Distributional analysis of Robin Hood linear probing hashing with buckets

Alfredo Viola[1†]

[1]*Pedeciba Informática, Casilla de Correo 16120, Distrito 6, Montevideo, Uruguay.*
E-mail: viola@fing.edu.uy
Part of the work was done while the author was at
LIPN - CNRS UMR 7030. Université de Paris-Nord. 93430 Villetaneuse, France.

This paper presents the first distributional analysis of a linear probing hashing scheme with buckets of size b. The exact distribution of the cost of successful searches for a $b\alpha$-full table is obtained, and moments and asymptotic results are derived. With the use of the Poisson transform distributional results are also obtained for tables of size m and n elements. A key element in the analysis is the use of a new family of numbers that satisfies a recurrence resembling that of the Bernoulli numbers. These numbers may prove helpful in studying recurrences involving truncated generating functions, as well as in other problems related with buckets.

Keywords: Distributional Analysis, Hashing, Linear Probing, Buckets

1 Motivation

The simplest collision resolution scheme for open addressing hash tables with hash function $h(x)$ is *linear probing* (Gonnet and Baeza-Yates (1991); Knuth (1998a); Sedgewick (1998)), which uses the cyclic probe sequence $h(K), h(K)+1, \ldots m-1, 0, 1, \ldots, h(K)-1$, assuming the table slots are numbered from 0 to $m-1$. Linear probing works reasonably well for tables that are not too full, but as the load factor increases, its performance deteriorates rapidly. Its main application is to retrieve information in secondary storage devices when the load factor is not too high, as first proposed by Peterson (Peterson (1957)). One reason for the use of linear probing is that it preserves locality of reference between successive probes, thus avoiding long seeks (Larson (1983)).

For each element x that gets placed at some location y, the circular distance between y and $h(x)$ (that is, $y-h(x)$ if $h(x) \leq y$, and $m+h(x)-y$ otherwise) is called its *displacement*. Displacement is both a measure of the cost of inserting x and of the cost of searching x in the table. *Total displacement* corresponding to a sequence of hashed values is the sum of the individual displacements of elements. and it determines the *construction cost* of the table.

Linear probing hashing has been the object of intense study; see the table on results and the bibliography in (Gonnet and Baeza-Yates, 1991, pp. 51-54). The first published analysis of linear probing was done by Konheim and Weiss (Konheim and Weiss (1966)). In addition, there is also special value for these problems since the first analysis of algorithms ever performed by D. Knuth (Knuth (1963)) was that of linear probing hashing. As Knuth indicates in many of his writings, the problem has had a strong influence on his scientific carreer. Moreover, the construction cost to fill a linear probing hash table connects to a wealth of interesting combinatorial and analytic problems. More specifically, the Airy distribution that surfaces as a limit law in this construction cost is also present in random trees (inversions and path length), random graphs (the complexity or excess parameter), and in random walks (area) (Knuth (1998b); Flajolet et al. (1998)).

Operating primarily in the context of double hashing, several authors (Brent (1973); Amble and Knuth (1974); Gonnet and Munro (1979)) observed that a collision could be resolved in favor of *any* of the keys involved, and used this additional degree of freedom to decrease the expected search time in the table. We obtain the standard scheme by letting the incoming key probe its next location. So, we may

[†]This work of was supported in part by proyecto CSIC fondos 2002-2004 and 2004-2006, from the Universidad de la República.

see this standard policy as a *first-come-first-served* (FCFS) heuristic. Later Celis, Larson and Munro (Celis (1986); Celis et al. (1985)) were the first to observe that collisions could be resolved having *variance reduction* as a goal. They defined the Robin Hood heuristic, in which each collision occurring on each insertion is resolved in favor of the key that is farthest away from its home location. Later, Poblete and Munro (Poblete and Munro (1989)) defined the *last-come-first-served* (LCFS) heuristic, where collisions are resolved in favor of the incoming key, and others are moved ahead one position in their probe sequences. These strategies do not look ahead in the probe sequence, since the decision is made before any of the keys probes its next location. As a consequence, they do not improve the average search cost.

It is shown in (Carlsson et al. (1987)) that the Robin Hood linear probing algorithm minimizes the variance of all linear probing algorithms that do not look ahead. This variance, for a full table, is $\Theta(m)$, instead of the $\Theta(m^{3/2})$ of the standard algorithm. They derived the following expressions for the variance of $C_{m,n}$, the successful search time

$$\begin{aligned}
\mathbf{Var}[C_{m,n}] &= \frac{1}{2}Q_1(m,n-1) - \frac{1}{4}Q_0(m,n-1)^2 - \frac{1}{6}Q_0(m,n-1) + \frac{1}{6}\frac{n-1}{m} - \frac{1}{12},\\
\mathbf{Var}[C_{m,\alpha m}] &= \frac{1}{4(1-\alpha)^2} - \frac{1}{6(1-\alpha)} - \frac{1}{12} + \frac{\alpha}{6} - \frac{1}{6m} - \frac{1+2\alpha}{3(1-\alpha^4)m} + O\left(\frac{1}{m^2}\right),\\
\mathbf{Var}[C_{m,m}] &= \frac{4-\pi}{8}m + \frac{1}{9} - \frac{\pi}{48} + \frac{1}{135}\sqrt{\frac{2\pi}{m}} + O\left(\frac{1}{m^2}\right).
\end{aligned} \tag{1}$$

Moreover, in (Jason (2003)) and (Viola (2004)), a distributional analysis for the FCFS, LCFS and Robin Hood heuristic is presented. More specifically they obtain

$$\begin{aligned}
Pr\{C_{m,n}=r\} &= \sum_{i=r}^{n+1}\binom{n+1}{i}\frac{(m-n-1+i)}{(n+1)m^i}\sum_{k=0}^{i+1-r}(-1)^{i+1-r-k}\binom{i+1}{r+k}k^i,\\
C_\alpha(z) &= z\frac{1-\alpha}{\alpha}\frac{e^{z\alpha}-e^{\alpha}}{ze^{\alpha}-e^{z\alpha}},
\end{aligned}$$

where $C_\alpha(z)$ is the probability generating function of the successful search time when $m, n \to \infty$ and $m/n = \alpha$, $0 \le \alpha < 1$.

These results consider a hash table with buckets of size 1. However, very little is known when we have tables with buckets of size b. In (Blake and Konheim (1977)), Blake and Konheim studied the asymptotic behavior of the expected cost of successful searches as the number of records and buckets tend to infinity with their ratio remaining constant. Mendelson (Mendelson (1983)) derived exact formulae for the same expected cost, but only solved them numerically. These papers consider the FCFS heuristic. In (Viola and Poblete (1998)) the first exact analysis of a linear probing hashing scheme with buckets of size b is presented. In that paper, they find the expected value and the asymptotic behavior of the average cost of successful searches when the Robin Hood heuristic is used. One of their main methodological contributions is the introduction of a new sequence of numbers $T_{k,d,b}$, that is one of the key components of the analysis presented in this paper.

In this paper we complete the work presented in (Viola and Poblete (1998)), and find the complete distribution for the search cost of a random element when we construct a linear probing hash table using the Robin Hood heuristic, in tables with buckets of size b. As far as we know this is the first distributional analysis of a hashing scheme with buckets of size b.

We give the distribution for $b\alpha$-full tables ($0 \le \alpha < 1$). These results can also be derived for tables of fixed length m and n elements inserted, by the use of the Poisson transform presented in section 2. However, the expressions are very complicated, so, in this extended abstract we work generally in the Poisson model, leaving for the full version all the complete derivations.

This paper is organized as follows. Section 2 refreshes the basic notions of the Poisson Transform and section 3 presents the new sequence of numbers $T_{k,d,b}$, key to perform this analysis. Finally in section 4 the exact distribution in the Poisson model is derived, and in section 5, the exact expressions for all the factorial moments are presented.

2 The Poisson Transform

There are two standard models that are extensively used in the analysis of hashing algorithms: the *exact filling* model and the *Poisson filling* model. Under the exact model, we have a fixed number of keys, n, that are distributed among m locations, and all m^n possible arrangements are equally likely to occur.

Under the Poisson model, we assume that each location receives a number of keys that is Poisson distributed with parameter λ, and is *independent* of the number of keys going elsewhere. This implies that the total number of keys, N, is itself a Poisson distributed random variable with parameter λm:

$$Pr\left[N = n\right] = \frac{e^{-\lambda m}(\lambda m)^n}{n!} \quad n \; = \; 0, 1, \ldots$$

This model was first considered in the analysis of hashing by Fagin *et al* (Fagin et al. (1979)) in 1979. The *Poisson transform* (also called *Poisson generating function* Flajolet et al. (1995); Jacquet and Régnier (1986)) of $f_{m,n}$, is

$$\mathbf{P}_m[f_{m,n}; \lambda] = \sum_{n \geq 0} Pr\left[N = n\right] f_{m,n}. \tag{2}$$

If $\mathbf{P}_m[f_{m,n}; \lambda]$ has a MacLaurin expansion in powers of λ, then we can retrieve the original sequence $f_{m,n}$ by the following inversion theorem (Gonnet and Munro (1984))

Theorem 1 *If* $\mathbf{P}_m[f_{m,n}; \lambda] = \sum_{k \geq 0} a_{m,k} \lambda^k$ *then* $f_{m,n} = \sum_{k \geq 0} a_{m,k} \frac{n^k}{m^k}$.

In this paper we consider $\lambda = b\alpha$.

3 A New sequence of numbers

We define $Q_{m,n,d}$ as the number of ways of inserting n records into a table with m buckets of size b, so that a given (say the last) bucket of the table contains more than d empty slots. Since the bucket size remains fixed as b, we do not include it as a subscript. There cannot be more empty slots than the size of the bucket so $Q_{m,n,b} = 0$. For each of the m^n possible arrangements, the last bucket has 0 or more empty slots, and so $Q_{m,n,-1} = m^n$. Observe that $Q_{m,n,0}$ gives the number of ways of inserting n records into a table with m buckets, so that the last bucket is not full. For notational convenience, we define $Q_{0,n,d} = [n = 0]$ (following the notation presented in Graham et al. (1989) we use $[S]$ to represent 1 if S is true, and 0 otherwise). In (Mendelson (1983)), Mendelson proves

Theorem 2 *For* $0 \leq d \leq b-1$, *and* $m > 0$,

$$Q_{m,n,d} = \begin{cases} \displaystyle\sum_{j=0}^{n} \binom{n}{j} Q_{m-1,j,d} & 0 \leq n < mb\text{-}d \\ 0 & n \geq mb\text{-}d \end{cases}$$

It does not seem possible to find a closed formula for $Q_{m,n,d}$. This sequence of numbers is important since $(Q_{m,n,b-d+1} - Q_{m,n,b-d})/m^n$ is the probability that a given bucket of the table contains exactly d elements.

A different approach to the study of the numbers $Q_{m,n,d}$, is presented in (Viola and Poblete (1998)), where a new sequence of numbers $T_{k,d,b}$ is introduced that satisfies a recurrence resembling that of the Bernoulli numbers. This new sequence may be helpful in solving problems involving recurrences with truncated generating functions.

Let

$$[A(z)]_n = \sum_{i=0}^{n} a_i z^i,$$

then Theorem 2 translates into the recurrence relation

$$\begin{aligned} Q_{0,d}(z) &= 1 \\ Q_{m,d}(z) &= [e^z Q_{m-1,d}(z)]_{bm-d-1} \qquad m \geq 1 \end{aligned} \tag{3}$$

for $Q_{m,d}(z) = \sum_{n \geq 0} Q_{m,n,d} \frac{z^n}{n!}$. The main problem is that we are dealing with a recurrence that involves truncated generating functions.

Their strategy is to find an exponential generating function $T_d(z)$ such that

$$Q_{m,d}(z) = [T_d(z) e^{mz}]_{bm-d-1} \tag{4}$$

where $T_d(z) = \sum_{k\geq 0} T_{k,b,d}\frac{z^k}{k!}$, for some coefficients $T_{k,b,d}$ to be determined, and independent of m. Here, b is an implicit parameter, and we use the expression $T_{k,d}$.

The intuition behind this idea is as follows. From (3), we obtain $Q_{m,d}(z)$ by multiplying the truncated generating function $Q_{m-1,d}(z)$ by the series e^z and then taking only the first $bm - d - 1$ terms of it. Moreover, $Q_{0,d}(z)$ is the first term of e^z. It is clear that without any truncations $Q_{m,d}(z)$ would be e^{mz}. However we have to consider a correcting factor originated by these truncations and this is the reason for defining this generating function $T_d(z)$. Then (4) gives a non recursive definition of $Q_{m,d}(z)$ that involves the truncated product of two series. The interesting aspect of this approach is that $T_d(z)$ does not depend on m. Furthermore, the only dependency on m is captured in the well known series that converges to e^{mz}.

These numbers $T_{k,d}$ satisfy some nice properties. The following can indeed be used as definition.

Theorem 3

$$\sum_j \binom{k}{j}\left(\left\lfloor\frac{k+d}{b}\right\rfloor\right)^{k-j} T_{j,d} = [k=0]. \tag{5}$$

A very curious property of these numbers is

Theorem 4

$$\sum_{d=0}^{b-1} T_{k,d} = \begin{cases} b & k=0 \\ -1 & k=1 \\ 0 & k>1. \end{cases}, \tag{6}$$

that translates into

$$\sum_{d=0}^{b-1} T_d(b\alpha) = \sum_{d=0}^{b-1}\left(\sum_{k\geq 0} T_{k,d}\frac{(b\alpha)^k}{k!}\right) = b(1-\alpha) \tag{7}$$

Equation (7) is very useful to simplify several expressions in the analysis. One of the key observations is that $T_d(b\alpha)$ is the Poisson transform of $Q_{m,n,d}/m^n$, the probability that a given bucket contains more than d empty slots, when $m, n \to \infty$ and $n = b\alpha m$.

4 Analysis of the Robin Hood Linear Probing Hashing Algorithm

We follow the ideas presented in (Viola (2004)) and (Viola and Poblete (1998)). Figure 1 shows the result of inserting records with the keys 36, 77, 24, 69, 18, 56, 97, 78, 49, 79, 38 and 10 in a table with ten buckets of size 2, with hash function $h(x) = x$ mod 10, and resolving collisions by linear probing using the Robin Hood heuristic.

69	10			24		36	77	18	78
79						56	97	38	49
0	1	2	3	4	5	6	7	8	9

Fig. 1: A Robin Hood Linear Probing hash table.

When there is a collision in location i, then the record that has probed the least number of locations, probes location $(i+1)$ mod m. In the case of a tie, we (arbitrarily) move the record whose key has largest value.

Figure 2 shows the partially filled table after inserting 58. There is a collision with 18 and 38. Since there is a tie (all of them are in their first probe location), we arbitrarily decide to move 58. Then, 58 is in its second probe location, 69 and 79 in their first one. So, 79 has to move to its final location.

The following properties are easily verified:

- At least one record is in its home location.

49	79			24		36	77	18	58
69	10					56	97	38	78
0	1	2	3	4	5	6	7	8	9

Fig. 2: The table after inserting 58.

- The keys are stored in nondecreasing order by hash value, starting at some location k and wrapping around. In our example k=4 (corresponding to the home location of 24).
- If a fixed rule is used to break ties among the candidates to probe their next probe location (eg: by sorting these keys in increasing order), then the resulting table is independent of the order in which the records were inserted (Celis (1986)).

4.1 Linear Probing Sort

To analyze Robin Hood linear probing, we first have to discuss some ideas presented in (Carlsson et al. (1987); Viola (2004); Viola and Poblete (1998)) and (Gonnet and Munro (1984)). When the hash function is order preserving (that is, if $x < y$ then $h(x) < h(y)$), a variation of the Robin Hood linear probing algorithm can be used to sort (Gonnet and Munro (1984)), by successively inserting the n records in an initially empty table. In this case, instead of letting the excess records from the rightmost location of the table wrap around to location zero, we can use an *overflow area* consisting of locations m, $m+1$, etc. The number of locations needed for this overflow area is an important performance measure for this sorting algorithm. In this section we study the distribution for the number of records that overflow.

This analysis is related to the study of the cost of successful searches in the Robin Hood linear probing algorithm, as follows. Without loss of generality, we search for a record that hashes to location 0. Moreover, since the order of the insertion is not important, we assume that this record is the last one inserted. If we look at the table after the first n records have been inserted, all the records that hash to location 0 (if any) will be occupying contiguous locations, near the beginning of the table. The locations preceding them will be occupied by records that wrapped around from the right end of the table, as can be seen in Figure 2. The key observation here is that those records are exactly the ones that would have gone to the overflow area. Furthermore, it is easy to see that the number of records in this overflow area does not change when the records that hash to 0 are removed. As a consequence, the cost of retrieving a record that hashes to 0 can be divided in two parts.

- The number of records that wrap around the table. In other words, the size of the overflow area.
- The number of records that hash to location 0.

In section 4.2 we study the distribution of the elements that overflow, and in section 4.3 we study the distribution for the successful search cost of a random element.

We work in the Poisson model, since the presentation is much simpler than in the exact model. Then, with the use of the Poisson transform, we obtain the exact results for fixed m, n, and b. It is important to note that in (Viola (2004)) the exact distribution of the elements that overflow when $b = 1$ and in (Viola and Poblete (1998)) the average number of elements that overflow (for general b) are calculated in the exact model. This analysis brings a new point of view on the problem.

4.2 Analysis of the Overflow Area

Let $\Omega(b\alpha, z)$ be the probability generating function for the number of elements that overflow from a given bucket. We first derive a recurrence for $\Omega(b\alpha, z)$, and then solve it.

Since we have a Poisson process, with probability $e^{-b\alpha}\frac{(b\alpha)^k}{k!}$ this given bucket receives, in addition to the elements that overflow from the previous bucket, k elements that hash to it. From these elements, all but b of them go to overflow, and their contribution to the recurrence is

$$e^{b\alpha(z-1)}\frac{\Omega(b\alpha, z)}{z^b}. \tag{8}$$

However, when this bucket receives less than b elements, there is no overflow, and so we need a correction term. This correction term, is

$$\sum_{s=1}^{b}\left(1-z^{-s}\right)P_s(b\alpha),$$

where $P_s(b\alpha)$ is the Poisson transform of the probability of having $b-s$ elements in a given bucket. As we have seen in section 3 this value is equal to $T_{s-1}(b\alpha)-T_s(b\alpha)$, and so the contribution of this correction term is

$$\sum_{s=1}^{b}\left(1-z^{-s}\right)\left(T_{s-1}(b\alpha)-T_s(b\alpha)\right)=\left(1-z^{-1}\right)\sum_{s=0}^{b-1}T_s(b\alpha)z^{-s}. \tag{9}$$

As a consequence, from equations (8) and (9) we obtain the following recurrence

$$\Omega(b\alpha,z)=e^{b\alpha(z-1)}\frac{\Omega(b\alpha,z)}{z^b}+\left(1-z^{-1}\right)\sum_{s=0}^{b-1}T_s(b\alpha)z^{-s},$$

and we finally get

$$\Omega(b\alpha,z)=\frac{(z-1)}{z^b-e^{b\alpha(z-1)}}\sum_{s=0}^{b-1}T_{b-1-s}(b\alpha)z^s. \tag{10}$$

When $b=1$ we rederive the result presented in (Viola (2004))

$$\Omega(\alpha,z)=\frac{(z-1)(1-\alpha)}{z-e^{\alpha(z-1)}}.$$

4.3 Analysis of Robin Hood Linear Probing

In this section we find the distribution of the cost of a successful search for a random record in a hash table of size $m\to\infty$ that contains $n=b\alpha m$ records with $0\le\alpha<0$.

Let $\Psi(b\alpha,z)$ be the probability generating function for the cost of a successful search for a random record in a $b\alpha$-full table with $0\le\alpha<1$.

As mentioned in section 4.1 the cost of retrieving a random record is composed by all the elements that hash to the same location (collisions), plus the number of records that overflow from the previous location. We first derive the generating function $C(b\alpha,z)$ for the total displacement, that is, the generating function for the total number of comparisons, without considering the fact we have to count only number of buckets probed. Then, if

$$C(b\alpha,z)=\sum_{n\ge0}c_n(b\alpha)z^n,$$

the probability generating function for the cost of a successful search is

$$\begin{aligned}\Psi(b\alpha,z) &= z\sum_{n\ge0}c_n(b\alpha)z^{\lfloor\frac{n}{b}\rfloor}=z\sum_{k\ge0}\left(\sum_{d=0}^{b-1}c_{bk+d}(b\alpha)\right)z^k\\ &= \frac{z}{b}\sum_{d=0}^{b-1}C\left(b\alpha,r^jz^{1/b}\right)\sum_{p=0}^{b-1}\left(r^jz^{1/b}\right)^{-p},\end{aligned} \tag{11}$$

where $r=e^{\frac{2\pi\mathrm{i}}{b}}$ is a b-th root of unity. As we may expect, the calculations are very involved and obscure, so we extract exact coefficients, when possible, and then use the Poisson Transform to find the results in the exact model. A similar approach is used to find higher moments and limit distributions.

If k elements collide with the searched one, the expected total displacement originated by these collisions for (separately) retrieving all these records is

$$\frac{1}{k+1}\sum_{r=0}^{k}z^r=\frac{1}{k+1}\left(\frac{1-z^{k+1}}{1-z}\right).$$

Since the probability of having k records colliding with the searched one is $e^{-b\alpha}\frac{(b\alpha)^k}{k!}$, we immediately see that the probability generating function of the displacement originated by these collisions is

$$\frac{e^{-b\alpha}}{1-z}\sum_{k\geq 0}\frac{(b\alpha)^k}{(k+1)!}\left(1-z^{k+1}\right)=\frac{1-e^{b\alpha(z-1)}}{b\alpha(1-z)} \tag{12}$$

To conclude the derivation we have to consider the cost originated by the elements that overflow, and so, by equations (10) and (12) we find

$$C(b\alpha,z)=\frac{1-e^{b\alpha(z-1)}}{b\alpha\left(z^b-e^{b\alpha(z-1)}\right)}\sum_{s=0}^{b-1}T_{b-1-s}(b\alpha)z^s. \tag{13}$$

When $b=1$, $T_0(\alpha)=(1-\alpha)$, and so we obtain

$$C(\alpha,z)=\frac{1-\alpha}{\alpha}\frac{1-e^{\alpha(z-1)}}{z-e^{\alpha(z-1)}},$$

as derived in (Viola (2004)) and (Jason (2003)).

To extract the coefficients $\psi_k(b\alpha)$ we first find $c_n(b\alpha)$. Towards this goal we expand equation (13)

$$\begin{aligned}
C(b\alpha,z) &= \left(1-z^{-b}\right)\left(\sum_{s=0}^{b-1}\frac{T_{b-1-s}(b\alpha)}{b\alpha}z^s\right)\sum_{k\geq 1}e^{-kb\alpha}\frac{e^{kb\alpha z}}{z^{bk}}\\
&= \left(1-z^{-b}\right)\left(\sum_{s=0}^{b-1}\frac{T_{b-1-s}(b\alpha)}{b\alpha}z^s\right)\sum_{n\geq 0}\left(\sum_{k\geq 1}e^{-kb\alpha}\frac{(kb\alpha)^{n+bk}}{(n+bk)!}\right)z^n\\
&= \sum_{n\geq 0}\left(\sum_{s=0}^{b-1}\frac{T_{b-1-s}(b\alpha)}{b\alpha}\sum_{k\geq 1}e^{-kb\alpha}\left(\frac{(kb\alpha)^{n-s+bk}}{(n-s+bk)!}-\frac{(kb\alpha)^{n-s+b(k+1)}}{(n-s+b(k+1))!}\right)\right)z^n
\end{aligned} \tag{14}$$

As a consequence, from equations (11) and (14) we have proved the following theorem

Theorem 5 *Let $\Psi_{b\alpha}$ be the random variable for the cost of searching a random element in a $b\alpha$-full table with buckets of size b using the Robin Hood linear probing hashing algorithm, and let $\Psi(b\alpha,z)$ be its probability generating function. Then*

$$\begin{aligned}
\Psi(b\alpha,z) &= \frac{z}{b}\sum_{d=0}^{b-1}C\left(b\alpha,e^{\frac{2\pi \mathrm{i} j}{b}}z^{1/b}\right)\sum_{p=0}^{b-1}\left(e^{\frac{2\pi \mathrm{i} j}{b}}z^{1/b}\right)^{-p}, \quad \text{with}\\
C(b\alpha,z) &= \frac{1-e^{b\alpha(z-1)}}{b\alpha\left(z^b-e^{b\alpha(z-1)}\right)}\sum_{s=0}^{b-1}T_{b-1-s}(b\alpha)z^s.
\end{aligned}$$

Moreover, the probability $\psi_n(b\alpha)$ that $n+1$ buckets have to be probed to retrieve a random element is

$$\begin{aligned}
\psi_n(b\alpha) &= Pr\{\Psi_{b\alpha}=n+1\}=\\
&\sum_{s=0}^{b-1}\frac{T_{b-1-s}(b\alpha)}{b\alpha}\sum_{k\geq 1}e^{-kb\alpha}\sum_{d=0}^{b-1}\left(\frac{(kb\alpha)^{b(n+k)+d-s}}{(b(n+k)+d-s)!}-\frac{(kb\alpha)^{b(n+k+1)+d-s}}{(b(n+k+1)+d-s)!}\right)
\end{aligned} \tag{15}$$

Even though the calculations are very involved and the expressions very complicated, we are able to present amazingly simple expressions for the first moment, in both models. First, the expected value is obtained in the Poisson model, and then by depoissonization the respective result is obtained in the exact model

Corollary 1 *Let $\Psi_{b,m,n}$ be the random variable for the cost of searching a random element when we insert $n+1$ elements in a hash table of m buckets size b using the Robin Hood linear probing hashing algorithm. Then for $0\leq\alpha<1$ we have*

$$\mathbf{E}[\Psi_{b,m,n+1}] = 1+\sum_{k=1}^{\lfloor n/b\rfloor}\sum_{i=kb}^{n}(-1)^{i-kb}\binom{i-1}{kb-1}\frac{(kb)^i}{(i+1)!}\frac{n^{\underline{i}}}{(bm)^i}, \tag{16}$$

$$\mathbf{E}[\Psi_{b\alpha}] = 1 + \sum_{k\geq 1} e^{-kb\alpha} \sum_{n\geq 1} n \frac{(kb\alpha)^{n+bk-1}}{(n+bk)!}, \tag{17}$$

$$b\mathbf{E}[\Psi_{b,m,bm-1}] = \frac{\sqrt{2\pi bm}}{4} + \frac{1}{3} + \sum_{d=1}^{b-1} \frac{1}{1 - T\left(e^{\frac{2\pi i d}{b}-1}\right)} + \frac{1}{48}\sqrt{\frac{2\pi}{bm}} + O\left(\frac{1}{bm}\right), \tag{18}$$

where $T(u)$ is the Tree Function ($T(u) = ue^{T(u)}$).

While equation (16) is a new result, the special case for a full table ($n = bm - 1$) is derived in (Viola and Poblete (1998)). Moreover equation (17) appears in (Knuth (1998a)) and (18) in (Viola and Poblete (1998)).

5 Higher Moments

From Theorem 5 we may derive exact expressions for the factorial moments. We present here the main results in the Poisson model, that generalize the results presented in (Viola (2004)) leaving the details of the derivations for the full version of the paper. The corresponding expressions in the exact model can be obtained with the use of the Poisson transform, although they are extremely complicated.

We first obtain the exact expression for all the factorial moments of $\Psi_{b\alpha}$. Given the probability generating function $\Psi(b\alpha, z)$, then $E[\Psi_{b\alpha}^{\underline{r}}]$ can be can be obtained by differentiating this generating function r times and setting $z = 1$.

Theorem 6 *Let $\Psi_{b\alpha}$ be the random variable for the cost of searching a random element in a $b\alpha$-full table with buckets of size b using the Robin Hood linear probing hashing algorithm, and let $\Psi(b\alpha, z)$ be its probability generating function. Then*

$$\begin{aligned}
E[\Psi_{b\alpha}^{\underline{r}}] &= r\sum_{s=0}^{b-1} \frac{T_{b-1-s}(b\alpha)}{b\alpha} \sum_{n\geq 1} n^{\underline{r-1}} \sum_{k\geq 1} e^{-kb\alpha} \sum_{d=0}^{b-1} \frac{(kb\alpha)^{b(k+n)+d-s}}{(b(k+n)+d-s)!} \\
&+ 1^{\underline{r}} \sum_{s=0}^{b-1} \frac{T_{b-1-s}(b\alpha)}{b\alpha} \sum_{k\geq 1} e^{-kb\alpha} \sum_{d=0}^{b-1} \frac{(kb\alpha)^{bk+d-s}}{(bk+d-s)!} \\
&= r\frac{1-\alpha}{\alpha} \sum_{n\geq 1} (n+1)^{\underline{r-1}} \sum_{k\geq 1} e^{-kb\alpha} \sum_{d=0}^{b-1} \frac{(kb\alpha)^{b(k+n)+d}}{(b(k+n)+d)!} \\
&- r(r-1) \sum_{n\geq 1} n^{\underline{r-2}} \sum_{k\geq 1} e^{-kb\alpha} \sum_{d=0}^{b-1} \frac{(kb\alpha)^{b(k+n)+d}}{(b(k+n)+d)!} \sum_{s=0}^{b-1-d} \frac{T_{b-1-s}(b\alpha)}{b\alpha} \\
&+ 1^{\underline{r-1}} r \sum_{d=0}^{b-1} \sum_{k\geq 1} e^{-kb\alpha} \frac{(kb\alpha)^{bk+d}}{(bk+d)!} \sum_{s=b-d}^{b-1} \frac{T_{b-1-s}(b\alpha)}{b\alpha} \\
&+ 1^{\underline{r}} \sum_{s=0}^{b-1} \frac{T_{b-1-s}(b\alpha)}{b\alpha} \sum_{k\geq 1} e^{-kb\alpha} \sum_{d=0}^{b-1} \frac{(kb\alpha)^{bk+d-s}}{(bk+d-s)!}.
\end{aligned} \tag{19}$$

It is important to notice that the main asymptotic contribution (needed to find the limit distribution) when $\alpha \to 1$ is given by the first sum of equation (19), while the last two sums vanish for moments greater than 2.

Limit distributions will be presented in the full version of the paper based on these derivations.

References

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

I. Blake and A. Konheim. Big buckets are (are not) better! *J. ACM*, 24(4):591–606, Oct. 1977.

R. Brent. Reducing the retrieval time of scatter storage techniques. *C. ACM*, 16(2):105–109, 1973.

S. Carlsson, J. Munro, and P. Poblete. On linear probing hashing. Unpublished Manuscript, 1987.

P. Celis. *Robin Hood Hashing*. PhD thesis, Computer Science Department, University of Waterloo, April 1986. Technical Report CS-86-14.

P. Celis, P.-A. Larson, and J. Munro. Robin Hood hashing. In *26th IEEE Sympusium on the Foundations of Computer Science*, pages 281–288, 1985.

R. Fagin, J. Nievergelt, N. Pippenger, and H. R. Strong. Extendible hashing - a fast access method for dynamic files. *ACM Transactions on Database Systems*, 4(3):315–344, 1979.

P. Flajolet, X. Gourdon, and P. Dumas. Mellin transforms and asymptotics : Harmonic sums. *Theoretical Computer Science*, 144(1–2):3–58, 1995.

P. Flajolet, P. Poblete, and A. Viola. On the analysis of linear probing hashing. *Algorithmica*, 22(4):490 – 515, 1998.

G. Gonnet and J. Munro. Efficient ordering of hash tables. *SIAM Journal on Computing*, 8(3):463–478, 1979.

G. Gonnet and J. Munro. The analysis of linear probing sort by the use of a new mathematical transform. *Journal of Algorithms*, 5:451–470, 1984.

G. H. Gonnet and R. Baeza-Yates. *Handbook of Algorithms and Data Structures: in Pascal and C*. Addison–Wesley, second edition, 1991.

R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley Publishing Company, 1989.

P. Jacquet and M. Régnier. Trie partitioning process: Limiting distributions. In P. Franchi-Zanetacchi, editor, *CAAP'86*, volume 214 of *LNCS*, pages 196–210, 1986. Proceedings of the 11th Colloquium on Trees in Algebra and Programming, Nice France, March 1986.

S. Jason. Individual displacements for linear probing hashing with different insertion policies. Preprint, July 2003.

D. E. Knuth. Notes on "open" addressing. Unpublished memorandum, 1963. (Memo dated July 22, 1963. With annotation "*My first analysis of an algorithm, originally done during Summer 1962 in Madison*". Also conjectures the asymptotics of the Q-function, with annotation "*Proved May 24, 1965*".).

D. E. Knuth. *The Art of Computer Programming*, volume 3 Sorting and Searching. Addison-Wesley Publishing Company, 1998a.

D. E. Knuth. Linear probing and graphs. *Algorithmica*, 22(4):561–568, 1998b.

A. Konheim and B. Weiss. An occupancy discipline and applications. *SIAM Journal on Applied Mathematics*, 6(14):1266–1274, 1966.

P.-A. Larson. Analysis of uniform hashing. *JACM*, 30(4):805–819, 1983.

H. Mendelson. Analysis of linear probing with buckets. *Information Systems*, 8(3):207–216, 1983.

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

P. Poblete and J. Munro. Last-come-first-served hashing. *Journal of Algorithms*, 10:228–248, 1989.

R. Sedgewick. *Algorithms in C, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching*. Addison–Wesley, Reading, Mass., third edition, 1998.

A. Viola. Exact distribution of individual displacements in linear probing hashing. 2004. Submitted to ACM Transactions on Algorithms.

A. Viola and P. V. Poblete. The analysis of linear probing hashing with buckets. *Algorithmica*, 21(2): 37–71, 1998.