Title
Uniform Hashing with Multiple Passbits

Authors
Martini, Paul
Burkhard, Walter

Publication Date
2000-08-18

Peer reviewed

Uniform Hashing with Multiple Passbits

Paul Martini Walter A. Burkhard*

Gemini Storage Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093−0114, USA
{pmartini, burkhard}@ucsd.edu

Abstract

We present a novel extension to passbits providing significant reduction to unsuccessful search lengths for open addressing collision resolution hashing. Both experimental and analytical results included demonstrate the dramatic reductions possible. This method does not restrict the hashing table configuration parameters and utilizes very little additional storage space per bucket. The runtime performance for insertion is essentially the same as for ordinary open addressing with passbits; the successful search lengths remain the same as for open addressing without passbits; the unsuccessful search lengths can be made to be arbitrarily close to one bucket access for any desired loading factor.

1 Introduction

Hashing is a well-known implementation technique for organizing data stored internally or externally. Numerous schemes exist for handling collisions such as open-addressing; each record determines the probe sequence used to store or retrieve it. To store a record, it is placed in the first non-full bucket of its probe sequence; to search for a record, buckets designated by its probe sequence are examined in order until finding it or a non-full bucket, not containing it, is encountered indicating the record is not present. Uniform hashing, introduced by Peterson [8], is an idealized model; double hashing is an efficient scheme to generate a record's probe sequence. Recently Lueker and Molodowitch [7] as well as Guibas and Szemeredi[4] have shown double hashing to be asymptotically equivalent to the ideal uniform hashing which maps records to random permutations.

In case the bucket capacity is one, Ullman [10] raised an optimality question and presented a model for discussing it. More recently, Yao [11] has shown that uniform hashing is optimal among all open-addressing schemes in the sense that the expected successful search length is at least $-\alpha^{-1}\log(1-\alpha)$ for loading factor α. For the open-addressing expected unsuccessful search length, Yao [11] poses the question – is uniform hashing also optimal among all open-addressing schemes in the sense that the expected unsuccessful search length is at least $1/(1-\alpha)$?

One approach to significantly reduce the unsuccessful search length is to include a passbit in each bucket as presented by Furukawa [3] and Amble and Knuth [1]. The passbit is initially set false for an empty structure, and it set true only when the bucket is full and at least one overflow occurs at the bucket. Now an unsuccessful search can terminate when the search accesses a bucket with a false passbit.

Technical Report CS00-xxx, Computer Science and Engineering, UCSD. * Sabbatic leave at IBM Almaden Research Center.

2 Multiple Passbits

We slightly modify the passbit approach in that each bucket will contain $g \geq 1$ passbits $pb_0, pb_1, \ldots, pb_{g-1}$. A table consisting of n buckets contains ng passbits. The hash value space is partitioned into g equal-sized blocks referred to as *varieties*. Accordingly, any probe sequence within variety i will be associated with passbit pb_i at each bucket. All passbits are initially set false for an empty structure. Passbit pb_i at a bucket is set true only when the bucket is full and at least one overflow occurs via a variety i probe. An unsuccessful search via a variety i probe sequence will terminate at a bucket where pb_i is false.

We present two table data type methods *insert* and *fetch* to demonstrate multiple passbits. These algorithms are essentially the same as for the single passbit configuration other than in the first step determining which passbit j to access.

insert (Data & data)

1. **determine probe sequence** $p_0, p_1, \ldots, p_{n-1}$ **and**
   determine passbit index j**;**
2. **for** ($i = 0$; **bucket** [p_i] **is full** ; $i{+}{+}$)
   check bucket [p_i] **for duplicate record; if found return false ;**
   bucket [p_i] . pb_j **= true;**
3. **check bucket** [p_i] **for duplicate record; if found return false ;**
   copy record into bucket [p_i] **; return true;**

fetch (Data & data)

1. **determine probe sequence** $p_0, p_1, \ldots, p_{n-1}$ **and**
   determine passbit index j**;**
2. **for** ($i = 0$; ; $i{+}{+}$)
   check bucket [p_i] **for record data;**
   if found, copy data and return true;
   if (bucket [p_i] **.** pb_j **is false) return false;**

The first step in either algorithm implementing double hashing would entail determining the initial probe p_0 as well as the step size; designating the hash value as `val`, p_0 is calculated as `val` % `n` and the step size is `val` % (`n` − 1) + 1. The passbit index is calculated as follows:

```
j = ( val + val/g ) % g.
```

We discuss the passbit index selection in section 4.

3 Multiple Passbits Analysis

Our table implementation algorithms are derived from open addressing with double hashing collision resolution in which there are n buckets each containing b slots and g passbits. The passbits dramatically reduce the expected unsuccessful search length as we will demonstrate here. A table configuration of m records is constructed using a sequence of k probes, referred to as the *k-construction sequence,* which is the concatenation of the prefixes of probe sequences for each of the records stored within the table. This terminology is exemplified in Figure 1. The 14-construction sequence of numbers conveys the record placement within the table. The records were inserted into an empty table in the order shown within the figure; cat, dog, sow, etc. The record cat contributes the first entry 1 of the construction sequence, dog contributes the first 2, and sow contributes the next entry 4. After two more records, bucket 2 is filled and when record ape is inserted, it collides with bucket 2 and moves to bucket 4; passbit pb_2, is set to true at this time. The process is recorded as the next two entries within the construction sequence. The remaining portion of the construction sequence is built in a similar fashion.

First we calculate the *loading factor* α given k. The loading factor measures the expected number of records per slot; α depends upon b and n but not g.

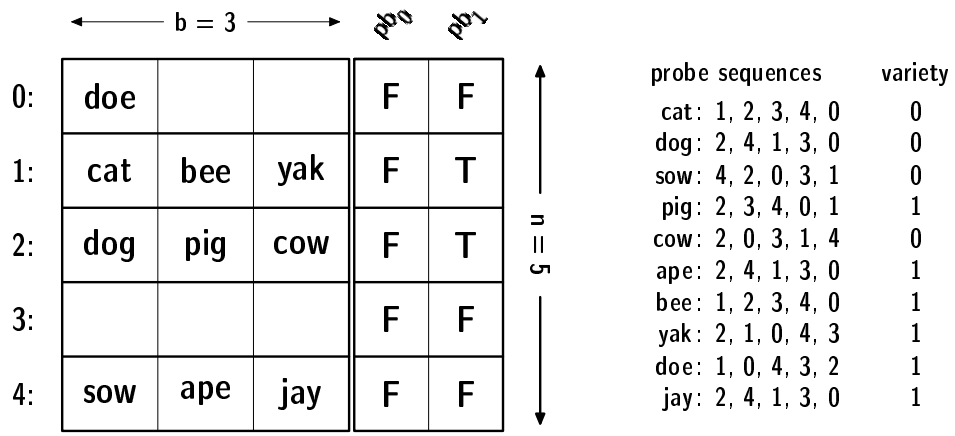

Figure 1: Configuration with $n = 5$ buckets, $b = 3$ slots per bucket, and $g = 2$ passbits per bucket.

P_i designates the probability that i of k probes, within the k-construction sequence, access a particular bucket. Since uniform hashing is assumed, all buckets are equally likely to be accessed –

$$P_i = \binom{k}{i} \left(\frac{1}{n}\right)^i \left(1 - \frac{1}{n}\right)^{k-i} \qquad 0 \le i \le k. \tag{1}$$

The expected number of records per bucket is

$$\alpha\, b \quad = \quad \sum_{i=0}^{b-1} i\, P_i \;+\; b \sum_{i\,=\,b}^{k} P_i\,.$$

Accordingly, we have

$$\alpha \quad = \quad 1 - \sum_{i\,=\,0}^{b-1} \left(\frac{b-i}{b}\right) P_i\,. \tag{2}$$

The loading factor α may be calculated given k and parameters b and n. We will have occasion to calculate k for a given α as well; in case b is one, it is possible to determine a closed-form expression for k in terms of α. Unfortunately, no other such inverse functions are known. The ratio k/n, referred to as the *probing factor*, measures the number of probes per bucket in the k-sequence. In case b is one, the probing factor may be approximated by $-\log(1-\alpha)$. That is,

$$\alpha = 1 - P_0 \;=\; 1 - \left(1 - \frac{1}{n}\right)^k.$$

Then letting $k, n \to \infty$ such that the ratio k/n is constant λ, we have $\alpha = 1 - e^{-\lambda}$ or $\lambda = -\log(1-\alpha)$. We will have occasion to consider similar limiting situations subsequently and we designate as the *asymptotic situation* the process of letting $k, n \to \infty$ while maintaining a constant probing factor λ.

3.1 Successful Search Length

The successful search length measures the number of probes per inserted record. The ratio k/nb measures the number of probes per slot. Thus the ratio

$$\frac{k}{n\,b\,\alpha}$$

measures the expected number of probes per record; that is, the expected successful search length. Accordingly, we have

$$\frac{k}{n\,(\,b - \sum_{i=0}^{b-1} (\,b-i\,)P_i\,)} \tag{3}$$

In case b is one, the asymptotic situation gives rise to the familiar formula for the expected successful search length $-\alpha^{-1}\log(1-\alpha)$. The average successful search length measuring the number of probes per record is given by the formula k/m; in Figure 1, the average successful search length is 1.4. The expected successful search length, calculated via (2) and (3), is given in Table 1.

3.2 Unsuccessful Search Length

Suppose L_u designates the number of buckets examined to conduct an unsuccessful search. The search termination condition depends upon whether passbits are used or not. Without passbits, the unsuccessful search terminates at an unfilled or open bucket; this condition provides the name for the general scheme. With passbits, the search terminates at a bucket with a false passbit.

The probability a bucket is filled (and the passbit is true if present) is q; we calculate q subsequently. The probability an unsuccessful search requires at least ξ probes is

$$\text{Prob}\,\{\,L_u \;\geq\; \xi\,\} \;=\; q^{\xi-1} \qquad \xi \geq 1 \tag{4}$$

and the expected unsuccessful search length is

$$E[L_u] \;=\; \sum_{\xi \geq 1} q^{\xi-1} \;=\; \frac{1}{1-q}. \tag{5}$$

Since $\hat{p} = 1 - q$ designates the probability the bucket is not filled or the passbit is false (if present), we calculate this value directly.

The average unsuccessful search length can be determined by calculating the length for each variety of unsuccessful search; that is, by considering a unsuccessful search to begin at each bucket and have every possible step size. The configuration of Figure 1 has an average unsuccessful search length of 1.2. Table 1 contains the expected unsuccessful search length calculated via (5) using our results of Section 3.3.

k	α	u.s.	s.s.
10	0.613	1.07	1.09
11	0.658	1.10	1.11
12	0.699	1.14	1.14
13	0.737	1.18	1.18
14	0.770	1.23	1.21

Table 1: Expected unsuccessful u.s. and successful s.s. search lengths for Figure 1 configuration.

3.3 Results

We present our general result for the unsuccessful search length within multiple passbit double hashing. Our results hold for any passbit; however, we will designate the passbits with indicies 1, 2, $\cdots$, g throughout our analysis.

THEOREM: *The probability p a passbit is false within a bucket is*

$$\hat{p} = \begin{cases} \displaystyle\sum_{\substack{j_1+j_2+\cdots+j_g \\ =\,0}}^{b} P_{j_1,j_2,\ldots,j_g} + \sum_{j_g=0}^{b} \sum_{\substack{j_1+j_2+\cdots+j_{g-1} \\ =\,b-j_g+1}}^{k-j_g} P_{j_1,j_2,\ldots,j_g} \binom{b}{j_g} \Big/ \binom{j_1+j_2+\cdots+j_g}{j_g} & k > b \\ 1 & \textit{otherwise} \end{cases} \tag{6}$$

where g, the number of passbits per bucket, is at least two, k is the number of probes, b is the number of slots per bucket, and p, the probability of accessing a bucket, is $1/n$ where n is the number of buckets and $P_{j_1, j_2, ,\ldots, j_g}$ designates the probability that j_i variety i probes access the bucket for $1 \leq i \leq g$

$$P_{j_1, j_2, ,\ldots, j_g} = \binom{k}{j_1\ j_2\ \cdots\ j_g} \left(\frac{p}{g}\right)^{j_1+j_2+\cdots+j_g} \left(1-p\right)^{k-j_1-j_2-\cdots-j_g}.$$

In case g is one, one passbit per bucket, the probability $\hat{p}$ is

$$\hat{p} = \begin{cases} \sum_{j=0}^{b} P_j & k > b \\ 1 & otherwise \end{cases}.$$

If no passbits are present, the probability $\hat{p}$ is

$$\hat{p} = \begin{cases} \sum_{j=0}^{b-1} P_j & k \geq b \\ 1 & otherwise \end{cases}. \qquad \blacksquare$$

This result is verified within the Appendix.

A less apparent phenomenon within the $\hat{p}$ expressions (6); for constant g, the expected unsuccessful search length for larger b configurations will exceed that of smaller b configurations for loading factors greater than approximately 0.85. This is intuitively clear; with more records per bucket, the net effect per passbit is diminished as each becomes more likely set true. This behavior are present within both our experimental data and our model.

4 Experimental and Analytical Results

We consider several configurations from simpler to more intricate in determining q the probability an unsuccessful search continues beyond a particular bucket. For each configuration, we present experimental data together with expected values in graphical form. The experimental data is derived from a table consisting of 131 buckets each with $b = 1, 2, 3, 4, 8, 16, 32, 64, 128$ and 256 slots; each data point is the average of 10,000 instances of the configuration. Expressions in closed-form for the expected unsuccessful search lengths will the derived for a few cases. Our results are presented within the graphs of Figures 2 through 11; each curve is for a particular bucket size. For small α, larger b produces smaller unsuccessful search lengths, while for large α the opposite is true. This observation allows us to avoid cluttering the graphs with individual labels for the curves; however, the crossing points are most visible within the graphs for large numbers of passbits.

Our analysis assumes independence of the initial bucket, step size, and the passbit. We calculate these parameters as follows assuming the hash value for a record is *value*. The initial bucket accessed b is

$$b \quad = \quad value\ \%\ n$$

where n is the number of buckets within the table, the step size s is

$$s \quad = \quad value\ \%\ (n-1)\ +\ 1$$

and

$$p \quad = \quad (\ value\ +\ value/g\ )\ \%\ g$$

is the passbit index p. Accordingly, an interval of $n\,(n-1)\,g^2$ consecutive (hash) values contains identical numbers of occurrences of each bucket, step, and passbit combination (b, s, p) since the least common multiple of n, $n-1$, and g^2 divides $n\,(n-1)\,g^2$. Thus, we have that each bucket, step, and passbit combination occurs with approximate probability $1/(n\,(n-1)\,g)$; the probability is approximate only because we make no assumption about the size of the hash value domain other than it be large.

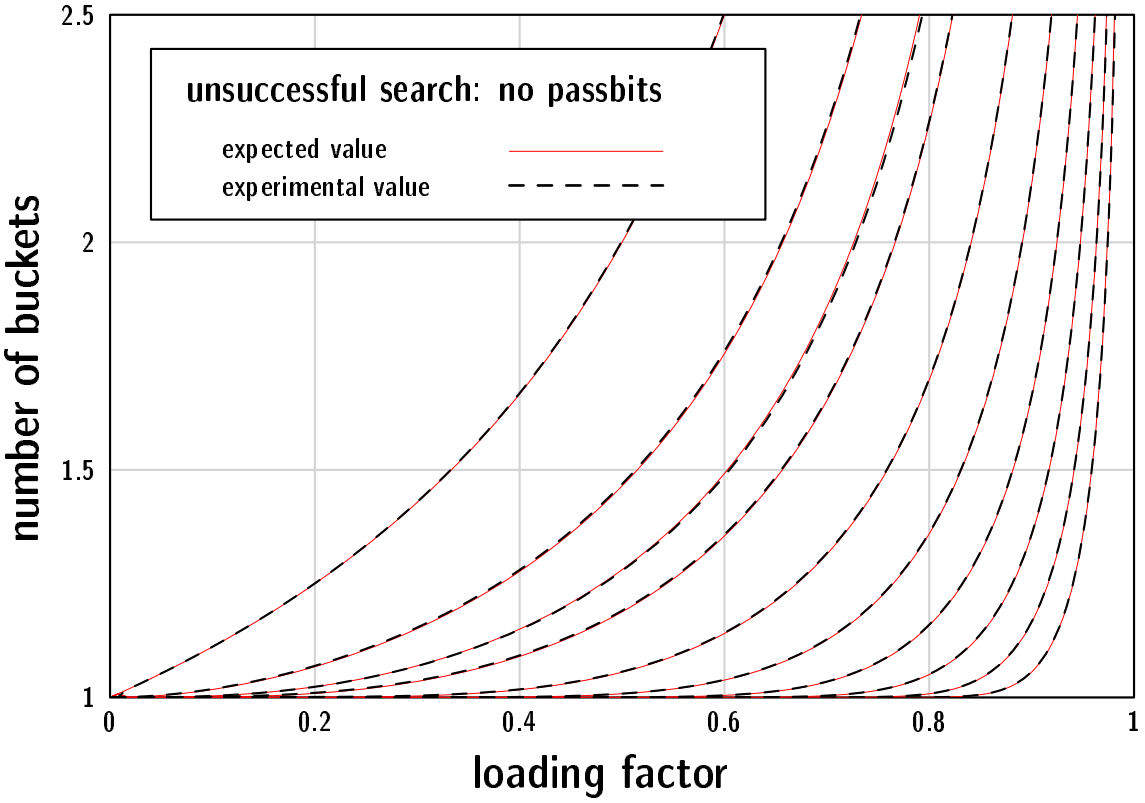

Figure 2: Unsuccessful search lengths with no passbits: $b = 1, 2, 3, 4, 8, 16, 32, 64, 128, 256$.

4.1 No passbits

We begin without passbits, that is ordinary double hashing with b slots per bucket. We calculate the probability $\hat{p}$ that a bucket is not filled, for $k > b$,

$$\hat{p} = P_0 + P_1 + \cdots + P_{b-1}$$

Then we have $q = 1 - \hat{p}$ and

$$E[L_u] \;=\; \frac{1}{1-q} \;=\; \frac{1}{\hat{p}}.$$

Here we have

$$\hat{p} \;=\; \sum_{i=0}^{b-1} \binom{k}{i} \left(\frac{1}{n}\right)^i \left(1 - \frac{1}{n}\right)^{k-i}.$$

Figure 2 presents a graph containing both experimental data together with expected values.

In the asymptotic situation, we have

$$\hat{p} \;=\; e^{-\lambda} \sum_{i=0}^{b-1} \frac{\lambda^i}{i\,!}$$

In case b is one, we have $\hat{p} = e^{-\lambda}$ and since the probing factor λ is $-\log(1-\alpha)$, we obtain the familiar $E[L_u] \;=\; 1/(1-\alpha)$.

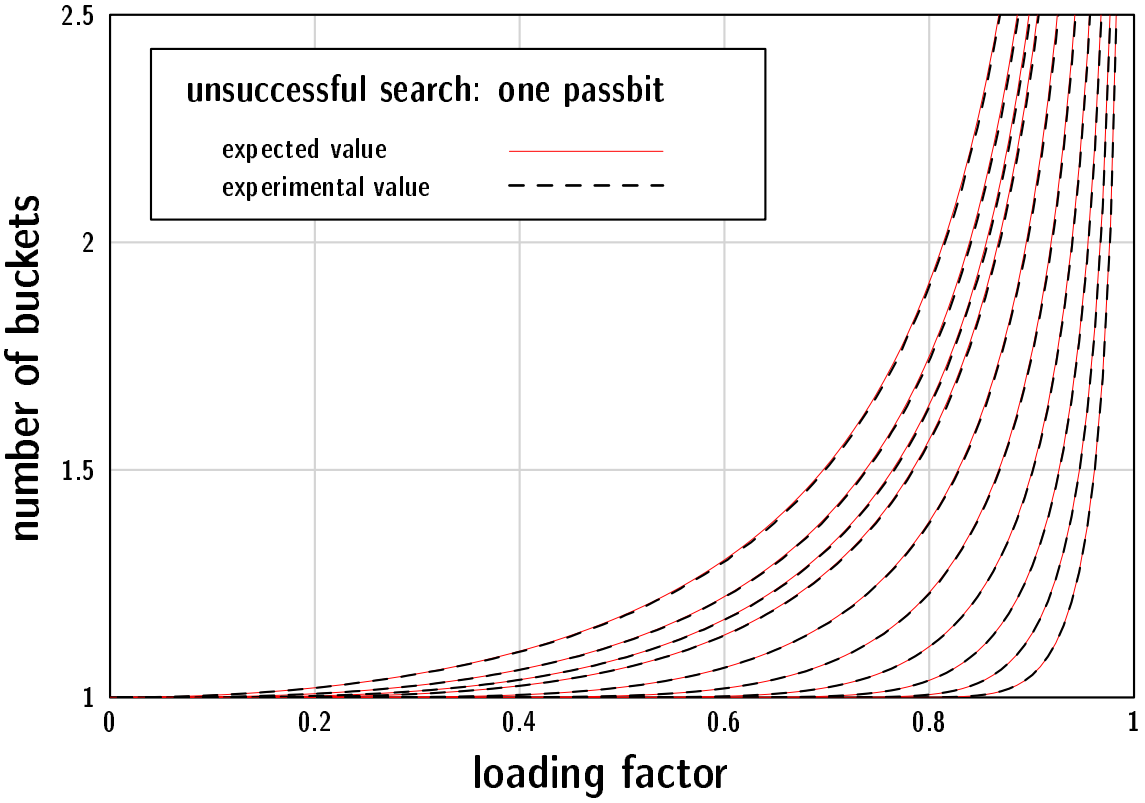

Figure 3: Unsuccessful search lengths with one passbit per bucket: $b = 1, 2, 3, 4, 8, 16, 32, 64, 128, 256$

4.2 Single Passbit

Next we consider a single passbit per bucket; the analysis of this configuration is similar to the previous analysis for no passbits. We calculate the probability $\hat{p}$ that a bucket is not filled or it is filled but its passbit is false; in this situation, at most b probes can be directed to the bucket, for $k > b$,

$$\hat{p} = P_0 + P_1 + \cdots + P_{b-1} + P_b$$

Since $E[L_u] = 1/\hat{p}$, we immediately obtain an improved unsuccessful search length due to the extra term P_b in the denominator. Figure 3 contains the experimental data as well as the expected values for this configuration.

In the asymptotic situation,

$$\hat{p} = e^{-\lambda} \sum_{i=0}^{b} \frac{\lambda^i}{i!}$$

In case b is one, $\hat{p} = P_0 + P_1 = e^{-\lambda} + \lambda e^{-\lambda}$ in the asymptotic situation and since $\lambda = -\log(1-\alpha)$, we have $\hat{p} = (1-\alpha)(1-\log(1-\alpha))$. Finally,

$$E[L_u] = \frac{1}{(1-\alpha)(1-\log(1-\alpha))} \tag{7}$$

an expression presented by Larson [6] who references Gunji [5] in this context.

4.3 Pair of Passbits

Now we consider a pair of passbits pb_1 and pb_2 per bucket. In this situation, we calculate p the probability that passbit pb_2 is false. The probability P_{j_1,j_2} that j_1 probes of variety 1 and j_2 probes of variety 2 access

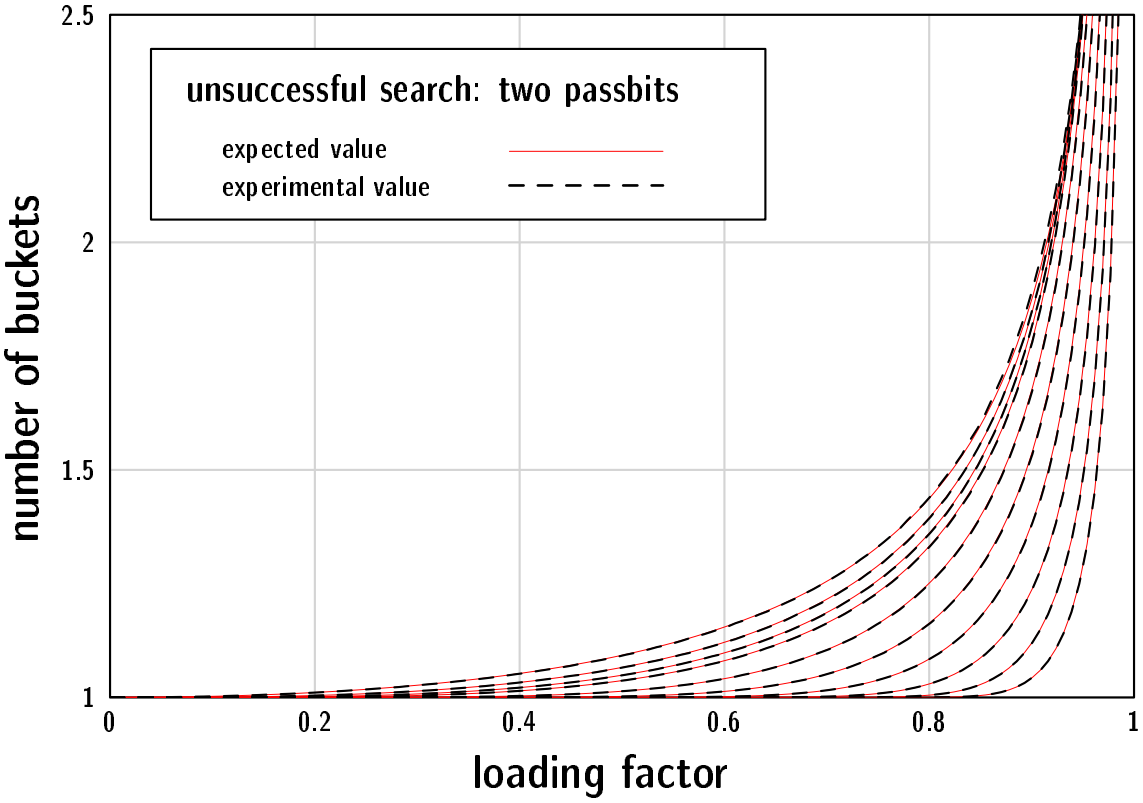

Figure 4: Unsuccessful search lengths with two passbits per bucket: $b = 1, 2, 3, 4, 8, 16, 32, 64, 128, 256$.

a bucket is given by

$$P_{j_1,j_2} = \binom{k}{j_1, j_2} \left(\frac{1}{2\,n}\right)^{j_1+j_2} \left(1 - \frac{1}{n}\right)^{k-(j_1+j_2)} \qquad j_1, j_2 \geq 0 \tag{8}$$

for k probes to construct the configuration; we utilize the uniform hashing assumption to evenly divide the probability of each variety of probe. Then $\hat{p}$ is derived from the two events "$pb_1 = \text{false} \wedge pb_2 = \text{false}$" and "$pb_1 = \text{true} \wedge pb_2 = \text{false}$" in which j_2 is no greater than b and the j_2 variety 2 probes must be among the first b of the k-sequence to access the bucket

$$\hat{p} = \sum_{j_1+j_2 \geq 0}^{min\,(\,b,\,k\,)} P_{j_1,j_2} + \sum_{j_2=0}^{b} \sum_{j_1=b+1-j_2}^{k-j_2} P_{j_1,j_2} \frac{\binom{b}{j_2}}{\binom{j_1+j_2}{j_2}}. \tag{9}$$

The double summation on the right captures "$pb_1 = \text{true} \wedge pb_2 = \text{false}$" and the single summation captures "$pb_1 = \text{false} \wedge pb_2 = \text{false}$." The ratio containing binomial coefficients measures the fraction of the sequences of length $j_1 + j_2$ containing j_2 probes of variety 2 within the first b probes; this ensures $pb_2 = \text{false}$. Figure 4 presents graphically the experimental and expected values.

In case b is one, we can obtain a closed-form expression for p. The two portions of the expression are presented; the first three terms capture pb_1 and pb_2 both false and the two sums capture pb_1 true and pb_2 false, for $k > 1$,

$$\hat{p} = P_{0,0} + P_{0,1} + P_{1,0} + \sum_{j \geq 2}^{k} P_{j,0} + \sum_{j \geq 1}^{k-1} P_{j,1} \frac{1}{j+1}.$$

Finally, we obtain

$$\hat{p} = 2\left(1 - \frac{1}{2\,n}\right)^k - \left(1 - \frac{1}{n}\right)^k.$$

In the asymptotic situation, we obtain $\hat{p} = 2e^{-\lambda/2} - e^{-\lambda}$. Since λ is $-\log(1-\alpha)$, we finally obtain

$$E[\,L_u\,] \;=\; \frac{1}{2\sqrt{1-\alpha} \;-\; (1-\alpha)}. \tag{10}$$

In case b is two, we can obtain an expression for $\hat{p}$ as well.

$$\hat{p} \;=\; P_{0,0} + 2P_{1,0} + 2P_{2,0} + P_{1,1} + \sum_{j\,\geq\,3}^{k} P_{j,0} + \sum_{j\,\geq\,2}^{k-1} P_{j,1}\frac{2}{j+1} + \sum_{j\,\geq\,1}^{k\,-\,2} P_{j,2}\frac{1}{\binom{j+2}{2}}.$$

Finally the asymptotic situation obtains

$$\hat{p} \;=\; e^{-\lambda}\,(\,4e^{\lambda/2} - 3 - \lambda\,)$$

and equation (2) provides the associated α values.

4.4 Four through 256 Passbits

We present our experimental and expected value results in figures 5, 6, 7, ..., 11; the y-axis scaling is magnified with increased passbit counts. In these configurations, we have no closed-form expressions in case b is one. However, the structure of $\hat{p}$ is similar to that presented for two passbits in the previous section and the details are present within the appendix.

The *crossing* phenomema can be observed in these graphs; for small load factors, larger numbers of slots per bucket produce smaller unsuccessful search lengths while for large load factors (0.85 or so) smaller numbers of slots per bucket obtain smaller unsuccessful search lengths.

5 Expected Value Calculations

We provide an effective $\hat{p}$ calculation scheme; without some care in this regard, the calculation of the nine curves of one graph with more than one or two passbits, will take several *days* on a dedicated 233 Mhz processor. As we note within the appendix, $\hat{p}$ can be expressed as the "tail" of a binomial distribution or as the sum of a pair of such tails.

We write

$$B\,(\,k;\,n,\,p\,) \;\;=\;\; \sum_{j\,=\,0}^{k} \binom{n}{j} p^j\,(\,1\,-\,p\,)^{n-j} \tag{11}$$

and note that

$$B\,(\,k;\,n,\,0\,) \;\;=\;\; 1\,.$$

Now we have

$$\frac{\partial B}{\partial p} \;=\; -n\binom{n-1}{k} p^k\,(\,1\,-\,p\,)^{n-k-1} \;=\; -\,(\,n\,-\,k\,)\binom{n}{k} t^k\,(\,1\,-\,t\,)^{n-k-1}$$

since the sum telescopes to a single term. Thus we have

$$\int_0^p \frac{\partial B}{\partial p}\,dp \;=\; B\,(\,k;\,n,\,p\,) \;-\; B\,(\,k;\,n,\,0) \;=\; -\,(\,n\,-\,k\,)\binom{n}{k} \int_0^p t^k\,(\,1\,-\,t\,)^{n-k-1}dt$$

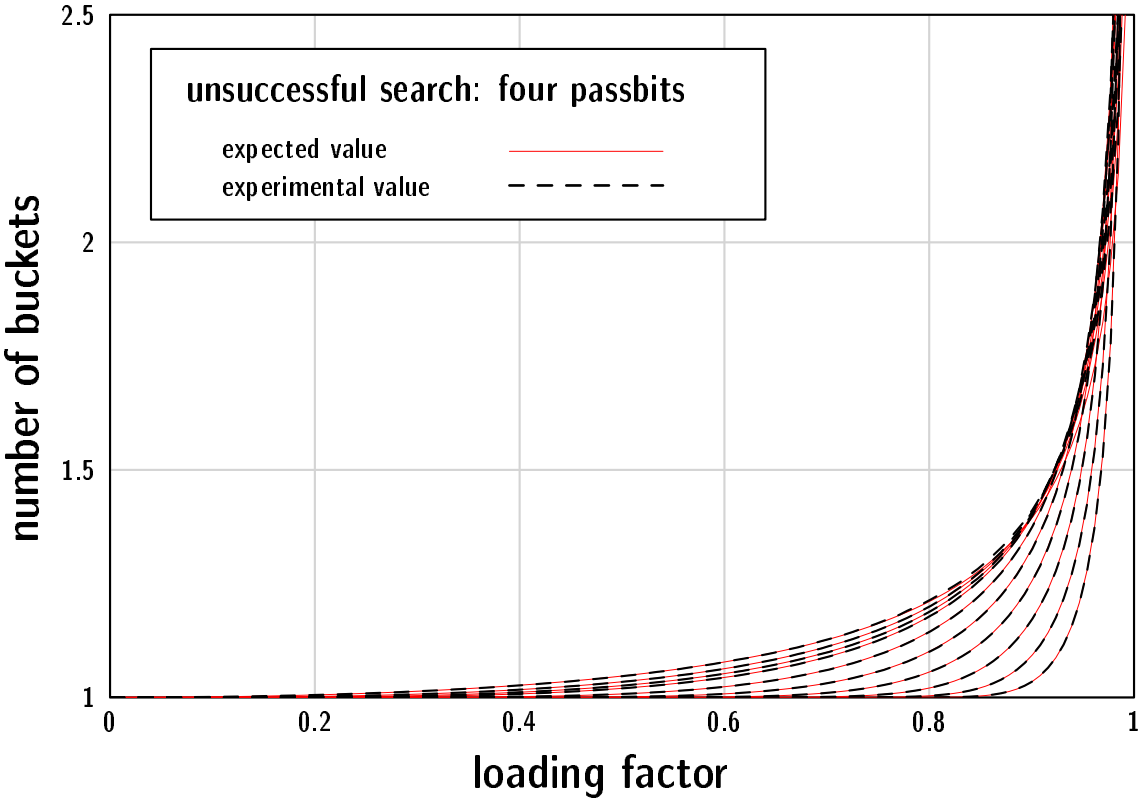

Figure 5: Unsuccessful search lengths with four passbits per bucket: $b = 1, 2, 3, 4, 8, 16, 32, 64, 128, 256$.

Finally we have

$$B\,(k;\,n,\,p\,) \quad = \quad 1 \;-\; (n-k)\binom{n}{k}\int_0^p t^k\,(1\,-\,t\,)^{n-k-1}\,dt \tag{12}$$

and by changing variables (t replaced by $1-t$) and $q = 1-p$

$$= \quad (\,n\,-\,k\,)\binom{n}{k}\int_0^q t^{n-k-1}\,(1\,-\,t\,)^k\,dt. \tag{13}$$

Since $B\,(k;\,n,\,0\,) = 1$, we have

$$\int_0^1 t^{n-k-1}\,(1\,-\,t\,)^k\,dt \;=\; (\,n\,-\,k\,)^{-1}\binom{n}{k}^{-1}.$$

Accordingly

$$B\,(k;\,n,\,p\,) \;=\; \int_0^{1-p} t^{n-k-1}\,(1\,-\,t\,)^k\,dt \;\Big/\; \int_0^1 t^{n-k-1}\,(1\,-\,t\,)^k\,dt$$

which has the structure of the incomplete beta function ratio [2, 9].

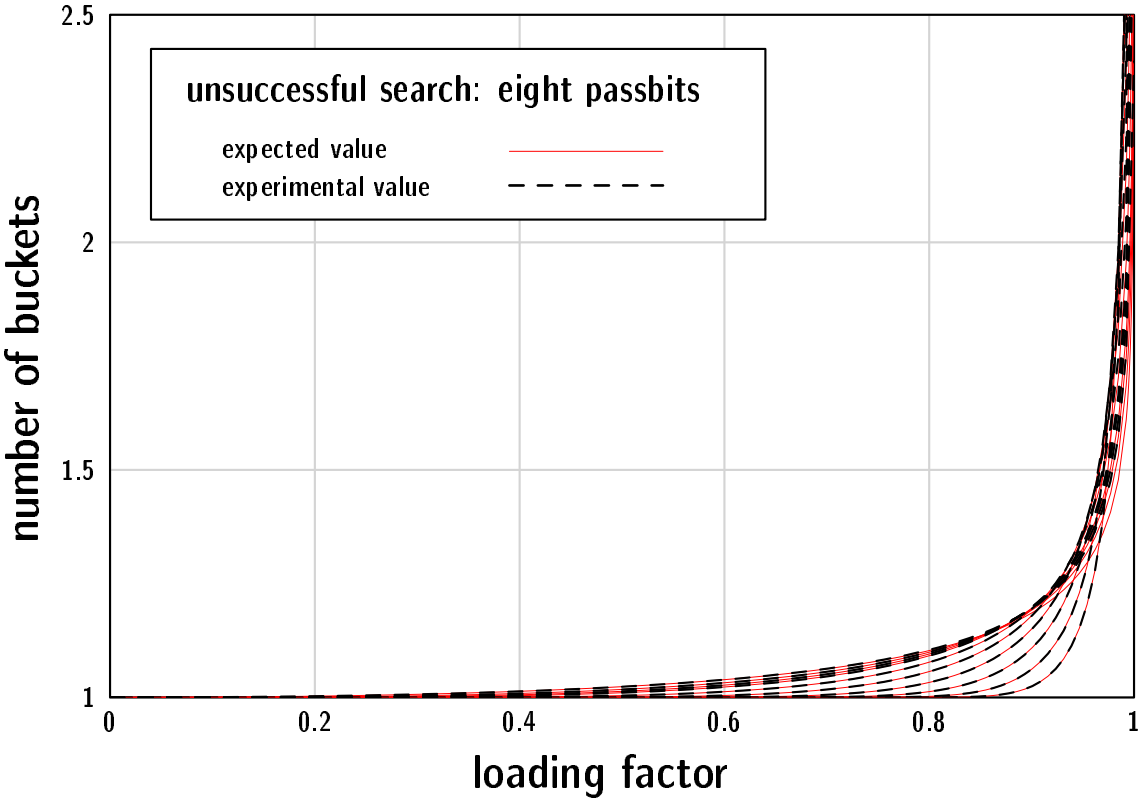

Figure 6: Unsuccessful search lengths with eight passbits per bucket: $b = 1, 2, 3, 4, 8, 16, 32, 64, 128, 256$

The *incomplete beta function ratio* $I_x(a,b)$ is defined as follows for $a, b > 0$ and $0 \leq x \leq 1$

$$I_x(a,b) \;=\; \int_0^x t^{a-1}(1-t)^{b-1}\,dt \;\Big/\; \int_0^1 t^{a-1}(1-t)^{b-1}\,dt.$$

The following boundary values are useful, $I_0(a\,,\,b) = 0$ and $I_1(a\,,\,b) = 1$, as is the symmetry relation

$$I_x(a\,,\,b) = 1 - I_{1-x}(b\,,\,a).$$

The cumulative binomial probability is related to the incomplete beta function ratio as we present above as follows

$$\sum_{j=b}^{k} \binom{k}{j} p^j (1-p)^{k-j} \;=\; I_p(b\,,\,k-b+1).$$

And using the symmetry relation, we have

$$\sum_{j=0}^{b} \binom{k}{j} p^j (1-p)^{k-j} \;=\; 1 - I_p(b+1\,,\,k-b) \;=\; I_{1-p}(k-b\,,\,b+1).$$

For large k values, calculation of I_x is much more efficient than calculating each of the binomial coefficients etc. of the sum [9].

In case there are no passbits, we have

$$\hat{p} \;=\; \sum_{j=0}^{b-1} P_j \;=\; I_{1-p}(k-b-1\,,\,b)$$

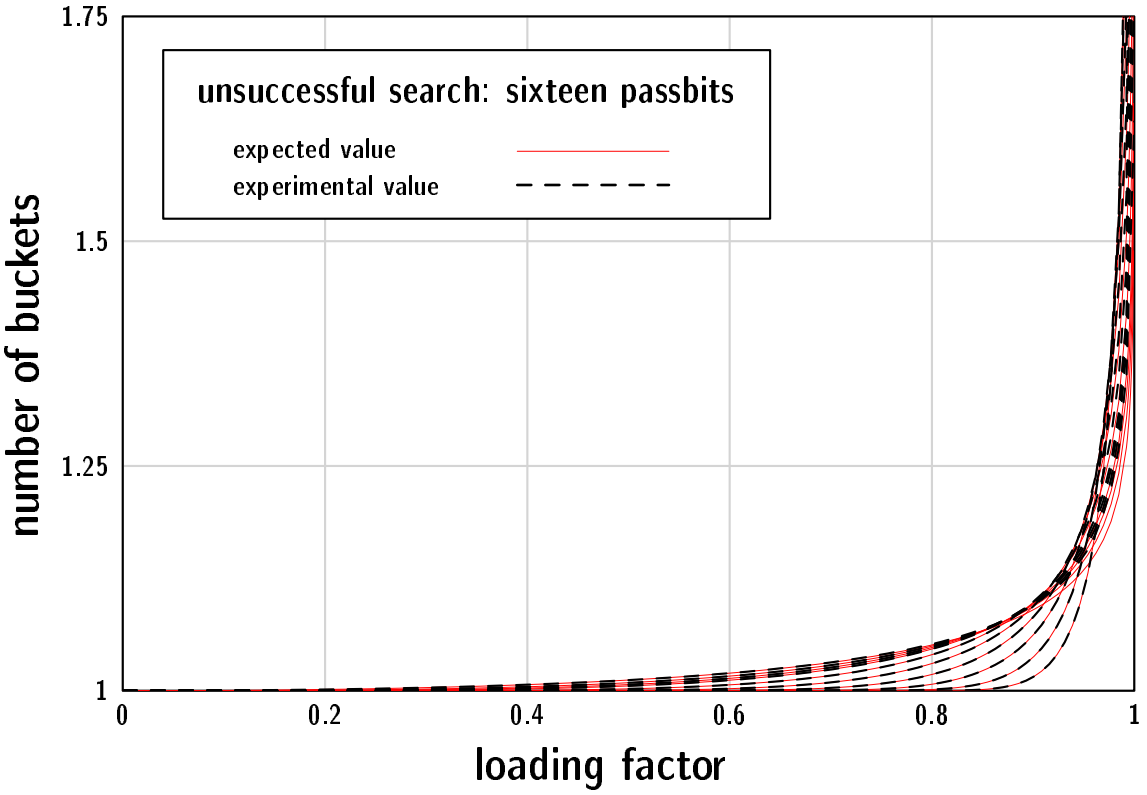

Figure 7: Unsuccessful search lengths with sixteen passbits per bucket: $b = 1, 2, 3, 4, 8, 16, 32, 64, 128, 256$

and in case g is one, we have

$$\hat{p} \;=\; \sum_{j\,=\,0}^{b} P_j \;=\; I_{1-p}\,(\,k-b\,,\,b+1\,)$$

For g greater than one, we show within the appendix that $\hat{p} = P_{\mathcal{A}} + P_{\mathcal{B}}$ and both terms are binomial distribution tails.

$$P_{\mathcal{A}} \;=\; \sum_{0}^{b} P_j \;=\; I_{1-p}(\,k-b,\,b+1\,) \tag{14}$$

and

$$\begin{aligned} P_{\mathcal{B}} \;&=\; \left(\frac{g}{g-1}\right)^{b} \left(1-\frac{1}{n\,g}\right)^{k} \sum_{j\,=\,b+1}^{k} \binom{k}{j} \left(\frac{g-1}{n\,g-1}\right)^{j} \left(\frac{g(n-1)}{n\,g-1}\right)^{k-j} \\ &=\; \left(\frac{g}{g-1}\right)^{b} \left(1-\frac{1}{n\,g}\right)^{k} I_{pg}\,(\,b+1\,,\,k-b\,). \end{aligned} \tag{15}$$

$$\hat{p} \;=\; P_{\mathcal{A}} + P_{\mathcal{B}} \;=\; \left(\frac{g}{g-1}\right)^{b} \left(1-\frac{1}{n\,g}\right)^{k} I_{pg}\,(\,b+1\,,\,k-b\,) \;+\; I_{1-p}\,(\,k-b\,,\,b+1\,) \tag{16}$$

where pg is $(g-1)/(n\,g-1)$ and p is $1/n$.

6 Appendix

We invoke uniform hashing to note that an event is equally likely to occur at all buckets. If there are $k \leq b$ probes then all passbits must be false and $\hat{p}$ is one. In case g is one and $k > b$, then at most b probes can

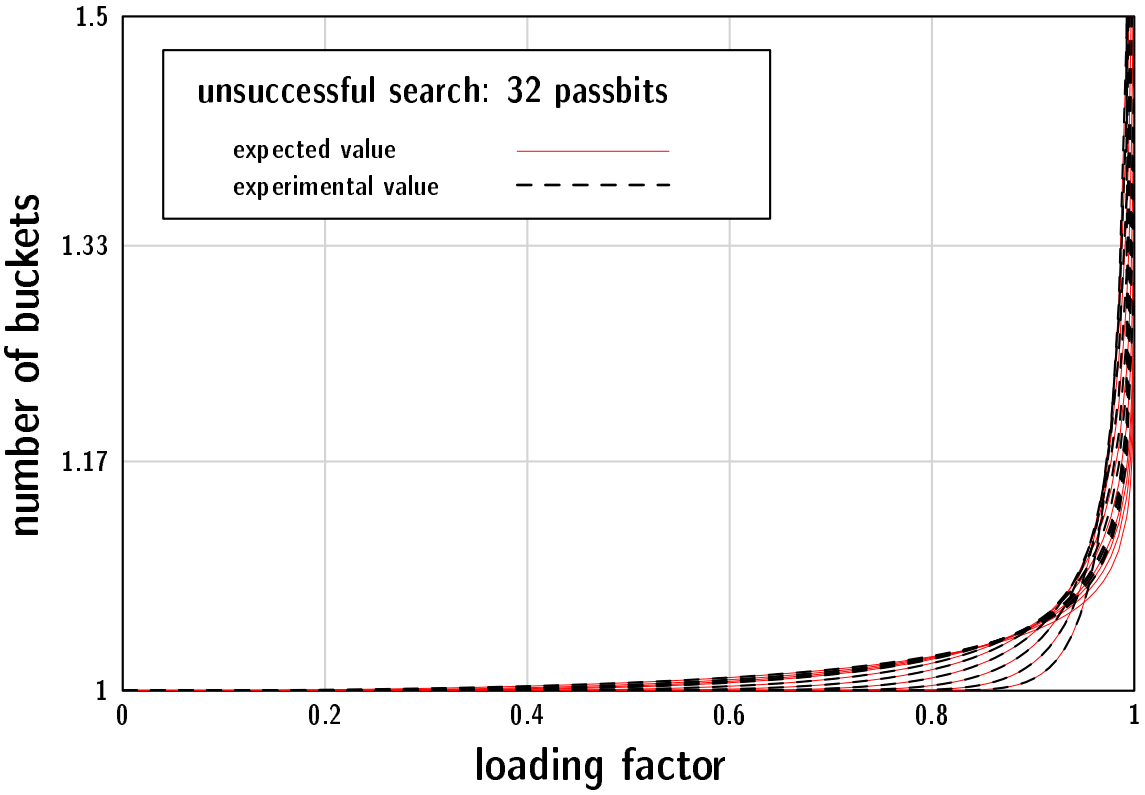

Figure 8: Unsuccessful search lengths with thirty-two passbits per bucket: $b = 1, 2, 3, 4, 8, 16, 32, 64, 128, 256$

access a bucket if its passbit is to remain false; the expression $\sum_{j=0}^{b} P_j$ measures the probability that passbit pb_1 is false.

We consider the cases g equal to zero and one together as these configurations are very similar. In case g is zero, we have, for $k \geq b$

$$\hat{p} = \sum_{j=0}^{b-1} P_j. \tag{17}$$

Thus $\hat{p}$ measures the probability a bucket is not filled. In case g is one, we have, for $k > b$

$$\hat{p} = \sum_{j=0}^{b} P_j. \tag{18}$$

Here $\hat{p}$ measures the probability a bucket is either empty or filled without overflow. The g is one result has been presented by Larson [6] (and Furukawa [3] check this!).

The $\hat{p}$ expression, for g greater than one, is more intricate. We calculate the probability that passbit pb_1 is false. The expression is derived by considering two varieties of events. The first event $\mathcal{A}$ gives rise to all g passbits pb_1, pb_2, ..., pb_g each being false; this is similar to the g equals one case. The second event $\mathcal{B}$ gives rise to pb_1 false while the expression $pb_2 \vee pb_3 \vee \cdots \vee pg_g$ is true; that is, at least one of these $g-1$ passbits must be true. The probability density function, generalizing (1), $P_{j_1, j_2, \ldots, j_g}$ designates the probability that j_i variety i probes access the bucket for $1 \leq i \leq g$

$$P_{j_1, j_2, \ldots, j_g} = \binom{k}{j_1\ j_2\ \cdots\ j_g} \left(\frac{1}{n\,g}\right)^{j_1+j_2+\cdots+j_g} \left(1 - \frac{1}{n}\right)^{k-j_1-j_2-\cdots-j_g}.$$

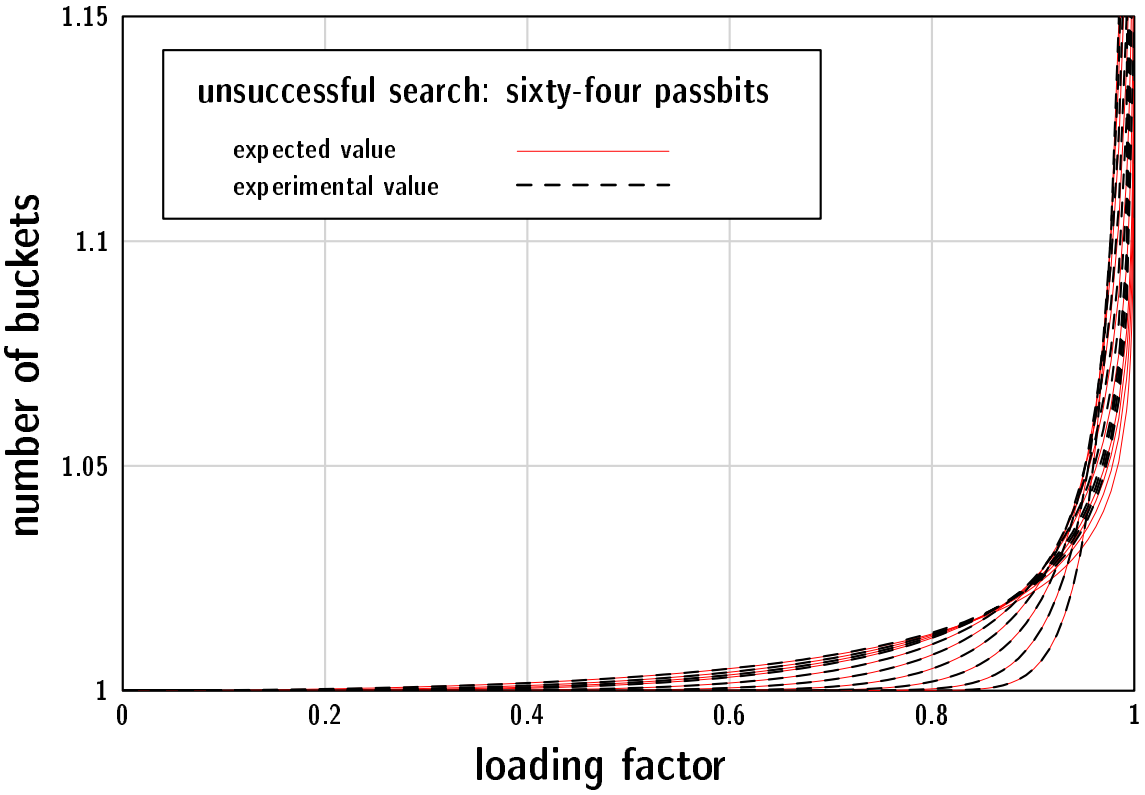

Figure 9: Unsuccessful search lengths with sixty-four passbits per bucket: $b = 1, 2, 3, 4, 8, 16, 32, 64, 128, 256$

Then the probability of event $\mathcal{A}$ is

$$P_{\mathcal{A}} = \sum_{\substack{j_1+j_2+\cdots+j_g \\ = 0}}^{b} P_{j_1, j_2, \ldots, j_g} \tag{19}$$

where the sum ranges over all g-compositions of zero through b.

The probability of event $\mathcal{B}$, designated $P_{\mathcal{B}}$, in which pb_1 is false and the expression $pb_2 \vee pb_3 \vee \cdots \vee pb_g$ is true, is measured with a "double" sum. The leftmost sum ensures that at most b variety 1 probes access the bucket. The remaining probes to the bucket are free to be of any other variety. The rightmost sum ensures that the total number of probes to the bucket is between $b+1$ and k thereby obtaining at least one true passbit. The probability of event B is

$$P_{\mathcal{B}} = \sum_{j_1=0}^{b} \sum_{\substack{j_2+j_3+\cdots+j_g \\ = b+1-j_1}}^{k-j_1} P_{j_1, j_2, \ldots, j_g} \binom{b}{j_1} \Big/ \binom{j_1 + j_2 + \cdots + j_g}{j_1} \tag{20}$$

where the rightmost sum ranges over all g–1 - compositions of $b+1-j_1$ through $k-j_1$. The ratio of binomial coefficients captures the fraction of these k-sequences in which *all* of the j_1 variety 1 probes are within the first b of the $j_1 + j_2 + \cdots + j_g$ probes to the bucket. If this condition does not hold, then pb_1 will be true.

There are

$$\binom{k}{j_1\, j_2\, \cdots\, j_g}$$

possible k-sequences. Similarly, there are

$$\binom{b}{j_1} \binom{j_2 + j_3 + \cdots + j_g}{j_2\, j_3\, \cdots\, j_g} \binom{k}{j_1 + j_2 + \cdots + j_g}$$

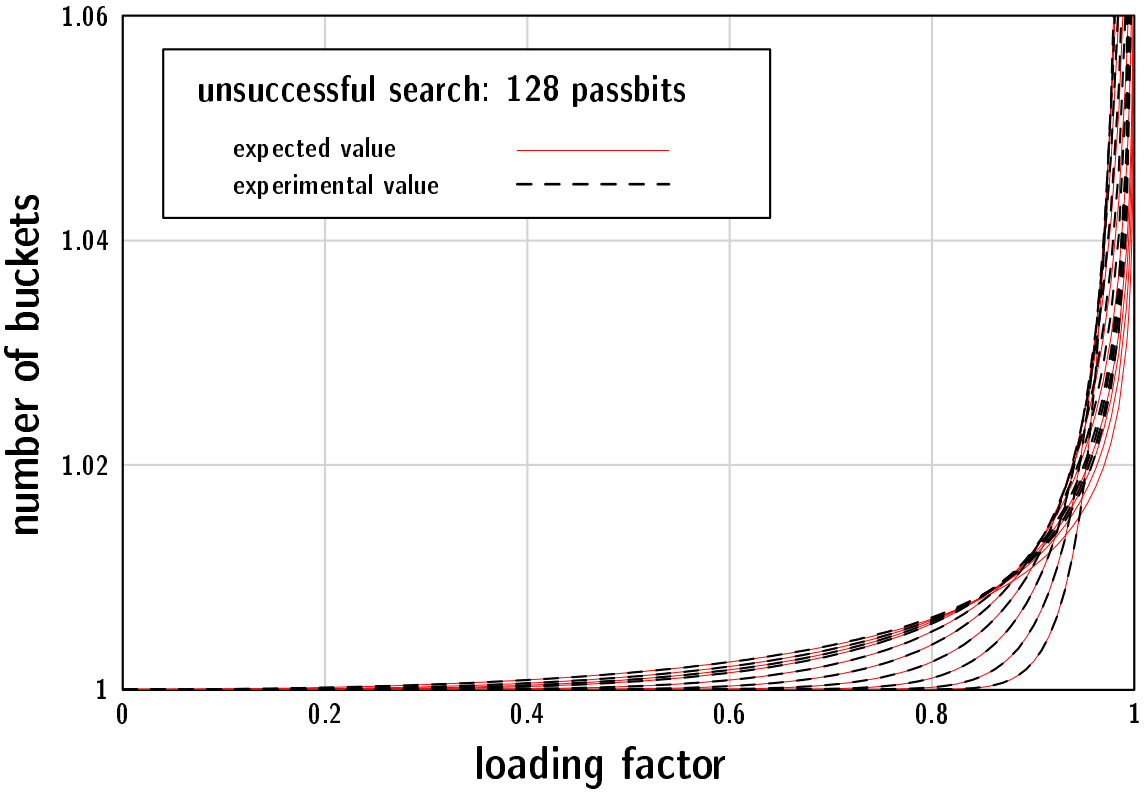

Figure 10: Unsuccessful search lengths with one hundred twenty eight passbits per bucket: $b = 1, 2, 3, 4, 8, 16, 32, 64, 128, 256$

k-sequences in which all of the variety g probes are among the first b of the $j_1 + j_2 + \cdots + j_g$ probes. This is true since once the positions among the first b are choosen for the variety g probes, the $j_2 + j_3 + \cdots + j_g$ probes within the sequence are obtained by suitable translations based upon the j_1 variety 1 positions; this multinomial coefficient counts the number of such sequences. Finally, the rightmost binomial coefficient counts the number of configurations in which to include the $j_1 + j_2 + \cdots + j_g$ probes to create the k-sequence. Thus the fraction of k-sequences in which all of the j_1 variety 1 probes within the first b of the $j_1 + j_2 + \cdots + j_g$ is

$$\frac{\binom{b}{j_1}\binom{j_2 + j_3 + \cdots + j_g}{j_2\ j_3\ \cdots\ j_g}\binom{k}{j_1 + j_2 + \cdots + j_g}}{\binom{k}{j_1\ j_2\ \cdots\ j_g}} = \binom{b}{j_1} \Big/ \binom{j_1 + j_2 + \cdots + j_g}{j_1}$$

as presented within the $P_{\mathcal{B}}$ expression.

Since events $\mathcal{A}$ and $\mathcal{B}$ are disjoint, $\hat{p}$ is $P_{\mathcal{A}} + P_{\mathcal{B}}$. We simplify the two expressions to obtain the result. Event $\mathcal{A}$ allows at most b probes and no bucket overflows. Accordingly we might expect

$$P_{\mathcal{A}} \ = \ \sum_{j=0}^{b} P_j$$

where P_j is specified within (1) since the variety of probes is of no consequence, only the total probe count per bucket. Indeed, this is the case as we show here. The multinomial theorem states

$$(\, x_0 + x_1 + x_2 + \cdots + x_g \,)^k = \sum_{\substack{j_1 + j_2 + \cdots + j_g \\ = \ 0}}^{k} \binom{k}{j_1 j_2 \cdots j_g} x_1^{j_1}\, x_2^{j_2} \cdots x_g^{j_g}\, x_0^{k - j_1 - j_2 - \cdots - j_g}$$

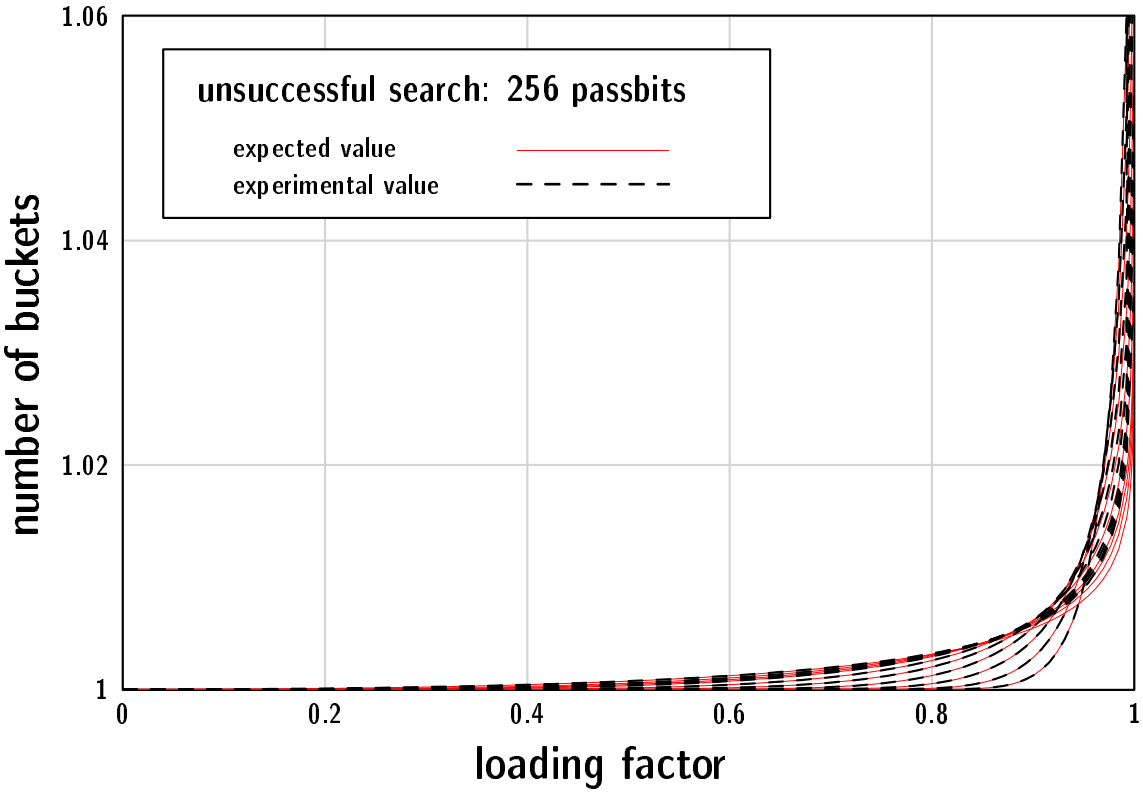

Figure 11: Unsuccessful search lengths with 256 passbits per bucket: $b = 1, 2, 3, 4, 8, 16, 32, 64, 128, 256$

where the sum is over all g-compositions of 0 through k. We have, using the binomial theorem as well,

$$(x_0 + g)^k = \sum_{i=0}^{k} \binom{k}{i} g^i x_0^{k-i} = \sum_{\substack{j_1+j_2+\cdots+j_g \\ =\ 0}}^{k} \binom{k}{j_1 j_2 \cdots j_g} x_0^{k-j_1-j_2-\cdots-j_g}$$

where i is $j_1 + j_2 + \cdots + j_g$. Thus we obtain

$$\binom{k}{i} g^i = \sum_{\substack{j_1+j_2+\cdots+j_g \\ =\ i}} \binom{k}{j_1 j_2 \cdots j_g} \tag{21}$$

with $j_1 + j_2 + \cdots + j_g$ ranging over all g-compositions of i. We could derive this combinatorially as well with k labelled balls and $g+1$ labelled cells. We use (21) to simplify the $P_{\mathcal{A}}$ expression (19) where p is $1/n$.

$$P_{\mathcal{A}} = \sum_{\substack{j_1+j_2+\cdots+j_g \\ =\ 0}}^{b} P_{j_1,j_2,\ldots,j_g} = \sum_{\substack{j_1+j_2+\cdots+j_g \\ =\ 0}}^{b} \binom{k}{j_1 j_2 \cdots j_g} \left(\frac{1}{ng}\right)^{j_1+j_2+\cdots+j_g} \left(1-\frac{1}{n}\right)^{k-j_1-j_2-\cdots-j_g}$$

$$= \sum_{j=0}^{b} \binom{k}{j} g^j \left(\frac{1}{ng}\right)^j \left(1-\frac{1}{n}\right)^{k-j} = \sum_{j=0}^{b} P_j. \tag{22}$$

The $P_{\mathcal{B}}$ expression (20) can be simplified in a similar fashion; however, we must limit the range of j_1. Beginning again with the multinomial theorem, we have

$$(x_0 + x_1 + x_2 + \cdots + x_g)^k = \sum_{\substack{j_1+j_2+\cdots+j_g \\ =\ 0}}^{k} \binom{k}{j_1 j_2 \cdots j_g} x_1^{j_1} x_2^{j_2} \cdots x_g^{j_g} x_0^{k-j_1-j_2-\cdots-j_g}$$

$$= \sum_{j_1=0}^{k} x^{j_1} \binom{k}{j_1} \sum_{\substack{j_2+j_3+\cdots+j_g \\ =0}}^{k-j_1} \binom{k-j_1}{j_2 j_3 \cdots j_g} x_2^{j_2} x_3^{j_3} \cdots x_g^{j_g} x_0^{k-j_1-j_2-\cdots-j_g} \quad (23)$$

Similarly, we have (omitting x_1) using both the binomial and the multinomial theorems,

$$(\,x_0 + (g-1)\,)^{k-j_1} = \sum_{\substack{j_2+j_3+\cdots+j_g \\ =0}}^{k-j_1} \binom{k-j_1}{j_2 j_3 \cdots j_g} x_0^{k-j_1-j_2-\cdots-j_g} = \sum_{j=0}^{k-j_1} \binom{k-j_1}{j-j_1} (g-1)^{j-j_1} x_0^{k-j} \quad (24)$$

where $j = j_1 + j_2 + j_3 + \cdots + j_g$. Now, combining (23) and (24), to determine the coefficient of $x_1^{j_1} x_0^{k-j}$ within

$$(\,x_0 + x_1 + (g-1)\,)^k = \sum \binom{k}{j_1\ j-j_1} x_1^{j_1} (g-1)^{j-j_1} x_0^{k-j}$$

we have

$$\sum_{\substack{j_1+j_2+j_3+\cdots+j_g \\ =j}} \binom{k}{j_1 j_2 j_3 \cdots j_g} = \binom{k}{j_1} \binom{k-j_1}{j-j_1} (g-1)^{j-j_1} = \binom{k}{j} \binom{j}{j_1} (g-1)^{j-j_1}. \quad (25)$$

We simplify (20) using (25) and $j = j_1 + j_2 + \cdots + j_g$

$$\begin{aligned}
P_{\mathcal{B}} &= \sum_{j_1=0}^{b} \sum_{\substack{j_2+j_3+\cdots+j_g \\ =b+1-j_1}}^{k-j_1} \binom{k}{j_1 j_2 \cdots j_g} \left(\frac{1}{ng}\right)^{j_1+j_2+\cdots+j_g} \left(1-\frac{1}{n}\right)^{k-j_1-j_2-\cdots-j_g} \binom{b}{j_1} \Big/ \binom{j}{j_1} \\
&= \sum_{j_1=0}^{b} \sum_{j=b+1}^{k} \binom{k}{j} \binom{j}{j_1} (g-1)^{j-j_1} \left(\frac{1}{ng}\right)^{j} \left(1-\frac{1}{n}\right)^{k-j} \binom{b}{j_1} \Big/ \binom{j}{j_1} \\
&= \sum_{j_1=0}^{b} \binom{b}{j_1} \left(\frac{1}{g-1}\right)^{j_1} \sum_{j=b+1}^{k} \binom{k}{j} \left(\frac{g-1}{ng}\right)^{j} \left(1-\frac{1}{n}\right)^{k-j} \\
&= \left(\frac{g}{g-1}\right)^{b} \sum_{j=b+1}^{k} \binom{k}{j} \left(\frac{g-1}{ng}\right)^{j} \left(1-\frac{1}{n}\right)^{k-j} \\
&= \left(\frac{g}{g-1}\right)^{b} \left(1-\frac{1}{n\,g}\right)^{k} \sum_{j=b+1}^{k} \binom{k}{j} \left(\frac{g-1}{n\,g-1}\right)^{j} \left(\frac{g(n-1)}{n\,g-1}\right)^{k-j}
\end{aligned} \quad (26)$$

Thus we have $\hat{p}$ summing (26) and (22)

$$\hat{p} = P_{\mathcal{A}} + P_{\mathcal{B}} \quad (27)$$

consists of a pair of binomial distribution tail sums.

Acknowledgements

We gladly acknowledge discussions with Leonard Haff, Paul Larson, Richard Olshen, and Gill Williamson during the early stages of our investigation.

References

[1] O. Amble and D.E. Knuth. Ordered hash tables. *Computer Journal*, 17(2):135–142, 1974.

[2] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume I. John Wiley, 1968.

[3] K. Furukawa. Hash addressing with conflict flag. *Information Procedings of Japan*, 13:13–18, 1973.

[4] L.J. Guibas and E. Szemeridi. The analysis of double hashing. *Journal of Computer And System Sciences*, 16:226–274, 1978.

[5] T. Gunji. Analysis of hash addressing methods. Technical report TR-76-03, University of Tokyo, Tokyo, Japan, 1976.

[6] P.-Å. Larson. Analysis of uniform hashing. *Journal of the Association for Computing Machinery*, 30(4):805–819, October 1983.

[7] G.S. Lueker and M. Molodowitch. More analysis of double hashing. *Combinatorica*, 13(1):83–96, 1993.

[8] W.W. Peterson. Addressing for random-access storage. *IBM Journal of Research and Development*, 1(2):130–146, 1957.

[9] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B. P. Flannery. *Numerical Recipes in C, The Art of Scientific Computing*. Cambridge University Press, second edition, 1992.

[10] J.D. Ullman. A note on the efficiency of hash functions. *Journal of the Association for Computing Machinery*, 19(3):569–575, July 1972.

[11] A.C. Yao. Uniform hashing is optimal. *Journal of the Association for Computing Machinery*, 32(3):687–693, July 1985.