†]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 will occasionally use minus signs to suggest that the worst case error is negative. Also, it is reasonable to write, say, $\frac{1}{1+O(h)} = 1 - O(h)$, for $h = o(1)$.

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

2a) Knowing the probe values for a small set of keys, in $DH_{\psi}^{(a)}$, gives no information about the probes for another key. Formally, let $\{z_j\}_{j=1}^{\zeta}$ be a set of $\zeta \le \psi/2$ distinct keys. Then

$$\mathrm{Prob}\{p(z_\zeta,j)=a_j, j=1,2,3,\ldots \ \Big|\ p(z_\ell,j),\ 1\le \ell<\zeta,\ j=1,2,\ldots\} = \mathrm{Prob}\{p(z_\zeta,j)=a_j, j=1,2,3,\ldots\}.$$

2b) For $DH_{\psi}^{(b)}$, the same statement holds, but the probe index j is uniformly bounded by $\kappa \log n$ for a sufficiently large constant κ:

$$\mathrm{Prob}\{p(z_\zeta,j)=a_j, j\le \kappa\log n \ \Big|\ p(z_\ell,j),\ 1\le \ell<\zeta,\ j\le\kappa\log n\} = \mathrm{Prob}\{p(z_\zeta,j)=a_j, j\le\kappa\log n\}.$$

2c) In $DH_{\psi}^{(c)}$, the listing of probe values is for sequences of consecutive probes that are altogether bounded by ψ. Formally, let Z be a set of $\zeta \le \psi/6$ distinct keys with associated probe bounds $\beta_1,\beta_2,\ldots,\beta_\zeta$, where $\sum_j \beta_j \le \psi$. Then

$$\mathrm{Prob}\{p(z_\zeta,j)=a_j,\ j\le\beta_\zeta \ \Big|\ p(z_\ell,i),\ i\le\beta_\ell,\ \ell\le\zeta-1\} = \mathrm{Prob}\{p(z_\zeta,j)=a_j, j\le\beta_\zeta\}.$$

The conditions given in part 2) need only hold for a subset of hash functions $\widehat{F}\subset F$, where $\widehat{F}$ depends on D, and $\frac{|\widehat{F}|}{|F|} = 1-\frac{O(1)}{n^2}$.

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

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

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

6) In $DH_{\psi}^{(b)}$, there is little chance that a probe sequence will revisit lots of locations:

$\mathrm{Prob}\{|\cup_{j=1}^{\kappa\log n}\{p(x,j)\}| < \frac{\kappa\log n}{2}\} = O(\frac{1}{n^3})$, where κ is as in 2)b.

In $DH_{\psi}^{(c)}$, analogous events of this type are even less likely. Formally, let $\mathcal{Z}=\{y_j\}_{j=1}^{\zeta}$ be any set of $\zeta<\frac{\psi}{6}$ keys with associated probe bounds $\{\beta_j\}_{j=1}^{\zeta}$. Let m and the probe bounds satisfy $\sum_j\beta_j+3m+3\le\psi$. Let $S=\cup_j\cup_{i\le\beta_j}\{p(y_j,i)\}$, and suppose that $x\notin\mathcal{Z}$. Then

$$\mathrm{Prob}\{|\cup_{1\le i\le 3m+3}\{p(x,i)\}-S|\le m\ \ |\ p(y_i,j),\ j\le\beta_i,\ i\le\zeta\} = O(\tfrac{1}{n^{m+3}}).$$

For DH, requirement 1) is worded to include both UH and standard double hashing, which happens to generate probe sequences $p(x,1), p(x,2), \ldots$ that are permutations.

Requirement 2) is needed for inclusion-exclusion arguments with limited independence, and especially for applications of the Chernoff-Hoeffding bound given in Lemma A2 of the Appendix. While the mathematical content of 2) is accurate, a few clarifications in plain (but imprecise) English are nevertheless appropriate. The statement says nothing about how random the probe sequencing is for any individual key. Rather, it says that if you have a bunch of distinct keys, each has statistics that, within various limits, appear to be independent of each other. Thus, these formulations define kinds of limited key-wise independence.

The formulations are also designed so that all of DH is included in $DH_\psi^{(a)}$ and $DH_\psi^{(b)}$. In addition, the model $DH_\psi^{(c)}$ contains UH, although DH does not satisfy some of its specifications. On the other hand, the requirements for multiple probes of a single key are fairly weak—much weaker than the usual assumptions of uniform distribution and full independence for, say, the first ψ probes. Further comments about these models and the reasons for formulating them can be found in [16].

It is essential to note that any family of hash functions $\mathcal{F}_\psi \in DH_\psi$ has a corresponding family $\mathcal{F} \in DH$, where these probe probabilities and their ψ-wise joint distributions are exactly the same. The family in DH is created by selecting, for each $x \in D$, a new function $f_x \in \mathcal{F}_\psi$, so that $p(x,j) = f_x(x,j)$. (If, in DH_ψ, property 2 holds only for a subset $\widehat{F}$ of the hash functions (depending on D), then this set should be used to define the hash functions in DH_∞^ψ. Thus, the model permits the hash functions to depend on D, but only in a very weak way.) In any case, the correspondence between DH_ψ and DH_∞^ψ simplifies the arguments for DH_ψ. We will show that all of DH and UH have the same performance, apart from statistically negligible errors. To extend this conclusion to DH_ψ, we need only show that each family in DH_ψ is comparable to its statistical representative in DH.

For specificity, let DH_∞^ψ be defined as follows.

- Given D and $\mathcal{F} \in DH_\psi$, let DH_∞^ψ be this special model in DH, which is constructed by independently selecting a hash function from $\mathcal{F}$ (or a suitable $\widehat{\mathcal{F}} \subset \mathcal{F}$) for each hash key in D.

The statistical differences between DH_ψ and DH_ψ^∞ will be understood to be between corresponding families.

In these formal models, a family of hash functions $\mathcal{F}$ comprises a finite set of functions. Given the data sequence D, a specific hash function is selected by choosing a member from $\mathcal{F}$ at random according to the uniform distribution. The statistical properties defined by UH, DH, and DH_ψ are with respect to $\mathcal{F}$. However, the DH and DH_ψ robustness requirement 3, which demands that the insertion process not fail for load factors $\alpha < 1$, can depend on the specifics of the hashing algorithm. While such a formulation is superfluous for true double hashing, it is necessary for the various abstract generalizations. In any case, there are many straightforward adaptations to ensure that the hashing algorithm will not fail as long as the table is not full. See [16] for a brief presentation of the details.

It is also appropriate to outline why, in property 2) of DH and DH_ψ, the multiplicativity is allowed to hold for a subset $\tilde{\mathcal{F}} \subset \mathcal{F}$ of that could depend on D. As explained in [16] and [18], a family of hash functions might be defined in a composite way so that, say, $D \to [0, n^4 - 1]$, and the specific functions $p(*,*)$ can then map $[0, n^4 - 1] \times [1, \infty]$ into $[0, n-1]^\infty$. The preliminary mapping reduces the size of the domain (and, thereby, the computational complexity) for the second mapping step. In this case, the probe characteristics for keys will not be completely independent because the preliminary mapping could map a pair of keys to the same alias in $[0, n^4 - 1]$. Of course, the expected performance degradation due to such events is negligible.

Bluntly put, this simple adaptation gives a completion to the computational model by accounting for actual preprocessing steps and other possible irregularities that would otherwise render the calculations more difficult. Moreover, the lower bound in [18] shows that universal classes of fast highly independent hash functions must have, for some data sets, a tiny fraction of such junk functions as otherwise their resource requirements in space and random bits will depend upon the size of the key space as well as the size of the data set. Thus, the models DH and DH_ψ are intended to facilitate both a tractable performance analysis and the use of formally feasible (albeit currently impractical) hash functions that can be evaluated in constant time. But the real value in quantifying these irregularities, is in the freedom to ignore them without reducing the validity of the performance results. They will not be mentioned again.

In [16], the analysis was based on dependency graphs, and multiplicative vacancy overestimators.

Definition 3.

- Given a sequence D of hash keys, let the hashing of D define a directed *dependency graph* $G(D)$ that is constructed as follows. The vertex set for $G(D)$ is D and the edge set is initially empty. Suppose that when inserting x into the hash table, x is placed in its k-th probe location l_k after probing $l_1, \ldots, l_{k-1}$. Then directed edges from x to the keys $x_{T_1}, \ldots, x_{T_{k-1}}$ residing in table locations $l_1, \ldots, l_{k-1}$ are included in $G(D)$. Each edge is labeled with its corresponding probe number (i.e., an edge from x to $y = L[p(x, j)]$ has the label j), and each vertex x_i bears the label i. Technically, these structures are multigraphs because a vertex can have multiple edges to the same table location. However, such multiple edges will cause no ambiguity since they all have different probe counts.

- The *dependency graph* for $x \in D$, $G(x, D)$, is the restriction of $G(D)$ to the vertex set comprising x and all nodes reachable from x in $G(D)$. Its edges and vertices are all labeled.

- The *dependency set* for x, $dep(x, D)$, is the set of keys in $G(x, D)$. We will typically view the dependency set as a subsequence of D instead of a true set.

- The *partial dependency* graphs for x_i are the $probe_i$ subgraphs $G_0(x_i, D)$, $G_1(x_i, D)$, $\ldots$, $G_{probe_i - 1}(x_i, D)$ of $G(x_i, D)$, where $G_r(x_i, D)$ comprises the portion of $G(x_i, D)$ that is reachable from the first r probes for x_i. Note that $G_0(x_i, D)$ comprises the singleton node x_i.

- A subsequence $S \subset D$, with $x = S_{|S|}$, is a *local dependency set* for x if $dep(x, S)$ comprises the elements of S, which is to say that S hashes by itself into a dependency set.

Thus, x_k will, when D is hashed, be the root of a unique dependency DAG, be the roots of $probe_k$ partial dependency DAGs, and even more local dependency DAGs. It should also be noted that the graphs $G_r(x, D)$ and $G_{r+1}(x, D)$ might have the same vertex set and only differ in the outdegree of the root x.

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 partial dependency DAGs rooted by $x_{\alpha n}$. The statistical closeness between hashing models is established by analyzing these structures in terms of their graph properties without reference to their specific embedding locations. In this formulation, a sample of keys, as shown in the forthcoming Lemma 1, forms a local dependency DAG with nearly identical probabilities in UH, DH, and DH_ψ. However, intermediate calculations will sometimes include the specific embedding locations — either for establishing the fact that they do not matter (much) or for expositional convenience.

The difficult part of the formulation is to estimate the probability that a local dependency DAG with vertices S is partial, which requires not only that S hash locally into a dependency DAG, but also that these locally determined hash locations are vacant at the actual insertion times of S as determined by the full sequence of keys D.

Definition 4.

- Let I be a sequence of distinct elements belonging to $\{0, 1, 2, \ldots, n-1\}$ and let $T \subset [1, 2, \ldots, \alpha n]$ be a sequence of $|I|$ insertion times. Define the predicate $M^{(h)}(T, I)$, to be true precisely when there is no tuple $S \subset (x_1, \ldots, x_{T_j - 1}) - D_T$ of size $|S| \le h$ that hashes into a dependency tree G rooted at location I_j, for $j = 1, 2, \ldots, |I|$. We call $M^{(h)}(T, I)$ a vacancy criterion.

- 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 in $(x_1, \ldots, x_{j-1}) - D_T$, that hashes by itself into a tree that is rooted at I_j. We choose to ignore non-tree DAGs in this definition. The most important property of a vacancy criterion is that it must be true whenever a location is actually vacant. Good criteria should also depend on a parameter that allows the vacancy overestimates to converge in measure to the true vacancy probability, and they should have at least somewhat tractable statistics under the restrictions of limited independence.

Calculations in [16] prove that for any fixed $\epsilon > 0$, fixed load factor $\alpha < 1$, and constant c, there are constants d and h so that $\psi = (d \log n)$-wise independence is sufficient to ensure that for $|T| \le c \log n$, $q^{(h)}_{DH_\psi}(T,I) = (1+O(\frac{|I|^3}{n}))q^{(h)}_{UH}(T,I)$. (In the preceding formulation, the subscripts denote the probabilities in the different hashing models.) Thus, the vacancy overestimates are almost independent of the underlying hashing model. Furthermore, it is established that $q^{(h)}_{UH}(T,I) = (1+O(\frac{|I|^3}{n}))\prod_{i=1}^{|I|} q^{(h)}_{UH}(T_i,i)$, which shows that the (overestimated) vacancy probability is almost independent of the locations in I. It is also proven that for suitable independence $\psi = d\log n$, $q^{(h)}_{UH}(T,I)$ and $q^{(h)}_{DH}(T,I)$ behave as stated when conditioned on the event that D_T is locally hashed into a specific dependency graph embedded in the locations specified by I. Now, the probability that D_T hashes into a given structure with specific embedding locations depends tremendously on the hashing model, but the vacancy probabilities do not. Thus, even in DH_ψ, the vacancy overestimates based on the hashing of $D - D_T$ are almost independent of the local hashing events defined by D_T, provided $|I| \le c\log n$, for some suitable constant c. Lastly, it is shown that the multiplicative vacancy overestimators $q^{(h)}(t,i)$, for scalars t and i, are nearly independent of i, and are within a factor of $1+\epsilon$ of the true vacancy estimate in UH, which is $q_{UH}(t,i) = 1 - \frac{t-1}{n}$. These properties, plus some large deviation bounds and a few other estimates are sufficient to establish Theorem 4 [16], which states that the expected number of probes to insert $x_{\alpha n}$ is at most

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

where $Q^{(h)}(k) = \frac{1}{n}\sum_{i=1}^{k-1} q^{(h)}(i)$ and, say, $q^{(h)}(t) = q^{(h)}(t,1)$. This bound, as a function of $Q^{(h)}$, is established for any hashing procedure that satisfies the statistical properties of DH_ψ, for $\psi \ge d\log n$, where d is some constant that is independent of n. The different models give numerical differences in the resulting overestimates for $\mathrm{E}[probe_{\alpha n}]$ that are all within $O(\frac{1}{n})$ of each other (and are all at most $\epsilon + O(\frac{1}{n})$ too large).

The proof is based on a summation, over all candidate dependency sets, of weighted probability estimates that the set hashes into a partial dependency set. The weighted probabilities have two factors: the probability that the set hashes locally into a dependency structure, and $\mathrm{Prob}\{M^{(h)}(T,I)\}$, which overestimates the probability that the locations as determined by the local hashing are vacant at their designated insertion times.

It is also worth observing that the true vacancy estimate $q(i) = 1 - \frac{i-1}{n}$ 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 the estimated expectation will be augmented by (too many) spurious probe statistics from DAGs other than the actual dependency graph for $x_{\alpha n}$. Yet the errors resulting from very accurate vacancy formulations appear to be rather difficult to bound satisfactorily.

2. Superdependency graphs

We now build another layer of structures on top of those in [16], appeal to inclusion-exclusion and apply more complex recurrences to establish an even sharper approximate statistical isomorphism between double hashing and uniform hashing. The inclusion-exclusion will, statistically, eliminate the overcount of spurious probe collisions between $x_{\alpha n}$ and previously processed keys that belong to local dependency sets with hashing locations that satisfy the weak vacancy criterion, but are actually occupied by other elements from D.

To analyze aggregates of local dependency DAGs, we extend the notion of a dependency DAG to larger ensembles of data. It will be convenient to use three different models for these aggregate collision structures. The first model is the most natural extension of dependency graphs. Another uses a bipartite DAG formulation, and yet another will comprise a somewhat ambiguous interpretation of the original dependency graph representation. The graphs are defined in two passes; the second pass builds a weak notion of the vacancy requirements into the structures. For the convenience of the reader, the bipartite construction is first introduced for an ordinary dependency DAG.

Definition 5.

- The superdependency graph $G_{sdep}(x, D)$ is defined as follows. The vertex set comprises the union of all local dependency sets for x. Let E_G be the edges in $G(D)$, and let E_x be the union of edges in local dependency DAGs for x. The edges in $G_{sdep}(x, D)$ comprise $E_x \cap E_G$. Let

$sdep(x, D)$ be the vertex set for $G_{sdep}(x, D)$.

- The bipartite dependency graph $BG(D)$ has two types of vertices: locations and keys. As D is hashed, $BG(D)$ is built as follows. If $x \in D$ is embedded in its r-th probe location $p(x, r) = \ell$, then $BG(D)$ will contain an edge from a vertex that represents ℓ to x, and from x to vertices representing the locations $p(x, 1)$, $p(x, 2)$, ..., $p(x, r - 1)$. The edge $(x, p(x, j))$ is labeled j, for $j = 1, 2, \ldots, r - 1$, and the edge (ℓ, x) is labeled r. Formally, such a graph is a series of equalities of the form $p(x, j) = p(y, k)$, and inequalities of the form $p(x, r) \neq p(y, i)$.

- The bipartite dependency DAG $G_{bdep}(x, D)$ is the subgraph reachable from x in $G_{bdep}(D)$. Its constituent vertex keys are the elements of the set $bdep(x, D)$.

- A local bipartite dependency DAG for x, is a graph $G_{bdep}(x, S)$ where $S \subset D$, and $bdep(x, S)$ comprises the keys in S.

- The bipartite superdependency graph $G_{bsdep}(x, D)$ is defined as follows. Its vertices comprise the union of vertices belonging to local bipartite dependency DAGs for x. The edges of $G_{bsdep}(x, D)$ comprise the union of edges belonging to local bipartite dependency DAGs for x.

- The r-th partial bipartite superdependency DAG for x comprises the restriction of the bipartite superdependency graph to the subgraph reachable from the first r probes for x, where r is less than the number of probes needed to insert x.

- The r-th partial superdependency DAG for x is defined analogously. It comprises the union of r-th partial local dependency DAGs with edges restricted to $Gsdep(x, D)$.

- A vertex that belongs to some local dependency set for x (and is not x), and which, when all of D is hashed, will actually reside in a later probe location than that indicated by the local dependency set, is called a *dummy*.

- We will call any type of superdependency DAG tree-like if its vertex count exceeds its edge count by one. The root of a tree-like superdependency DAG for x will be x.

The superdependency graphs $G_{sdep}(x, D)$ are similar to the regular dependency DAGs, but have a more complex structure. For example, suppose that y belongs to $dep(x, D)$, and suppose that y will be embedded (as an element in D) in location ℓ. There might be other nodes in local dependency sets $dep(x, D')$ that would reside in location ℓ and would belong to the dependency set for x, were y (and/or other keys) removed from D. These vertices are dummies. Evidently, a vertex (other than x) with indegree zero in $G_{sdep}(x, D)$ must be a dummy, since no key in the superdependency set is hashed in a way that depends upon its presence. Other dummies can be embedded in a superdependency DAG, which means that they can have a positive indegree. In this case, some probe other than the key's last creates the false local embedding that can be mistakenly used to create a local dependency DAG. A key could also play a dummy role for some local dependency sets for root $x_{\alpha n}$, and still belong to $x_{\alpha n}$'s actual dependency set. In the bipartite formulation, the dummies are key vertices that serve as the head of locally defined edges that are not in $BG(D)$. Of course, all information about the actual embedding locations of keys is suppressed to ensure that similar structures in, say, DH and UH will have nearly identical probabilities of occurring.

Most of the analysis will concern the case where the superdependency DAGs are tree-like. In this case, the bipartite superdependency structures have a very natural interpretation.

Definition 6.

We say that T is an AND_OR superdependency tree if

1) T is finite.

2) The root of T is an AND node.

3) Each AND node has zero or more children that are OR nodes.

4) Each OR node has one or more children that are AND_OR superdependency trees.

5) Each AND node stores a key $x_i \in D$, and the insertion time label i as well.

6) Each OR node stores represents a unique index in the hash table L.

7) The insertion time for the key in each AND node is larger than the times associated with its descendent AND nodes.

8) No represented index or key value appears in two nodes of T.

9) The edges exiting each AND node v are labeled $1, 2, 3, \ldots, outdegree(v)$, and are sorted in increasing order by label.

10) The edge entering an AND node v bears a label equal to $outdegree(v) + 1$.

11) The edges exiting each OR node are sorted in decreasing order of the insertion times for the keys in their destination vertices.

An AND_OR tree is just an interpretation of a tree-like bipartite superdependency DAG. Each key, apart from the root, is inserted in the location indicated by the tail of the edge that points to the key node. Each key has descending pointers to vertices representing the table locations unsuccessfully probed during the search to insert the key. Each child of an OR node, apart from the last, is a dummy.

The issue of non-tree-like structures is postponed.

For counting purposes, an alternative tree representation turns out to be especially useful. This change turns an AND_OR tree A into an annotated dependency graph as follows. Let each dummy key in the AND_OR DAG carry the label "dummy." Then every OR node can be replaced by a linked list as follows. The list contains the children of the node, where the edge (in A) into the child is replaced by an edge from the child's left sibling, if present. The new edge has the same label as the one it replaces. The left sibling's new edge is appended to its edge list. The edge into the parent OR node in A is redirected to point to the first node in the OR node's child list, and this edge again has the same label as the one it replaces. Evidently, this transformation is one to one, and the total number of collisions in the new structure, when viewed as a normal dependency DAG, is the same as that in the AND_OR representation. All location nodes are lost, and the resulting structure is, effectively, an ordinary dependency tree where any node that is neither the root nor a leaf might carry a dummy label.

If the dummy labels are erased in a dependency tree structure with w keys and ℓ leaves, it will then represent $2^{w-\ell-1}$ different superdependency graphs bundled together.

Definition 7.

- Let a tree-like non-bipartite labeled dependency structure G be called a bundled superdependency DAG if it with its labels are interpreted as representing the collection of AND_OR trees that, when processed as above, yield this dependency structure, apart from the missing dummy labels.

To analyze superdependency DAGs that are not tree-like, we will give a simple algorithm to map each such DAG to a canonical tree-like structure that is contained within the DAG. In the traversal, a vertex's outgoing edges to new vertices are saved at preorder time to maximize the vertex's probe count in the resulting structure. To keep the structure tree-like, one cross edge is saved for each vertex d that (apart from the root) has an indegree of zero. The cross edge will be the last probe edge exiting the dummy d in the superdependency DAG. It should be noted that an embedded node in a superdependency DAG G must be the first to probe its embedding location. Moreover, any node d in G that is not the root and has indegree 0 must have a last probe in G to some destination vertex x, which is probed by some other node z that has an insertion time exceeding that of d. Otherwise there will be no subset of vertices in G that could form a local dependency DAG with d occupying x's embedding location. The tree-like structure will define, for each embedded vertex x, the parent of x to be the vertex that, among all vertices in G that probe x's embedding location, has the largest insertion time.

The construction of the tree-like subgraph will be top-down. To facilitate this processing, let the vertices of G have an indexing that is in the reverse order of the insertion times and consecutive, with the root being vertex 1.

Because the vertices are explored in order, each node will indeed have the lowest numbered vertex to probe its location as its parent. Furthermore, each dummy d with indegree zero will have a last probe in G to a vertex x that has already been marked (and adopted) by its parent y, since y must be less than d.

The details are given below. The reverse order sequencing ensures that the cross edges are as described.

Definition 8.

Let $G = (V, E)$ model a superdependency DAG with $|V| = \{1, 2, \ldots, j\}$. Let 1 be the root, and the identifier of each parent exceed that of its children.

The following procedure produces a tree-like subset of G that contains V. The selected tree-like edges are saved in E_{Tr}.

```
make all vertices unmarked;
mark 1;
for j ← 1 to k do
    for childdex ← 1 to |GAdj[j]| do
        jchild ← GAdj[childex]
        if jchild unmarked then
            mark jchild
            include (j, jchild) in E_Tr
        endif
    endfor
    if j not marked then include the cross edge (j, jchild) in E_Tr endif
endfor;
```

- Let the tree-like subgraph (V, E_{Tr}) as extracted by the above procedure be G's *omnidirectional spanning tree.*

The bipartite and bundled superdependency DAG representations can be applied to the omnidirectional spanning tree that is derived from a general superdependency DAG.

Lemma 1 below shows that a v-vertex e-edge superdependency graph has essentially the same local probability distribution as its dependency counterpart. For example, it will turn out that v keys hash into a given v-vertex $(v-1)$-edge superdependency graph with probability $\frac{1+O(v^2/n)}{n^{v-1}}$. More precisely, we have the following.

Lemma 1. Suppose $\psi = O(\log n)$, $6k \leq \psi$ and $S = (S_1, S_2 \ldots S_k)$ is a sequence of k distinct elements in U. Let a superdependency DAG be degenerate, in $DH_{\psi}^{(b)}$, if it contains a vertex with an outdegree of at least $\kappa \log n$. Let there be no degenerate DAGs in the other models.

1) Let Tr be a tree-like superdependency DAG of k vertices. Then the probability that the local hashing of S yields the superdependency structure Tr is, in DH_ψ, $\frac{1+O(k^2/n)}{n^{k-1}}$.

2) The set of all k-vertex superdependency DAGs that are not degenerate can be partitioned into equivalence classes so that each class is represented by an omnidirectional spanning tree that is contained as a subgraph of each element in the corresponding class, and the probability that S is hashed into a structure that is not tree-like, and that belongs to the class corresponding to a given k-element tree-like superdependency DAG is bounded by $O(\frac{k^3}{n^k})$ in DH_ψ.

3) The probability S hashes into a non-degenerate structure that belongs to a given equivalence class as defined in 2), and that has at least $2k$ probe edges emanating from the root is, in DH_ψ, bounded by $O(\frac{k^3}{n^{k+1}})$.

Proof: Let $\widehat{S} = (\widehat{S_1}, \widehat{S_2}, \ldots, \widehat{S_k}) = (S_k, S_{k-1}, \ldots, S_1)$ be the reversed ordering of the sequence S. The lemma will be established by implicitly identifying all possible embeddings of Tr, and estimating the probability that the hashing of S builds a superdependency structure that has its canonical omnidirectional spanning tree isomorphic to Tr. By definition, the vertex set for Tr is $\{1, \ldots, k\}$, and Tr's adjacency lists are ordered. Let, h_j, for $j = 1, 2, \ldots, k$, be the outdegree of vertex j in Tr, where any (terminating) cross edge, if present, is excluded from the count. Let the intended mapping (graph subisomorphism) between the hashing of S and Tr map $\widehat{S_j}$ to j. The statistical hashing of S into a superdependency DAG will proceed one node at a time.

For specificity, we will embed the (potential) root $\widehat{S_1}$, but will impose no constraint for the location. It will be evident that this embedding will have a nominal asymptotic effect on the overall probabilities for the embedding of all other nodes of $\widehat{S}$: the same formulations would result if this first embedding step were omitted.

1) We begin by considering the case where the hashing of S is into a tree-like structure. First, the root $\widehat{S_1}$ is embedded in L. Tr prescribes that $\widehat{S_1}$ must be embedded in its (h_1+1)-st probe location, which is any one of n target locations, and then the h_1 locations for the children

are specified. Of course, the root is not really embedded, and the asymptotics do not change if we embed the root at random into an empty table or not. Similarly, a dummy root is not embedded, but we can use language to describe its terminating cross edge as a virtual embedding requirement, since our processing will ensure that each terminating cross edge will be to a predefined location. With this understanding, we can say that whenever vertex is embedded, its last probe location will have been prespecified (unless the vertex is the root). Its earlier probe locations are used to specify the target destinations for the subsequent embedding of its children. The probability of the actual embeddings are computed sequentially, for the vertices $1, 2, 3, \ldots, k$. For completeness, it should be pointed out that a tree-edge, in Tr, always exits the lowest numbered vertex to probe a given location, and its destination vertex is the highest numbered such vertex. Dummy roots have names that lie between these extreme values.

For $\widehat{S_1}$, the probability that its first $h_1 + 1$ probes are to distinct locations is between 1 and $1 - |O(h_1^2/n)|$. The hashing of subsequent nodes will have two constraints: a node with h_j children must have its $(h_j + 1)$-st probe location prespecified, and the first h_j probes must be to h_j distinct locations that have not yet become specified targets for embedding. Let R_j be the set of target embedding locations that are specified prior to the hashing of $\widehat{S_j}$, and let r_j be the location for $\widehat{S_j}$.

The probability that the hashing of a node $\widehat{S_j}$ satisfies the location specification for its (h_j+1)-st probe, and has, as prescribed by Tr, h_j children specifications is:

$$\mathrm{Prob}\left\{(p(\widehat{S_j}, h_j + 1) = r_j) \bigwedge_{1 \le i < \ell \le h_j} (p(\widehat{S_j}, i) \ne p(\widehat{S_j}, \ell)) \bigwedge_{1 \le i \le h_j} (p(\widehat{S_j}, i) \notin R_j)\right\},$$

which is at most $\mathrm{Prob}\{(p(\widehat{S_j}, h_j + 1) = r_j)\} \le \frac{1+O(1/n)}{n}$, and at least

$$\mathrm{Prob}\left\{(p(\widehat{S_j}, h_j + 1) = r_j)\right\} - \sum_{1 \le i < \ell \le h_j} \mathrm{Prob}\left\{(p(\widehat{S_j}, h_j + 1) = r_j), (p(\widehat{S_j}, i) = p(\widehat{S_j}, \ell))\right\}$$
$$- \sum_{\substack{1 \le i \le h_j \\ r \in R_j}} \mathrm{Prob}\left\{(p(\widehat{S_j}, h_j + 1) = r_j), (p(\widehat{S_j}, i) = r)\right\},$$

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

2) If S is to hash into a structure that is not tree-like, then we must bound the probability that the hashing produces a DAG that has Tr as its canonical omnidirectional spanning tree. The probability considerations are similar to those in the preceding case, but some nodes $x \in \widehat{S}$, will have gaps in the probe sequences $p(x,1), p(x,2), \ldots$ of edges that correspond to those in Tr. Of course, any k keys have initial probe sequences that are mutually independent of each other as long as the total number of probes is bounded by ψ, in $DH_\psi^{(c)}$. The other DH_ψ models, will be seen to require less independence. So first suppose that the total number of probes is at most ψ. In this case, the probability that V is hashed into a DAG that (is Tr or) contains Tr as its omnidirectional spanning tree 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 number of probes to locations that are available for its spanning tree children, and has a terminating probe that satisfies its prescribed probe constraint.

To compute pr_j, it is convenient to distinguish among three subcases: i) $\widehat{S_j}$ has exactly h_j exiting edges (plus a terminating cross edge, if present in Tr), ii) $\widehat{S_j}$ has, in addition to the h_j or $h_j{+}1$ edges prescribed by Tr, at least 1 at most $h_j + 1$ other exiting edges, and iii) $\widehat{S_j}$ has at least $h_j + 2$ such additional edges. Note that if no location can appear twice in a probe sequence — as is the case for real double hashing — then ii) and iii) can be combined into the event that $\widehat{S_j}$ has at least 1 additional edge, and no more than $k-1$ edges altogether.

i) This subcase has the same analysis as 1), and contributes a probability of at most 1 for pr_1, and at most $\frac{1}{n}(1 + O(1/n))$ for pr_j, for $j > 1$.

ii) In this subcase, there are different DAG structures, depending on which probe count for $\widehat{S_j}$ is terminal. The count must lie within the sequence $h_j{+}2, \ldots, 2h_j{+}2$. Summing over all possible last probe counts, over the possible probe counts that might† correspond to the first edge that

†Technically, the event

is not tree-like (and which must be 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 $\frac{1}{n}(1+O(\frac{1}{n}))+O(\frac{h_jk}{n^2})$ as an overestimate for the probability contributed to pr_j by case 2.

iii) In this subcase, there must be two consecutive edges that are not tree-like among the first $2h_j+2$ probes of $\widehat{S_j}$. These edges might go to previously embedded keys or collide with earlier probes of $\widehat{S_j}$. To estimate this contribution to pr_j, we ignore the prespecified location requirement for the terminating probe of $\widehat{S_j}$, and simply overcount the expected number of ways a consecutive pair of such probes could occur, which is bounded by $(2h_j+1)\frac{O(k^2)}{n^2}$. There are, however, a few possible complications that we defer to case iv).

Combining terms from the three cases into factors and multiplying gives the following overestimate for the probability that S hashes into a DAG structure that is represented by (or is the same as) Tr:

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

Hence, the probability that S hashes into a DAG that is not tree-like but is represented by Tr is at most $(\frac{1}{n})^{k-1}(1+O(k^3/n)) - (1-\frac{|O(k^2)|}{n})(\frac{1}{n})^{k-1} = O(k^3/n^k)$. Moreover, it follows from the reasoning in iii) that the probability, in all three models of DH_ψ, is $O(k^3/n^{k+1})$ that the k vertices form such a DAG with the root having more than $2k$ edges.

iv) This multiplicative formulation, however, is not valid if the total number of probes needed to embed the structure exceeds ψ. This issue is resolved differently for different models of DH_ψ. The multiplicativity even deserves some justification for $DH_\psi^{(a)}$; see [16], where an analogous argument is given (for dependency DAGs) in Lemma 2.

$\langle$*$p(\widehat{S_j},i)$ is the first non-tree probe, $p(\widehat{S_j},\ell)$ is to the prespecified terminal probe location for $\widehat{S_j}$, and there are exactly h_j probes among the first ℓ to as yet unspecified locations*$\rangle$

is outside of the kinds of events we can specify with two degrees of freedom. Instead, we use overlapping superset decompositions of the event space (rather than a true partition) of the form:

$\langle$*$p(\widehat{S_j},i)$ is to a previously specified location for some vertex in the DAG, and $p(\widehat{S_j},\ell)$ is to the prespecified location for the terminating probe of $\widehat{S_j}$ in Tr*$\rangle$.

For $DH^{(b)}_{\psi}$ and $DH^{(c)}_{\psi}$, it is possible that there are so many non-tree edges that the allocated independence can be used up before all of a superdependency set's embedding locations have been discovered. In $DH^{(b)}_{\psi}$, the problem has been eliminated by the requirement of non-degeneracy, which introduces certain statistical undercounts for the expected number of probes by ignoring probe-excessive superdependency DAGs. The expected error due to this simplification will be bounded by the term Err_b as introduced in Definition 14.

For $DH^{(c)}_{\psi}$, an amortized attribution of probabilistic cause is needed when the superdependency DAG G comprises $|V| \le \frac{\psi}{6}$ vertices and $|E| > \psi$ probes. To bound the probability that a candidate superdependency set S is hashed into a probe-excessive superdependency DAG, we redefine the partitioning of all such DAGs into equivalence sets indexed by omnidirectional spanning trees. The difficulty with Tr as extracted is that important tree edges might be omitted from some subsets of initial probe sequences for each key. The following procedure begins with a candidate superdependency DAG G and its omnidirectional spanning tree Tr as previously defined. It then substitutes edges one-by-one to construct a sequence of omnidirectional spanning trees, and to reallocate the limited resource of available probes per vertex.

The idea is to view the edges as undirected. Each restructuring step will begin by adding a new edge to Tr. Whenever a new edge (a, b) is added to Tr, the modified graph will contain a cycle, when the edges are viewed as undirected. Consequently, there will be a unique edge (c, b), with $c \ne a$, that can be deleted from the new structure to get new omnidirectional spanning tree, which we will still call Tr. For convenience,

- let the function procedure $Swap(Tr, e, b)$ modify the tree that results when $e = (a, b)$ is inserted, and the removed edge is of the form (c, b), with $c \ne a$. Let c be returned by $Swap$. The call $Swap(Tr, e, b)$ will be an error if b is not an endpoint of e.

Recall that the vertex j has h_j children in Tr, and this count is defined to exclude the terminating cross edge from j, if j is a dummy root. With this definition, j will have $h_j + 1$ probes specified by the tree Tr regardless of whether j is embedded or is a dummy root. (Vertex 1 requires just h_1 probes, since it is technically unembedded. But we will view it as embedded just for the uniformity of exposition. The following restructuring algorithm begins with a k-vertex superdependency DAG $G = (V = \{1, 2, \ldots, k\}, E)$ and its omnidirectional spanning tree Tr as extracted according to Definition 10.

```
for j ← 1 to k do
    β_j ← 3h_j + 3
endfor
repeat
    mark all vertices untouched
    for j ← 1 to k do
        childex ← 1
        while childex ≤ β_j and childex ≤ |GAdj[j]| do
            childj ← (GAdj[j])[childex]
            if childj not touched then
                mark childj touched;
                if (j, childj) ∉ Tr then
                    x ← Swap(Tr, (j, childj), childj)
                    β_x ← β_x − 3
                    β_j ← β_j + 3
                endif
            endif
            childex ← childex + 1
        endwhile
    endfor
until there are no changes in any β_*
```

We now show that the procedure terminates. A first step is to observe that β_1 is maximal at its initialization, since vertex 1 will, at the first round, adopt every vertex that is probed within its first β_1 probes for its children, and these edges will all already be in Tr. Thereafter, β_1 can only decrease as vertex 1 surrenders untouched children to other vertices. Consequently, β_1 is non-increasing, and must eventually stabilize at its minimum value. The induction hypothesis supposes that after some iteration, β_i, for $i \le \ell$ are all constant. Subsequently, $\beta_{\ell+1}$ might be changing, but must attain some value that is maximum, among its assignments that follow the stabilization of $\beta_1, \ldots \beta_\ell$. At that point, vertex $\ell + 1$ has adopted all neighboring vertices in its first $\beta_{\ell+1}$ probes as its own (and as destinations of exiting edges in Tr), except for those already touched and adopted by the restricted probes of vertices $1, \ldots, \ell$. Thereafter, $\beta_{\ell+1}$ can only be nonincreasing from iteration to iteration, since vertex $\ell + 1$ will never have, within its first $\beta_{\ell+1}$ probes, any additional vertices available for adoption. It follows that its count must also stabilize at some iteration. Hence all β_* counts stabilize, and the algorithm must terminate. So let β_j be these finalized values for.$j = 1, 2, \ldots, k$.

If $\beta_j > |GAdj[j]|$. Then all of j's probes are revealed within its allotment of β_j probes, and case i), ii), or iii) is applicable to establish pr_j. But there is a mild catch that we momentarily defer. If, on the other hand, $\beta_j \le |GAdj[j]|$, then the number of keys that j probes within its first β_j probes, and which are not among the probe-restricted discoveries for vertices $1, \ldots, j-1$ is at most $\frac{\beta_j}{3} - 1$. Consequently, the excessive probe bound of Definition 2 shows that the probability that $\widehat{S_j}$ hashes to achieve this characteristic, given the initial probe sequences of $\widehat{S_1}, \ldots, \widehat{S_{j-1}}$, is $O(\frac{1}{n^{\beta_j/3+2}})$. This value is so small that we can award a factor of $\frac{1}{n}$ to each true child of vertex j in Tr, and still allocate a factor $O(\frac{1}{n^3})$ to j's terminating probe. With this allocation of probabilities, it does not matter if vertex j reveals the terminal probe prespecifications for its children or not. And it does not matter if j's last probe in G is independent of the location constraint as issued by its parent.

The mild catch is that when $\beta_j > |GAdj[j]|$, the processing of the edge set by cases i), ii), or iii) is based on the probability that one or two probes of vertex j are to locations that have been prespecified by the probing of earlier keys. But if these one or two other keys do not have sufficient probe allotments to reveal the prespecified location(s), we cannot assert that the constraint(s) and the hashing of vertex j are independent. The solution is straightforward. Think of a vertex as owning probe locations rather than vertices in its adjacency list for Tr. Let x be an abstract hashing location that would be prespecified by a probe from x-s owner — name it vertex i — and now suppose that $\beta_i \leq |GAdj[i]|$. Let Σ be the collection of all probes to x that are selected to define probability constraints as described for the cases i), ii), or iii). These probes are required to be from vertices that have exiting edge counts in G that are less than their corresponding β assignments. For example, the vertex that is supposed to reside in location x as defined by Tr would have its last probe belonging to Σ if its probes as recorded in G satisfy the requirements of cases i) or ii). Other vertices can have selected probes to location x based on cases ii) or iii). (Σ was defined in terms of probes rather than keys because a key could have two constraining probes to the same location. Thus, a probe in Σ should comprise the vertex name and its probe number for each probe to x in Σ.) We use the probe in Σ that originates from the key with the smallest index (in the reverse ordering of keys, and smallest probe number, if the key has two such probes in Σ) to specify the location constraint for the $|\Sigma| - 1$ other probes in Σ. Σ will now represent $|\Sigma| - 1$ real constraints, which is one too few. But the missing factor of $\frac{1}{n}$ is available from x's owner — vertex i — due to its excessive probe characteristic. A more detailed argument would retrace the same reasonings given for cases i)—iii), but now applied to the members of Σ.

It is now easy to estimate the probability that $\widehat{S}$ hashes into a DAG that results with the spanning tree Tr, in all cases where $|V| = k \leq \frac{\psi}{6}$ and $|E|$ might or might not exceed ψ. There are four cases per vertex. If $|GAdj[j]| < 3h_j+3$, then case i), ii), or iii) is applicable, and requires at most $3h_j + 3$ degrees of freedom for the hashing of $\widehat{S_j}$. If not, the rare probe requirement of Definition 2.6) for $DH_\psi^{(c)}$ will apply (and use $3h_j + 3$ degrees of freedom). Combining all of these events gives a bound of

$$p_j = (\frac{1}{n} + O(\frac{h_j k^2}{n^2}) + O(\frac{1}{n^3})),$$

whose product then bounds the probability that S can hash into a structure that is Tr or has Tr as its omnidirectional spanning tree. In more concrete terms, the equivalence class for

Tr can be viewed as being contained in the refinement of a Cartesian product space. In this product space, (a superset of) the successful events correspond to events that map vertex j into either the integer $h_j + 1$ and the designator i, or a designator in the set $\{ii, iii\}$ and pair of integers bounded, respectively, by k and $2h_j + 2$, or the designator "excessive probes." The probability that S hashes into some member of the equivalence class is bounded by the sum of the probabilities that S hashes into one of the decompositions, which in aggregate, is bounded by Πp_j. Claim 2) now follows from subtracting the lower bound for the probability that S hashes into the exact structure Tr.

3 Claim 3) is a straightforward application of case iii) above.

Lastly, we must account for the necessary limited independence. For tree-like superdependency DAGs, each vertex uses a probe count equal to its number of children plus one. Of course, this sum is bounded by $2k$ degrees of freedom for a k-vertex tree. The analysis for DAGs that are not tree-like used additional independence. The worst-case instances in these accounting schemes use $3h_j + 3$ probes for each vertex $j \in Tr$. Summing $3(|V| + |E_{Tr}|)$ for an omnidirectional spanning tree Tr gives less than $6|V|$ degrees of freedom that must be reserved from ψ, in $DH_\psi^{(c)}$. The accounting for $DH_\psi^{(a)}$ is less severe, and $DH_\psi^{(b)}$ just requires that $\kappa \log n$ be at least as large as $2k$, for a k-key superdependency set.

Altogether, the independence requirement is bounded by $6k$. ∎

While the proof of Lemma 1 sometimes did not attribute any randomness to certain tree-like edges, our counting will pretend otherwise. That is, the recurrence equations will count the expected number of tuples that hash into various structures by sequencing over the tuples and the probe collisions that give all omnidirectional spanning trees. The countings will give each local insertion a probability of $\frac{1}{n}$ even where our argument did not. However, the net effect is that in the exceptional circumstances where this uniform distribution might not hold, the probabilities for these pathologies correspond to asymptotically negligible errors that introduce, in aggregate, a multiplicative factor of the form $(1 + O(\frac{k^3}{n}))$ to the coefficients of the ensuing recurrence equations.

From a less detailed perspective, Lemma 1 shows that the probability that a k-tuple hashes into a k-node tree-like superdependency structure DAG is changed by a factor of $(1+O(\frac{k^3}{n}))$, when the non-tree-like superdependency DAGs that it represents are included as additional hashing structures. This nominal change 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, the exponential blowup in the number of interpretations of a bundled superdependency DAG will have to be eliminated.

A way to control this growth is by extending the definitions of superdependency graphs to include the vacancy criterion. The vacancy criterion requires that any dummy root z in a local superdependency set S, which locally hashes into a location l, must not have h or fewer items preceding it in D that locally hash into a tree-like structure rooted at l.

The following definitions (9 and 10) incorporate the vacancy criterion into dependency graphs and superdependency graphs.

Definition 9.

- The sequence $S = D_T$ is a local h-dependency set for $x = S_{|S|}$ if $S = dep(x, D')$, for some $D' \subset D$, and for each S_j, its locally determined hash location I_j satisfies the vacancy criterion $M^{(h)}(T_j, I_j)$, which is based on $D - D_T$. In this case, we say that $S = dep^{(h)}(x, D')$.

- The h-superdependency set for x, $sdep^{(h)}(x, D)$ is the union of all local h-dependency sets for x:

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

- The *h-superdependency graph* for x, $G^{(h)}_{sdep}(x, D)$, comprises $sdep^{(h)}(x, D)$ plus the edges in $G(D)$ that are restricted to vertices in $sdep^{(h)}(x, D)$.

- Similarly, a sequence $S \subset D$ is a local h-superdependency set for x if $sdep^{(h)}(x, S) = S$. The vacancy criterion $M^{(h)}$ however, is always taken with respect to $D - S$.

- Let $\mathcal{G}^{(h)}_{sup}(x, D)$ be the set containing, for each prefix of the probe sequence for x, the resulting partial h-superdependency graph.

- Let these definitions also apply to the tree-like bipartite, AND_OR tree, and bundled superdependency DAG representations, and to all partial and local superdependency formulations as well.

When the true vacancy criterion is used, x_j will obviously only root $probe_j$ different local h-superdependency graphs, which are global. While the size of the graphs grows when weaker criteria are used, we still have the following formulation for any h:

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

where $degr(G^{(h)}_{sdep}(x_i, D))$ is the degree of x_i in the graph.
The reason this formulation still holds is that the h-superdependency graphs are global constructs. When local sampling is used, the numbers of such structures will grow considerably. The first step in controlling the growth of these DAGs is to show that the expected count of such local structures, for fixed $\alpha < 1$ and sufficiently large constant h, is bounded. The counting argument will be based on bundled local h-superdependency DAGs. The recursive counting formulation will be simplified by appealing to a vacancy criterion for the graph structures that has been substantially weakened for mathematical tractability, but is still adequate to ensure that the expected number of restricted hashing structures is bounded.

As a weak consequence of the structural constraints posed by the vacancy criterion, a node z in a tree-like bundled h-superdependency graph $G^{(h)}_{sdep}(x, D)$ cannot be a dummy unless z has more than h descendents in the bundled representation for $G^{(h)}_{sdep}(x, D)$.

Definition 10.

- Let this weak consequence be the *polyexclusion principle*, and let bundled h-superdependency structures that satisfy the polyexclusion principle be ph-superdependency graphs. By definition, these structures require all vertices to have locally determined embeddings that satisfy the global vacancy criterion $M^{(h)}$ in UH, DH, and DH_{ψ}. The definition is applicable to partial, local, and global structures.

- We extend this notion to superdependency DAGs that are tree-like but not bundled. In this case, the dummies are clearly identified, and the requirement for a dummy is that a vertex w cannot be a dummy unless its corresponding vertex in a bundled representation satisfies the polyexclusion principle.

Local *ph*-superdependency DAGs are determined solely with respect to their own dependency set, and will serve as templates for the hashing of small data sets. Since these structures are defined without a larger context of keys, they are intrinsically partial. Furthermore, there is only an asymptotically negligible difference between the hashing probabilities for such a structure based on whether or not its root is (locally) unembedded. This fact actually deserves further quantification and a proof for cases where the structure is not tree-like. (However, we will resolve the issue implicitly, by using extravagant overcharges for the probe counts in all cases where the structure is not tree-like.)

To reiterate, these templates are used to to quantify events where a specific tuple D_T hashes into one of these structures, which requires that the locally determined embedding locations satisfy $M^{(h)}$ with respect to $D - D_T$. However, our counting of these structures will not meet such stringent standards. In particular, the estimates will not require dummy roots to have virtual embedding locations that satisfy $M^{(h)}$.

Theorem 1. Let α and r be fixed with $\alpha < 1$. Let $s(k, \alpha n)$ be the recursively defined overestimate, as quantified below, of the expected number of tree-like local partial *ph*-superdependency DAGs that are rooted by $x_{\alpha n}$, and contain k keys. Then for c, d and h sufficiently large and fixed, with $\psi = d \log n$:

1) The expected number of such structures 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 structures with $c \log n$ nodes or less is bounded even when a structure 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 DAGs that have a node count in the interval $[c\log n, 2c\log n]$ is bounded by $O(n^{-r-2})$: $\sum_{c\log n\le k\le 2c\log n} s(k,\alpha n) = O(n^{-2-r})$.

Proof: The counting will be over bundled structures. Let $s(k,a)$ be the weighted sum, over all subsequences $x_{T_1},\ldots,x_{T_k}$, with $T_k = a$, of the (over)estimated probability that the tuple hashes locally into a bundled tree-like ph-superdependency DAG, so that all non-dummy nodes other than x_a find locations satisfying $M^{(h)}$. Nodes that can be dummies yield alternative embeddings (ph-superdependency structures), depending upon whether they are interpreted as dummies or not. Dummy nodes will be given an embedding probability (vacancy estimate) of 1 because they represent keys that are not embedded in the the graph structure, and we must avoid multiplying the dependent estimates that a given single hash location satisfies the vacancy criterion $M^{(h)}$ at different (locally successful globally unsuccessful) insertion times. The polyexclusion principle says that a local bundled substructure with root x_j cannot have x_j counted as a dummy unless the substructure holds at least h keys. The counting convention does not require the current root x_b to be successfully embedded so that partial ph-superdependency structures will be counted individually. The consequence of this convention is that the vacancy overestimator q should be used to multiply the recursively computed count for the (previously unembedded) root of a subtree bundle as represented by $s(j,b)$ below. If the recurrence is to be applicable in DH_ψ, we must prove that there are acceptable estimates for $q^{(h)}(b)$ in these hashing models. This fact was the substance of Theorem 3 and its precursor lemmas in [16].

Let μ be the constant 1. (We will later have cause to solve the same system with $\mu = 3$.) Recursing on the subtree bundle rooted at the most recent probe location of x_a gives:

$$s(1,a) = 1,$$
$$s(k,a) \le \sum_{\substack{h<j\le k-1\\ 0<b\le a-1}} s(j,b)\frac{\mu+q^{(h)}(b)}{n_1}s(k-j,a) + \sum_{\substack{1\le j\le h\\ 0<b\le a-1}} s(j,b)\frac{q^{(h)}(b)}{n_1}s(k-j,a), \quad k>1. \qquad (3)$$

When $j\le h$, the quotient $\frac{\mu+q^{(h)}(b)}{n_1}$ is replaced by $\frac{q^{(h)}(b)}{n_1}$ because the root of a subtree bundle of size h or less cannot be interpreted to be a local dummy, according to the polyexclusion principle. The factor $\frac{1}{n_1}$ denotes an upper bound for the probability $\frac{1}{n-O(1)}$ of probing a given location. It is also worth noting that (3) has positive coefficients so that overestimates for $s(j,b)$ result in overestimates for $s(k,a)$. It is helpful to rewrite (3) as:

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

We now suppress the subscript in n_1 and use the simpler value n, which can only increase the resulting value for $s(k,a)$ by a factor of $(1+\frac{O(1)}{n})^k$. This modification will turn out to be negligible, and subsumed by larger factors in all cases. Let $\tau(k,a) = s(k,a)q^{(h)}(a)$, $\sigma(k,a) = s(k,a)(\mu+q^{(h)}(a))$. These adaptations give:

$$\begin{aligned}
&\tau(1,a) = q^{(h)}(a);\\
&\tau(k,a) = \sum_{1\le j\le k-1} \mathcal{T}(j,a)\tau(k-j,a), \quad 1<k\le h;\\
&\mathcal{T}(j,a) = \frac{1}{n}\sum_{0<b\le a-1} \tau(j,b), \quad j\le h;\\
&T(k,a) = \frac{1}{n}\sum_{0<b\le a-1} \frac{\tau(k,b)}{q^{(h)}(b)}, \quad k\le h;\\
&\sigma(1,a) = q^{(h)}(a)+\mu;\\
&\sigma(k,a) = \sum_{0<j\le k-1} \mathcal{S}(j,a)\sigma(k-j,a) - \mu \sum_{1\le j\le \min(h,k-1)} T(j,a)\sigma(k-j,a), \quad 1<k;\\
&\mathcal{S}(k,a) = \frac{1}{n}\sum_{0<b\le a-1} \sigma(k,a).
\end{aligned} \tag{4}$$

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:

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

$$\tilde{\mathcal{T}}_\beta(1,\beta) = \tilde q^{(h)}(\beta),$$

$$\tilde{\mathcal{T}}_\beta(k,\beta) = \sum_{1\le j\le k-1} \tilde{\mathcal{T}}(j,\beta)\tilde{\mathcal{T}}_\beta(k-j,\beta), \quad k>1, \tag{5}$$

$$\tilde T(k,\beta) = \int_{0\le\alpha\le\beta} \frac{\tilde{\mathcal{T}}_\alpha(k,\alpha)}{\tilde q^{(h)}(\alpha)}d\alpha, \tag{6}$$

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

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

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

$$\tilde{\mathcal{S}}_\beta(k,\beta) = \sum_{1\le j\le k-1} \tilde{\mathcal{S}}(j,\beta)\tilde{\mathcal{S}}_\beta(k-j,\beta) - \mu \sum_{1\le j\le h} \tilde{T}(j,\beta)\tilde{\mathcal{S}}_\beta(k-j,\beta), \quad k > 1. \tag{7}$$

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 $\sigma(k,\beta n) < \tilde{\mathcal{S}}_\beta(k,\beta)$.

Let

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

and put

$$g(x,\beta) = \sum_{0<k} \tilde{\mathcal{T}}(k,\beta)x^{k-1}. \tag{9}$$

Define the truncated function

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

and define

$$G(x,\beta) = \sum_{0<k} \tilde{T}(k,\beta)x^{k-1}. \tag{11}$$

Differentiating (8) with respect to β and expanding via (7) and (10) gives

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

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

Manipulating (12) gives,

$$w_\beta(x,\beta) = \frac{\mu + \tilde{q}(\beta)}{1 + \mu x G^{(h)}(x,\beta) - xw(x,\beta)}, \tag{13}$$

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

Differentiating (9) with respect to β and expanding via (5) gives

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

$$g_\beta(x,\beta) = xg(x,\beta)g_\beta(x,\beta) + \tilde{q}(\beta). \tag{14}$$

Integrating (14) as is, and applying the quadratic formula gives,

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

where

$$\widetilde{Q}(\beta) = \int_0^\beta \tilde{q}(\gamma)d\gamma, \tag{15}$$

and hence

$$g_\beta(x,\beta) = \frac{\tilde{q}(\beta)}{\sqrt{1-2x\widetilde{Q}(\beta)}}. \tag{16}$$

In view of (6) and (9), we see that $G(x,\beta)$ as defined in (11) can also be expressed as:

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

Finally, let

$$\zeta(x,\beta) = \int_0^\beta g_b(x,b)\frac{\mu+\tilde{q}(b)}{\tilde{q}(b)}db \tag{18}$$

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

$$= g(x,\beta) + \mu G(x,\beta). \tag{19}$$

Then (18) gives

$$\zeta_\beta(x,\beta) = g_\beta(x,\beta)\frac{\mu+\tilde{q}(\beta)}{\tilde{q}(\beta)}, \tag{20}$$

and expanding with (14) gives

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

whence applying (20) gives

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

so that (19) can be used to yield

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

whereupon collecting the ζ_β terms results with

$$\zeta_\beta(x,\beta) = \frac{\mu+\tilde{q}(\beta)}{1+\mu x G(x,\beta) - x\zeta(x,\beta)}, \text{ which has the same form as (13).} \tag{21}$$

From (20) and (16), we see that ζ_β can also be expressed as:

$$\zeta_\beta(x,\beta) = \frac{(\mu + \tilde{q}(\beta))}{\sqrt{1 - 2x\widetilde{Q}(\beta)}}. \tag{22}$$

Since the interpolated $\tilde{q}(\beta)$ converges uniformly to $1-\beta$ on $[0,\alpha]$ as h (and n) $\to \infty$, $\widetilde{Q}(\beta)$ as defined in (15) converges uniformly to $\beta - \beta^2/2$, and $\zeta_\beta(x,\beta)$ as expressed in (22) will converge uniformly to $\frac{(\mu+1-\beta)}{\sqrt{1-2x\beta+x\beta^2}}$ on, say, domain $\Gamma = [0, \frac{2+(1-\alpha)^2}{2}] \times [0,\alpha]$, since the denominator is bounded away from zero on Γ.

Evidently, $g_\beta(x,\beta)$ as expressed in (16) will converge uniformly on the same domain Γ, to $\frac{(1-\beta)}{\sqrt{1-2x\beta+x\beta^2}}$, and $G^{(h)}(x,\beta)$ will converge uniformly and coefficient wise on Γ (identity (17)). But $G^{(h)}$ is a truncation of the Taylor's series for G. Hence $|G(x,\beta) - G^{(h)}(x,\beta)|$ is uniformly bounded by some ϵ_h, where $\epsilon_h \to 0$ as h (and n) $\to \infty$. The point is that G is analytic in x, and its Taylor terms $a_i x^i$ (based on (17) and an expansion of (16)) will be uniformly exponentially decreasing once h and n exceed some fixed constants, for $(x,\beta) \in \Gamma$.

From (13) and (21), it follows that w_β must 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 $|x| \le 1 + \frac{(1-\alpha)^2}{2}$ 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 partial ph-superdependency DAGs that are rooted by $x_{\alpha n}$, are tree-like, and have $c\log n$ nodes or less is bounded by

$$O(w_\beta(1,\beta)\mid_{\beta=\alpha}) = O(1).$$

2) For large constant h, the expected number of weighted local partial ph-superdependency DAGs that are rooted by $x_{\alpha n}$, are tree-like, and have $c\log n$ nodes or less is bounded by $O((x\frac{d}{dx})^r w_\beta(x,\beta)\mid_{\beta=\alpha,x=1}) = O(1)$, even when each k-vertex DAG has a weight of k^r.

3) For large enough constant h, the expected number of local partial ph-superdependency DAGs that are rooted by $x_{\alpha n}$, are tree-like, and have between $c\log n$ and $2c\log n$ nodes 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 overestimator also has a statistical behavior equivalent, up to a factor of $(1+O(\frac{1}{n}))$, to the fully independent case (c.f. Theorems 3 and 4 and Lemma A2 of [16]). These results give $\psi = d\log n$ for a suitable constant d. ∎

It is tempting to believe that the calculations of Theorem 1 show that the presence of non-tree-like ph-superdependency DAGs is negligible, but such a conclusion would be premature. Lemma 1 shows that a k-tuple is, up a factor of $(1+O(\frac{k^2}{n}))$, equally likely to hash into any k-vertex tree-like ph-superdependency structure. However, the polyexclusion principle allows certain candidate structures to be excluded. But the presence of non-tree edges might, for $j<h$, be from a j-node substructure with root x_b into another portion of a k-node ph-superdependency DAG, which could thereby increase the number of vertices reachable from x_b to a count exceeding h. In this case, the polyexclusion principle would not be applicable.

This issue is resolved below by adapting recurrence equation (3) as follows. For a count $s(j,b)$, with $j\le h$, we could restore the vacancy factor $\frac{q^{(h)}}{n_1}$ to the bundled vacancy factor $\frac{1+q^{(h)}}{n_1}$ if the DAG rooted by x_b has a non-tree edge that is to some other location beyond the $j-1$ tree vertices reached by tree edges as represented by the recursive formulation for $s(j,b)$. As a convenience and to ensure that the logical consequences (i.e., dependencies) are properly modeled, we simply add a weighted count of $y(j,b)$ probes in the recurrence below. The recurrence for y uses the bundled factors exclusively. The probability that $y(j,b)$ should be added in the recurrence below for $w(k,a)$ is, according to Lemma 1, $O(\frac{(k-j)^2 j}{n})$.

Theorem 2. Let, as in Theorem 1, α, and r be fixed with $\alpha<1$. Let $g(k,\alpha n)$, for $k\le 2c\log n$, be the number of local partial ph-superdependency sets that are rooted by $x_{\alpha n}$, contain k items and define structures that are not tree-like. Let c be a sufficiently large constant. Then for sufficiently large constant h, in UH, DH, and $DH_{d\log n}$, for sufficiently large constant d,

$$\mathrm{E}[\textstyle\sum_{k\le c\log n} k^r g(k,\alpha n)] = O(\frac{1}{n}), \text{ and } \mathrm{E}[\textstyle\sum_{c\log n\le k\le 2c\log n} g(k,\alpha n)] = O(n^{-r-1}).$$

Proof: We follow the proof schema of Theorem 1. Let $v(k,a)$ be the sum, over all subsequences of $k-1$ keys in $x_1, \ldots, x_{a-1}$, of the (over)estimated probability that the tuple plus x_a hashes locally into a ph-superdependency DAG whose non-dummy nodes other than x_a find vacant locations, according to $M^{(h)}$. The counting of structures will be over bundled tree-like structures that represent omnidirectional spanning trees. It is convenient to overcount the number of tuples that hash into either tree-like structures or non-tree-like DAGs and subtract an overcount for the tree-like events. To justify the subtraction of the such overcounts, it suffices to use a recurrence formulation that introduces a tag variable z for DAGs that have extra edges, and take $(g(z) - g(0)) \mid_{z=1}$ as an overestimate for the number of omnidirectional spanning trees that result from non-tree-like DAGs.

The counting will again follow the recursive schema of Theorem 1, but will include additional factors to account for non-tree-like edges as in Lemma 1. Let $v(k,a)$ be an overestimate of the expected number of local ph-superdependency sets where the root x_a is not embedded, and which have k nodes. This prescription gives the following.

$$
\begin{aligned}
y(1,a) &= 1;\\
y(k,a) &= \sum_{\substack{1\le j<k\\ 0<b\le a-1}} y(k-j,a)\frac{q^{(h)}(b)+1}{n}y(j,b), \quad k\le h;\\
w(1,a) &= 1;\\
w(k,a) &= \sum_{\substack{1\le j\le h\\ 0<b\le a-1}} w(k-j,a)\frac{q^{(h)}(b)}{n}w(j,b) + zO(\frac{k^2h}{n}) \sum_{\substack{1\le j\le h\\ 0<b\le a-1}} w(k-j,a)\frac{1+q^{(h))}(b)}{n}y(j,b)\\
&\quad + \sum_{\substack{h<j\le k-1\\ 0<b\le a-1}} w(k-j,a)\frac{q^{(h)}(b)+1}{n}(1+zO(k^3/n))w(j,b);\\
v(k,a) &= (1+zO(\frac{k^3}{n}))w(k,a).
\end{aligned}
$$

To eliminate the need for y in the above system, let

$$\kappa = \max_{\substack{1\le j\le h\\ b\le\alpha}} \{(\frac{y(j,b)}{(1-\alpha)s(j,b)})\}.$$

The factor $\frac{1}{1-\alpha}$ is an overestimate for $\frac{1}{q^{(h)}(b)}$.

Gross overestimates for $w(k,a)$ and $v(k,a)$ are defined as follows:

$$
\begin{aligned}
w(1,a) &= 1;\\
w(k,a) &= \sum_{\substack{1\le j\le h\\ 0<b\le a-1}} w(k-j,a)\frac{(1+z\kappa\frac{k^3}{n})q^{(h)}(b)}{n}w(j,b)\\
&\quad+\sum_{\substack{h<j\le k-1\\ 0<b\le a-1}} w(k-j,a)\frac{1+q^{(h)}(b)}{n}(1+zO(k^3/n))w(j,b);\\
v(k,a) &= (1+zO(\frac{k^3}{n}))w(k,a).
\end{aligned}
$$

It is straightforward to see that

$$v(k,a)\,|^{z=1} \le (1+z\kappa O(\frac{k^3}{n}))^k s(k,a)\,|^{z=1},$$

and

$$v(k,a)\,|_{z=0}^{z=1} \le (1+z\kappa O(\frac{k^3}{n}))^k s(k,a)\,|_{z=0}^{z=1},$$

where we have used the standard notation $f(x)\,|_a^b\equiv f(b)-f(a)$, and the rather less standard $f(x)\,|^b\equiv f(b)$.

By construction, h is a constant, and hence κ is some constant. We can take n to be so large that $2c\log n<(n/\kappa)^{1/4}$. In this case, $(1+z\kappa O(\frac{k^3}{n}))^k=1+z\frac{O(k^4)}{n}$, for $z\in[0,1]$.

Evidently, the tagged recurrence equation shows that $\mathrm{E}[g(k,a)]=O(\frac{k^4}{n})s(k,a)$, provided $k\le 2c\log n$. Theorem 1 now shows that

$$\mathrm{E}[\sum_{k\le 2c\log n} k^r g(k,a)]=O(\frac{1}{n}),$$

and

$$\mathrm{E}[\sum_{c\log n<k\le 2c\log n} g(k,a)]=O(n^{-r-2}).$$

The size of ψ necessary to compute $\mathrm{E}[g(k,a)]$ and $\mathrm{E}[k^r g(k,a)]$ come from Theorem 3 in [16], which ensures that for a suitable constant d, we can set $\psi=d\log n$, ∎

Corollary 1. Let α, and r be fixed with $\alpha<1$. Let c be a sufficiently large constant. Then for h and d sufficiently large and fixed, with $\psi=d\log n$: the probability, in UH, DH, and DH_ψ, that $x_{\alpha n}$ roots a local ph-superdependency DAG that contains $2c\log n$ nodes or more is $O(n^{-r})$.

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

Lemma 2. Let α, r and c be fixed with $\alpha < 1$. Let $e_{T,G}$ be the predicate for the event that D_T hashes locally into the ph-superdependency structure G, and the hashing satisfies the vacancy criterion $M^{(h)}$. Let $\mathcal{F}(\alpha n)$ be the predicate for the event that the ph-superdependency DAG for $x_{\alpha n}$, which must satisfy $M^{(h)}$, is tree-like and has at least $f\log n$ keys. Then f, d, and h can be assigned constant values so that the following holds in DH and DH_ψ, with $\psi = d\log n$: Suppose that $T \subset (1, 2, \ldots, \alpha n)$, $|T| \le c\log n$, and $T_{|T|} = \alpha n$. Let G be an unembedded tree-like local ph-superdependency structure with $|T|$ nodes. Then

$$\text{Prob}\{e_{T,G} \bigwedge \mathcal{F}(\alpha n)\} = \text{Prob}\{e_{T,G}\}O(n^{-r}).$$

Proof: Let Γ be the ph-superdependency DAG for $x_{\alpha n}$, and suppose that Γ is tree-like. It is helpful to view these structures, for the moment, as a 2-level bipartite graph, with all edges directed from vertices to probe locations. Suppose that $e_{T,G}$ is true, so that $D_T \subset sdep^{(h)}(x_{\alpha n}, D)$. Let Y be the resulting collection of tree-like bipartite ph-superdependency structures that occur when $G - x_{\alpha n}$ is removed from Γ. In particular, all of the bipartite edges in G are deleted, along with each key in G, except for $x_{\alpha n}$. This construction yields $|T|$ tree-like structures (that would be connected components if the edges were undirected). Root $x_{\alpha n}$ remains, and is attached via any additional probes that are not part of G.

- Let S^i be the structure implicitly rooted at the locally determined insertion location for D_{T_i}, and let S^i have ξ_i vertices in $G_{sdep}(\alpha n, D)$. The implicit root is the highest indexed key in S^i to have a probe to D_{T_i}.

There are three types of structures. First, one of them has the root $x_{\alpha n}$. For it, $x_{\alpha n}$ effectively begins its probes afresh after those counted by its degree in G. The conditioning imposed by G is that the preceding probes of $x_{\alpha n}$ were to distinct locations Consequently, $(1 - O(\frac{|T|^2}{n}))^{-1}s(k, \alpha n)$

gives an overestimate of the probability that the resulting structure rooted by $x_{\alpha n}$ and defined by $x_{\alpha n}$'s subsequent probes comprises k additional vertices in Γ. The other two types are distinguished as follows. The structure S^i is subject to some conditioning because x_{T_i} was known to have found its embedding location *ph*-compliant: For vertex $x_{T_i} \in G$, its virtual local embedding location must satisfy $M^{(h)}(T_i)$. Furthermore, if x_{T_i} is in a subtree of h or fewer keys in G, then the vacancy criterion will prohibit further growth in the structure: the location will have no probes eligible for S_i from potential dummies x_s with $s > T_i$. The other type of structure occurs if x_{T_i} does not belong to such a subtree. In this case, the location will be available for some further accumulation of dummies and their substructures.

In any case, we claim that

$$\mathrm{Prob}\Big\{\xi_i = k \;\Big|\; e_{T,G}\Big\} < \frac{(1 - O(\frac{|T|^2}{n}))^{-1}}{1-\alpha} s(k, \alpha n), \tag{23}$$

and this inequality holds in DH_ψ, for the parameters as quantified in the lemma. Some justification is necessary.

First, the function $s(k,t)$ is an overestimate for the number of partial *ph*-superdependency sets that occur with k keys. Second, the insertion process is required to succeed, so that the dominating statistical process is guaranteed to exist. Third, the hashing of D_T causes some conditioning, but factor $\frac{1}{1-\alpha}$ accounts for the conditioning that the root location of S^i meets the criterion $M^{(h)}$ at time T_i.

A fourth and implicit claim is that the consequences of limited independence are insignificant. A brief explanation is that the calculations for $s(k,t)$ (that are based on developments in [16], which included not only results analogous to Lemma 1, but also an analysis of global constraints as represented by witness graphs) can be readily extended to include the predicate $e_{T,G}$ with no significant changes other than a constant factor increase in the independence parameter d. Simply put, these probe-restricted hashing events are all within the estimation schema presented in Lemmas 4, 5 and 6, and Theorems 2 and 3 of [16]. And we have proven, in [16], that these events are statistically equivalent in DH_ψ, DH and UH, up to a factor of $1 + O(\frac{1}{n})$.

In summary, we can view $\Gamma - (G - x_{\alpha n})$ as a collection of structures that are statistically dominated by a forest of partial ph-superdependency DAGs whose vertex distributions are bounded by the function $(1-O(\frac{|T|^2}{n}))^{-1}s(k,\alpha n)/(1-\alpha)$. We seek to bound the probability that the sum of the vertex counts, in these $|T|$ graphs, exceeds $f\log n-|T|$.

Let $\mathcal{X}=\sum_i \xi_i$. Then we can write:

$$\begin{aligned}\text{Prob}\Big\{e_{T,G}\bigwedge(\mathcal{X}>f\log n-|T|)\Big\} &< \sum_{i=1}^{|T|}\text{Prob}\Big\{e_{T,G}\bigwedge(\xi_i>f\log n-|T|)\Big\}\\ &+\sum_{j=1}^{|T|}\text{Prob}\Big\{e_{T,G}\bigwedge(f\log n-|T|<\sum_{i=1}^{j}\xi_i<2f\log n-2|T|)\Big\}.\end{aligned}$$

Let E_j be the event that x_j roots a partial ph-superdependency DAG that is tree-like, and has between $f\log n-|T|$ and $2f\log n-2|T|$ nodes, none of which belong to $D_T-x_{\alpha n}$. Then

$$\begin{aligned}\text{Prob}\Big\{e_{T,G}\bigwedge(\mathcal{X}>f\log n-|T|)\Big\} &\le \sum_{i=1}^{|T|}\text{Prob}\Big\{e_{T,G}\bigwedge(f\log n-|T|<\xi_i<2f\log n-2|T|)\Big\}\\ &+\sum_{j=1}^{|T|}\text{Prob}\Big\{e_{T,G}\bigwedge(f\log n-|T|<\sum_{i=1}^{j}\xi_i<2f\log n-2|T|)\Big\}\\ &+\sum_{j=1}^{\alpha n}\text{Prob}\{e_{T,G}\bigwedge E_j\}.\end{aligned}$$

To determine $E_{\alpha n}$, the probe locations for $x_{\alpha n}$ should begin after the probes represented by links from $x_{\alpha n}$ to its children in G.

The point of this exercise was to attain expressions that can be bounded in DH_ψ by the sum of expected numbers of partial ph-superdependency structures of fixed sizes. Then inequality (23) can be used to estimate the likelihood of very large superdependency structures arising from such small ones. The last formulation almost meets this objective. Proceeding further gives:

$$\begin{aligned}\text{Prob}\{e_{T,G}\bigwedge\mathcal{X}>f\log n-|T|\} &< \text{Prob}\{e_{T,G}\}\Big(\sum_{i=1}^{|T|}\text{Prob}\{f\log n-|T|<\xi_i<2f\log n-2|T|\mid e_{T,G}\}\\ &+\sum_{j=1}^{|T|}\text{Prob}\Big\{f\log n-|T|<\sum_{i=1}^{j}\xi_i<2f\log n-2|T|\mid e_{T,G}\Big\}\\ &+\sum_{j=1}^{\alpha n}\text{Prob}\{E_j\mid e_{T,G}\}\Big).\end{aligned}$$

At this point, we are almost ready to drop all references to DH_ψ, and simply use

$$n\text{Prob}\{e_{T,G}\}\text{Prob}_{DH}\{f\log n-|T|<\mathcal{X}\}$$

as a crude overestimate upper bound for

$$\sum_{i=1}^{|T|}\text{Prob}\Big\{e_{T,G}\bigwedge(f\log n-|T|<\xi_i<2f\log n-2|T|)\Big\}$$
$$+\sum_{j=1}^{|T|}\text{Prob}\Big\{e_{T,G}\bigwedge(f\log n-|T|<\sum_{i=1}^{j}\xi_i<2f\log n-2|T|)\Big\}+\sum_{j=1}^{\alpha n}\text{Prob}\{e_{T,G}\bigwedge E_j\}.$$

The remaining arguments that must be made are straightforward. First, it should be noted $s(k,t)$ is monotone in t, so we can certainly set $t=\alpha n$ in these upper bounds. In addition, the $|T|$ random variables that comprise $\mathcal{X}$ are, technically, not independent, since the constituent vertices of each structure cannot have any keys (or locations) in common. But the recursive formulations for these structures shows that we can simulate their construction simply by suppressing events where new "copies" of the structure have collisions that cause non-tree-like structures. Thus, treating them as independent simply further overestimates the probability that $\mathcal{X}$ is large.

In concrete terms, $\text{Prob}\{e_{T,G}\wedge\mathcal{F}(\alpha n)\}$ can be bounded by the statistics encapsulated in $s(k,\alpha n)$. Intuitively, we can think of $\frac{s(*,\alpha n)}{1-\alpha'}$ (where the use of α' is intended to include the factor of $(1-O(\frac{|T|^2}{n}))^{-1}$) as an approximate probability distribution, and seek to apply a Chernoff-type bound for the sum of $|T|$ random variables that are characterized by this pseudo-distribution function. Of course, $\frac{s(k,\alpha n)}{1-\alpha'}$ is not a probability distribution because $\frac{\sum_k s(k,\alpha n)}{1-\alpha'}$ will be some constant that is considerably larger than 1. But the analyticity results of Theorem 1 ensure that for all $k\geq 0$, $\frac{s(k,\alpha n)}{1-\alpha'}<c\gamma^k$ for some constants $\gamma<1$ and c. And while $\frac{s(k,\alpha n)}{1-\alpha'}$ might not model the probabilistic behavior for *ph*-superdependency structures with sizes, say, $k>2f\log n$, we do not care; this case is outside of the formulation as quantified for DH_ψ.

In any case, it follows that for some constants $\ell>1$ and $\gamma<1$,

$$\frac{s(k,\alpha n)}{1-\alpha'}<(1-\gamma)\gamma^{k-\ell},$$

for all relevant k. Let $g(x)=(1-\gamma)\sum_{i\geq 0}\gamma^i x^i$. By construction, $g(x)$ is the moment generating function for the geometric distribution with success parameter γ. Let X_i, for $i=1,2,\ldots,|T|$, be identically distributed independent random variables with the geometric distribution and moment generating function $g(x)$. Let $X=X_1+\ldots+X_{|T|}$. In view of the preceding discussion,

$$\text{Prob}\{f\log n-|T|<\mathcal{X}\}=\text{Prob}\{f\log n-(\ell+1)|T|<\mathcal{X}-\ell|T|\}<\text{Prob}\{f\log n-(\ell+1)|T|<X\},$$

and we can now use a standard Hoeffding-Chernoff deviation bound for the latter.

We momentarily suppress the $(\ell+1)|T|$ term (which could even be dropped, since f is an arbitrary constant, and $|T| \le c\log n$.) There follows, for $z > 1$:

$$\mathrm{Prob}\{X > f\log n\} = \mathrm{Prob}[z^X > z^{f\log n}] \le \mathrm{E}[z^{-f\log n}z^X] = z^{-f\log n}(g(z))^{|T|} = z^{-f\log n}(\frac{1-\gamma}{1-\gamma z})^{|T|}.$$

Let $z = \frac{f\log n}{\gamma(|T|+f\log n)}$, and set $f\log n = w|T|$. Then

$$\begin{aligned}\mathrm{Prob}\{X > f\log n\} &< \frac{\gamma^{f\log n}(|T|+f\log n)^{f\log n+|T|}}{(f\log n)^{f\log n}}\frac{(1-\gamma)^{|T|}}{|T|^{|T|}}\\ &< (\gamma^w(1+w)(\frac{1+w}{w})^w(1-\gamma))^{|T|}\\ &< (\gamma^w(1+w)e(1-\gamma))^{|T|}\end{aligned}$$

We need to find w large enough that $(\gamma^w(1+w)e(1-\gamma)) < n^{-r/|T|}$.

Observe that for $a > 1$, and $b > 0$:

$$\log(1+a+b) < 1+\log a+\frac{b}{2}. \tag{24}$$

We can suppose that $\gamma > 1/e$, so that $\log(1/\gamma) < 1$, which is assumed below.

Let

$$w = \frac{\frac{r}{|T|}\log n + 1 + \log(1+w)}{\log\frac{1}{\gamma}}. \tag{25}$$

Then

$$w = \frac{\frac{r}{|T|}\log n + 1 + \log(1+\frac{\frac{r}{|T|}\log n+1+\log(1+w)}{\log\frac{1}{\gamma}})}{\log\frac{1}{\gamma}} \tag{26}$$

$$w < \frac{\frac{r}{|T|}\log n + 1 + \log(1/\log\frac{1}{\gamma}) + \log(1+1+\frac{r}{|T|}\log n + \log(1+w))}{\log\frac{1}{\gamma}}$$

Applying inequalty (24) to $\log(1+(1+\frac{r}{|T|}\log n)+\log(1+w))$ in (26) gives:

$$w < \frac{\frac{r}{|T|}\log n + 1 + \log(1/\log\frac{1}{\gamma}) + 1 + \log(1+\frac{r}{|T|}\log n) + \frac{\log(1+w)}{2}}{\log\frac{1}{\gamma}} \tag{27}$$

Whence doubling inequality (27) and subtracting (25) gives:

$$w < \frac{\frac{r}{|T|}\log n + 3 + 2\log(1/\log\frac{1}{\gamma}) + 2\log(1+\frac{r}{|T|}\log n)}{\log\frac{1}{\gamma}}.$$

So it suffices to set $w = \frac{\frac{r+1}{|T|}\log n + 3 + 2\log(1/\log\frac{1}{\gamma}) + 2\log(1+\frac{r+1}{|T|}\log n)}{\log\frac{1}{\gamma}}$. The $(\ell+1)|T|$ term is now convenient to restore, whence

$$\mathrm{Prob}\{e_{T,G} \bigwedge \mathcal{F}(\alpha n)\} = \mathrm{Prob}\{e_{T,G}\}O(n^{-r}),$$

provided $f \geq \frac{w|T|+(\ell+1)|T|}{\log n}$. Substituting for w establishes that f can be a constant. Given f and $|T|$, the value for the constant d is a consequence of Theorem 4 in [16]. ∎

Lemma 3. Let, as in Theorem 1, α, r and c be fixed with $0 < \alpha < 1$. Let $e_{T,G}$ be the predicate for the event that D_T hashes locally into G, and the hashing satisfies the vacancy criterion $M^{(h)}$. Let $\aleph(\alpha n)$ be the predicate for the event that the h-superdependency DAG for $x_{\alpha n}$ is not tree-like or has more than $f\log n$ vertices.

Then h, f, and d can be assigned constant values so that the following holds in DH and DH_ψ, where $\psi = d\log n$: Suppose $T \subset (1,2,\ldots,\alpha n)$, $|T| \leq c\log n$, and $T_{|T|} = \alpha n$. Let G be an unembedded tree-like ph-superdependency structure with $|T|$ nodes. Then

$$\mathrm{Prob}\{e_{T,G} \bigwedge \aleph(\alpha n)\} = \mathrm{Prob}\{e_{T,G}\}O(\frac{|T|^2}{n}).$$

Proof: The reasoning is similar to that of Lemma 2. Let D^* be the elements of D where the elements $(D_T - x_{\alpha n})$ are removed, and $x_{\alpha n}$ is replaced by a symbolic key $x^*_{\alpha n}$ that has the following probe sequence: For $i = 1,2,\ldots,|T|$, $p(x^*_{\alpha n}, j) = p(x_{\alpha n}, j+\tau)$, where $x_{\alpha n}$ has outdegree τ in G.

It is easy to see that $\mathrm{Prob}\{e_{T,G} \wedge \aleph(\alpha n)\}$ is bounded by the probability that D_T hashes as prescribed, and at least one of the following holds:

1) A subset $S \subset (D^*)$ hashes into a local ph-superdependency DAG that has $f\log n$ keys or less, is not tree-like and is rooted at some locally determined embedding location of G or of $x^*_{\alpha n}$.

2) Some pair of subsequences $R, S \subset D^*$, hash into two local ph-superdependency DAGs that are rooted at different embedding locations of G or $x^*_{\alpha n}$, and which have either a key in common or (more generally) an embedding probe location in common.

3) In the hashing of D, key $x_{\alpha n}$ roots a partial ph-superdependency DAG of $f\log n$ keys or more.

Reasoning as before, we count the expected number of local ph-superdependency DAGs for each case. These counts give joint probabilities of, respectively,

1) $\text{Prob}\{e_{T,G}\}|T|\sum_{k=1}^{f\log n} O(\frac{k^3}{n})s(k,\alpha n) = O(\frac{|T|}{n})\text{Prob}\{e_{T,G}\}$,

2) $\text{Prob}\{e_{T,G}\}\binom{T}{2}\sum_{k=1}^{f\log n}\sum_{j=1}^{f\log n} O(\frac{kj}{n})s(k,\alpha n)s(j,\alpha n) = O(\frac{|T|^2}{n})\text{Prob}\{e_{T,G}\}$ and

3) (by the reasoning of Lemma 2) $O(n^{-r})\text{Prob}\{e_{T,G}\}$. ▮

It will now be convenient to switch our focus to bundled tree-like h-superdependency DAGs that are represented by AND_OR trees. These structures will be required to have embeddings that satisfy the vacancy criterion $M^{(h)}(T,I)$, and have structures that are consistent with this requirement (as opposed to the polyexclusion principle).

The purpose of our previous counting exercises was to bound the expected sizes and numbers of these structures, which has been done. The next step is to eliminate the overcounting that arises from spurious dependency graphs.

Definition 11.

- Let Tr be an AND_OR tree. Then S is an *implicant* of Tr if S is an AND_OR tree, its edges and vertices are contained in Tr, its root is the same as Tr's, and each AND node in S has the same outdegree as in Tr.

- Let $S \subseteq \mathit{Tr}$ denote the property that S is an implicant subtree of Tr.

Inclusion-exclusion for AND_OR trees turns out to be particularly clean, as will be shown in the following lemma.

Definition 12.

- Let Tr be an AND_OR tree. Let R be the set of OR nodes in Tr. Define Tr's *degree of freedom* $\mathit{fr}(\mathit{Tr})$ by:

$$\mathit{fr}(\mathit{Tr}) = \sum_{v\in R}(\mathit{outdegree}(v) - 1).$$

Lemma 4. Let Tr be an AND_OR tree. Then

$$\sum_{S\subseteq \mathit{Tr}} (-1)^{\mathit{fr}(S)} = 1.$$

Proof: The proof is by structural induction over Tr. The formula holds if Tr is a single node. If the root has only one child, which has r children, then the induction hypothesis can be applied to each of these r descendents individually, which gives a value of 1 for the evaluation of the subtree rooted at each grandchild. Since the implicants of Tr comprise, in this case, the root, its child and the Cartesian products of the set of implicants for each grandchild taken in groups of j grandchildren, for $1 \le j \le r$, we can formulate the net result as $\sum_{S\subseteq \mathit{Tr}} (-1)^{\mathit{fr}(S)} = \sum_{j=1}^{r} \binom{r}{j} (-1)^{j-1} \cdot 1^j = -((1-1)^r - 1)$, since all (non-empty) combinations of the OR node's subtrees can be used to construct $S \subseteq \mathit{Tr}$.

If the root has $r \ge 2$ children, then we simply note that $\sum_{S\subseteq \mathit{Tr}} (-1)^{\mathit{fr}(S)} = \prod_{j=1}^{r} \sum_{S\subseteq \mathit{Tr}_j} (-1)^{\mathit{fr}(S)}$, where Tr_j comprises the root of Tr and the root's j-th child as its only child. Here, the formula is assumed to hold for each lesser structure Tr_j,whence $\sum_{S\subseteq \mathit{Tr}} (-1)^{\mathit{fr}(S)} = \prod_{j=1}^{r} \sum_{S\subseteq \mathit{Tr}_j} (-1)^{\mathit{fr}(S)} = 1^r$. ∎

The errors that result from the inclusion-exclusion formula will be especially important to bound, but one minor issue has to be resolved before this can be done.

It is easy to believe that if Tr is an AND_OR tree that satisfies $M^{(h)}$, then each implicant subtree $S \subseteq \mathit{Tr}$ also satisfies the requirement, since true vacancy is defined with respect to the keys in D. In point of fact, this statement is not quite true for two reasons. Recall that in DH_ψ, the joint probability that a collection of locations I, at various insertion times T, satisfies the weak vacancy criterion $M^{(h)}(T, I)$ is analyzed in terms of witness graphs for dependency sets (cf. [Definition 16, 16]). The construction extends to superdependency DAGs with no modifications whatsoever. Given a superdependency DAG $G = (V, E)$, the witness graph is the random structure that contains a subset of keys $W_V \subset D - V$, and abstract embedding locations $\mathcal{L}$ that comprise the locally determined embedding locations for G and (more importantly) a particular subset of the first h probes of the keys in W_V. In semantic terms, the witness graph, as quantified in Definition 16, is a superset of vertices and probe locations comprising the keys and probe locations of (almost) all (potential) witness subsequences in $D - V$ that contain j keys, for $j \le h$, and that might hash into a local dependency tree with a root placement which shows that G does not satisfy $M^{(h)}$. Witness sets are built one key at a time, starting with the last in $D - V$. The specifics of the construction exclude keys that have two or more restricted probes to $\mathcal{L}$ as the witness set is built.

So let $G = (V, E)$ be a local h-superdependency DAG that satisfies $M^{(h)}$, and let G have the witness set W_V and associated probe locations $\mathcal{L}$. Let G_1 be an implicant of G with vertex set S, and witness set W_S.

It is possible that G_1 does not satisfy $M^{(h)}$ because of keys in W_S that are not eligible for membership in W_V. There is only reason such keys could exist: $W_S \not\subset W_V$. There are only two possible causes for this:

1) $V - S$ has keys in W_S;

2) There is at least one key in $W_S - W_V$ that has two of its first h probes in $\mathcal{L}$.

Fortunately, likelihood of such events is small.

Lemma 5. Let f be a fixed constant, and suppose that $G = (D_T, E)$ is a tree-like ph-superdependency DAG, with $|T| \leq f \log n$. Then in DH_ψ, for $\psi = d \log n$, where d is a sufficiently large constant, the probability that G has an implicant that does not satisfy $M^{(h)}$ is

$$O(\frac{|T|^2}{n}).$$

Proof: By definition, G satisfies $M^{(h)}$. Let the witness set for G comprise the keys W, and let $\mathcal{L}$ be its set of restricted probe locations. Theorem 4 of [16] shows that $\mathrm{E}[|W|] = O(|T|)$, $\mathrm{E}[|\mathcal{L}|] = O(|T|)$, and $\mathrm{E}[|\mathcal{L}|^2] = O(|T|^2)$, and these relationships even hold when the random variables $|W|$, $|L|$ and $|L|^2$ are computed with respect to the conditional probabilities that result from G satisfying $M^{(h)}$. Consequently, the probability is $O(\frac{|T|^2}{n})$ that any key in $D - W$ has two of its first h probes in $\mathcal{L}$. Similarly, the probability that some key $x_t \in W$ has, among its first h probes, a collision with one of the first h probes of D_{T_i}, for some $T_i < t$, is $O(\frac{|T|^2}{n})$. (These collisions are "too late" to establish non-vacancy for D_{T_i}.) Finally, it should be noted that these collision statistics can all be bounded by applying the properties of pairwise independence, since each structure is tree-like, which is to say that each key in D_T and each key in W is bound by only one constraint in the probabilistic construction of the superdependency DAG and its witness set. ∎

Definition 13.

- Call a tree-like ph-superdependency DAG $G = (V, E)$ *perfect* if every implicant satisfies $M^{(h)}$ with respect to $D - S$, where S is the implicant's vertex set.

The following corollary is the last preliminary step in the proof. It is the workhorse that eliminates what would otherwise be an exponential explosion of error terms due to DAGs that exhibit various structural or statistical irregularities.

Corollary 2. Let $w(k, t)$ be the sum of the expected number of AND_OR implicants in local ph-superdependency AND_OR trees with k vertices and root x_t. That is, $w(k, t)$ is the expected count of weighted tree-like k-element local ph-superdependency DAGs, where the weighting factor its number of implicants in the DAG. Let α and r be constant, with $\alpha < 1$. Then there are constants c, ρ and h such that given any constant f, the following holds:

In UH, DH, and DH_ψ where $\psi = \rho f \log n$, $w(k, \alpha n) = O(n^{-r})$, provided $c \log n < k < f \log n$.

In UH, DH, and DH_ψ where $\psi = \rho f \log n$, $\sum_{k=1}^{f \log n} k^2 w(k, \alpha n) = O(1)$.

Proof: We again appeal to the recursive formulation of Theorem 1, and the earlier representation of bundled superdependency DAGs with dummies. The only difference is that a dummy x_b in the partial ph-superdependency DAG gives rise to many implicants. In the counting for Theorem 1, a dummy had two interpretations, which corresponded to two different DAG structures. In this case, there are the two interpretations as before, but in addition, a dummy can be completely eliminated along with its substructure. Furthermore, a non-dummy implicant can be eliminated provided it has a dummy sibling to take its place. Thus, a gross overcount for the number of implicants can be obtained by setting μ to 3 in the analysis of $s(k, t)$ for Theorem 1.

So by recurring on the subDAG rooted at the most recent probe location of x_a, we get:

$$\begin{aligned} &w(1, a) = 1, \\ &w(k, a) \le \sum_{\substack{h<j\le k-1 \\ 0<b\le a-1}} w(j, b) \frac{\mu + q^{(h)}(b)}{n_1} w(k-j, a) + \sum_{\substack{1\le j\le h \\ 0<b\le a-1}} w(j, b) \frac{q^{(h)}(b)}{n_1} w(k-j, a), \quad k > 1, \end{aligned} \tag{28}$$

where $\mu = 3$.

The analysis of this system is the same as in Theorem 1. The constants h and ψ will be larger than before because the increase in μ must slow down the convergence. The convergence, however, is still to the system with $\mu = 0$. ∎

At this point, a reality check is most appropriate. (The reader might wish to pause and puzzle out the paradox that seems to be lurking behind the implications of Theorem 1 and the following discussion.) The recurrence for $w(k, a)$ counts a huge number of trees with k vertices. The vertex names are distinct integers in $[1..\alpha n]$, and the descending edge names are $1, 2, 3, \ldots$. There are a few other characteristics that we omit to keep the commentary brief. In the counting, each vertex (other than the root) is given a weighting that comprises two factors. The first factor is $\frac{1}{n}$. The other factor for vertex x_b is the modified vacancy weighting $\mu + q^{(h)}(b)$ or $q^{(h)}(b)$ as determined by recurrence equation (28). The root gets an unmodified vacancy weighting of $q = 1$. A tree's weight is just the product of the weights assigned to its vertices. The coefficient $w(k, b)$ is the sum of all weighted trees with k vertices and root x_b. Sadly, if all of the modified vacancy weightings were set to 1, then $\sum_k w(k, \alpha n)$ would diverge as $n \to \infty$ unless α is smaller than some small fraction. On the other hand, if the modified vacancy weighting for x_b were set to $q^{(h)}(b)$ for all b, then the sum would remain bounded provided α is bounded below 1, and $q^{(h)}(b)$ is close enough to $1 - \frac{b}{n}$. The "closeness error" can be some small fixed constant, as long as each q factor is bounded by 1. These assertions are all true.

However, we now propose to set all of the weighting multipliers to be, say, 4, except for those in subtrees of h or fewer vertices, where h is a large enough constant. So a "fringe" vertex x_i (i.e., that is in a subtree of h or fewer vertices) will get the vacancy weighting $q^{(h)}(i)$, which is good, but the non-fringe the vertices will get, say, a vacancy weighting of 4, which is very bad. According to Theorem 1, $\sum_k w(k, \alpha n)$ will again converge provided h is a large enough constant.

To understand why this should be true, first observe that the product of h fringe weightings $q^{(h)}(b)$ can become arbitrarily small, on average: It is just a matter of setting h to be large enough. Second, these trees are, on average very bushy. So there are more leaves than internal vertices, and likewise more of the small fringe subtree clusters than internal nodes. This argument should not be fully convincing because there are a huge number of these trees; this reduction of one tree to a much smaller tree where each fringe subtree is counted as a leaf is a hugely many-to-one transformation. But just consider: we can require each fringe multiplier $q^{(h)}(b)$ to be as close to $1-\frac{b}{n}$ as we like, up to a fixed error, and then increase (all but a small fraction of) these values by a factor of $1+\delta$, for some small δ. Then the overwhelming majority of fringe subtrees will have more than $h/2$ of their vertices offering a free factor of $(\frac{1}{1+\delta})$ to, collectively, wipe out the tiny fraction of factors that were set to 4. The result is no contest; although there will be some trees with very sparse fringes, and some with too many fringe elements among, say, the first $2\delta n$ keys in D, their relative density might well be so exponentially small that the overall sum of all such weighted trees, one might anticipate, should converge if h is large enough. Theorem 1 proves that this is so.

The infrastructure necessary to account for the expected insertion cost in DH_ψ is now complete.

Theorem 3. For fixed load $\alpha < 1$ and suitably large constant d, 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 give an elaborate expression that, in UH, must equal $\frac{1}{1-\alpha}+O(\frac{1}{n})$, and observe that the same formulation in DH and DH_ψ must differ by just $O(\frac{1}{n})$ from its counterpart in UH.

Let c, f and h be sufficiently large constants.

At this point, the proof is just a matter of defining various probe-counting events plus associated error terms, and applying the estimates that have already been derived. In the following definitions, entities that begin with the letter P will denote random variables, and those of the form Err_j will be expectations of various probe-related expressions that account for the model-dependent errors as well as the numerous counting mistakes attributable to our simplified analyses.

Definition 14.

- Let E_{tree} denote the event: $\langle$ the (globally defined) ph-superdependency DAG $G^{(h)}_{sdep}(x_{\alpha n}, D)$ is tree-like, perfect as an AND_OR tree, and has at most $f \log n$ vertices$\rangle$.

- Let F_{tree} denote the event: $\langle$ the (globally defined) ph-superdependency DAG $G^{(h)}_{sdep}(x_{\alpha n}, D)$ is tree-like, and has at most $f \log n$ vertices$\rangle$.
- Let $e_{T,\mathcal{T}}$ be the predicate for the event that D_T hashes locally into the ph-superdependency structure $\mathcal{T}$, and all locally determined embedding locations satisfy $M^{(h)}$.
- Let $e^{\neg Tr}_{T,\mathcal{T}}$ be the predicate for the event that D_T hashes locally into a ph-superdependency structure that is not tree-like, but has $\mathcal{T}$ as its canonical omnidirectional spanning tree, and all locally determined embedding locations satisfy $M^{(h)}$.
- Let Tr_k be the set of all tree-like local h-superdependency DAGs of k vertices where all dummies are unambiguously identified as in the AND_OR representation. The locality characteristic means that the structure satisfies the vacancy criterion $M^{(h)}$ locally, with respect to the k vertices, and is partial, when used as a structural template for hashing a k-tuple in D.
- Let $Tr_* = \cup_{k \le f \log n} Tr_k$.
- Given an event E, let $\mathcal{X}(E)$ be the indicator function for E, which is one if E occurs, and zero otherwise.
- Let

$$P_1 = \sum_{\substack{T \subset (1,2,\ldots,\alpha n) \\ |T| < c \log n \\ \mathcal{T} \in Tr_*,\ T_{|T|} = \alpha n}} (-1)^{fr(\mathcal{T})} \mathcal{X}(e_{T,\mathcal{T}}).$$

We will prove that in all hashing models,

$$\mathrm{E}_{\bullet}[probe_{\alpha n}] = \mathrm{E}_{UH}[P_1] + O(\frac{1}{n}).$$

- Let

$$P_{1E} = \sum_{\substack{T \subset (1,2,\ldots,\alpha n) \\ |T| < c \log n \\ \mathcal{T} \in Tr_*,\ T_{|T|} = \alpha n}} (-1)^{fr(\mathcal{T})} \mathcal{X}(e_{T,\mathcal{T}}) \mathcal{X}(E_{tree}).$$

Equation (2) and Lemma 4 combine to ensure that P_{1E} achieves the correct probe count when the formulation is restricted to the portion of the event space where the $G^{(h)}_{sdep}(\alpha n, D)$ is tree-like and

$|sdep^{(h)}(x_{\alpha n}, D)| \leq c \log n$. (To see that this is so, recall that the root of a tree-like superdependency DAG is an AND node, whence Lemma 4 will apply to each of the $probe_{\alpha n}$ partial h-superdependency DAGs that occur, along with their corresponding implicants.)

- Let

$$P_{1F} = \sum_{\substack{T \subset (1,2,\ldots,\alpha n) \\ |T| < c \log n \\ \mathcal{T} \in \mathcal{T}\!r_*,\ T_{|T|} = \alpha n}} (-1)^{\mathcal{f}\!r(\mathcal{T})} \mathcal{X}(e_{T,\mathcal{T}}) \mathcal{X}(F_{tree}).$$

- Let

$$P_{EF} = \sum_{\substack{T \subset (1,2,\ldots,\alpha n) \\ |T| < c \log n \\ \mathcal{T} \in \mathcal{T}\!r_*,\ T_{|T|} = \alpha n}} \mathcal{X}(e_{T,\mathcal{T}})(\mathcal{X}(F_{tree}) - \mathcal{X}(E_{tree})).$$

- For $\mathcal{T} \in \mathcal{T}\!r*$, let $Imp(\mathcal{T})$ be the number of implicants in $\mathcal{T}$.

- Let

$$P_{c-f} = \sum_{\substack{T \subset (1,2,\ldots,\alpha n) \\ c \log n < |T| \leq f \log n \\ \mathcal{T} \in \mathcal{T}\!r_*,\ T_{|T|} = \alpha n}} Imp(\mathcal{T}) \mathcal{X}(e_{T,\mathcal{T}}).$$

- Let

$$P_{\neg F}^{loc\mathcal{T}\!r} = \sum_{\substack{T \subset (1,2,\ldots,\alpha n) \\ |T| < c \log n \\ \mathcal{T} \in \mathcal{T}\!r_*,\ T_{|T|} = \alpha n}} \mathcal{X}(e_{T,\mathcal{T}})(1 - \mathcal{X}(F_{tree})).$$

In view of Equation (2), Lemma 4, and the various definitions, it follows that

$$P_1 - P_{\neg F}^{loc\mathcal{T}\!r} - P_{EF} - P_{c-f} \leq \mathcal{X}(F_{tree}) probe_{\alpha n} \leq P_1 + P_{\neg F}^{loc\mathcal{T}\!r} + P_{EF} + P_{c-f}. \tag{29}$$

We still have to quantify the probe performance when F_{tree} is false, and the differences in probe behavior across the different models. Meanwhile, it is worth noting that all of the irregular probe counts defined so far are small on average.

By Lemma 5 and Corollary 2, the expected number of implicants omitted from the formula for P_{1F} is bounded as follows.

$$\mathrm{E}[P_{EF}] = \sum_{\substack{T \subset (1,2,\ldots,\alpha n) \\ |T| \leq c \log n \\ \mathcal{T} \in \mathcal{T}\!r_*,\ T_{|T|} = \alpha n}} O(\frac{|T|^2}{n}) Imp(\mathcal{T}) \mathcal{X}(e_{T,\mathcal{T}}) = \frac{1}{n} \sum_{k=1}^{c \log n} O(k^2) w(k, \alpha n) = O(\frac{1}{n}).$$

By Lemma 3, and Theorem 1 and Corollary 1,

$$\mathrm{E}[P^{loc Tr}_{\neg F}] = \sum_{\substack{T \subset (1,2,\ldots,\alpha n) \\ |T| < c \log n \\ \mathcal{T} \in Tr_*,\ T_{|T|} = \alpha n}} O(\frac{|T|^2}{n}) \mathrm{Prob}\{e_{T,\mathcal{T}}\} = \sum_{k \le c \log n} O(\frac{k^2}{n}) s(k, \alpha n) = O(\frac{1}{n}).$$

By Corollary 2,

$$\mathrm{E}[P_{c-f}] = O(n^{-r}).$$

The first irregular probe count to reconcile arises from local h-superdependency DAGs that are not tree-like, but where the root has no probing of any location more than once.

- Let

$$P^{\neg Tr}_1 = \sum_{\substack{T \subset (1,2,\ldots,\alpha n) \\ |T| < f \log n \\ \mathcal{T} \in Tr_*,\ T_{|T|} = \alpha n}} \mathcal{X}(e^{\neg Tr}_{T,\mathcal{T}}).$$

Because of limited independence, and the use of probe axioms that permit a fair amount of deviant probe behavior, there are additional error terms. For large h-superdependency sets, the axioms of limited independence are not adequate to characterize the resulting interactions.

- Let

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

  Corollary 1 ensures that for suitably large constants f, h and d, $Err_{c_0} = O(n^{-r})$ in DH_ψ, for $\psi = d \log n$.

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

$$Err_{++} = \sum_{t \ge c_0 n} \mathrm{Prob}\{probe_{\alpha n} > t\}.$$

Note that Err_{c_0}, which charges a cost of $c_0 n$ probes to the event, when combined with Err_{++} and P_{c-f}, accounts for dependency sets and probe-counting errors that might occur for dependency DAGs with more than $f \log n$ keys. We now resolve the remaining expected number of probes attributable non-tree-like DAGs and related irregularities. The errors $\mathrm{E}[P^{\neg Tr}_F]$, Err_{c_0} and Err_{++} capture some of these events. The next two terms account for the extra probes in non-tree edges of dependency sets with $f \log n$ or fewer elements.

- Let, for $k < f \log n$, $B(k)$ be the indicator function for the event $(|sdep^{(h)}(x_{\alpha n}, D)| = k)$, and the root of $G^{(h)}_{sdep}(x_{\alpha n}, D)$ has non-tree edges. Set

$$Err_1 = \sum_{k \leq f \log n} 2k\mathrm{E}[B(k)],$$

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

- Let, for $k < f \log n$, $ER(k)$ be the indicator function for the event: $(|sdep^{(h)}(x_{\alpha n}, D)| = k)$, and $degr(G^{(h)}_{sdep}(x_{\alpha n}, D)) \geq 2k - 1$. Let

$$Err_2 = c_0 n \times \sum_{k \leq f \log n} \mathrm{E}[ER(k)],$$

so that we charge an insertion cost 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.

- Let X_b be the number of keys in D that, in $DH^{(b)}_\psi$, have their first $\kappa \log n$ probes hit just $\frac{\kappa \log n}{2}$ or fewer locations as formalized in Definition 2.6), and set

$$Err_b = c_0 n\mathrm{E}[X_b] + Err_{++}.$$

Thus, the penalty for $x_{\alpha n}$'s probe count is $c_0 n$ if any vertex has this excessive local probing, plus the expected number of probes for $x_{\alpha n}$ that exceed the first $c_0 n$ regardless of the cause.

Among all of the preceding probe counts and error estimates, only P_1 has error contributions that remain to be addressed for the different models. That is, $P_1^{\neg Tr}$, P_{c-f} and the previous Err formulations already include the modeling errors for all DAGs that are not tree-like.

- Let

$$Err_\psi = \sum_{\substack{T \subset (1,2,\ldots,\alpha n) \\ |T| < c \log n \\ \mathcal{T} \in Tr_*,\ T_{|T|} = \alpha n}} |\ \mathrm{Prob}_{DH_\psi}\{e_{T,\mathcal{T}}\} - \mathrm{Prob}_{UH}\{e_{T,\mathcal{T}}\}\ | \ .$$

- Let

$$Err_{tot} = Err_{c_0} + Err_{++} + Err_1 + Err_2 + Err_b + Err_\psi.$$

Inequality (29) and the preceding arguments show that

$$\mathrm{E}_\bullet[P_1]-Err_{tot}-\mathrm{E}[P_{\neg F}^{locTr}+P_{EF}+P_{c-f}] \le \mathrm{E}[probe_{\alpha n}] \le \mathrm{E}_\bullet[P_1]+Err_{tot}+\mathrm{E}[P_{\neg F}^{locTr}+P_{EF}+P_{c-f}]+\mathrm{E}[P_1^{\neg Tr}].$$

We have already shown that $\mathrm{E}[P_{\neg F}^{locTr}+P_{EF}+P_{c-f}] = O(\frac{1}{n})$. Is now straightforward to check that $\mathrm{E}[P_1^{\neg Tr}] = O(\frac{1}{n})$, and likewise that $Err_{c_0}+Err_{++}+Err_1+Err_2+Err_b+Err_\psi = O(\frac{1}{n})$. ∎

3. Extensions

We now apply the techniques from Section 2 to see what else can be deduced about double hashing, and to illustrate how the construction of statistically comparable processes can extend other bounds from models with full independence to circumstances with limited randomness.

3.1 Higher moments

As formulated, the hashing bounds for DH_ψ do not extend to higher moments. This restriction is due to the very weak prohibition of irregular cases (parts 3) and 6) of Definition 2. There are many reformulations of these conditions that would enable analogous results to be obtained for higher moments of the probe statistics. In the (belated) interest of brevity, we omit them, and state a much narrower formulation for models such as genuine double hashing with $\frac{\psi}{2}$-wise independent functions f and d.

Corollary 3. Let α and r be fixed, with $\alpha < 1$. Then for sufficiently large fixed d,

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

provided $\psi \ge d\log n$, and DH_ψ models a hashing scheme such as true double hashing. ∎

Of course, the bound also holds for more general formulations.

3.2 Tree statistics

The statistics for dependency structures have meaningful limits as $n \to \infty$. Thus, there is limit a probability distribution for the structures $G(x_{\alpha n}, D^n)$, where D^n are a sequence of data sets of, say, size αn. Moreover, this distribution is asymptotically the same in UH, DH and DH_ψ, for sufficiently large $\psi = O(\log n)$. A reasonably representative distribution theorem is the following.

Theorem 4. Let Tr be a fixed tree-like dependency DAG with vertices $[1,..k]$ and ordered adjacency lists. 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 Tr is $\frac{1}{(k-1)!}(1-\alpha)(\alpha - \frac{\alpha^2}{2})^{k-1}(1+\frac{O(k^2)}{n})$.

Proof: (Sketch) The error factor $(1+\frac{O(k^2)}{n})$ comes from Lemma 1 above, or from Lemma 2 of [16]. The principal term can be derived from the proof of Theorem 1 in [16], or direct calculation (in UH) that the probability is

$$(1-\alpha)\int_{0<\alpha_{k-1}<\alpha}\int_{0<\alpha_{k-2}<\alpha_{k-1}}\cdots\int_{0<\alpha_1<\alpha_2}\prod_{i=1}^{k-1}(1-\alpha_i)\,d\alpha_1 d\alpha_2\cdots d\alpha_{k-1}.$$

The validity of this calculation for DH_ψ follows from the proof of Theorem 3. ∎

In retrospect, it is fair to suggest that this commonality of form unites these different hashing schemes, and underlies the actual performance analysis that we used.

3.3 Introducing $DH_\psi^{(d)}$ Retrospective analysis also shows that the constraints of $DH_\psi^{(c)}$ can be weakened so that requirement 6) of Definition 2 can be the same as that for $DH_\psi^{(b)}$. The reason, in brief, is as follows. Let c be the implicitly defined constant c used in Theorem 3. The exception covered by requirement 6) occurs if a candidate partial ph-superdependency set contains at most $c\log n$ vertices but uses more than $6c\log n$ probes, and satisfies the vacancy criterion $M^{(h)}$. If this event occurs, let G be the smallest subDAG still exhibits this exception. It follows that G must represent at most $12c\log n + 1$ probes, as otherwise there is a smaller structure that exhibits this characteristic. Then by doubling the independence ψ as used in Theorem 3, we see that G will have a probability of occurring that can be bounded in $DH_\psi^{(d)}$. Since the number of nodes is bounded by $c\log n$, either there are three nodes in G with non-tree DAG edges, or one node has at least $2c\log n$ probes. Upon setting $\kappa = 2c$, it will then follow that the probability that such a DAG exists is bounded by $nO(\frac{1}{n^3}) = O(\frac{1}{n^2})$.

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

As indicated earlier, the probability distribution for the vacancy overestimator $M^{(h)}$ 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 basic facts are that $q^{(2)}$ can be explicitly computed to get $q^{(2)}(\alpha n) = e^{1-2\alpha-e^{-\alpha}}(1+O(\frac{1}{n}))$, and $Q^{(2)}(\alpha n) = (1+O(\frac{1}{n}))(1+e^{-\alpha})e^{1-e^{-\alpha}} - 2$. The value .669 is obtained from a numerical solution to the rescaled differential equation associated with equation (3) and $q^{(2)}$ as stated. The details are omitted.

3.5 Tertiary clustering

In tertiary clustering, the probe 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 $[0, n^2-1] \times \text{probe index} \mapsto [0, n-1]$. 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.

3.6 Uniform hashing

Uniform hashing is generally viewed as an idealized model because of its computational requirements. Each probe requires the computation of an independent random probe index. 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, standard constructions (cf. Section 1.1 of [16]) give 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 family F_ψ with domain $[0, \psi p - 1]$, 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. A more robust formulation of $p(x,j)$ would 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, which must be qualified by robustness requirements analogous to those stated in the definitions of DH_ψ.

Corollary 4. Let r and α be fixed with $\alpha < 1$. Then for sufficiently large constant c:

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

provided $\psi \geq c \log n$, and the hashing includes suitable robustness guarantees.

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 might 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 item onto $[1, 2I]$, when the first probe index is $I+1$, and k keys have already been inserted among the locations $[1, 2I]$ according to linear probing. These k keys are assumed to take on initial probe locations in $[1, 2I]$ independently and according to the uniform distribution. From the definition of $X(I)$, it follows that with probability (exceeding) $1 - X(I)$, keys inserted outside of $[1, 2I]$ cannot affect the hashing of x. We require ψ to be so large that $W(k)$ is the same in LP and LP_ψ, for relevant k. Then by inclusion-exclusion,

$$\begin{aligned}\mathrm{E}[probe_{\alpha n}] = &\sum_{\substack{k<2I\\ 0\le j\le 4\lceil eI\rceil-k}} W(k)\binom{\alpha n-1}{k+j}\left(\frac{2I}{n}\right)^{k+j}\binom{k+j}{k}(-1)^j\\ &+O\left(nX(I)+\sum_{k<2I}W(k)\binom{\alpha n-1}{4\lceil eI\rceil}\left(\frac{2I}{n}\right)^{4\lceil eI\rceil}\binom{4\lceil eI\rceil}{k}\right),\end{aligned}\tag{31}$$

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, $X(I) < n\mathrm{Prob}\{$at least I items in D 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 (31) are bounded by the inclusion-exclusion terms where

$j = 4\lceil eI \rceil - k$, which are bounded by

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

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 possible, in principle, to program hash functions that exhibit, with high probability, $c \log n$-wise independence on the subset of keys D, execute just 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 data sets. Nevertheless, there is still a gap between the use of $\log n$-wise independent hash functions, and real implementation. One might well wonder if real data, when hashed by standard hash functions, might, in fact, exhibit the statistics of $\log n$-wise independence. Similarly, it would seem reasonable to conjecture that there are weaker formulations of high independence that might also be adequate to characterize the randomness necessary for optimal hashing.

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.

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, *JACM*, **37**(2), April, 1990, pp. 224–237.

[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, *Combinatorica*, 13, 1993, pp. 83–96.

[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, *SIAM J. on Discrete Mathematics*, 8(2), 1995, pp. 223–250.

[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.