Combinatorics, Probability and Computing (2019) 28, 600–617. © Cambridge University Press 2018
doi:10.1017/S0963548318000408

Analysis of Robin Hood and Other Hashing Algorithms Under the Random Probing Model, With and Without Deletions

P. V. POBLETE[1†] and A. VIOLA[2‡]

[1] Department of Computer Science, University of Chile, Casilla 2777, Santiago, Chile
(e-mail: `ppoblete@dcc.uchile.cl`)

[2] Instituto de Computación, Universidad de la República, Montevideo 11300, Uruguay
(e-mail: `aviola@fing.edu.uy`)

Received 12 September 2016; revised 18 July 2018 ; first published online 14 August 2018

Thirty years ago, the Robin Hood collision resolution strategy was introduced for open addressing hash tables, and a recurrence equation was found for the distribution of its search cost. Although this recurrence could not be solved analytically, it allowed for numerical computations that, remarkably, suggested that the variance of the search cost approached a value of 1.883 when the table was full. Furthermore, by using a non-standard mean-centred search algorithm, this would imply that searches could be performed in expected constant time even in a full table.

In spite of the time elapsed since these observations were made, no progress has been made in proving them. In this paper we introduce a technique to work around the intractability of the recurrence equation by solving instead an associated differential equation. While this does not provide an exact solution, it is sufficiently powerful to prove a bound of $\pi^2/3$ for the variance, and thus obtain a proof that the variance of Robin Hood is bounded by a small constant for load factors arbitrarily close to 1. As a corollary, this proves that the mean-centred search algorithm runs in expected constant time.

We also use this technique to study the performance of Robin Hood hash tables under a long sequence of insertions and deletions, where deletions are implemented by marking elements as *deleted*. We prove that, in this case, the variance is bounded by $1/(1-\alpha)$, where α is the load factor.

To model the behaviour of these hash tables, we use a unified approach that we apply also to study the First-Come-First-Served and Last-Come-First-Served collision resolution disciplines, both with and without deletions.

2010 *Mathematics subject classification*: Primary 68W40
Secondary 68P05, 68P10, 68P20

† Supported in part by NIC Chile.
‡ Partially supported by Project CSIC I+D ‘Combinatoria Analítica y aplicaciones en criptografía, comunicaciones y recuperación de la información’, fondos 2015-2016.

1. Introduction

In 1986, Celis, Larson and Munro [6, 7] introduced the Robin Hood collision resolution strategy for open addressing hash tables. Under this discipline, collisions are decided in favour of the element that is farthest from its home location. While this does not change the expected search cost, it turns out to have a dramatic effect on its *variance*. In effect, unlike other disciplines where the variance tends to infinity as the table becomes full, the variance of Robin Hood seems to remain constant, and very small. This fact, conjectured from numerical computations, has not been proved in the years since it was observed, and is the main focus of our work. This problem has been hard to solve because the distribution of the search cost obeys a nonlinear recurrence equation for which no successful line of attack has been found.

To show the kind of recurrence involved, we now quote Theorem 3.1 from [6] (our notation will be slightly different).

Theorem (from [6]). *In the asymptotic model for an infinite Robin Hood hash table with load factor α $(\alpha < 1)$, the probability $p_i(\alpha)$ that a record is placed in the ith or further position in its probe sequence is equal to*

$$p_1(\alpha) = 1, \quad p_{i+1}(\alpha) = 1 - \left(\frac{1-\alpha}{\alpha}\right)\left(e^{\alpha(p_1(\alpha)+\cdots+p_i(\alpha))} - 1\right). \tag{1.1}$$

He then goes on to define another function $r_i(\alpha) = \alpha(p_i(\alpha) + \cdots + p_\infty(\alpha))$, in terms of which the variance can be expressed as

$$V(\alpha) = \frac{2}{\alpha}\sum_{i=1}^{\infty} r_i(\alpha) + \frac{\ln(1-\alpha)}{\alpha} - \frac{\ln^2(1-\alpha)}{\alpha^2}. \tag{1.2}$$

He shows that $r_i(\alpha)$ satisfies the following recurrence equation:

$$r_i(\alpha) - r_{i+1}(\alpha) = 1 - e^{-r_i(\alpha)}, \tag{1.3}$$

with $r_1(\alpha) = -\ln(1-\alpha)$. By leaving the '$(\alpha)$' implicit and using the Δ operator (defined as $\Delta r_i = r_{i+1} - r_i$), this can be rewritten as $\Delta r_i = f(r_i)$ where f is the function $f(x) = -1 + e^{-x}$.

This seemingly simpler equation has, nonetheless, so far remained unsolved.

In this paper, we will introduce a technique applicable to equations of this form, and we will use it first to prove a bound on the variance of Robin Hood hashing. Then we will use it to study another recurrence equation of the same type arising from the problem of hashing with deletions.

2. Modeling hashing algorithms

In this paper we will study the search cost of a random element in a hash table, using the *random probing model*. This is an open addressing hashing scheme in which collisions are resolved by additional probes into the table. The sequence of these probes is considered to be random and depends only on the value of the key. The difference from uniform probing is that positions may be repeated in this sequence. We use the *asymptotic model* for a hash table with load factor α [7, 11, 12, 14], where we assume that the number of keys n and the table size m both tend to infinity, while keeping their ratio $\alpha = n/m$ constant.

Each element has associated with it an infinite probe sequence consisting of i.i.d. integers uniformly distributed over $\{0,\ldots,m-1\}$, representing the consecutive places of probes for that element. The probe sequence for element x is denoted by $h_1(x), h_2(x), \ldots$. Elements are inserted sequentially into the table. If element x is placed in position $h_j(x)$, then we say that element x has age j, as it requires j probes to reach the element in the case of a search. When an element x of age j and an element y of age k compete for the same slot ($h_j(x) = h_k(y)$), a collision resolution strategy is needed.

In the standard method [13], a collision is resolved in favour of the incumbent key, so the incoming key continues probing to its next location. We call this a First-Come-First-Served (FCFS) collision resolution discipline. Several authors [2, 4, 10] have observed that a collision could be resolved in favour of *any* of the keys involved, and used this additional degree of freedom to decrease the expected search time in the table.

Celis, Larson and Munro [6, 7] were the first to observe that collisions could be resolved having *variance reduction* instead as a goal. They defined the Robin Hood (RH) heuristic, in which each collision occurring during an insertion is resolved in favour of the key that is farthest away from its home location (*i.e.* oldest in terms of *age*). Later, Poblete and Munro [16] defined the Last-Come-First-Served heuristic, where collisions are resolved in favour of the *incoming* key.

In both cases, the variance is reduced, and this can be used to speed up searches by replacing the standard search algorithm by a *mean-centred* one that first searches in the vicinity of where we would expect the element to have *drifted* to, rather than in its initial probe location. This *mean-centred* approach was introduced in [6] (and called ‘organ-pipe search’) to speed up successful searches in the Robin Hood heuristic, with expected cost bounded by the standard deviation of this random variable. Numerical computations in [6] suggest that for full tables the variance of the search cost for RH is constant, but no formal proof is given.

In this paper we formally settle this conjecture, by proving that this is in fact the case, and give an explicit upper bound (although not as tight as the numerical results seem to suggest). As a consequence we prove that the mean-centred searching algorithm in [6] has constant expected cost for full tables.

In Section 4 we extend this approach to perform the analysis of hashing with deletions. Deletions in open addressing hash tables are often handled by marking the cells as *deleted* instead of *empty*, because otherwise the search algorithm might fail to find some of the keys. The space used by deleted cells may be re-used by subsequent insertions. Intuitively, search times should deteriorate as tables become contaminated with deleted cells and, as Knuth [13] points out, in the long run the average successful search time should approach the average *unsuccessful* search time.

In this paper we analyse the effect of a long sequence of insertions and deletions in the asymptotic regime (α-full tables with $0 \leqslant \alpha < 1$) and prove a bound for the variance of RH with deletions that is close to numerical results.

There is an alternative algorithm designed to keep variance low in the presence of deletions. This method marks cells as deleted, but keeps the key values (these cells are called *tombstones*). In this paper we do not study the algorithm with tombstones. We note that [14] derives equations for this algorithm, but only obtains numerical solutions.

3. Analysis without deletions

To analyse the cost of searching for a random element, we begin by presenting a general framework, based on the one used in [8]. As we will see, this framework applies to FCFS and LCFS and RH. and in this paper we will focus mainly on RH, whose analysis has been a long-standing open problem. As stated before, we use the asymptotic model for a hash table with load factor α and random probing.

Under this model, if collisions are resolved without 'looking ahead' in the table, the cost of inserting a random element is 1 plus a random variable that follows a geometric distribution with parameter $1-\alpha$, and therefore its expected cost is $1/(1-\alpha)$, independent of the collision resolution discipline used.

Let $p_i(\alpha)$ be the probability that a randomly chosen key has age i when the table has load factor α.

Suppose we insert a new element. Depending on the insertion discipline used, a number of keys will change locations and therefore increase their ages as a consequence of the arrival of the new element. Let us call $t_i(\alpha)$ the expected number of elements that probe their ith location during the course of an insertion. It is easy to see that

$$t_1(\alpha)=1, \quad \sum_{i\geqslant 1} t_i(\alpha)=\frac{1}{1-\alpha}. \tag{3.1}$$

Before the insertion, the expected number of keys of age i is $\alpha m p_i(\alpha)$. After the insertion, it is

$$(\alpha m+1)p_i\left(\alpha+\frac{1}{m}\right)=\alpha m p_i(\alpha)+t_i(\alpha)-t_{i+1}(\alpha). \tag{3.2}$$

If we write $\Delta\alpha=1/m$ and $q_i(\alpha)=\alpha p_i(\alpha)$, this equation becomes

$$\frac{q_i(\alpha+\Delta\alpha)-q_i(\alpha)}{\Delta\alpha}=t_i(\alpha)-t_{i+1}(\alpha) \tag{3.3}$$

and, as $\Delta\alpha\to 0$ (*i.e.* $m\to\infty$),

$$\partial_\alpha q_i(\alpha)=t_i(\alpha)-t_{i+1}(\alpha), \tag{3.4}$$

where ∂_α denotes a derivative with respect to α, and with the initial condition $q_i(0)=0$.

We introduce a notation that we will use throughout the paper. For any sequence a_i we define its *tail* $\overline{a}_i$ as

$$\overline{a}_i=\sum_{j\geqslant i} a_j. \tag{3.5}$$

Also, to simplify notation we will leave the '(α)' implicit whenever there is no ambiguity. Using these conventions, equation (3.4) can be rewritten as

$$\partial_\alpha \overline{q}_i=t_i. \tag{3.6}$$

We note that this equation is valid for all three collision resolution strategies, and it generalizes formula (10) in [14], where it is proved only for RH.

The mean of the search cost can be obtained using the tail notation, as

$$\mu_\alpha=\overline{\overline{p}}_1=\frac{1}{\alpha}\overline{\overline{q}}_1 \tag{3.7}$$

and the variance as

$$\sigma_\alpha^2 = 2\overline{\overline{\overline{p}}}_1 - \mu_\alpha - \mu_\alpha^2 = \frac{2}{\alpha}\overline{\overline{\overline{q}}}_1 - \mu_\alpha - \mu_\alpha^2. \tag{3.8}$$

We note that we can already compute the expected search cost, without needing to know the exact form of the function t_i. Taking tails in both sides of (3.6), we have $\partial_\alpha \overline{\overline{q}}_i = \overline{t}_i$. Now setting $i = 1$ and using (3.7), we obtain $\partial_\alpha(\alpha\mu_\alpha) = 1/(1-\alpha)$, and from this we obtain

$$\mu_\alpha = \frac{1}{\alpha}\ln\frac{1}{1-\alpha} \tag{3.9}$$

independent of the collision resolution discipline used.

The fact that the mean search cost is independent of the collision resolution discipline used does not necessarily carry over to higher moments or to the distribution of the search cost. To compute them, we need to know the t_i for the specific discipline.

3.1. Analysis of FCFS

For FCFS we have simply $t_i = \alpha^{i-1}$, and therefore (3.6) becomes $\partial_\alpha \overline{q}_i(\alpha) = \alpha^{i-1}$ with $\overline{q}_i(0) = 0$. The solution to this equation is $\overline{q}_i = \alpha^i/i$, and therefore from (3.8) we obtain the following result.

Theorem 3.1. *Under the asymptotic model for an infinite hash table with random probing and FCFS collision resolution discipline, the probability distribution of the search cost of a random element is*

$$p_i = \frac{\alpha^{i-1}}{i} - \frac{\alpha^i}{i+1} \quad \text{for } i \geqslant 1, \tag{3.10}$$

and the variance of the distribution is

$$\sigma_\alpha^2 = \frac{2}{1-\alpha} - \frac{1}{\alpha}\ln\frac{1}{1-\alpha} - \frac{1}{\alpha^2}\ln^2\frac{1}{1-\alpha}. \tag{3.11}$$

3.2. Analysis of LCFS

For LCFS, every time a key tries to occupy a location and finds it occupied by some other key, it forces this other key to try its next probe location. Since the 'victim' is chosen randomly, we have

$$t_{i+1}(\alpha) = \frac{1}{1-\alpha}q_i(\alpha) \quad \text{for } i \geqslant 1. \tag{3.12}$$

Introducing the generating functions $q(\alpha,z) = \sum_{i\geqslant 1} q_i(\alpha)z^i$ and $t(\alpha,z)$, defined similarly, we have

$$t(\alpha,z) = z\left(1 + \frac{1}{1-\alpha}q(\alpha,z)\right). \tag{3.13}$$

On the other hand, (3.4) implies that

$$\partial_\alpha q(\alpha,z) = \left(1 - \frac{1}{z}\right)t(\alpha,z) + 1 \tag{3.14}$$

with $q(0,z) = 0$. Combining these two equations, we have

$$\partial_\alpha q(\alpha,z) + \frac{1-z}{1-\alpha} q(\alpha,z) = z. \tag{3.15}$$

This can be solved to obtain

$$p(\alpha,z) = \frac{1}{\alpha}((1-\alpha)^{1-z} - (1-\alpha)), \tag{3.16}$$

from where we can compute the distribution and the variance.

Theorem 3.2. *Under the asymptotic model for an infinite hash table with random probing and LCFS collision resolution discipline, the probability distribution of the search cost of a random element is*

$$p_i = \left(\frac{1}{\alpha} - 1\right) \frac{\ln^i(1/(1-\alpha))}{i!} \quad \text{for } i \geqslant 1, \tag{3.17}$$

and the variance of this distribution is

$$\sigma_\alpha^2 = \frac{1}{\alpha} \ln \frac{1}{1-\alpha} - \frac{1-\alpha}{\alpha^2} \ln^2 \frac{1}{1-\alpha}. \tag{3.18}$$

The distribution of the search cost is a *positive Poisson distribution* (*i.e.* a Poisson distribution with the zero state suppressed, and then renormalized), with parameter $\lambda = \ln 1/(1-\alpha)$.

3.3. Analysis of RH

For RH, a key will be forced to try its $(i+1)$st probe location or higher each time there is a collision between an incoming key of age i or higher and another key in the table that is also of age i or higher. Therefore, we have

$$\bar{t}_{i+1} = \bar{t}_i \bar{q}_i. \tag{3.19}$$

Together with equation (3.4), this implies $\partial_\alpha \bar{q}_i = (1-\bar{q}_i)\partial_\alpha \bar{\bar{q}}_i$. Then, after integrating both sides of the equation we have $\ln 1/(1-\bar{q}_i) = \bar{\bar{q}}_i$ from where we obtain $\bar{q}_i = 1 - e^{-\bar{\bar{q}}_i}$. Moreover, by expressing $\bar{q}$ as the difference of two $\bar{\bar{q}}$, we arrive at the following result.

Theorem 3.3. *Under the asymptotic model for an infinite hash table with random probing, and Robin Hood collision resolution discipline, the double tail of the probability distribution of the search cost of a random element satisfies the recurrence*

$$\bar{\bar{q}}_{i+1} = \bar{\bar{q}}_i - 1 + e^{-\bar{\bar{q}}_i} \tag{3.20}$$

with the initial condition $\bar{\bar{q}}_1 = \ln 1/(1-\alpha)$.

This is exactly equation (1.3), which we quoted from [6], but we obtained it via a completely different derivation. As we mentioned above, numerical computations performed in [7] indicate that as $\alpha \to 1$, the variance converges to a small constant, approximately equal to 1.883. Figure 1 compares the shape of the distributions of search costs for the three disciplines.

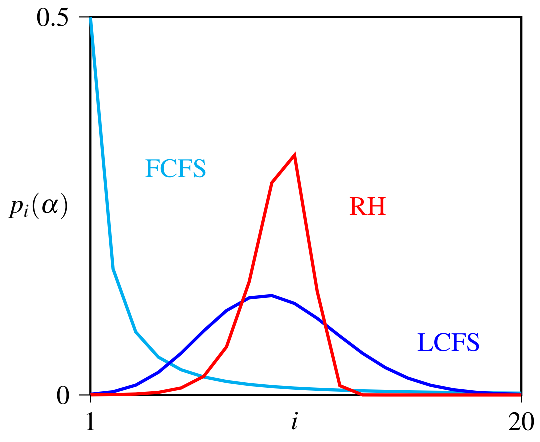

Figure 1. Density function for the distribution of search costs for FCFS, LCFS and RH for $\alpha = 0.9999$.

3.4. Bounding the variance of RH

We begin by rewriting equation (3.20) as

$$\Delta\overline{\overline{q}}_i = -1 + e^{-\overline{\overline{q}}_i}, \tag{3.21}$$

with $\overline{\overline{q}}_1 = \ln 1/(1-\alpha)$. This equation is of the form

$$\Delta\overline{\overline{q}}_i = f(\overline{\overline{q}}_i), \tag{3.22}$$

where f is the function $f(x) = -1 + e^{-x}$. This recurrence equation seems very hard to solve exactly, but we will be able to obtain useful information about its solution by studying instead the differential equation

$$Q'(x) = f(Q(x)) \tag{3.23}$$

with the same initial condition $Q(1) = \ln 1/(1-\alpha)$.

We note that both equation (3.22) and (3.23) are of the type called *autonomous* equations, defined as follows.

Definition. A differential or a difference equation is called autonomous if it does not depend explicitly on the independent variable.

Theorem 3.4. *Let the q_i satisfy the autonomous difference equation $\Delta\overline{\overline{q}}_i = f(\overline{\overline{q}}_i)$ with the initial condition $\overline{\overline{q}}_1 = \alpha\mu_\alpha$, let $Q(x)$ satisfy the autonomous differential equation $Q'(x) = f(Q(x))$ with the same initial condition $Q(1) = \alpha\mu_\alpha$ and let*

$$g(u) = \int \frac{u}{f(u)} du.$$

Then

$$\sigma_\alpha^2 < h(\alpha), \tag{3.24}$$

where

$$h(\alpha) = \frac{2}{\alpha}(g(0) - g(\alpha\mu_\alpha)) - \mu_\alpha^2. \tag{3.25}$$

The proof of this theorem will be given in Section 3.5.

Theorem 3.4 allows us to obtain the main result of our paper, to prove the conjecture presented in [6].

Theorem 3.5. *Under the asymptotic model for an infinite α-full hash table with random probing and RH collision resolution discipline, the variance of the search cost of a random element satisfies*

$$\sigma_\alpha^2 < \frac{\pi^2}{3}. \tag{3.26}$$

Proof. For the function $f(x) = -1 + e^{-x}$ we have

$$g(u) = \int \frac{u}{f(u)} du = -\frac{1}{2}u^2 - u\ln(1 - e^{-u}) + \operatorname{dilog}(1 - e^{-u}), \tag{3.27}$$

where the dilog function is defined as

$$\operatorname{dilog}(x) = \int_1^x \frac{\ln t}{1-t} dt. \tag{3.28}$$

Substituting in (3.24) and simplifying using the identity 27.7.3 from [1],

$$\operatorname{dilog}(x) + \operatorname{dilog}(1-x) = -\ln x \ln(1-x) + \frac{\pi^2}{6}, \tag{3.29}$$

we can show that

$$h(\alpha) = \frac{2\operatorname{dilog}(1-\alpha)}{\alpha} - \frac{(1-\alpha)\ln^2(1-\alpha)}{\alpha^2}. \tag{3.30}$$

Figure 2 compares the exact value of σ_α^2 (computed numerically) and the upper bound $h(\alpha)$. To see that $h(\alpha)$ is an increasing function, we use formula 27.7.2 from [1],

$$\operatorname{dilog}(x) = \sum_{k \geqslant 1} (-1)^k \frac{(x-1)^k}{k^2}, \tag{3.31}$$

to compute its Taylor expansion,

$$h(\alpha) = \frac{4}{3} + \sum_{i \geqslant 1} \frac{2H_{i+1}}{(i+1)(i+2)} \alpha^i, \tag{3.32}$$

where H_i denotes a harmonic number. As all coefficients are positive, $h(\alpha)$ is increasing, and it takes its maximum value at $\alpha = 1^-$, with $h(1^-) = \pi^2/3$. □

This gives us an upper bound of approximately 3.29 for the variance of Robin Hood Hashing. Although a numerically computed value of approximately 1.883 has been known for a long time, this is the first proof that this variance is bounded by a small constant as $\alpha \to 1$.

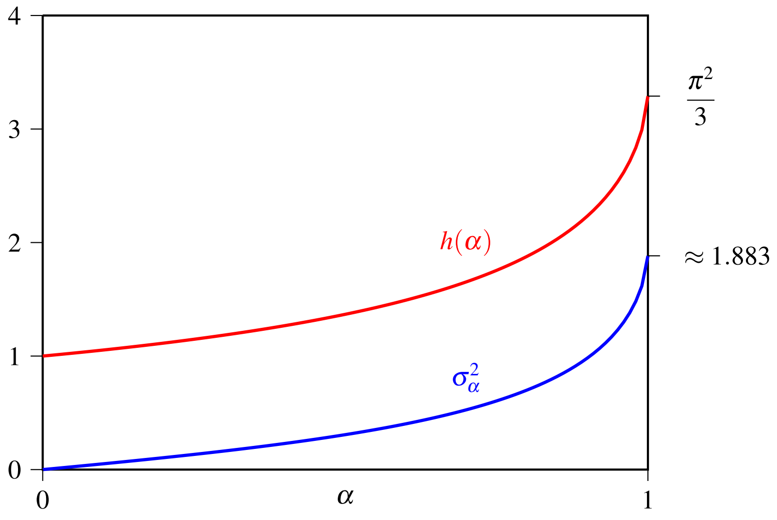

Figure 2. Comparison of σ_α^2 and $h(\alpha)$.

As Celis, Larson and Munro observed, the fact that the variance is very small can be used to carry out a more efficient *mean-centred search*. If we call X the random variable 'search cost of a random key', the expected cost of this modified search is $\Theta(\mathbb{E}|X-\mu_\alpha|)$. But Jensen's inequality implies that

$$\mathbb{E}|X-\mu_\alpha| = \mathbb{E}\sqrt{(X-\mu_\alpha)^2} \leqslant \sqrt{\mathbb{E}(X-\mu_\alpha)^2} = \sigma_\alpha, \tag{3.33}$$

so the *mean value* of the search cost of a mean-centred search is proportional to the *standard deviation* of the cost of a standard search. Theorem 3.5 then implies that this search algorithm runs in expected constant time in a full table.

3.5. Using autonomous equations to prove an upper bound

We will begin by proving some results that we will use to prove Theorem 3.4.

Throughout this section we will consider, as an example, the case of $f(x) = -1+e^{-x}$ with $\overline{\overline{q}}_1 = Q(1) = \ln 1/(1-\alpha)$. The solution to this equation is

$$Q(x) = \ln\left(\frac{1}{1-\alpha} - 1 + e^{x-1}\right) - x + 1. \tag{3.34}$$

Figure 3 compares the solution $\overline{\overline{q}}_i$ (defined only for integer i) of recurrence equation (3.22) to the solution $Q(x)$ of differential equation (3.23). This plot suggests that at each integer point $i \geqslant 1$, $Q(i)$ is an upper bound for $\overline{\overline{q}}_i$.

In what follows, we will be able to show that a stronger result holds. Let us extend $\overline{\overline{q}}_i$ to all $x \geqslant 1$ by linear interpolation, and let us call the resulting function $q(x)$. This function satisfies

$$q(x) = \overline{\overline{q}}_i + (x-i)f(\overline{\overline{q}}_i) \tag{3.35}$$

for all $x \in [i, i+1]$ and $i \geqslant 1$.

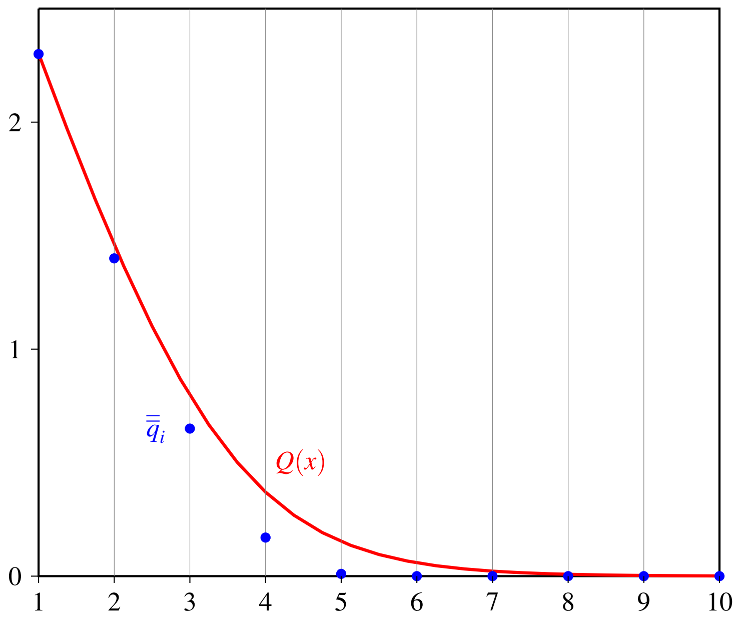

Figure 3. Comparison of $\overline{\overline{q}}_i$ and $Q(x)$ for $\alpha = 0.9$.

Figure 3 compares $q(x)$ and $Q(x)$, and from this we can conjecture that $Q(x) > q(x)$ for all $x > 1$. If we define $d(x) = Q(x) - q(x)$ (shown in Figure 5), this is equivalent to conjecturing that $d(x) > 0$ for all $x > 1$. The following lemma proves this conjecture.

Lemma 3.6. *Let a_i satisfy the recurrence equation $\Delta a_i = f(a_i)$ for all $i \geqslant 1$, and $A(x)$ satisfy the differential equation $A'(x) = f(A(x))$ for all $x \geqslant 1$, where $f : [0, +\infty) \to (-\infty, 0]$ is a continuous strictly decreasing function. Let $a(x)$ be defined by $a(i) = a_i$ for integer i, and $a(x) = a(i) + (x - i) f(a(i))$ for $x \in (i, i+1)$. Then*

$$A(i) \geqslant a(i) \implies A(x) > a(x) \tag{3.36}$$

for all $x \in (i, i+1]$.

Proof. Let us consider x restricted to a given interval $[i, i+1]$ and define $d(x) = A(x) - a(x)$. We will show that $d(i) \geqslant 0 \implies d(x) > 0$ for $x > i$. We note that $A(x)$ and $a(x)$ are strictly decreasing functions, because $f(x)$ is negative, and $d'(x) = f(A(x)) - f(a(i))$ is strictly increasing. Therefore, $d(x)$ is a convex function in the interval $[i, i+1]$.

By hypothesis, $d(i) = A(i) - a(i) \geqslant 0$, and therefore $d'(i) = f(A(i)) - f(a(i)) \leqslant 0$. Moreover, $d(i) = 0$ if and only if $d'(i) = 0$ and, equivalently, $d(i) > 0$ if and only if $d'(i) < 0$. We will consider the following cases.

(a) Case $d(i) = 0$, $d'(i) = 0$. In this case d has a minimum at $x = i$, which implies that $d(x) > 0$ for all other x in the interval. An example of this is the interval [0, 1] in Figure 5.

(b) Case $d(i) > 0$, $d'(i) < 0$. To prove that $d(x) > 0$ in the interval $(i, i+1]$, we reason by contradiction. Assume function d has a leftmost zero at $x^* \in (i, i+1]$. Then d cannot have a

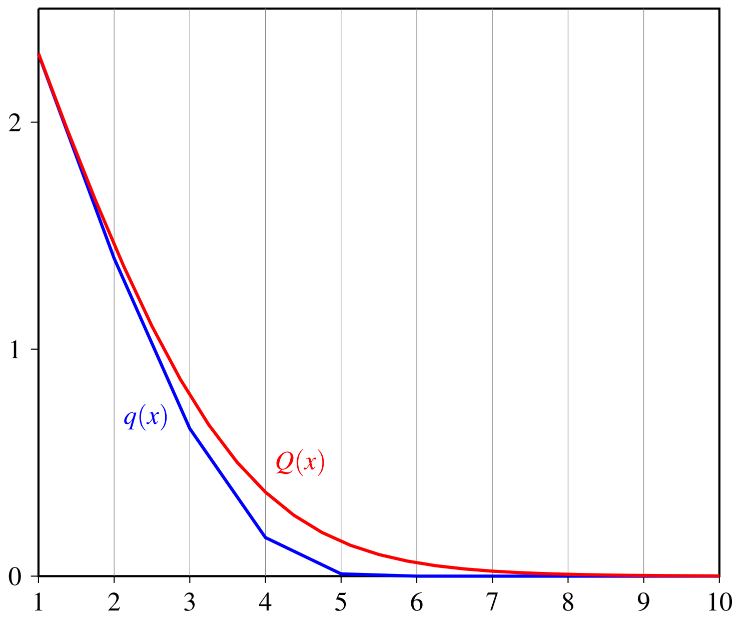

Figure 4. Comparison of $q(x) = \overline{\overline{q}}_i + (x-i)f(\overline{\overline{q}}_i)$ and $Q(x)$ for $\alpha = 0.9$.

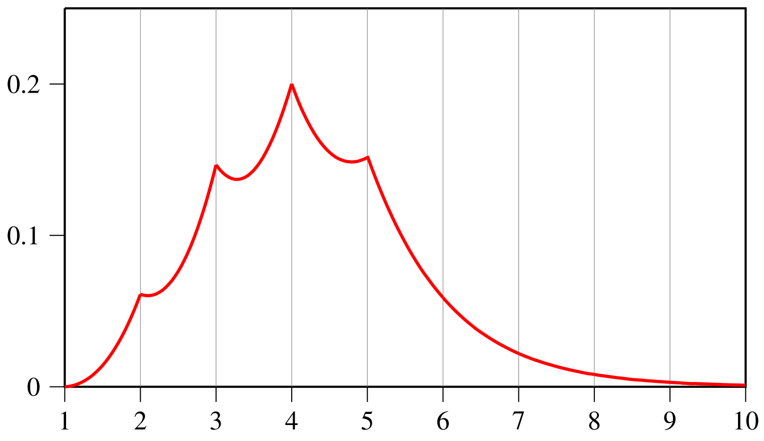

Figure 5. The function $d(x)$ for $\alpha = 0.9$.

minimum in $[i,x^*)$ and therefore $d'(x)<0$ in all that interval, and $d'(x^*)\leqslant 0$ by continuity. On the other hand, $d(x^*)=0$ if and only if $A(x^*)=a(x^*)$. But a is decreasing, therefore $A(x^*)<a(i)$ and $f(A(x^*))>f(a(i))$, or equivalently, $d'(x^*)>0$, a contradiction. Therefore, $d(x)$ cannot have zeros in $[i,i+1]$, and $d(x)>0$ in the interval. □

Corollary 3.7.

$$q(x)<Q(x) \quad \textit{for all } x>1. \tag{3.37}$$

Proof. By induction using $q(1)=Q(1)$ and Lemma 3.6. □

We are now ready to present the proof of Theorem 3.4:

Proof of Theorem 3.4. To bound the right-hand side of equation (3.8) we note that

$$\overline{\overline{\overline{q}}}_1=\frac{\overline{\overline{q}}_1}{2}+\sum_{i\geqslant 1}\frac{\overline{\overline{q}}_i+\overline{\overline{q}}_{i+1}}{2}, \tag{3.38}$$

but the summation is the area under the $q(x)$, which is bounded above by the area under $Q(x)$. Noting that $\overline{\overline{q}}_1=\alpha\mu_\alpha$, we have

$$\sigma_\alpha^2<\frac{2}{\alpha}\int_1^\infty Q(x)dx-\mu_\alpha^2. \tag{3.39}$$

To evaluate the integral, if $g(u)$ is defined as

$$g(u)=\int\frac{u}{f(u)}du,$$

then it is easy to prove by differentiation that

$$\int Q(x)dx=g(Q(x)).$$

The result follows. □

3.6. Bounding the tail of RH

We focus now on the tail of the distribution of the search cost. Recall that X is the random variable ‘search cost of a random element’. Using our notation we have

$$\mathbb{P}\{X\geqslant i\}=\overline{p}_i=\frac{1}{\alpha}\overline{q}_i. \tag{3.40}$$

We proved earlier that $\overline{\overline{q}}_i\leqslant Q(i)$. By applying f to both sides and recalling that f is a decreasing function, we have $f(\overline{\overline{q}}_i)\geqslant f(Q(i))$. Using equations (3.22) and (3.23), we have $\Delta\overline{\overline{q}}_i=-\overline{q}_i\geqslant Q'(i)$, and therefore

$$\mathbb{P}\{X\geqslant i\}=\overline{p}_i\leqslant-\frac{1}{\alpha}Q'(i)=\frac{1}{\alpha+(1-\alpha)e^{i-1}}. \tag{3.41}$$

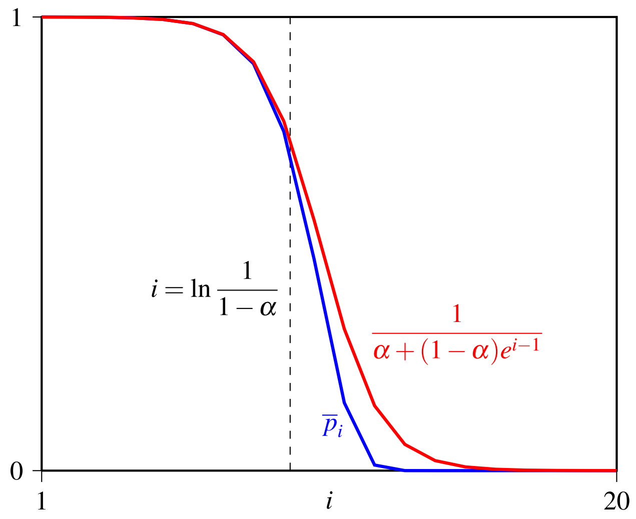

Figure 6. Comparison of $\overline{p}_i$ and the upper bound for $\alpha = 0.9999$.

If in this inequality we substitute $i = \ln 1/(1-\alpha) + k$, we obtain the probability that the search cost will exceed the threshold $\ln 1/(1-\alpha)$ by a given number of steps k,

$$\mathbb{P}\left\{X \geqslant \ln \frac{1}{1-\alpha} + k\right\} \leqslant \frac{1}{\alpha + e^{k-1}} \leqslant e^{-(k-1)}, \tag{3.42}$$

and we can see that the upper bound depends only on k. Therefore, as the table becomes full, the mass of the distribution moves to the right without bound, but it remains tightly packed to the right of the threshold, and the probability that the search cost exceeds it by a given amount decreases exponentially with the distance.

Figure 6 compares the actual value of the tail $\overline{p}_i$ with the value of the upper bound.

Finally, it is interesting to note that the upper bound $1/(\alpha + (1-\alpha)e^{x-1})$ can be interpreted as the tail of a continuous distribution whose density function is

$$\frac{(1-\alpha)e^{x-1}}{(\alpha + (1-\alpha)e^{x-1})^2}, \tag{3.43}$$

and that has its mode at $x = 1 + \ln(\alpha/(1-\alpha))$. If we shift this distribution to the left so it is centred around zero, the density becomes

$$\frac{1}{\alpha}\frac{e^x}{(1+e^x)^2}. \tag{3.44}$$

As $\alpha \to 1$, this converges to $e^x/((1+e^x)^2)$, or equivalently $e^{-x}/((1+e^{-x})^2)$, which is the density function of a Logistic(0,1) distribution [3].

4. Analysis with deletions

We assume a process where we first insert keys until the table reaches load factor α, and then we enter an infinite cycle where we alternate one random insertion followed by one random deletion.

If the distribution of the retrieval cost is given by $p_i(\alpha)$ and a random element is inserted, the effect is described by equation (3.2). If we then perform a random deletion, the following classical lemma[9] shows that the distribution remains unchanged.

Lemma 4.1. *Suppose a set contains n balls of colours* $1, 2, \ldots, k$, *such that the probability that a ball chosen at random is of colour i is* p_i. *Then, if one ball is chosen at random and discarded, the* a posteriori *probability that a random ball is of colour i is still* p_i.

Proof. Call p'_i the probability that a random ball is of colour i after the deletion. The expected number of balls of colour i afterwards is $(n-1)p'_i$, but that number can also be obtained as the expected number before, np_i, minus the expected number of balls of colour i lost, that is,

$$(n-1)p'_i = np_i - 1 \cdot p_i. \tag{4.1}$$

The result follows. □

Therefore, equation (3.2) describes also the probability distribution after one insert–delete step. Now, assume the process reaches a steady state. In that case, the distribution after the insert–delete must be equal to the distribution before, *i.e.* $p_i(\alpha + 1/m) = p_i(\alpha)$, and replacing this in (3.2) we have

$$p_i(\alpha) = t_i(\alpha) - t_{i+1}(\alpha), \tag{4.2}$$

and equivalently

$$\overline{p}_i(\alpha) = t_i(\alpha). \tag{4.3}$$

These equations play the role that equation (3.4) did for the case without deletions. Taking tails in both sides of this equation and setting $i = 1$, we can obtain the expected search cost μ_α as

$$\mu_\alpha = \overline{\overline{p}}_1 = \bar{t}_1 = \frac{1}{1-\alpha}, \tag{4.4}$$

confirming the prediction that the expected successful search cost should approach the expected *unsuccessful* search cost when deletions are allowed.

4.1. Analysis for FCFS

Recall that for FCFS we have $t_i = \alpha^{i-1}$, and therefore (4.2) becomes $p_i = \alpha^{i-1} - \alpha^i$, hence obtaining the following result.

Theorem 4.2. *Under the asymptotic model for an infinite* α*-full hash table with random probing and FCFS collision resolution discipline, in the steady state of a sequence of insert–delete operations, the distribution of the search cost of a random element is given by*

$$p_i = (1-\alpha)\alpha^{i-1} \tag{4.5}$$

and its variance is

$$\sigma_\alpha^2 = \frac{\alpha}{(1-\alpha)^2}. \tag{4.6}$$

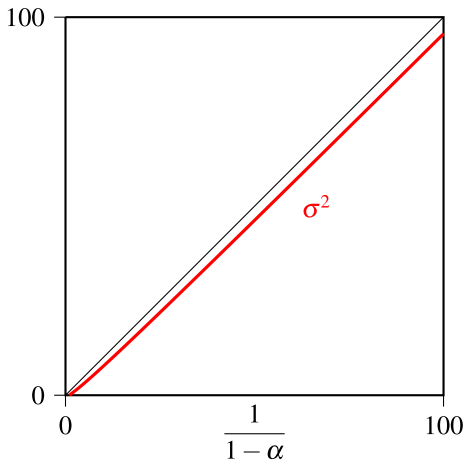

Figure 7. The variance of RH with deletions as a function of $1/(1-\alpha)$.

4.2. Analysis for LCFS

Recall that for LCFS we have $t_{i+1} = q_i/(1-\alpha)$, and therefore (4.2) becomes

$$p_i = \frac{\alpha}{1-\alpha}(p_{i-1} - p_i).$$

Solving this equation, we obtain the following result.

Theorem 4.3. *Under the asymptotic model for an infinite α-full hash table with random probing and LCFS collision resolution discipline, in the steady state of a sequence of insert–delete operations, the distribution of the search cost of a random element is given by*

$$p_i = (1-\alpha)\alpha^{i-1} \tag{4.7}$$

and its variance is

$$\sigma_\alpha^2 = \frac{\alpha}{(1-\alpha)^2}. \tag{4.8}$$

These results, first obtained in ([17], show that when deletions are allowed, FCFS and LCFS become indistinguishable!

4.3. Analysis for RH

For RH, from (4.3) we get $\overline{\overline{p}}_i = \overline{t}_i$, and combining this with (3.19) we obtain

$$\overline{\overline{p}}_1 = \frac{1}{1-\alpha}, \quad \overline{\overline{p}}_{i+1} = \frac{\alpha\overline{\overline{p}}_i^2}{1+\alpha\overline{\overline{p}}_i}. \tag{4.9}$$

We can use this recurrence to compute numerically the distribution for RH.

Figure 7 shows the value of the variance of RH as a function of $1/(1-\alpha)$, and from the plot we may see that the variance is very close to $1/(1-\alpha)$. Moreover, Figure 8 shows the distribution of the search cost for the three methods, for $\alpha = 0.99$. As we have just proved, FCFS and LCFS

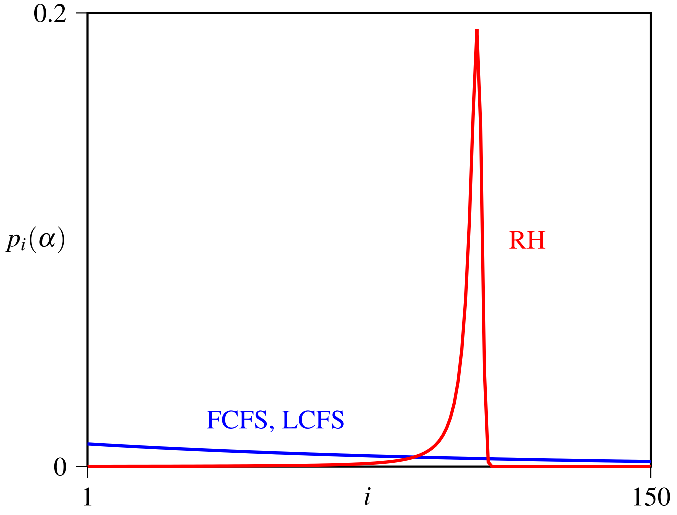

Figure 8. Density function for the distribution of search costs for FCFS, LCFS and RH for $\alpha = 0.99$.

are now identical and have very large dispersion, while RH retains a much more concentrated shape. We prove that this is indeed the case.

4.4. Bounding the variance of RH with deletions

We begin by rewriting the recurrence equation (4.9) as

$$\bar{\bar{q}}_1 = \frac{\alpha}{1-\alpha}, \quad \Delta\bar{\bar{q}}_i = -\frac{\bar{\bar{q}}_i}{1+\bar{\bar{q}}_i}. \tag{4.10}$$

This equation is of the form $\Delta\bar{\bar{q}}_i = f(\bar{\bar{q}}_i)$ for $f(x) = -x/(1+x)$, and therefore we can apply Theorem 3.4 to obtain the following result.

Theorem 4.4. *Under the asymptotic model for an infinite α-full hash table with random probing and RH collision resolution discipline, in the steady state of a sequence of insert–delete operations, the variance of the search cost of a random element satisfies*

$$\sigma_\alpha^2 < \frac{1}{1-\alpha}. \tag{4.11}$$

Proof. For $f(x) = -x/(1+x)$ we have $g(u) = -u(u+2)/2$. The result follows by substituting into formula (3.24). □

This proves our earlier conjecture that the variance was very close to $1/(1-\alpha)$.

5. Conclusions

In this paper we have presented in a unified approach the analysis of hashing algorithms with random probing under the asymptotic model. Three heuristics are analysed in the same way:

FCFS, LCFS and Robin Hood. Moreover the same equation (3.2), with the help of Lemma 4.1, can be used to study these heuristics when deletions are allowed. The new results include very tight bounds on the variance for Robin Hood both with and without deletions, and a complete analysis of FCFS and LCFS when deletions are allowed. Our results agree very well with previous numerical computations.

The key methodological contribution of the paper is Theorem 3.24. The use of autonomous equations has allowed us to formally prove good bounds for sequences that satisfy non linear relations that are very difficult to analyse exactly. Using this, we have obtained a formal proof of a 30 year-old conjecture about the variance of Robin Hood. Furthermore, this approach allows the analysis of the tail distribution of the search cost for Robin Hood when the load factor $\alpha \to 1$. The main algorithmic consequence of our analysis is the proof that the 'organ-pipe' search algorithm presented in [6] gives an average *constant* search time of a random element for *full* tables. Numerical evidence for this had been presented in [6]. The running time of the algorithm is proportional to the standard deviation of the displacement of a random element, and it is proved to be constant in Theorem 3.5.

In our theoretical model we assume that the hash function $h(x)$ used distributes the elements uniformly. That is, if the table has size m, for any element x the probability that $h(x) = k$ is $1/m$ for any table slot k. This is called the uniform hashing assumption. Since the introduction of Universal Hash Functions [5], several researchers have studied to what extent this theoretical ideal can be realized by hash functions that do not take up too much space and can be evaluated quickly. A practical solution has been proposed in [15].

Acknowledgements

We are grateful to the anonymous reviewers for their valuable comments and suggestions, which helped us improve the paper.

References

[1] Abramowitz, M. and Stegun, I. A. (1964) *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*, Courier Corporation.

[2] Amble, O. and Knuth, D. E. (1974) Ordered hash tables. *Comput. J.* **17** 135–142.

[3] Balakrishnan, N. (2013) *Handbook of the Logistic Distribution*, CRC Press.

[4] Brent, R. P. (1973) Reducing the retrieval time of scatter storage techniques. *Comm. Assoc. Comput. Mach.* **16** 105–109.

[5] Carter, J. L. and Wegman, M. N. (1979) Universal classes of hash functions. *J. Comput. Syst. Sci.* **18** 143–154.

[6] Celis, P. (1986) Robin Hood hashing. PhD thesis, University of Waterloo. Technical Report CS-86-14.

[7] Celis, P., Larson, P.-A. and Munro, J. (1985) Robin Hood hashing. In *26th IEEE Symposium on the Foundations of Computer Science*, IEEE, pp. 281–288.

[8] Cunto, W. and Poblete, P. V. (1988) Two hybrid methods for collision resolution in open addressing hashing. In *SWAT 88: Scandinavian Workshop on Algorithm Theory* (R. Karlsson and A. Lingas, eds), Springer, pp. 113–119.

[9] Feller, W. (1968), *An Introduction to Probability Theory and Its Applications*, Vol. 1, Wiley.

[10] Gonnet, G. H. and Munro, J. I. (1979) Efficient ordering of hash tables. *SIAM J. Comput.* **8** 463–478.

[11] Guibas, L. J. (1976) *The Analysis of Hashing Algorithms*, STAN-CS, Stanford University.

[12] Guibas, L. J. (1978) The analysis of hashing techniques that exhibit k-ary clustering. *J. Assoc. Comput. Mach.* **25** 544–555.

[13] Knuth, D. E. (1998), *The Art of Computer Programming*, Vol. 3: *Sorting and Searching*, second edition, Addison-Wesley.

[14] Mitzenmacher, M. (2014) A new approach to analyzing Robin Hood hashing. arXiv:1401.7616

[15] Pagh, A. and Pagh, R. (2008) Uniform hashing in constant time and optimal space. *SIAM J. Comput.* **38** 85–96.

[16] Poblete, P. V. and Munro, J. I. (1989) Last-Come-First-Served hashing. *J. Algorithms* **10** 228–248.

[17] Poblete, P. and Viola, A. (2001) The effect of deletions on different insertion disciplines for hash tables. In *Brazilian Symposium on Graphs, Algorithms and Combinatorics (GRACO)*.