Chapter 1
On the statistical dependencies of coalesced hashing and their implications for both full and limited independence*

Extended Abstract

Alan Siegel†

1 Summary.

This paper gives the first optimal bounds for coalesced hashing schemes in the case of limited randomness, and thereby establishes the analytic performance of these schemes in a model that supports formal randomized computation.

As a byproduct of this work, we attain a much simpler analysis of coalesced hashing schemes, which provides more information about the statistics of the underlying processes.

We present the generating functions that govern the chain distribution and probe performance for coalesced hashing schemes, including asymptotic formulations when cellars are used.

2 Background.

In coalesced hashing, a sequence of distinct keys $D = (x_1, x_2, \ldots, x_{\alpha(C+n)})$ that belong to a universe $[0..m]$ is stored in a two-part table $T[-C..n-1]$. A hash function h, that maps $[0..m]$ into the probe region $[0..n-1]$ is used to insert a key x as follows. If table slot $T[h(x)]$ is vacant, x then is stored in $T[h(x)]$. Otherwise x is called a collider, and is stored in the left-most vacant location (i.e., with lowest valued index) in T. To facilitate search, chaining is used to link the collider x somewhere in the list that will emanate from the record stored in location $h(x)$.

There are several variations. If we forbid colliders from entering the table region $T[0..n-1]$, then we have open hashing (a.k.a. closed addressing) with separate chaining. This scheme is not as efficient nor as robust; it fails when the cellar region $T[-C..-1]$ overflows, even though a significant fraction of $T[0..n-1]$ will still have vacant slots (with high probability). Coalesced hashing is defined to allow colliders to enter the probe range of h in order to be more efficient and to tolerate loads of up to $\alpha = 100\%$. If C is set to 0, we have standard coalesced hashing, with no cellar. If $C > 0$, we have the coalesced hashing schemes studied by Vitter [11], and Vitter and Chen [12]. These works introduce three variations of coalesced hashing, which are characterized by their chaining policies for colliders.

In LICH, a colliding key x is inserted at the tail of the list (Late-Insertion) that emanates from $T[h(x)]$. In EICH, the insertion occurs immediately following $T[h(x)]$ (Early-Insertion). In VICH, it occurs immediately following the last cellar item in the list, or as in EICH, if no cellar items are present (Varied-Insertion). In all variations of coalesced hashing, a sequence of $\alpha(C+n)$ distinct search keys is inserted item-by-item in a greedy manner. Data is not relocated. The parameter α, which is the ratio of data to storage locations, is called the load factor.

The basic hashing scheme, where no cellar is used, dates at least as far back as Williams [14], and is analyzed in Knuth [5]. An alternative analysis due to Paterson is presented in Greene and Knuth [3]. The general cases with optimized cellars are analyzed in [11] and [12]. The analyses include a study of how to partition a fixed memory into $[-C..-1]$ and $[0..n-1]$ to achieve optimal performance. [12] also includes a discussion of the advantages and disadvantages of coalesced hashing. Among the more interesting facts is that its expected performance is excellent: even at a load of 100%, the expected unsuccessful search time is only 1.79 probes [12]. Successful search is a little better. All of this is hard to beat, on average, although we must acknowledge that a wealth of applications are now in a context where memory hierarchies and overlapped and even superscalor instruction execution affect performance tradeoffs. The fact that pointers are used in coalesced hashing implies that a 100% load corresponds to something less (perhaps 90% or more for real applications) than the comparable load for pointerless methods such as double hashing, but this difference in no way diminishes its robust performance advantage at high loads. Even perfect hashing – which in modern form was introduced by Fredman, Kolmos and Szemerédi [2], and has since been amortized, dynamized, optimized and Sanforized in a sequence of works – will experience performance

* Supported in part by NSF grants CCR-8906949 and CCR–9204202.

† Courant Institute, NYU, NYC, NY 10012.

degradation when loads are not nominal (above, say, 50%). Furthermore, this scheme is actually a hierarchical multiprobe method since the hash location itself must be computed from several probes into function tables of n bits or more.

Unfortunately, the formal use of a fully random hash function h leaves our understanding of computable hashing in an unsatisfactory state, since such a function has a program size (Kolmogorov complexity) of $|U| \log n$ bits, on average. This size is larger than the space allotted to the hash table. We could use polynomials of degree $n-1$ to implement n-wise independent random functions (as defined below) to reduce the spatial cost to $O(\log\log m + n \log n)$, while raising the time needed for evaluating the hash function to n, which is no better than the time needed to search an unordered list.

A more constructive perspective on the traditional analyses such as those of Knuth [5] and Vitter and Chen [12] is that they establish optimal performance bounds for hash schemes that use programmable functions provided the measure of running time is taken to be the performance averaged over all possible input sequences. As is well known, this is not the same as a randomized performance bound, where the expected running time for any fixed sequence of data is shown to be optimal.

In terms of its use of random resources, hashing exerts especially unforgiving requirements, since we demand that a hash address be evaluated by a constant number of computations, and are obliged to retain enough of this random information to be able to recompute a key's probe location at search time. This last feature is somewhat different from the needs of many probabilistic algorithms, where we might suppose that some source of randomness could provide an unlimited number of random bits that could be discarded upon use, or where the requisite amount of randomness is a nominal fraction of the storage capacity or the processing time.

Carter and Wegman contributed to our understanding of limited randomness and randomized performance by introducing the notion of universal hash functions, and showing that these functions, when used for open hashing with separate chaining, work as well as fully random ones, [1], [13]. In particular, Carter and Wegman exhibited the universal classes F_h of h-wise independent hash functions, where F_h is defined to be the following set:

$$\{f \mid f(x) = (\sum_{0 \le j < h} a_j x^j \bmod p) \bmod n,\ a_j \in [0, p-1]\},$$

where $p \ge m$ is prime. They showed that if, for any $D \subset U$, a hash function is randomly selected from F_h (independent of D), then the sum of the expected j-th moments of the chain (i.e., list) lengths, in separate chaining, is essentially the same as that resulting from fully random functions, for $j \le h$. Their point is that since the second moment determines the expected retrieval time, pairwise independence guarantees optimal expected performance.

Carter and Wegman also posed as an open question whether a comparable result could be achieved for any form of closed hashing, such as double hashing. Schmidt and Siegel [9] resolved the double hashing question affirmatively if, for a sufficiently large c, $c \log n$-wise independent hash functions are used. For both double hashing, and this study of coalesced hashing, the evaluation time for the $c \log n$-wise independent hash functions presented by Carter and Wegman is too high, but the results of Siegel, [10] establish formally comparable families featuring a constant evaluation time for a reasonably standard word model Random Access Machine.

The double hashing bound of Schmidt and Siegel shows that with very high probability, the performance of double hashing can be quantified by statements about the hashing of many, many aggregates of $O(\log n)$ items. Indeed, the whole work is essentially a quantification of the probabilistic expressiveness of the underlying $\log n$ deep inclusion-exclusion algebra for double hashing.

In coalesced hashing, such an approach appears to be hopeless. A few more facts are necessary to explain why coalesced hashing introduces intrinsic difficulties into the milieu of limited independence. When the cellar size C is properly tuned for optimum performance, the cellar will (with overwhelming probability) overflow, and the colliders will start winding up within portion of the table that is directly accessed by the hash function.

The reason that such overflow is good is that the hashing range $[0..n-1]$ will be large if C is small, which reduces the number of collisions. The reason that overflow is bad is that a new entry can collide with a collider that overflowed into $T[0..n-1]$ and the collision chains can start growing with the larger ones growing more rapidly, due to a multiplicity of entries in $T[0..n-1]$. The fact, though, is that an optimal tuning of C will have a nontrivial percentage of the data as overflow in $T[0..n-1]$. Moreover, a nontrivial fraction of these colliders will have collided with previously inserted overflow items, which means that coalesced collision lists will exist and grow. Moreover, a non-trivial fraction of the data will participate in these events.

There are two consequences of this coalescing. The first is that the analysis gets more difficult. The second is that the probability that key x collides with the (overflow) collider x_i at location ℓ appears to be outside of the events that can be accurately estimated by our probability algebra for limited independence.

Let us focus, momentarily, on the second difficulty. The event that collider x_i is in $T[\ell]$, for some specific $\ell \geq 0$ is quantified as follows:

a. Among the keys $x_1, x_2, \ldots, x_{i-1}$, the number of colliders that have probes in the interval $[\ell+1..n-1]$ plus the number of keys with probes in $[0..\ell-1]$ is equal to $C+\ell$.
b. None of these items probes location ℓ.
c. Key x_i is a collider.

Thus, the placement of colliders within T depends on a linear fraction of the data, and even a random reassignment the probe value for any key in $x_1 \ldots, x_i - 1$ will have an $\Omega(1)$ probability of changing the resulting key that winds up in $T[\ell]$. Consider the following gedanken experiment. We pick LICH for simplicity (even though it is not optimal). Let H comprise all functions with suitable domain and range, so that h is fully random when selected at random from H. Let D be a fixed set of N keys. Partition H as follows. Let H_1 be the subset of $h \in H$ where the hashing of D causes the following to occur.

a. The hash function h causes x_N to probe a location that already contains a collider, which we shall name x_τ. (Of course the index τ depends on h.)
b. Let key $x_{\tau+}$ be the first collider that occurs after x_τ is inserted, for hash function h, and let $x_{\tau-}$ be the last collider that occured prior to the processing of x_τ. Finally, we require, for $h \in H_1$, that x_τ's collision list be shorter than the lists for $x_{\tau+}$ and $x_{\tau-}$, by the time D is fully hashed.

Let, for a given h, S be the subset of all keys in D that are linked into a list that contains either x_τ, $x_{\tau-}$, or $x_{\tau+}$, and let L be the set of probe locations that are hit by any of the members of S. Now pick a random key y from $D-S$, and modify h so that $h(y)$ is now (a probably different) random value in $[0..n-1]-L$. What happens? Key x_N now discovers that $x_{\tau-}$, x_τ, or $x_{\tau+}$ will reside in $T[\ell]$, depending on what probe location is assigned to y. Moreover, each of the three possibilities will occur with a non-negligible probability, and H_1 will comprise a non-negligible fraction of H.

It is not particularly difficult to imagine that the hashing statistics of our modified H still appear to be about the same, by the standards of limited independence. Yet the hashing performance for the key x_N has increased by a fixed amount, when we so modify H_1, and leave $H - H_1$ unchanged. In this sense, we cannot guarantee that coalesced hashing, with limited independence will statistically emulate the same process with full independence. This sensitivity to subtle dependencies stands in sharp contrast to such schemes as double hashing, where no such sensitivity is possible.

On the other hand, it may be possible to show that while invidious dependencies can conspire to degrade the performance of specific keys, the overall performance will not degrade. We resolve this difficulty differently, by introducing additional randomization into the hash functions, which prevent such performance degrading dependencies. Whether any standard form of $O(\log n)$-wise independent hash functions is alone sufficient to achieve optimal performance for coalesced hashing remains open.

An additional problem is that despite the analyses in [5], [3], [11] and [12], it is not clear whether the largest chain is bounded by $O(c \log n)$ with, say, probability $1 - 1/n^k$, for fixed k and suitable constant c. We would like to be able to establish this probability as a precursor to a proof that limited randomness in the range of $O(c \log n)$-wise independence performs well, once the hash functions are fixed to break unspeakably invidious dependencies. (The arbitrary fixed $k > 2$ are needed for analyzing higher moments of the probe performance.)

At issue, though, is how to go about finding the generating function or equivalent that governs the coalesced hashing process. Rather than do this directly, Knuth suggests a complicated procedure [5] for finding the expected number of probes and more complicated still for the variance. Other works give a magic moment cookie monster method based on eigenoperators that allows, with some ad hoc combinatorial calculations, probe moments to be generated sequentially one-at-a time. The material is tricky, and it is not clear if enough general information can be gathered in this manner. Formally at least, we need to bound a sum of all but the first $O(\log n)$ coefficients of an unknown generating function for a problem where, essentially, getting the first two or three coefficients is rather challenging.

Now the work by Vitter and Chen is a tour de force. They follow a derivation procedure that is similar to the coefficient-by-coefficient procedures established for the cellarless case, but is yet more difficult. The work is also notable for the completeness of the study; three variations of coalesced hashing are analyzed, optimality is

essentially established, deletion methods are analyzed, and subtle considerations that impinge upon the performance of each are exposed. The calculations comprise a daunting battle amidst a sea of recurrences to produce means and some variances (one or two coefficients) for several coalesced hashing processes.

3 Coalesced calculations.

It is interesting to note that our method of analyzing time ensembles gives a powerful asymptotic analysis for standard coalesced hashing, in the case of full randomness. Moreover, this analysis is complete enough to attain the comparable formulations for the cases where cellars are used. Although the analysis is not simple, we do believe that it is much simpler than previously published approaches [3], [5], [11], [12]. For standard coalesced hashing, define the generating function $F(x,\alpha) = \sum_{j>0} x^j p(\alpha n, j)$, where $p(\alpha n, j)$ is the probability that the αn-th record hits a chain that increases to size j, when the new item is included in the count. We shall call F^+ the asymptotic generating function, and use F to denote the exact formulation.

THEOREM 1.3.1 *For standard coalesced hashing, F^+ has the following asymptotic formulation as $n \to \infty$:*

$$F^+(x,\beta) = \frac{e^\beta x}{x - e^\beta(x-1)} + \frac{x}{x-1}\log(1 + (e^\beta - 1)(1-x)).$$

Proof. (Sketch) See brief discussion in Appendix.

THEOREM 1.3.2 *For standard coalesced hashing, F has the following exact formulation in a neighborhood of $x = 1$:*

$$F(x,\beta) = x(1 - \beta + \tfrac{1}{n}) + \sum_{1\le k}(((1+\tfrac{k}{n})^{n\beta-1} - 1)(1-\tfrac{1}{k}) + \beta - \tfrac{1}{n})(\tfrac{x-1}{x})^{k-1}.$$

Proof. (Sketch) See brief discussion in Appendix.

A comparable expression can be attained for a neighborhood of $x = 0$.

The power of these things is that expanding them in a Taylor Series about $x = 1$ (i.e. $\sum_i a_i(x-1)^i$) exposes the expected moments of the probe counts, while expanding about 0 reveals the probability of hitting chains of given lengths. The former is just what is needed for computing performance while the latter gives us a critical cutoff (more or less) for how many keys need to behave independently of each other. Theorem 1.3.2 is only valid near $x = 1$. Notice that it establishes that the k-th moment is just a finite combination of the first $k+1$ coefficients of $(\frac{x-1}{x})^j$, for $j \le k$. This is why we know that the convergence of coefficients inside the summation is valid for a finite number of Taylor coefficients of $(x-1)^j$.

The formulation for F^+ will turn out to be asymptotically exact both as an expansion about 0 and about 1. We still need

THEOREM 1.3.3 *For $\beta < 1$, the first $\sqrt{n}$ coefficients in the Taylor expansion of F about $x = 0$ zero are bounded by the corresponding coefficients of $4F^+$.*

Proof. (Sketch) See brief discussion in Appendix.

For $\beta = 1$, Theorem 1.3.3 is still valid, except that F^+ predicts a probability of 0 for the $n'th$ item to find a vacant slot.

We may now use F^+ to show that given c we may find a d so that with probability exceeding $1 - 1/n^c$, none of the coefficients in the indexed in $[d\log n, \sqrt{n}]$ are bigger than $1/n^c$. Well, this being the case, we may infer that with probability $1/n^{c-1}$, no chain will have as many as $\sqrt{n}$ elements.

COROLLARY 1.3.4 *Given c, there is a fixed d where the following holds. With probability $1 - 1/n^c$, no chain in coalesced hashing is of size $d\log n$ or more.*

Proof. From the formulation for F^+ in Theorem 1.3.1, we can read off that (the function is fine at $x = 1$;) its algebraic singularity is at $x = \frac{e}{e-1}$ or more, which ensures that the coefficients of the Taylor expansion about the origin are exponentially decreasing. Theorem 1.3.3 completes the proof. ∎

Notice that the asymptotic function F^+ is actually the derivative of the generating function for the distribution of chains, multiplied by x^2 and added to $(1-\alpha)x$. Interestingly, if try to reverse these steps to recover the original generating function, a polylogarithm ($\int \frac{\log(x)}{1-x}$) results. On the other hand, it turns out that **all** of the calculations for expected successful search time, the variance, in the hashing variations, both with and without cellars, have elementary evaluations. This includes calculations not provided by Vitter and Chen. (They say, for example, that the calculation for the variance in EICH has a derivation that is similar to those they give in the text, but becomes terribly messy due to complicated recurrences.) We show that the level of calculation for these and all moments is in fact simpler than the simplest calculations they use, thanks to the generating function formulations.

COROLLARY 1.3.5 *For standard coalesced hashing, F has the following exact formulation in a neighborhood of $x = 1$:*

$$F(x,\beta) = x(1-\beta+\tfrac{1}{n}) + \beta - \tfrac{1}{n} - \sum_{0\le k\le j} (((1+\tfrac{k+2}{n})^{n\beta-1} - 1)\tfrac{k+1}{k+2} + \beta - \tfrac{1}{n}) \times \binom{j}{k}(-1)^{j-k}(x-1)^{j+1}.$$

4 Randomizing probe functions.

For coalesced hashing, we seek a family of hash functions that feature limited randomness and provide a performance equivalent to the mathematically idealized fully random formulation. A construction that requires, perhaps, the simplest proof is based on the observation that coalesced hashing even works well at loads above 100%. To be a little more specific, suppose the hash table came with an attic as well as a cellar. Once the table becomes completely full, we place subsequent items in the attic, and chain them as usual. This is not very efficient, but if the attic were asymptotically small then the performance would not be materially damaged, since chains grow, on average, only in proportion to their size. For completeness, we note that the use of cellars as attics was formally analyzed, in the case of full randomness by Guibas [4], who showed that they are not efficient.

The hash functions are essentially the following: partition the addressable portion of the hash table into $\frac{\sqrt{n}}{\log^3 n}$ contiguous equal sized pieces, $I_1, I_2, \ldots, I_{\frac{\sqrt{n}}{\log^3 n}}$. (A skewed partitioning is actually better.) Our hash functions are modified so that each interval I_j, will have, essentially, no items probing at either end region. In particular, items that probe I_j are actually hashed into a local interval of size $|I_j| - \log^{10} n$. For each index j, two random numbers are needed, o_j and s_j. The offset o_j is uniformly selected from $[0, \log^{10} n]$, and is used to translate, by o_j, the location of the actual zone within I_j where probing occurs. The shift parameter s_j is selected uniformly from $[0, |I_j| - 1 - \log^{10} n]$. It is used to cause a circular shift among the available probe addresses within the local zone.

The parameters o_j and l_j, for $j = 1, 2, \ldots$ need to be $c \log n$-wise independent, where we have ensured that no chains are $c \log n$ in size or more. To meet space concerns, they need not be stored in tables but can be recomputed (in constant time) whenever necessary. The entire hash table, including the "attic" portions unavailable for direct probing is used on an availability basis for fill-in items during the ongoing left-to-right sweep.

The reason for the offsets is explained by observing that, for example, the first fill-in location in a table zone I_j is not uniformly distributed once slots are randomly filled. That is, the cyclic shifts don't get the job done. We can see this by noting that a sequence of Vacant and Occupied slots VOOOVVOO is four times more likely, despite circular shifts, to have the first fill-in land in the second (vacant) V location than the third, thanks to the left-to-right sweep rule. We can randomize this skewing by peeling off a random initial segment, so that the first item to enter is also somewhat randomized.

Large deviation bounds for our limited independence can ensure that with overwhelming probability, no item will, due to unfavorable VO configurations, land in a given fill-in a probe location with probability that is more than $(1 + 1/\log^2 n)$ above the average. Remaining irregularities include the slight overloads of probe locations, and possible biasing in which items use an interval I_j for fill-in.

There are more irregular cases; for example, a chain may have several items in the same interval. These cases are handled by giving heavier weightings to these events and noting that there are simply not enough of them to matter: they occur but are too improbable. The net result is that almost all events where hash chains occur will be multiplied by at most $(1 + 1/\log^2 n)^{c \log n}$, which gives an ensuing performance penalty of $\frac{1}{\log n} = o(1)$.

Notice that this revision of the hash function does not alter the left-to-right insertion rule for colliders, which is part of the algorithm's specification, and which keeps the amortized insertion time very low.

5 Cellar statistics.

When a cellar is present, LICH and EICH must be augmented with VICH, which inserts keys at the end of the list as long as the key is placed within the cellar, and otherwise inserts the key at the first chain location that follows the lead item and cellar members, if any. Now Vitter and Chen derive what asymptotic percentage of storage should be reserved for the cellar, for a given load density. Here we show that as a spinoff, we can calculate the generating function for the probe performance. We believe the calculation to be much simpler than the sea of recurrences approach, and capable of providing more information, such as the variance with little additional calculation, or an understanding of the performance for small numbers of items without resorting to simulation or intuitive reasoning.

Let, for notational convenience, the cellar have a size C, and the rest of the table be of size n, so that storage constraints keep $C+n$ fixed. In this formulation, we will represent the number of items in D as $|D| = \alpha n$, so that α depends on n. Let L be the number of table entries when the cellar becomes full.

Large deviation bounds show that even in the case of limited independence the filling time $(L + C)$ for a fixed cellar exhibits a very narrow fluctuation.

Due to the $O(\log n)$-wise independence, we may perform all of our calculations as if we had full independence and then verify that the resulting errors are small, even in aggregate.

Following this paradigm, we see that the expected number of items in the cellar after T insertions is $T - n(1-(1-\frac{1}{n})^T) \approx T - n(1-e^{T/n})$. Thus the cellar becomes full, approximately, at time $T = C + L$, where $T = \beta n$, $C \approx (1-e^{-\beta})n$, and L = $betan - (1-e^{-\beta})n$. Vitter and Chen, in fact show, that for full randomness, T turns out to be, with probability $1 - 1/n^c$, within $[\beta - O(\sqrt{n}\log n), \beta + O(\sqrt{n}\log n)]$. Similar properties can be established in the case of limited randomness; The errors due to the limited randomness and the uncertainty of L are not difficult to bound. The fully random case is studied in greater detail in the next section.

The point of these remarks is that we may momentarily treat L as a constant. The results, evidently, will not be any different from those of Vitter and Chen where full randomness is assumed.

The difference is that we can write down an asymptotic approximate generating function for the chain or probe distribution resulting from this process; the errors turn out to be bounded quite naturally. No new kinds of calculations are needed, although a few variations are.

Consider, for example, the total number of comparisons for successful search (the most interesting measure) when αn items are inserted in VICH or LICH.

THEOREM 1.5.1 *In VICH and LICH, the expected total number of probes can be expressed as*
$E\big[C_{tot} + C + L \times W(\alpha n - C - L + 1, 1)$
$+ \sum_{L<t\le \alpha n - C} W(\alpha n - C, t) + (\alpha n - C - L)\frac{L}{n}\frac{C}{L}\big]$,
where C_{tot} is the sum of the probes within the cellar that are needed to locate cellar items, C, is added as an extra cost for finding each cellar item since the search begins with a probe to the non-cellar region of the table. The function $W(w,t)$ is calculated for the cellarless case with w items inserted into a table size of n, and equals the expected total number of probes to locate the items belonging to the chain that originates with the t-th item.

Discussion: The probe cost for $W(w,t)$ is counted as zero, should the t-th item belong to some extant chain gang. Observe that an asymptotic formula for W (for LICH, and VICH) has already been computed as a natural course of analyzing cellarless hashing in the Appendix. (The exact form for LICH is also readily available, and we also present the exact from for VICH.) Finally, the last term is simply the expected additional search cost that must be sustained, due to the presence of cellar items, by those items that are inserted after the cellar becomes full, and have the misfortune of colliding with one of the first L table residents.

Apart from some error, $\mathrm{E}[C_{tot}] = \binom{C+L}{2}/n$, where L is viewed as a constant. Errors are addressed in the next section, where sharp asymptotic expansions are presented. For VICH, where the αn items are inserted so that cellar items are placed immediately following the lead item, and most recent items are chained in the first position that follows the original lead item and all cellar items (if any), we have the following asymptotic formula.

$W(w,t) = (1-t/n)\frac{e^{\frac{w-t}{n}}}{(e^{\frac{w-t}{n}} - 1 - 1/(e^{\frac{w-t}{n}}))^2}$, which, as is done in the Appendix, can be readily summed with great precision. It is straightforward to perform the appropriate substitutions and estimate the summations. The generating function $\sum_i a_i x^i$, where a_i is the expected number of items requiring i probes during search is characterized as follows.

THEOREM 1.5.2 *The generating function for the distribution of probe lengths in VICH or LICH is*
$E[x^{2\frac{C_{tot}(1)-C_{tot}(x)}{1-x}} + L \times W(\alpha n - C - L + 1, 1, x)$
$+P(\alpha n - C - L, x) \times x(C_{tot}(x) - C_{tot}(0))$
$+\sum_{L<t\le \alpha n - L - C} W(\alpha n - C, t, x)]$,
where $C_{tot}(x) = \sum c_i x^i$ where c_i is the expected number of chains with cellar length i, and $P(t,x)$ is the generating function for the probe counts (minus 1) of successful searches for chained items that collide with item 1, after $t+1$ items have been inserted: $P(t,x) = \sum p_i x^i$, where p_i is the expected number of items that collide with the first hash key, and that require i+1 probes during search.

Discussion: For EICH, the recurrence for P is
$P(0,x) = 0$
$P(t,x) = P(t-1,x) - \frac{x}{n}P_x(t-1,x)$
$+\frac{x^2}{n}P_x(t-1,x) + \frac{1}{n}x,\ t > 0.$

This is readily solved with the aid of ϕ as defined and used at length in the Appendix:
$P(x,t) = \frac{1}{n}\sum_{0\le j\le t-1}\phi^j x = \frac{t}{n} + \sum_{1\le k}\frac{(1+\frac{k}{n})^t - 1}{k}(\frac{x-1}{x})^k$.
$W(t,s,x)$ is the generating function governing probe lengths in the cellarless case. For EICH (VICH), we get

$$W(t,s,x) = (1-\frac{s-1}{n})W(t-s,1,x)$$
$$= (1-\frac{s-1}{n})xf(t-s,x), \text{ where}$$

$f(1,x) = 1$,
$f(t,x) = f(t-1,x) - \frac{x}{n}f_x(t-1,x) + \frac{x^2}{n}f_x(t-1,x)$
$+\frac{x}{n}g_x(t-1,1)$,
where g is the generating function for the length of the chain that originates with the first probe:

$g(1,x) = x$,
$g(t,x) = g(t-1,x) - \frac{x}{n} g_x(t-1,x) + \frac{x^2}{n} g_x(t-1,x)$.
So $g(x,t) = \phi^{t-1}x$, and
$f(t,x) = \phi^{t-1}1 + \sum_{0\le k\le t-2} \phi^k \frac{x}{n} E_1 D\phi^{t-2-k}x$
$= 1 + \sum_{0\le k\le t-2} \phi^k (1+\frac{1}{n})^{t-2-k}\frac{x}{n}$
$= 1 + (1+\frac{1}{n})^{t-2} \sum_{0\le k\le t-2} \phi^k (1+\frac{1}{n})^{-k}x$
$= 1 + (1+\frac{1}{n})^{t-2}(1-(1+\frac{1}{n})^{-t+1})$
$\times \left[(n+1) + (t-1)(\frac{x-1}{x})\right.$
$\left. + \sum_{1\le j} \frac{((1+\frac{j}{n+1})^{t-1}-1)(n+1)}{j}(\frac{x-1}{x})^{j+1}\right]$.

For LICH, $P(t,x) = \int_0^x \frac{f(1)-f(x)/x}{1-x}dx$, where $f(x)$ is the generating function for the chain length, at time t, of the chain that is headed by the item inserted at time 1, and operator transformations give

$$W(x,t) = xf_x(0) + x^2 \frac{\int_x^1 \frac{1-\frac{f(x)}{x}}{1-x}dx}{1-x}.$$

For this formulation, $C_{tot}(x)$ is the generating function for the distribution of lengths of the L chains residing in the cellar (some of which are 0 items long).

This is the only portion of the generating function that is presently not expressed in a way that gives sharp asymptotics; there is little error, however, because L is nearly constant. It is very likely that the same style of asymptotics as in the next section can give the errors for this calculation as well. Indeed, as noted in the next section, some of the errors are remarkably small. For a fixed stopping time $C+L$, the generating function is $C_{tot}(x) = n\frac{(1-\frac{1}{n}+\frac{x}{n})^{C+L} - (1-\frac{1}{n})^{C+L}}{x}$, where we have stripped out the lead (non-cellar) item from each chain. (Notice that $C_{tot}(0) = L$.)

We may put these observations together with asymptotic generating functions where we formally take L to be fixed. The errors will turn out to be small, for times $t >> T$,

5.1 Nailing down the cellar.

There is a thematic reason why we can anticipate asymptotic results with a multiplicative accuracy of $1+O(\frac{1}{n})$, rather than $1+O(\frac{1}{\sqrt{n}})$. We briefly summarize this reason at the end of this section. Further expectations are that the coefficients for these corrections can be found, and the analysis will apply even for small values of n. (C.f [12], where weaker bounds are attained, and plausible but informal reasons are given to explain why optimizing approximate results gives such good answers for modest sizes of n.)

More precise calculations for these hashing schemes require the probability distribution $p(T,C)$ for the stopping time T when the cellar becomes full with C items. Let $L = T - C$. Inclusion-exclusion (plus standard, albeit alighted reasoning) gives:

$$p(T,C) = \binom{n}{L}(\frac{L}{n})^T \sum_j \binom{L}{L-j}(-1)^j(\frac{L-j}{L})^{T-1}$$
$$= \binom{n}{L} L \frac{1}{n^T} D_x^{T-1}(e^x-1)^L|_{x=0}$$
$$= \binom{n}{L} L \frac{1}{n^T}(T-1)! \frac{1}{2\pi}\int_{-\infty}^{\infty} \frac{(e^{x+iy}-1)^L}{(x+iy)^T}dy.$$

Saddle point methods (used implicitly here) lead to explicit calculations that we choose to omit. The consequence of these calculations is that the stopping time T when the cellar is full turns out to be a nearly perfect Gaussian with mean $n\beta$. As a weak consequence:

$$\mathrm{E}[T] = n\beta + O(1),\ \mathrm{E}[T^2] = n^2\beta^2 + O(n).$$

The coefficients of the main contributions of error can all be readily exposed.

It is worth observing that hardest error calculations are not due to difficulties with coalesced hashing calculus, but are entirely attributable to approximating the behavior of the almost constant (almost Gaussian) random variable T, and its particular consequences for the portion of the algorithm that emulates separate chaining.

The errors due to the fluctuations of the stopping time are quite small and are slowly varying, which explains why they can be so successfully ignored. More formally, for a smooth, well behaved function f and well behaved random (nearly constant) variable X with variance σ^2, we may expect that $\mathrm{E}[f(X)] \approx f(\mathrm{E}[x]) + \frac{1}{2}\sigma^2 f''(\mathrm{E}[X])$, which is what gives such small errors.

This last observation, it turns out, does hold for the quantity C_{tot}, which has an expected contribution that, up to a factor of $1+O(\frac{1}{n})$, is the same as the analogous work done up to the constant stopping time $\mathrm{E}[T]$, according to omitted calculation.

6 Conclusions.

A natural way to establish bounds for limited independence is to show a statistical equivalence between collision events for limited and full independence:
If a set of keys S_1 is hashed in a way that does not affect the location of key x, and S_2 is hashed in a way that does affect the outcome, then we would like to prove that x and S_2 will have comparable chances of achieving this outcome when hashed by either a fully independent

hash function or by some suitable universal family of k-wise independent hash functions. We would also like to show that S_1 is equally likely to behave as described with either hashing model.

This paper discusses what is involved to ensure that such claims can be made. Due to space limitations and the level of technical material involved, it does not explicitly piece the arguments together, nor does it give a complete characterization of each piece. Nevertheless, the critical issues are exposed, and the pieces can indeed be combined to show that coalesced hashing works well for universal families of $c \log n$-wise independent hash function that are crafted to include our extra form of statistical randomness. The estimates and randomization procedures are strong enough to attain, up to a factor of $(1 + O(\frac{1}{\log n}))$, the same asymptotic performance for the full suite of hashing variations studied by Vitter and Chen.

For limited independence, performance calculations are typically somewhat personalized, in the sense that the participants S_1 and S_2 are named. See, for example the Appendix, where the participants in a chain are recursively named (along with their probability of appearance).

The process of formalizing the personalized calculations leads to slightly different recurrences, which turn out to have natural generating functions. Moreover, these formulations capture the precise quantities that need to be calculated for general coalesced hashing with a cellar, even in the case of full independence.

7 Acknowledgments.

It is a pleasure to thank Jeff Vitter for stimulating conversation and helpful suggestions.

8 Appendix: some of the calculations.

The basic calculations are below.

Gist of Theorem 1.3.1 and Theorem 1.3.3

For coalesced hashing, let $\tau(\kappa, a, b)$ be the probability that a collision chain originates with the a-th item, and increases to size κ because the b-th item joined the chain as the κ-th item. By recurring on item the hit by x_a, we get:

$$\tau(\kappa,a,b) = \begin{cases} \frac{1}{n}\sum_{a\le c<b}(\kappa-1)\tau(\kappa-1,a,c)(1-\frac{\kappa-1}{n})^{b-c-1}, & \text{if } \kappa>1; \\ \tau(1,a,b) = 1-\frac{a-1}{n}, & \text{if } a=b; \\ 0 & \text{otherwise.} \end{cases}$$

Notice that τ is exact. The calculation is personalized in the sense that the other items in the chain are recursively identified, beginning with a. We solve

$t(\kappa,a,b) = \frac{1}{n}\sum_{a\le c<b}(\kappa-1)t(\kappa-1,a,c)e^{\frac{-(\kappa-1)(b-c)}{n}}$, $\kappa > 1$.

$t(1,a,b) = 1-\frac{a}{n}$, if $a = b$, and 0 otherwise.

It is not difficult to bound the solution for τ by the solution for t, for $k \le \sqrt{n}$, which is Theorem 1.3.3, in part. Put

$t(\kappa,a,b) = r(\kappa,a,b)e^{\frac{-\kappa(b-a)}{n}}$ then

$r(\kappa,a,b)e^{\frac{-b+a}{n}} = \frac{1}{n}\sum_{a\le c<b}(\kappa-1)r(\kappa-1,a,c)$, $\kappa > 2$.

$r(2,a,b) = \frac{1-a/n}{n}e^{\frac{b-a}{n}}$, if $a < b$, and 0 otherwise.

Momentarily ignoring errors, we put $\alpha = a/n$, $\beta = b/n$, and set $f(x,\beta) = \sum_{j\ge2}\sum_{\alpha n<c<\beta n}\frac{r(j,\alpha n,c)}{n}x^j$.

Then

$e^{-\beta+\alpha}\frac{d}{d\beta}f(x,\beta) = x^2\frac{d}{dx}f(x,\beta) + x^2(1-\alpha)/n$, $\beta > \alpha$.

Using the method of characteristics, we put $\frac{dx}{dz} = -x^2$, $\frac{d\beta}{dz} = e^{-\beta+\alpha}$. So $z = c + \frac{1}{x}$, and $e^{\beta-\alpha} = z$ whence $\frac{df}{dz} = \frac{1-\alpha}{n(z-c)^2}$. The initial condition is $f|_{z=1} = 0$, since $\alpha = \beta$ when $z = 1$.

So $f = -\frac{1-\alpha}{n(z-c)} + G(c)$. Setting $G(c) = -\frac{1-\alpha}{n(c-1)}$ satisfies the boundary condition.

Now $c = z - 1/x = e^{\beta-\alpha} - 1/x$, so $f = -\frac{1-\alpha}{n}(x + \frac{1}{e^{\beta-\alpha}-1/x-1})$. So $\frac{d}{d\beta}f(x,\beta) = \frac{1-\alpha}{n}(\frac{e^{\beta-\alpha}}{(e^{\beta-\alpha}-1-1/x)^2})$ and the generating function for the size of the chain hit by the βn-th item is given by

$$\begin{aligned} F^+(x,\beta) &= x(1-\beta) + \sum_{0\le a<\beta n}\sum_{k\ge2} t(k,a,n\beta)x^k \\ &= x(1-\beta) + \sum_{0\le n\alpha<\beta n}\sum_{k\ge2} r(k,\alpha,n\beta)(xe^{\alpha-\beta})^k \\ &\to x(1-\beta) + \int_0^\beta \frac{(1-\alpha)e^{\beta-\alpha}}{(e^{\beta-\alpha}-1-1/(xe^{\alpha-\beta}))^2}\,d\alpha \\ &= \frac{1-\beta}{1-x} - \frac{x^2}{(x-1)^2e^\beta - x^2 + x} \\ &\quad - \frac{x(\beta - \log((1-x)e^\beta + x))}{x-1}. \end{aligned}$$

Hence we have the estimate

$$F^+(x,\beta) = \frac{e^\beta x}{x - e^\beta(x-1)} + \frac{x\log(1+(e^\beta-1)(1-x))}{x-1}.$$

So

$$\begin{aligned} F^+(x,\beta) &= 1 + \tfrac{1+e^{2\beta}}{2}(x-1) \\ &\quad + \textstyle\sum_{2\le j}(\frac{2j-1}{j} + \frac{je^\beta - j}{j+1} + \frac{1}{e^\beta-1})(e^\beta-1)^j)(x-1)^j \\ &= 1 + \tfrac{e^{2\beta}+1}{2}(x-1) + \tfrac{4e^{3\beta}-3e^{2\beta}-1}{6}(x-1)^2 \\ &\quad + \tfrac{9e^{4\beta}-16e^{3\beta}+6e^{2\beta}+1}{12}(x-1)^3 \\ &\quad + \tfrac{16e^{5\beta}-45e^{4\beta}+40e^{3\beta}-10e^{2\beta}-1}{20}(x-1)^4 + \ldots \end{aligned}$$

It is not difficult to verify by direct substitution that F^+ in fact dominates the actual solution for $t(k,a,b)$.

Gist of Theorem 1.3.2

It is also possible to get an exact pseudogenerating function for the table distribution as follows. Let
$f(x,t) =$
$\sum_{1\le k} \mathrm{E}[\#\text{ of chains of size } k \text{ after } t \text{ insertions}]x^k$.
So $f(x,0) = 0$. Now it is easy to see that the time evolution operator $\Phi(t)$: $f(x,t) = \Phi(t)f(x,t-1)$ is given by
$\Phi(t)f(x,t-1) =$
$x^2 f_x(x,t-1)/n + (f - xf_x(x,t-1)/n) + x(1-\frac{t-1}{n})$.
So $\Phi = I + \frac{x^2-x}{n}\frac{d}{dx}$, and $f(t,x) = \sum_{0\le s<t} \frac{n-s}{n}\Phi^{t-s-1}x$.
Solving for the eigenfunctions of Φ gives
$\Phi g = \lambda g$ so $(1-\lambda)n/(x-x^2) = g_x/g$ and $g(x) = (\frac{1-x}{x})^{(\lambda-1)n}$. Expanding $x-1 = \sum a_k(\frac{1-x}{x})^k = \sum a_k w^k$ gives $x-1 = 1/(w+1)-1 = -w/(1+w)$. So

$$x-1 = \sum_{1\le k}(\frac{x-1}{x})^k \text{ and}$$

$$\begin{aligned}
f(t,x) &= \sum_{0\le s<t}\frac{n-s}{n}(1+\sum_{1\le k}(\frac{x-1}{x})^k(1+k/n)^{t-s-1}) \\
&= t - t(t-1)/2n \\
&\quad + \sum_{1\le k}(\sum_{0\le s<t}\frac{n-s}{n}(1+k/n)^{t-s-1})(\frac{x-1}{x})^k \\
&= t - t(t-1)/2n \\
&\quad + \sum_{1\le k}(\sum_{0\le s<t}\frac{n-s}{n}(1+k/n)^{t-s-1})(\frac{x-1}{x})^k \\
&= t - t(t-1)/2n \\
&\quad + \sum_{1\le k}\Big(\frac{n(1+k/n)^t - n}{k} \\
&\qquad - n\frac{(1+k/n)^t - 1 - tk/n}{k^2}\Big)(\frac{x-1}{x})^k \\
&= t - t(t-1)/2n \\
&\quad + \sum_{1\le k}(\frac{n(1+\frac{k}{n})^t - n}{k}(1-1/k) + \frac{t}{k})(\frac{x-1}{x})^k.
\end{aligned}$$

This formula is good enough to define fixed moments of the expected probe count precisely. We use it to compute $x(1-\frac{t}{n}) + x^2 f_x(x,t)/n$, which gives the generating function for the chain encountered when item $t+1$ is inserted, and switch to asymptotics. We get:
$x(1-\beta) + \sum_{1\le k}((e^{k\beta}-1)(1-1/k)+\beta)(\frac{x-1}{x})^{k-1}$
$= x(1-\beta) + \frac{e^\beta x}{x-e^\beta(x-1)} + (\beta-1)x$
$\quad + \frac{x\log(1-e^\beta(x-1)/x)}{x-1} - \frac{\log(1-\frac{x-1}{x})}{x-1}$
$= \frac{e^\beta x}{x-e^\beta(x-1)} + x\log(1-e^\beta(x-1))/(x-1)$, which agrees with F^+.

Given F^+, we may, for example, readily compute the mean and variance for the successful insertion time in EICH, where chain items are inserted immediately following the record occupying their probe location, or in LICH, probes are inserted at the end of the list. In either case, the chain length distribution is the same, the only difference is the expected value to be assigned to a chain of length j.

It is convenient to determine the probe cost functionals in terms of the generating function for the actual chain distribution $f(x) = \sum_j a_j x^j$, where a_j is the expected number of chains of length j.

The calculation for LICH is simple. Let $M^{(1)}(j)$ be the expected number of probes to insert all j items that comprise a chain of size j. Then $M^{(1)}(1) = 1$ and $M^{(1)}(j) = M^{(1)}(j-1) + \frac{j}{2} + 1$, since each of the $j-1$ items is equally likely to be hit by the j-th item. Thus $M^{(1)}(j) = \frac{j+(j-1)(j+2)}{4} = j^2/4 + 5j/4 - 1/2$, for $j > 1$.

Let $M^{(2)}(j) = \mathrm{E}[\sum_1^j probe(k)^2]$, where $probe(k)$ is the number of probes used to insert the k-th entry in the chain. This second moment satisfies $M^{(2)}(1) = 1$ and $M^{(2)}(j) = M^{(2)}(j-1) + \frac{2^2+3^2+\ldots(j)^2}{j-1}$. Solving gives $M^{(2)}(j) = \frac{j^3}{9} + \frac{7j^2}{12} + \frac{53j}{36} - \frac{7}{6}$.

For early insertion, EICH, cookie calculus and sliding operators show that $M^{(1)}(j) = j(1+\frac{1}{2}+\ldots+\frac{1}{j})$. $M^{(2)} = j^2 - j + j(1+\frac{1}{2}+\ldots+\frac{1}{j})$. Of course EICH is the same as VICH in the cellarless case, which is all that matters.

In terms of operators, these mappings are functionals on x^j; that is, we wish to map x^j into $j^2/4 + 5j/4 - 1/2$, $\frac{j^3}{9} + \frac{7j^2}{12} + \frac{53j}{36} - \frac{7}{6}$, $j(1+\frac{1}{2}+\ldots+\frac{1}{j})$, or $j^2 - j + j(1+\frac{1}{2}+\ldots+\frac{1}{j})$. Then the generating function for the chain distribution is mapped into the expected respective moment for each of the hashing strategies.

Clearly the mapping $x^j \to poly(j)$ is done by applying differential operators and evaluating the result at 1. They can be readily applied to both the exact and the asymptotic formulations. For EICH, there is a slight problem. The operator $x^j \to j(\frac{x}{1} + \frac{x^2}{2} + \ldots + \frac{x^j}{j})$ is achieved by applying the transformation $f \to \int_0^x \frac{f_x(1) - xf_x(x)}{1-x}dx$. While the result is readily attained in closed form for the asymptotic function F^+

(i.e., $\int_0^1 \frac{poly(x)-F^+}{x(1-x)}dx$), the analogous computation for the exact formulation does not appear to give the constant term due to the restricted domain of convergence, and the constant is precisely the value sought. (We can apply $f \to \int_1^x \frac{f_x(1)-xf_x(x)}{1-x}dx$ to get $\int_1^x \frac{xf_x(x,t)}{1-x}dx = -t\log(x) + \sum_{2\le k}(\frac{n(1+k/n)^t-n}{k} + \frac{t}{k-1})(\frac{x-1}{x})^{k-1}$, but it is not clear how to adjust the computation to integrate from 0.)

Cookie calculus can give us the precise result, but so can generating functions and straightforward eigenfunction expansions. In this case, we solve for the generating function $g(x,t) = \sum a_i x^i = xf(t,x)$, where a_i is the expected number of successful searches requiring i probes after t insertions. We get

$$f(1,x) = 1,$$

$$\begin{aligned} f(t,x) &= f(t-1,x) - \frac{x}{n}f_x(t-1,x) + \frac{x^2}{n}f_x(t-1,x) \\ &\quad + \frac{t-1}{n}x + 1 - \frac{t-1}{n}. \\ &= \sum_{0\le j\le t-1} \phi^j\left(\frac{t-1-j}{n}x + 1 - \frac{t-1-j}{n}\right), \end{aligned}$$

where $\phi = (I + \frac{x^2-x}{n}d/dx)$.

Hence

$$\begin{aligned} f(t,x) &= t - \frac{(t-1)t}{2n} \\ &\quad + \sum_{0\le j\le t-1} \frac{t-1-j}{n}\sum_{0\le k}(1+\frac{k}{n})^j(\frac{x-1}{x})^k \\ &= t + \sum_{1\le k}(\frac{x-1}{x})^k\left(\frac{t((1+\frac{k}{n})^t-1)}{k}\right. \\ &\quad \left. -\frac{(t+1)(1+\frac{k}{n})^t}{k} + \frac{n((1+\frac{k}{n})^{t+1}-1)}{k^2}\right) \\ &= t + \sum_{1\le k}\left(-\frac{t}{k} + \frac{n(1+\frac{k}{n})^t}{k^2} - \frac{n}{k^2}\right)(\frac{x-1}{x})^k. \end{aligned}$$

So $g(x,t) = xt + x\sum_{1\le k}\left(-\frac{t}{k} + \frac{n(1+\frac{k}{n})^t}{k^2} - \frac{n}{k^2}\right)(\frac{x-1}{x})^k$.

For the function g, moments are achieved for EICH by evaluating derivatives at $x = 1$.

The function $W(w,t)$ in Theorem 1.5.1 is defined by a derivation ($W(w,t) = g_x(w,t,1)$) comparable to that given for $g(x,t)$, but restricted to the chain born at time t, so that $g(t,w,x) = (1-\frac{t-1}{n})xf(w,x)$, where $f(s,x) = 1$, if $s = t$,

$$\begin{aligned} f(s,x) &= f(s-1,x) - \tfrac{x}{n}f_x(s-1,x) + \tfrac{x^2}{n}f_x(s-1,x) \\ &\quad + \tfrac{x}{n}f_x(s-1,1) + \tfrac{x}{n},\ s > t, \\ &= (\psi^{s-t}1) + \textstyle\sum_{0\le j<s-t}\psi^j\frac{x}{n}, \end{aligned}$$

where $\psi = \phi + xE_{x=1}\frac{D_x}{n}$.

So $\psi^j = \phi^j + \sum_{0\le i<j}\phi^i x E_{x=1}\frac{D_x}{n}(1+\frac{2}{n})^{j-i-1}$, and

$$\begin{aligned} f(w,x) &= 1 + \textstyle\sum_{0\le j<w-t}(\phi^j\frac{x}{n} + \sum_{0\le i<j}(1+\frac{2}{n})^{j-i-1}\phi^i\frac{x}{n^2}) \\ &= 1 + \textstyle\sum_0^{w-t-1}(1+\frac{1}{2}((1+\frac{2}{n})^{w-t-1-j}-1))\phi^j\frac{x}{n} \\ &= 1 + \textstyle\frac{1}{2n}\sum_k\sum_{j=0}^{w-t-1}\big(1 + ((1+\frac{2}{n})^{w-t-1-j} \\ &\qquad \times(1+\tfrac{k}{n})^j\big)(\tfrac{x-1}{x})^k \\ &= 1 + \frac{w-t}{2n} + \frac{((1+\frac{2}{n})^{w-t}-1)}{4} \\ &\quad + (\frac{w-t}{2n} + \frac{(1+\frac{2}{n})^{w-t}-(1+\frac{1}{n})^{w-t}}{2})\frac{x-1}{x} \\ &\quad + \frac{w-t+(w-t)(1+\frac{2}{n})^{w-t-1}}{2n}(\frac{x-1}{x})^2 \\ &\quad + \textstyle\sum_{k>2}(\frac{w-t}{2n} + \frac{((1+\frac{k}{n})^{w-t})-(1+\frac{2}{n})^{w-t}}{2k-4})(\frac{x-1}{x})^k. \end{aligned}$$

References

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

[2] M. L. Fredman, J. Komlós and E. Szemerédi, *Storing a Sparse Table with $O(1)$ Worst Case Access Time*, Journal of the Association for Computing Machinery, Vol 31, No. 3, (1984) pp. 538–544.

[3] D. H. Greene and D. E. Knuth, Mathematics for the Analysis of Algorithms, Birkhauser, 1982.

[4] L. J. Guibas, The Analysis of Hashing Algorithms, Technical Report STAN-CS-76-556, Stanford Univ., 1976.

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

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

[7] 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.

[8] ——, *Data Structures and Algorithms 1: Sorting and Searching*, Springer-Verlag, Berlin Heidelberg, 1984.

[9] J. P. Schmidt and A. Siegel, *The analysis of closed hashing under limited randomness*, 22nd STOC, 1990.

[10] A. Siegel, *On universal classes of fast hash functions, their time-space tradeoff, and their applications*, 30th FOCS, 1989, pp. 20–25. pp. 224–234.

[11] J. S. Vitter, *Analysis of the Search Performance of Coalesced Hashing*, JACM, Vol 30, No. 2, 1983, pp. 231–258.

[12] ——, and W.-C. Chen, Design and Analysis of Coalesced Hashing, Oxford University Press, 1987.

[13] M. N. Wegman and J. L. Carter, *New Classes and Applications of Hash Functions*, 20th FOCS, 1979, pp. 175–182.

[14] F. A. Williams, *Handling Identifiers as Internal Symbols in Language Processors*, CACM, Vol 2, No. 6, (1959), pp. 21–24.