Regular Paper

An Alternative Analysis of Dynamic Hashing Algorithms

Ryozo Nakamura,† Ayad Soufiane†† and Tsuyoshi Itokawa†

Linear and spiral hashing are well-known dynamic hashing methods. Traditional analyses of these two search algorithms have been proposed under the assumption that all keys are uniformly accessed; however, in general the search cost is defined as the product of the number of probes and the frequency of access to a key. Therefore, in this paper, we present a discrete analysis of the average search cost in consideration of the frequency of access to each key for these two dynamic hashing methods. In the proposed discrete analyses, the number of probes itself is regarded as a random variable and its probability distribution is derived. The evaluate formulae derived from the proposed analyses can exactly evaluate the average search cost in conformity with any probability distribution of the frequency of access.

1. Introduction

Hashing is a well-known and efficient technique used mainly in applications that require dictionary operations such as insert, search, and delete. Hashing algorithms can be divided in two principal groups: static and dynamic. In the static scheme, the size of the hash table is invariant, no matter how many records are stored in it. This method has the problem that the cost of all basic operations increases, and the memory space becomes hard to manage when the number of records is not known in advance. To overcome these undesirable features, dynamic hashing methods have been proposed. In these methods, the space address has the property of growing or shrinking in coordination with the number of records stored. The most important points in the construction of such methods is to provide well-designed hash and rehash functions. In addition, fast search should still be possible, and the storage space should be well managed.

Two dynamic hashing algorithms, spiral and linear hashing, have been proposed [2),4),6)], and a mathematical analysis of the average search cost has also been presented. This analysis was derived under the assumption that the frequency of access to each key is uniform, which means that each key in the hash table is probed with equal probability.

The search cost is defined generally as the product of the number of probes and the frequency of access to a key. In this paper, therefore, we propose a discrete analysis of the expected cost of a successful search for both the linear and spiral hashing algorithms, taking account of the frequency of access to each key.

† Department of Computer Science, Faculty of Engineering, Kumamoto University
†† Graduate School of Science and Technology, Kumamoto University

2. Preliminary

Before we turn to the main subject, let us examine more closely the chaining technique, since both linear and spiral techniques of dynamic hashing use the linked list as their basic data structure. We presume that N keys are to be stored in a hash table, indexed by $0, 1, 2, \ldots, M-1$, where M represents the number of buckets in the hash table. Each bucket holds a pointer to a linked list, which contains all the keys that have the same hash address.

2.1 Basic Concepts Used in the Analysis

First, let us assume that we have a hash table of M buckets with N keys stored and that the probability that a key hits bucket i is p_i for $0 \le i \le M-1$, with $\sum_{i=0}^{M-1} p_i = 1$ (the case $p_i = 1/M$ for $0 \le i \le M-1$ has been studied extensively). Next, let P_{Nik} denote the probability that bucket i contains k keys after all N keys have been scattered. Thus we have the following binomial probability distribution:

$$P_{Nik} = \binom{N}{k} p_i^k (1-p_i)^{N-k}. \tag{1}$$

Here, we introduce a generating function $P_{Nik}(z)$ for the probability P_{Nik}:

$$\begin{aligned} P_{Nik}(z) &= \sum_{k=0}^{N} P_{Nik} z^k \\ &= \sum_{k=0}^{N} \binom{N}{k} (p_i z)^k (1-p_i)^{N-k} \end{aligned}$$

$$= (1 + p_i(z-1))^N. \quad (2)$$

From this generating function we get

$$P_{Nik}(0) = (1-p_i)^N \ ,$$
$$P_{Nik}(1) = 1 \ ,$$

and we obtain its derivative as follows:

$$P'_{Nik}(z) = N(1 + p_i(z-1))^{N-1} p_i$$
$$= \sum_{k=1}^{N} \binom{N}{k} k p_i^k (1-p_i)^{N-k} z^{k-1}$$
$$= \sum_{k=1}^{N} k P_{Nik} z^{k-1}. \quad (3)$$

For $z = 1$,

$$P'_{Nik}(1) = \sum_{k=1}^{N} k P_{Nik}$$
$$= N p_i \ . \quad (4)$$

An analysis has been proposed for a more general situation, taking account of the frequency of access to an individual key [5]. In this case, the sequential order of a key's insertion plays a crucial role. Therefore, we must clarify the relationship between the insertion order of a key and its position in a list. For this, we derive the probability $\alpha_{\iota jk}$ that the ι-th key inserted will be located in the j-th position from the head of a list containing k keys. Assuming that keys are inserted successively at the head of a list, the probability $\alpha_{\iota jk}$ is given by

$$\alpha_{\iota jk} = \frac{\binom{N-\iota}{j-1}\binom{\iota-1}{k-j}}{\binom{N-1}{k-1}}. \quad (5)$$

On the other hand, assuming that keys are inserted at the tail of a list, $\alpha_{\iota jk}$ is similarly given by:

$$\alpha_{\iota jk} = \frac{\binom{\iota-1}{j-1}\binom{N-\iota}{k-j}}{\binom{N-1}{k-1}}. \quad (6)$$

If the keys are inserted either at the head or at the tail of a list, we have $\sum_{j=1}^{k} \alpha_{\iota jk} = 1$.

In this paper, a new key is inserted by merely placing it at the head of a list.

In the analysis of the algorithm, let ρ_ι be the probability that the ι-th key inserted will be retrieved, and γ_{kj} be the probability that the j-th key from the head of a list containing k keys is probed. The probability can then be expressed as follows:

$$\gamma_{kj} = \sum_{\iota=1}^{N} \alpha_{\iota jk} \rho_\iota, \qquad \text{where} \ \ k \geq j \ . \quad (7)$$

In the next step, we derive the probability q_{Nh} that the h-th key from the head of a list of any length is probed; namely, q_{Nh} is the probability that the h-th key from the head of a list of length h or longer is probed,

$$q_{Nh} = \sum_{i=0}^{M-1} p_i \sum_{k=h}^{N} \gamma_{kh} P_{Nik} \ , \quad (8)$$

and q_{Nh} effectively represents the probability distribution:

Proof:

$$\sum_{h=0}^{N} q_{Nh} = \sum_{h=0}^{N} \sum_{i=0}^{M-1} \left(p_i \sum_{k=h}^{N} \gamma_{kh} P_{Nik} \right)$$
$$= \sum_{i=0}^{M-1} p_i \sum_{h=0}^{N} \sum_{k=0}^{h} \gamma_{hk} P_{Nih}$$
$$= \sum_{i=0}^{M-1} p_i$$
$$= 1 \ . \quad (9)$$

2.2 Average Search Cost

The final step is to derive the formula for the average search cost S_N, assuming the number of probes h to be a random variable in the case of a successful search.

$$S_N = \frac{\sum_{h=1}^{N} h q_{Nh}}{\sum_{i=0}^{M-1} p_i \sum_{k=1}^{N} P_{Nik}} \ . \quad (10)$$

In the above average search cost, Eq. (10), we indicate that the number of probes itself is regarded as a random variable h. The denominator of Eq. (10) is introduced because only lists containing more than one key must be probed for a successful search.

The notions introduced in section 2 will help us later in our analyses of the algorithms of dynamic hashing: linear and spiral hashing.

3. Linear Hashing

Linear hashing has more flexibility and can be implemented more easily than spiral hashing. If we presume that we have a hash table formed by M buckets, namely $0, 1, \ldots, M-1$, a pointer "p" keeps track of the next bucket to be split. Initially, this pointer travels from bucket 0 to bucket $M-1$, and when the load factor reaches or exceeds a predefined threshold value α, $(\alpha > 0)$, a split is performed: a new bucket is attached to the end of the table and approximately half of the keys in the bucket pointed to are moved to the new bucket. After each split, the pointer points to the next bucket in the table. When all M buckets have been split, the table size has doubled to $2M$, the pointer

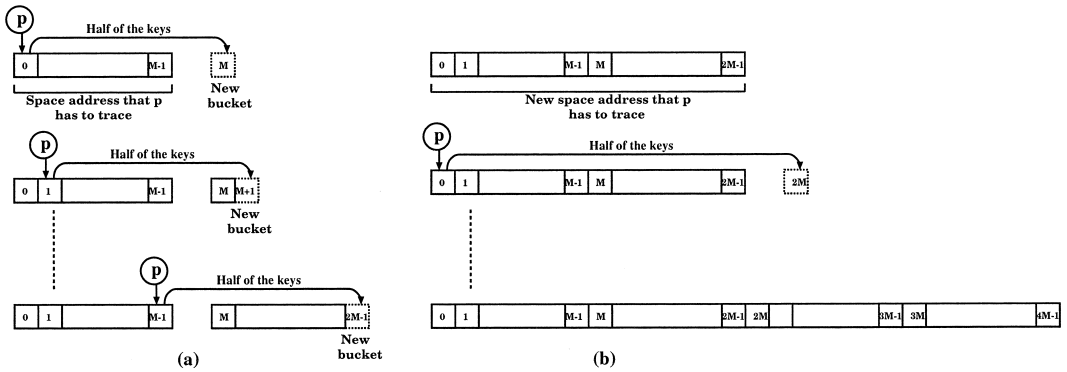

Fig. 1 Splitting process and expansion of the hash table in linear hashing.

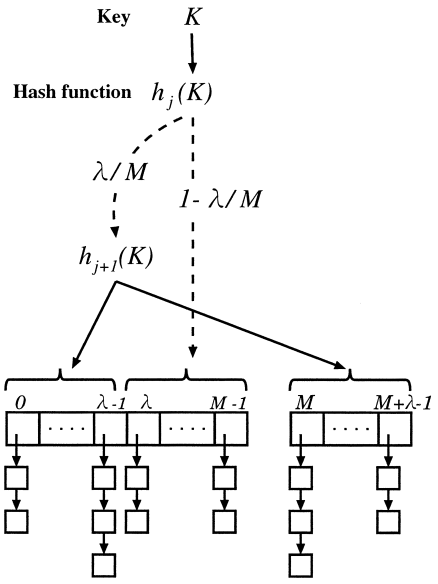

Fig. 2 Illustration of the process of inserting keys.

is reset to zero, and the splitting process starts over again. This splitting process is illustrated in **Fig. 1**.

In the insertion algorithm, we assume that j is the number of times the table size has doubled, λ is the number of buckets that have been split, and M is the number of buckets at the initial step of the current process. We show in **Fig. 2** that a new key K is to be inserted into the hash table. First, a hash address is computed by means of the hash function $h_j(K)$. If this key falls into one of the buckets that are not yet split, then the insertion is made normally; otherwise, the hash address is computed by using the hash function $h_{j+1}(K)$.

We now discuss the details of the hash function used for insertion and retrieval of keys. Initially we have a bucket table of size M, and when the number of times the table size has doubled is j, then the hashing functions are given as follows:

$$h_j(K) = K \bmod (2^j * M) \ . \qquad (11)$$

To analyze the linear hashing, we make the following three assumptions:

(1) The keys are uniformly mapped into the hash table by using a hash function.

(2) The overall load factor is defined as the number of keys in the table divided by the current number of buckets. For a large table, the overall load factor will be constant and equal to α (the threshold value). It is also assumed that there are no deletions.

(3) The number of buckets at the initial step of the current process is M, and the number of buckets to have been split is λ.

3.1 Traditional Analysis

The traditional analysis has already proposed the average successful search cost on the assumption that all keys are uniformly accessed; that is, uniform probing is presumed [6].

A table in linear dynamic hashing can be viewed as consisting of two kinds of buckets: the buckets that have not yet been split during the current expansion, and the buckets that have been split plus the new buckets created during the current expansion. If we let $x = \lambda/M$, $0 \le x \le 1$, be the fraction of buckets that have been split during the current expansion, and z be the expected number of keys in each bucket, the expected number of keys in a split or in new bucket is $z/2$. For the overall load factor to be equal α, the following relationship must hold for the number of keys:

$$z\lambda + z(M - \lambda) = \alpha(M + \lambda)$$

and

$$zx + z(1 - x) = \alpha(1 + x) \ .$$

Finally, we obtain the following relationship:

$$z = \alpha(1+x) \ .$$

Here the expected number of keys z in an unsplit bucket grows linearly from α to 2α.

Thus the average successful search cost $S(\alpha, x)$ is [6]

$$\begin{aligned} S(\alpha,x) &= xs(z/2) + (1-x)s(z) \\ &= xs(\alpha(1+x)/2) \\ &\quad +(1-x)s(\alpha(1+x)) \ , \end{aligned} \tag{12}$$

where $s(y)$ is the expected number of comparisons for a successful search in the case of a traditional separate chaining method with load factor y. It has been derived by Knuth as follows [3]:

$$s(y) = 1 + y/2 \ . \tag{13}$$

The average search cost is represented concisely by the function of the load factor α and the fraction of buckets to have been split x as follows:

$$S(\alpha,x) = 1 + \alpha(2 + x - x^2)/4 \ . \tag{14}$$

3.2 Proposed Analysis

The traditional analysis has assumed that all keys in the hash table are probed with equal probability. But in general, the search cost is defined as the product of the number of probes and the frequency of access to a key; therefore, an analysis that takes account of the frequency of access to each key is proposed.

In the case of linear hashing, the probability p_i of Eq. (1) is $1/M$, since the keys are uniformly scattered into the hash table. Thus the probability $p(N,k,M)$ that a list holds k keys out of N, where the hash table contains M buckets, is

$$p(N,k,M) = \binom{N}{k}\left(\frac{1}{M}\right)^k\left(1-\frac{1}{M}\right)^{N-k} . \tag{15}$$

Its generating function $P_{NM}(z)$ can be expressed as follows, from Eq. (2):

$$P_{NM}(z) = \left(1 + \frac{z-1}{M}\right)^N . \tag{16}$$

We first determine the number of keys in both the unsplit buckets and the split buckets. When λ splits are performed, the probability $t_1(w_1, N, \lambda)$ that w_1 out of N keys stored in the overall hash table are stored in buckets that have not yet been split is:

$$t_1(w_1,N,\lambda) = \binom{N}{w_1}\left(\frac{M-\lambda}{M}\right)^{w_1}\left(\frac{\lambda}{M}\right)^{N-w_1} . \tag{17}$$

Similarly, the probability $t_2(w_2, N, \lambda)$ that w_2 out of N keys are stored in buckets that have been split and/or the new buckets created during the current expansion is

$$t_2(w_2,N,\lambda) = \binom{N}{w_2}\left(\frac{\lambda}{M}\right)^{w_2}\left(\frac{M-\lambda}{M}\right)^{N-w_2} . \tag{18}$$

We can now derive the probability $p_1(N,k,\lambda)$ that a list among the buckets that have not yet been split contains k keys as follows:

$$\begin{aligned} p_1(N,k,\lambda) &= \sum_{w_1=k}^{N} t_1(w_1,N,\lambda)\ p(w_1,k,M-\lambda) \\ &= \sum_{w_1=k}^{N}\binom{N}{w_1}\left(\frac{M-\lambda}{M}\right)^{w_1}\left(\frac{\lambda}{M}\right)^{N-w_1} \\ &\quad *\binom{w_1}{k}\left(\frac{1}{M-\lambda}\right)^k\left(1-\frac{1}{M-\lambda}\right)^{w_1-k} \\ &= \sum_{w_1=k}^{N}\binom{N}{w_1}\binom{w_1}{k}\frac{(M-\lambda)^{w_1}}{M^{w_1}}\frac{1}{(M-\lambda)^k} \\ &\quad *\left(\frac{M-\lambda-1}{M-\lambda}\right)^{w_1-k}\left(\frac{\lambda}{M}\right)^{N-w_1} \\ &= \sum_{w_1=k}^{N}\binom{N}{k}\binom{N-k}{w_1-k}\frac{1}{M^k}\frac{(M-\lambda)^{w_1-k}}{M^{w_1-k}} \\ &\quad *\left(\frac{M-\lambda-1}{M-\lambda}\right)^{w_1-k}\left(\frac{\lambda}{M}\right)^{N-w_1} \\ &= \binom{N}{k}\left(\frac{1}{M}\right)^k\sum_{w_1=k}^{N}\binom{N-k}{w_1-k} \\ &\quad *\left(\frac{M-\lambda-1}{M}\right)^{w_1-k}\left(\frac{\lambda}{M}\right)^{N-w_1} \\ &= \binom{N}{k}\left(\frac{1}{M}\right)^k\sum_{w'=0}^{N-k}\binom{N-k}{w'} \\ &\quad *\left(\frac{M-\lambda-1}{M}\right)^{w'}\left(\frac{\lambda}{M}\right)^{N-k-w'} \\ &= \binom{N}{k}\left(\frac{1}{M}\right)^k\left(\frac{M-\lambda-1}{M}+\frac{\lambda}{M}\right)^{N-k} \\ &= \binom{N}{k}\left(\frac{1}{M}\right)^k\left(1-\frac{1}{M}\right)^{N-k} . \end{aligned}$$

We can then infer that

$$p_1(N,k,\lambda) = p(N,k,M), \tag{19}$$

and the generating function of $p_1(N,k,\lambda)$ is the same as Eq. (16).

In the same manner, we find the probability $p_2(N,k,\lambda)$ that k keys are stored in a list among those buckets on which a split has been

performed:

$$p_2(N,k,\lambda) = \sum_{w_2=k}^{N} t_2(w_2,N,\lambda)\ p(w_2,k,2\lambda) = \binom{N}{k}\left(\frac{1}{2M}\right)^k\left(1-\frac{1}{2M}\right)^{N-k}. \quad (20)$$

Thus,

$$p_2(N,k,\lambda) = p(N,k,2M), \quad (21)$$

and the generating function of $p_2(N,k,\lambda)$ is

$$P_{N\ 2M}(z) = \left(1+\frac{z-1}{2M}\right)^N. \quad (22)$$

When the frequency of access, ρ_ι ($\iota = 1,2,\ldots,N$), is non-uniform, the probability that the j-th key from the head of a list with k keys will be probed is γ_{kj} (see Eq. (7)).

The probability $q_1(N,k,\lambda)$ that the number of probes to search for those buckets that have not yet been split during the current expansion is then equal to k, and similarly the probability $q_2(N,k,\lambda)$ for those that have been split plus the new buckets created during the current expansion is given as follows:

$$q_1(N,k,\lambda) = \sum_{j=k}^{N}\gamma_{jk}p_1(N,j,\lambda), \quad (23)$$

$$q_2(N,k,\lambda) = \sum_{j=k}^{N}\gamma_{jk}p_2(N,j,\lambda). \quad (24)$$

Finally, let $S(N,\lambda)$ denote the expected cost of a successful search when N keys have been scattered into the hash table and λ buckets have been split. In this case, for a successful search there must be at least one key in a list: hence, the evaluation formula $S(N,\lambda)$ is represented according to Eq. (10) assuming the number of probes k to be a random variable, as follows:

$$S(N,\lambda) = \left(1-\frac{\lambda}{M}\right)\frac{\sum_{k=1}^{N}kq_1(N,k,\lambda)}{\sum_{k=1}^{N}p_1(N,k,\lambda)} + \left(\frac{\lambda}{M}\right)\frac{\sum_{k=1}^{N}kq_2(N,k,\lambda)}{\sum_{k=1}^{N}p_2(N,k,\lambda)}. \quad (25)$$

The above analysis has been derived under the conditions that the frequency of access to each key could be any probability distribution.

3.3 Comparison of the Two Analyses

In this section, the proposed analysis is compared with the traditional one under the assumption that the probability of access to each key is equal. We first give the formula for the average search cost $S(N,\lambda)$, in the case where the frequency of access to keys is assumed to be uniform, $\rho_1 = \rho_2 = \cdots = \rho_N = 1/N$. γ_{jk} in both Eq. (23) and Eq. (24) is transformed as follows. From Eq. (7),

$$\begin{aligned}\gamma_{jk} &= \sum_{\iota=1}^{N}\alpha_{\iota kj}\rho_\iota \\ &= \frac{1}{N}\sum_{\iota=1}^{N}\binom{N-\iota}{k-1}\binom{\iota-1}{j-k}\Big/\binom{N-1}{j-1} \\ &= \frac{1}{N\binom{N-1}{j-1}}\sum_{\iota=1}^{N}\binom{N-\iota}{k-1}\binom{\iota-1}{j-k} \\ &= \frac{\binom{N}{j}}{N\binom{N-1}{j-1}} \\ &= \frac{1}{j}.\end{aligned}$$

Therefore, from Eq. (23), Eq. (24), and Eq. (25), we get

$$S(N,\lambda) = \left(1-\frac{\lambda}{M}\right)\sum_{k=1}^{N}k\sum_{j=k}^{N}\frac{1}{j}p_1(N,j,\lambda)\Big/\sum_{k=1}^{N}p_1(N,k,\lambda) + \frac{\lambda}{M}\sum_{k=1}^{N}k\sum_{j=k}^{N}\frac{1}{j}p_2(N,j,\lambda)\Big/\sum_{k=1}^{N}p_2(N,k,\lambda).$$

From Eq. (19) and Eq. (21),

$$S(N,\lambda) = \left(1-\frac{\lambda}{M}\right)\sum_{k=1}^{N}k\sum_{j=k}^{N}\frac{1}{j}p(N,j,M)\Big/\sum_{k=1}^{N}p(N,k,M) + \frac{\lambda}{M}\sum_{k=1}^{N}k\sum_{j=k}^{N}\frac{1}{j}p(N,j,2M)\Big/\sum_{k=1}^{N}p(N,k,2M). \quad (26)$$

Here, from the generating functions Eq. (16) and Eq. (22), we obtain the following derivatives:

$$P'_{NM}(z) = \frac{N}{M}\left(1+\frac{z-1}{M}\right)^{N-1}$$

and

$$P'_{N\ 2M}(z) = \frac{N}{2M}\left(1+\frac{z-1}{2M}\right)^{N-1} .$$

Therefore,

$$\begin{aligned}\sum_{k=1}^{N} p(N,k,M) &= \sum_{k=0}^{N} p(N,k,M) \\ &\quad -p(N,0,M) \\ &= 1-\binom{N}{0}\left(1-\frac{1}{M}\right)^{N} \\ &= 1-P_{NM}(0) \qquad (27)\end{aligned}$$

$$\begin{aligned}&\sum_{k=1}^{N} k\sum_{j=k}^{N}\frac{1}{j}p(N,j,M) \\ &\quad = \sum_{j=1}^{N}\frac{1}{j}P(N,j,M) + \sum_{j=2}^{N}\frac{2}{j}p(N,j,M) \\ &\qquad + \sum_{j=3}^{N}\frac{3}{j}p(N,j,M) + \cdots \\ &\qquad\quad \cdots + \sum_{j=N-1}^{N}\frac{N-1}{j}p(N,j,M) \\ &\qquad + \frac{N}{N}p(N,N,M) \\ &\quad = p(N,1,M) + \frac{1}{2}p(N,2,M) \\ &\qquad + \frac{1}{3}p(N,3,M) + \cdots \\ &\qquad\quad \cdots + \frac{1}{N-1}p(N,N-1,M) \\ &\qquad\quad + \frac{1}{N}p(N,N,M) \\ &\qquad + p(N,2,M) + \frac{2}{3}p(N,3,M) + \cdots \\ &\qquad\quad \cdots + \frac{2}{N-1}p(N,N-1,M) \\ &\qquad\quad + \frac{2}{N}p(N,N,M) \\ &\qquad \vdots \\ &\qquad + p(N,N-1,M) + \frac{N-1}{N}p(N,N,M) \\ &\qquad + p(N,N,M) \\ &\quad = \frac{(N+1)}{2}p(N,N,M) \\ &\qquad + \frac{N}{2}p(N,N-1,M) \\ &\qquad\quad + \frac{N-1}{2}p(N,N-2,M) \\ &\qquad\quad + \cdots + p(N,1,M)\end{aligned}$$

$$\begin{aligned}&= \frac{1}{2}\left\{\sum_{j=1}^{N} jp(N,j,M) + \sum_{j=1}^{N} p(N,j,M)\right\} \\ &= \frac{1}{2}\left\{P'_{NM}(1) + 1 - P_{NM}(0)\right\}. \qquad (28)\end{aligned}$$

Similarly

$$\sum_{k=1}^{N} p(N,k,2M) = 1 - p_{N\ 2M}(0) \qquad (29)$$

and

$$\begin{aligned}&\sum_{k=1}^{N} k\sum_{j=k}^{N}\frac{1}{j}p(N,j,2M) \\ &\qquad = \frac{1}{2}\left\{P'_{N\ 2M}(1) + 1 - P_{N\ 2M}(0)\right\} .\end{aligned} \qquad (30)$$

Here, using Eqs. (27) through (30), Eq. (26) is rewritten as follows:

$$\begin{aligned}S(N,\lambda) &= \frac{1}{2}\left(1-\frac{\lambda}{M}\right) \\ &\quad * \frac{P'_{NM}(1) + 1 - P_{NM}(0)}{1 - P_{NM}(0)} \\ &\qquad + \frac{1}{2}\left(\frac{\lambda}{M}\right) \\ &\quad * \frac{P'_{N\ 2M}(1) + 1 - P_{N\ 2M}(0)}{1 - P_{N\ 2M}(0)} \\ &= \frac{1}{2}\left(1-\frac{\lambda}{M}\right)\left\{\frac{N/M}{1-(1-1/M)^N} + 1\right\} \\ &\quad + \frac{1}{2}\left(\frac{\lambda}{M}\right)\left\{\frac{N/2M}{1-(1-1/2M)^N} + 1\right\}.\end{aligned}$$

We also approximate $(1-\frac{1}{M})^N \simeq \exp(-\frac{N}{M})$ when $N \to +\infty$; thus, we get

$$\begin{aligned}S(N,\lambda) &\simeq \frac{1}{2}\left(1-\frac{\lambda}{M}\right)\left\{\frac{N/M}{1-\exp(-N/M)} + 1\right\} \\ &\quad + \frac{1}{2}\left(\frac{\lambda}{M}\right)\left\{\frac{N/2M}{1-\exp(-N/2M)} + 1\right\} .\end{aligned} \qquad (31)$$

Here, the fraction of the buckets that have been split is $x = \lambda/M$ and the number of keys N becomes $\alpha(M+\lambda)$. We get $N/M = \alpha(1+x)$, and we can therefore simplify the above formula to

$$\begin{aligned}\tilde{S}(\alpha,x) &\simeq \frac{1}{2}(1-x)\left\{\frac{\alpha(1+x)}{1-\exp(-\alpha(1+x))} + 1\right\} \\ &\quad + \frac{1}{2}x\left\{\frac{\alpha(1+x)/2}{1-\exp(-\alpha(1+x)/2)} + 1\right\}.\end{aligned} \qquad (32)$$

The derived Eq. (32) differs from Eq. (14), which we derived from the traditional analysis in Section 3.1. The difference is caused by the way in which a random variable and its probability distribution are defined, as shown in reference 5; that is, in the proposed analysis the number of probes itself is regarded as a random variable, but in Eq. (13) derived by Knuth in the traditional analysis, the random variable is denoted as the number of probes per key. In addition, in the proposed analysis, only lists containing more than one key are probed for a successful search. Formula (32) is a simple form of formula (25) in which the frequency of access is considered to be uniform. Thus, the average search cost is represented concisely by the function of the load factor α and the fraction of buckets to have been split during the current expansion x.

Following another approach under the assumption that the probability of access to each key is equal, we have already derived the expected number of comparisons of a successful search for a separate chaining method with load factor y as follows [5]:

$$s(y) = \frac{1}{2}\left\{1 + \frac{y}{1-\exp(-y)}\right\} . \qquad (33)$$

The above Eq. (33) differs from Eq. (13) derived by Knuth [3] for the reason given in the above analysis.

If we substitute Eq. (33) into Eq. (12) in the same way as in the traditional analysis, we can also obtain the same formula Eq. (32) in the result.

3.4 Numerical Results

To indicate that the proposed formulae can evaluate the search cost in accordance with any probability distribution of the frequency of access to a key, we consider the following probability distribution for the frequency of access to a key.

(1) Uniform: the probability of access to each key is equal; in this case, the probability ρ_ι satisfies the relation $\rho_\iota = 1/N \ \ (\iota = 1, 2, \ldots, N)$.

(2) Zipf's law: the probability of access to a key is reduced harmonically according to the order in which a key is inserted. The $\rho_\iota = c/\iota \ \ (\iota = 1, 2, \ldots, N)$, where $c = 1/H_N$ and $H_N = \sum_{k=1}^{N} 1/k$.

Table 1 represents a comparison between the formulae (14) and (32). The upper row of each cell presents the cost of a successful search in the case of the traditional analysis, while the lower row presents the cost of a successful search derived by the proposed analysis. The numerical results in Table 1 show that the traditional analysis overestimates to a greater or lesser extent.

We can also see that whenever α increases, the value of $\tilde{S}(\alpha, x)$ also increases, but that the fraction of buckets to have been split x does not have a large influence on the search cost. We can also observe that the average search cost varies cyclically.

Table 2 shows the numerical results obtained by the proposed evaluation formula (25) with the probability distribution given by Zipf's law. The numerical behavior expected in Table 2

Table 1 Comparison of the average number of probes in successful search with uniform probing.

Load Factor α	Fraction of buckets to have been split (x)									
	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1.0	1.500	1.523	1.540	1.553	1.560	1.563	1.560	1.553	1.540	1.523
	1.291	1.307	1.320	1.329	1.336	1.338	1.337	1.332	1.322	1.309
5.0	3.500	3.613	3.700	3.763	3.800	3.813	3.800	3.763	3.700	3.613
	3.017	3.132	3.222	3.286	3.324	3.336	3.323	3.284	3.220	3.131
10.0	6.000	6.225	6.400	6.525	6.600	6.625	6.600	6.525	6.400	6.225
	5.500	5.726	5.902	6.026	6.101	6.126	6.101	6.026	5.900	5.725

Table 2 Average number of probes in successful search with Zipf's law probability distribution ($M = 50$; $N = 50, 100, 150$).

Load Factor α	Fraction of buckets to have been split (x)									
	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1.0	1.1118	1.1118	1.1055	1.0992	1.0928	1.0865	1.0802	1.0738	1.0675	1.0612
2.0	1.241	1.2276	1.2141	1.2007	1.1872	1.1738	1.1603	1.1469	1.1334	1.1200
3.0	1.373	1.3518	1.3305	1.3093	1.2881	1.2668	1.2456	1.2243	1.2031	1.1818

shows that it is possible to evaluate the average search cost appropriately in accordance with the frequency of access to a key by the proposed analysis.

4. Spiral Hashing

Spiral hashing is an interesting extension of the traditional hashing algorithm. In this technique, the data items are mapped into the bucket address space unevenly, with the result that records tend to be stored more densely at the beginning of the bucket address space than at the end. We predefine a value β to be the upper limit of the load factor $\alpha = N/M$. Whenever the load factor reaches or exceeds the predefined constant value β, a split is performed, the keys of the first bucket, which is the densest bucket, are split into g buckets that are created and attached to the end of the table, the integer g is called the growth factor, and the first bucket is deleted from the table. Initially the space address is considered to be formed by $g-1$ buckets, namely $\{1, 2, \ldots, g-1\}$ or equivalently $\{g^0, g^0+1, \ldots, g^1-1\}$. When the bucket load factor α is greater than the predefined value β, bucket 1 is chosen to be split. We create g new buckets, which are buckets g through $2g-1$. The data items that were stored in bucket 1 are rehashed by a new hashing function and stored in one of the new buckets g through $2g-1$. After the split, bucket 1 no longer exists. The current bucket address space is $\{2, 3, \ldots, 2g-1\}$ or equivalently $\{g^c, g^c+1, \ldots, g^{c+1}-1\}$ with $c = \log_g 2$. In general, when bucket λ is chosen to be split in the λ-th split, the bucket address space after λ splits can always be represented in the form $\{g^c, g^c+1, \ldots, g^{c+1}-1\}$ with $c = \log_g(\lambda+1)$ and buckets λg through $(\lambda+1)g-1$ are created [2]. Thus, each time a split is performed, the table will grow by g buckets. The process of splitting and the expansion of the table can be represented as in **Fig. 3**.

The main contribution to the construction of the algorithm is a well-defined hash function that permits an uneven distribution of the keys over the table. A hash function $H(K, \lambda)$ is not only a function of the given key K, but also a function of λ, which is the number of splits that have been performed. It is implemented as follows:

- Map key K uniformly into the interval $[0, 1)$:
  $x = h1(K) \in [0, 1)$, that is, $0 \le h1(K) < 1$.
- Map x uniformly into the interval $[c, c+1)$:
  $y = h2(x) \in [c, c+1)$, where $c = \log_g(\lambda+1)$,
  E.g., $h2(x) = \lceil c - x \rceil + x$.
- Let the hashing function be
  $H(K, \lambda) = \lfloor g^y \rfloor$.

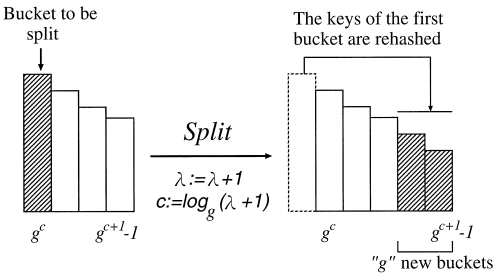

Fig. 3 Splitting process and expansion of the table after λ splits.

The above hashing function has the property that $H(K, \lambda+1) = H(K, \lambda)$ for all buckets $g^c, g^c+1, \ldots, g^{c+1}-1$ which exist at stage λ except bucket $g^c = \lambda+1$, which is no longer in use at stage $\lambda+1$.

4.1 Traditional Analysis

Two traditional analyses of the spiral hashing algorithm have been derived on the assumption that all keys are uniformly accessed. One, an approximate analysis based on a heuristic approach is described in Ref. 6), and the other is given in Ref. 2).

In Ref. 2), the average cost S_N of a the successful search was presented as follows:

$$S_N = \frac{1}{N} \sum_{i=1}^{M} \sum_{k=1}^{N} \frac{k(k+1)}{2} P_{Nik}, \qquad (34)$$

and its approximate formula is represented concisely by the load factor α and the growth factor g as follows:

$$\begin{aligned} S_N &\simeq 1 + \frac{N-1}{2} \left\{ \frac{1}{(\ln\ g)^2} \frac{(g-1)^2}{gM} + O\left(\frac{g^2}{M^2}\right) \right\} \\ &\simeq 1 + \frac{(N-1)(g-1)^2}{2gM(\ln\ g)^2} + O\left(\frac{g^2 N}{M^2}\right) \\ &\simeq 1 + \alpha \frac{(g-1)^2}{2g(\ln\ g)^2}. \end{aligned} \qquad (35)$$

4.2 Proposed Analysis

We propose an alternative analysis of spiral hashing, taking account of the frequency of access to each key. Ours is a discrete analysis in which the number of probes itself is regarded

as a random variable and its probability distribution is derived concretely. The probability q_{Nh} that the number of probes to search a key in the table will be equal to h after λ splits is represented as follows:

$$q_{Nh} = \sum_{i=g^c}^{g^{c+1}-1} p_i \left(\sum_{k=h}^{N} \gamma_{kh} P_{Nik} \right), \qquad (36)$$

where $c = \log_g(\lambda + 1)$, the probability p_i that a key hits bucket i is $p_i = \log_g(i+1) - \log_g(i)$ for $g^c \le i < g^{c+1}$, and P_{Nik} and γ_{kh} are defined in Eq. (1) and Eq. (7), respectively.

The average search cost S_N, taking account of the fact that the number of probes h is a random variable in the successful search, is derived in the same way as Eq. (10), as follows:

$$S_N = \frac{\sum_{h=1}^{N} h q_{Nh}}{\sum_{i=g^c}^{g^{c+1}-1} p_i \sum_{k=1}^{N} P_{Nik}} . \qquad (37)$$

4.3 Approximated Formula for S_N

When the frequency of access to keys is considered to be uniform, i.e., when $\rho_1 = \rho_2 = \cdots = \rho_N = 1/N$, an approximated formula for S_N of Eq. (37) is derived as follows:

$$\begin{aligned} S_N &= \frac{\sum_{h=1}^{N} h q_{Nh}}{\sum_{i=g^c}^{g^{c+1}-1} p_i \sum_{k=1}^{N} P_{Nik}} \\ &= \frac{\sum_{h=1}^{N} \sum_{i=g^c}^{g^{c+1}-1} \sum_{k=h}^{N} h p_i \frac{1}{k} P_{Nik}}{\sum_{i=g^c}^{g^{c+1}-1} p_i \sum_{k=1}^{N} P_{Nik}} \\ &= \frac{\sum_{i=g^c}^{g^{c+1}-1} p_i \sum_{h=1}^{N} h \sum_{k=h}^{N} \frac{1}{k} P_{Nik}}{\sum_{i=g^c}^{g^{c+1}-1} p_i \sum_{k=1}^{N} P_{Nik}} . \end{aligned} \qquad (38)$$

Here we can indicate the difference between Eq. (34), derived by the traditional analysis, and the Eq. (38), derived by the proposed analysis. The difference is caused by the same factors as mentioned previously.

In what follows, we will derive the approximate formula of Eq. (38).

In the above Eq. (38),

$$\begin{aligned} \sum_{k=1}^{N} P_{Nik} &= \sum_{k=1}^{N} \binom{N}{k} p_i^k (1-p_i)^{N-k} \\ &= \sum_{k=0}^{N} \binom{N}{k} p_i^k (1-p_i)^{N-k} \\ &\quad - \binom{N}{0} (1-p_i)^N \\ &= 1 - (1-p_i)^N \\ &= 1 - P_{Nik}(0) \end{aligned} \qquad (39)$$

$$\begin{aligned} \sum_{h=1}^{N} h \sum_{k=h}^{N} \frac{1}{k} P_{Nik} &= \frac{1}{2} \left[\sum_{k=1}^{N} k P_{Nik} + \sum_{k=1}^{N} P_{Nik} \right] \\ &= \frac{1}{2} \left[P'_{Nik}(1) + 1 - P_{Nik}(0) \right] . \end{aligned} \qquad (40)$$

In the above transformation, we have used Eq. (2), Eq. (4), and Eq. (39). Thus Eq. (38) can be rewritten as follows:

$$\begin{aligned} S_N &= \frac{\sum_{i=g^c}^{g^{c+1}-1} p_i \frac{1}{2} \left[P'_{Nik}(1) + 1 - P_{Nik}(0) \right]}{\sum_{i=g^c}^{g^{c+1}-1} p_i \left[1 - P_{Nik}(0) \right]} \\ &= \frac{1}{2} \left\{ \sum_{i=g^c}^{g^{c+1}-1} p_i P'_{Nik}(1) \right. \\ &\quad \left. + \sum_{i=g^c}^{g^{c+1}-1} p_i \left(1 - P_{Nik}(0) \right) \right\} \\ &\quad \Big/ \sum_{i=g^c}^{g^{c+1}-1} p_i \left(1 - P_{Nik}(0) \right) \\ &= \frac{1}{2} \left\{ \frac{\sum_{i=g^c}^{g^{c+1}-1} p_i P'_{Nik}(1)}{\sum_{i=g^c}^{g^{c+1}-1} p_i \left(1 - P_{Nik}(0) \right)} + 1 \right\} \\ &= \frac{1}{2} N \frac{\sum_{i=g^c}^{g^{c+1}-1} p_i^2}{\sum_{i=g^c}^{g^{c+1}-1} p_i \left(1 - (1-p_i)^N \right)} + \frac{1}{2} . \end{aligned} \qquad (41)$$

Here we will approximate the above Eq. (41). First we use the following approximation given in Ref. 2):

$$\begin{aligned} \sum_{i=g^c}^{g^{c+1}-1} p_i^2 &= \sum_{i=\frac{M}{g-1}}^{\frac{gM}{g-1}-1} \left(\log_g (1 + \frac{1}{i})^2 \right) \\ &\simeq \frac{1}{(\ln(g))^2} \frac{(g-1)^2}{gM} , \end{aligned} \qquad (42)$$

where $M = g^{c+1} - g^c = (g-1)g^c$.

Next, we approximate $(1-p_i)^N \simeq \exp(-Np_i)$ when $N \to +\infty$. Thus

$$\begin{aligned} &\sum_{i=g^c}^{g^{c+1}-1} p_i (1 - (1-p_i)^N) \\ &\quad \simeq \sum_{i=g^c}^{g^{c+1}-1} p_i (1 - \exp(-Np_i)) \\ &\quad = \sum_{i=g^c}^{g^{c+1}-1} p_i - \sum_{i=g^c}^{g^{c+1}-1} p_i \exp(-Np_i) \end{aligned}$$

$$= 1 - \sum_{i=g^c}^{g^{c+1}-1} p_i \exp\left(-Np_i\right) . \quad (43)$$

Furthermore, we use the approximation $p_i = \frac{1}{\ln(g)} \ln(1 + \frac{1}{i}) \simeq \frac{1}{i \ln(g)}$. Thus we have

$$\sum_{i=\frac{M}{g-1}}^{\frac{gM}{g-1}-1} p_i \exp(-Np_i) \simeq \sum_{i=\frac{M}{g-1}}^{\frac{gM}{g-1}} \frac{1}{i \ln(g)} \exp\left(-\frac{N}{i \ln(g)}\right) . \quad (44)$$

In the above equation, we also approximate the summation on the right-hand side by the summation of a value of its middle term $\left(i = \frac{M(g+1)}{2(g-1)}\right)$; thus,

$$\sum_{i=\frac{M}{g-1}}^{\frac{gM}{g-1}} \frac{1}{i \ln(g)} \exp\left(-\frac{N}{i \ln(g)}\right) \simeq \frac{M}{\ln(g)} \frac{2(g-1)}{M(g+1)} \exp\left(-\frac{N}{\ln(g)} \frac{2(g-1)}{M(g+1)}\right) = \frac{2(g-1)}{\ln(g)(g+1)} \exp\left(-\frac{2\alpha(g-1)}{\ln(g)(g+1)}\right) . \quad (45)$$

Finally, we obtain an approximation of S_N using Eqs. (42) through (45), as follows:

$$S_N \simeq \frac{1}{2} \frac{\alpha(g-1)^2(g+1)}{g \ln(g)} \Bigg/ \left\{ (g+1)\ln(g) - 2(g-1) \exp\left(-\frac{2\alpha(g-1)}{(g+1)\ln(g)}\right) \right\} + \frac{1}{2} . \quad (46)$$

The above approximate formula Eq. (46) is represented concisely by the load factor α and the growth factor g in the same way as in the traditional analysis.

4.4 Numerical Results

We show that the proposed Eq. (37) can evaluate the average search cost in accordance with any probability distribution of the frequency of access to each key. We consider the same probability distribution for the frequency of access to each key as mentioned in section 3.4.

Figures 4 and **5** show the numerical results obtained by the proposed Eq. (37), with uniform and Zipf's law probability distributions for the frequency of access to the key. These graphs of each figure show the average search cost for the different values assigned to g and for $\beta = 1$.

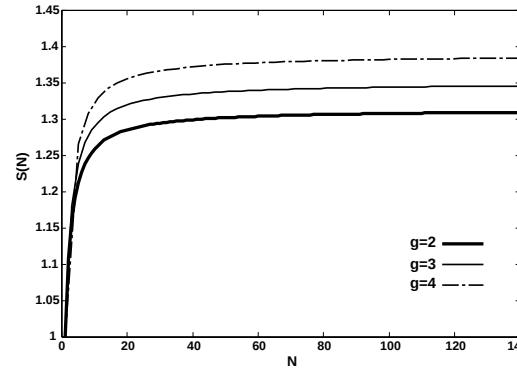

Fig. 4 Average successful search cost for uniform probing: $\beta = 1$, $g = 2, 3, 4$, and $\rho_\iota = 1/N$ for $\iota = 1, 2, \ldots, N$.

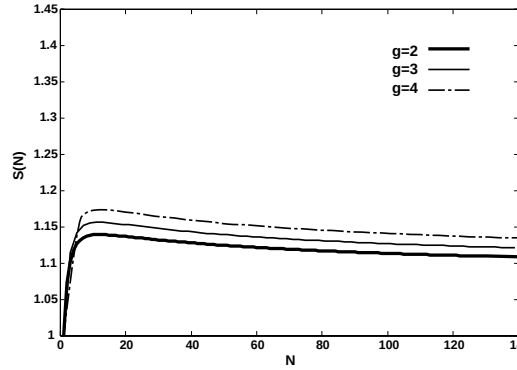

Fig. 5 Average successful search cost for Zipf's law probability distribution: $\beta = 1$, $g = 2, 3, 4$, and $\rho_\iota = c/\iota$ for $\iota = 1, 2, \ldots, N$, where $c = 1/H_N$ and $H_N = \sum_{k=1}^{N} 1/k$.

From these graphs we can see that the average search cost for spiral hashing will reach a steady state very fast, and that the average search cost will increase if the value of g increases.

The numerical results of the simplified formula in Eq. (46) for the average successful search cost are shown in **Fig. 6**. Comparing these graphs to those in Fig. 4 we can see that the simplified formula Eq. (46) is a good approximation of Eq. (38).

5. Conclusions

We have proposed a discrete analysis of the average cost of successful searching using two dynamic hashing algorithms, linear hashing and spiral hashing, taking account of the frequency of access to an individual key.

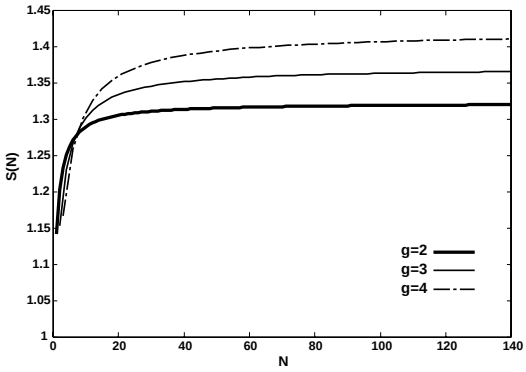

Fig. 6 Approximated average successful search cost for uniform probing: $\beta = 1$, $g = 2, 3, 4$, and $\rho_\iota = 1/N$ for $\iota = 1, 2, \ldots N$.

In the analysis, we clarified the relationship between the sequential order in which a key is inserted and its position in a list.

The proposed analysis make it possible to evaluate the average search cost for both algorithms in accordance with any probability distribution of the frequency of access to a key. The difference between the formulae derived by the proposed analysis and those obtained by the traditional analysis is caused by the way in which a random variable and its probability distribution are defined. In the proposed analysis, the number of probes is regarded as a random variable and its probability is derived; in addition, only lists containing more than one key must theoretically be probed for a successful searching.

Acknowledgement

The authors would like to thank the anonymous referees, editors and adviser for their valuable comments to improve the clarity of this paper.

References

1) Knott, G.D.: Hashing functions, *The Computer Journal*, Vol.18, No.3, pp.265–278 (1972).
2) Chu, J.H. and Knott, G.D.: An analysis of spiral hashing, *The Computer Journal*, Vol.37, No.8, pp.715–719 (1994).
3) Knuth, D.E.: *The art of computer programming, Vol.3: Sorting and searching*, 2nd edition, Addison Wesley (1998).
4) Mullin, J.K.: Spiral storage: Efficient dynamic hashing with constant performance, *The Computer Journal*, Vol.28, No.3, pp.330–334 (1985).
5) Nakamura, R.: An alternative analysis of the algorithm for separate chaining technique of the hashing method, *Journal of the IPSJ*, Vol.34, No.1, pp.10–15 (1993).
6) Larson, P.A.: Dynamic hash tables, *Comm. ACM*, Vol.31, No.4, pp.446–457 (1988).
7) Sedgewick, R. and Flajolet, P.: *An introduction to the analysis of algorithms*, Addison-Wesley (1996).
8) Ramamohanarao, K. and Lloyd, J.W.: Dynamic hashing schemes, *The Computer Journal*, Vol.25, No.4, pp.478–485 (1982).

(Received July 21, 2000)
(Accepted January 7, 2003)

Ryozo Nakamura received the M.E. degree from Kumamoto University in 1968 and the D.E. degree in computer science from Kyushu University in 1985. From 1968 to 1974, he joined Chubu Electric Power Company. Since 1975 he has joined in Faculty of Engineering of Kumamoto University, and is presently a professor in Department of Computer Science. His current research interests include the design and analysis of algorithms and data structures.

Ayad Soufiane received the B.E. degree in Operational Research and Statistics from Mohammed 1st University (Morocco) in 1994, and received the M.E. in Computer Science from Kumamoto University in 1998. He is presently working for the D.E. degree in the Graduate School of Science and Technology, Kumamoto University. His current research interests include the design and analysis of algorithms and data structures.

Tsuyoshi Itokawa received the B.E. degree from Kumamoto University in 1993. From 1993 to 1994 he joined Fujitsu Kyushu System Engineering LTD., and received the M.E. degree from Kumamoto University in 1997, and the D.E. degree from Kumamoto University in 2001. He is presently a research associate in Department of Computer Science. His current research interests include the design and analysis of algorithms and data structures.