Nordic Journal of Computing

ANALYSIS OF LINEAR HASHING REVISITED

RICARDO A. BAEZA-YATES
Depto. de Ciencias de la Computación
Universidad de Chile
Blanco Encalada 2120
Santiago, Chile
rbaeza@dcc.uchile.cl

HÉCTOR SOZA-POLLMAN
Depto. de Ciencias de la Computación
Universidad Católica del Norte
Av. Angamos 0610, Casilla 1280
Antofagasta, Chile
hsoza@socompa.cecun.ucn.cl

Abstract. In this paper we characterize several expansion techniques used for linear hashing and we present how to analyze any linear hashing technique that expands based on local events or that mixes local events and global conditions. As an example we give a very simple randomized expansion technique, which is easy to analyze and implement. Furthermore, we obtain the analysis of the original hashing technique devised by Litwin, which was unsolved until now, comparing it to the later and more widely used version of Larson's. We also analyze one hybrid technique. Among other results, it is shown that the control function used by Litwin does not produce a good storage utilization, matching known experimental data.

CR Classification: F.2.2, E.5, E.2.

Key words: external hashing, linear hashing, analysis of algorithms, optimal bucketing.

1. Introduction

External hashing is a very efficient technique used to obtain a fast organization and retrieval of information in big size files whose contents change dynamically [5]. Among the different schemes developed in the last decade there is Linear Hashing, due to Litwin [8], with two variants: globally controlled or locally controlled bucket division, depending on whether the storage utilization of the file is fixed or not. A generalization of controlled division, called Linear Hashing with partial expansions, corresponds to Larson [6]. Based on the latter, other methods of hashing have been developed, called dynamic hashing [9, 4], which allow supporting modern database systems, since they provide flexibility for handling dynamic files and preserve the very fast expected time of hashing access.

Received January 9, 1998.

We first classify in three classes all the control functions that have been proposed for Linear Hashing and we analyze several of them. Until today, only the scheme proposed by Larson [7] was analytically studied expressing the results as infinite sums that can be numerically computed. In this paper we give a simple differential equation that allows to analyze a class of linear hashing techniques. We apply this equation to a simple randomized technique and to Litwin's original scheme. For this purpose, the mathematical model developed by Larson [7] is used for the analysis, also obtaining infinite sums (seems difficult to obtain real closed forms for the results). Furthermore, the optimal size of the overflow bucket is computed for Larson's control function in typical cases. For the analysis, one (or several) differential equations had to be solved, with boundary conditions over an unknown parameter, which led to the use of several numerical techniques, such as the Runge-Kutta method to solve differential equations, the Newton method to determine roots, numerical integration and the Maple V symbolic algebra system [2]. A simulation of the methods studied was developed and the results agreed with the ones obtained analytically. Some of these results are part of Soza-Pollman [10].

2. Linear Hashing

Linear Hashing refers to the dynamic hashing algorithm which allows storing records in a file without changing significantly its access time, independently of the number of record insertions or deletions that occur in it. The access time of a hashing algorithm is understood as the number of accesses to secondary memory needed in order to find a record in the file.

The records have an identifier called primary key and they are inserted, according to a given hashing function, into a hash table where each entry is a called a *primary bucket* (which has a capacity for b records). We define a *bucket* as a unit of storage in secondary memory (consisting of one or more physical disk pages). When a primary bucket is full, it is said that there is an *overflow*, and an overflow resolution method is used, consisting of chaining a new bucket to the primary one, called *overflow bucket* (which has a capacity for c records), and the new record (called overflow record) is added to this one. More overflow buckets are created if needed, chaining them. All the records in a primary bucket and its overflow buckets are called a *group* (that is, all the records whose keys fall in the same hash table entry).

If all the overflows are resolved just creating overflow records, the access time deteriorates quickly. In order to avoid this, a file expansion is made to decrease the number of overflow records, allowing to keep the method performance at the desired level. This is achieved by adding a bit to the hashing function which affects only the group that we decide to be expanded (not necessarily the group with the primary bucket in overflow) and leaves the others intact [7]. During the growth of the file, the groups are divided in a sequence ranging between 0 and $N-1$, N being a power of 2. When

the group j is divided, the hash table entry $j + N$ is added to the file and the records are distributed between both entries, thus transferring about half of records of group j into the entry $j + N$. This process ends when the group $N - 1$ is divided, and the file reaches size $2N$, which corresponds to a complete expansion, and the process may start again. We use the variable x to denote the fraction of groups that has been expanded. An extension, devised by Larson [7], is linear hashing with partial expansions, where a complete file expansion is reached through an arranged succession of n_0 partial expansions before duplicating the file size, $n_0 \geq 1$.

To control the file size, we define the load factor and the storage utilization as follows (see Table Table I):

$$z = \frac{n}{b\ N} \quad , \qquad U = \frac{n}{b\ N_{bp} + c\ N_{ov}}$$

The file expansion is directed by a control function, which is a set of rules used to determine when an expansion takes place, from which a relationship can be established between the load factor z and the fraction x of expanded groups of the file. That is, $x = g(z)$, a function to be determined. Some of these control functions are:

(1) If an overflow occurs after inserting a record [8]. It has been studied only through experimental analysis.

(2) If the storage utilization exceeds a given limit when an overflow takes place [8], which has been analyzed experimentally.

(3) If the storage utilization exceeds a given limit, which was analytically and experimentally analyzed by Larson [7]. It is called constant or limited storage utilization control function.

(4) If the expected successful search cost exceeds a given limit [1]. It has been empirically analyzed.

(5) If the number of inserted records reaches a multiple of a fixed constant K.

Figure Fig. 1 shows the expansion of a file using linear hashing and control function 1. We can divide control functions in three classes:

I. Based on local events (originally called uncontrolled case). This is the case of functions 1 and 5. We give the differential equation that allows analyzing any hashing technique on this class.

II. Based on a condition for a global performance measure (also called the controlled case). This is the case of control functions 3 and 4. The analysis for this class follows Larson [7], that basically solves the equation given by the global condition. We give an example at the end of the next section.

III. Hybrid control function. Both a local and a global condition are used, as in control function 2. We also outline the generic solution for this case through an example.

In this paper we present the analysis for control functions 1 and 2, a randomized version of function 5 and we extend the analysis of function 3. The detailed analysis of control functions 1, 2 and 4 is part of Soza-Pollman [10].

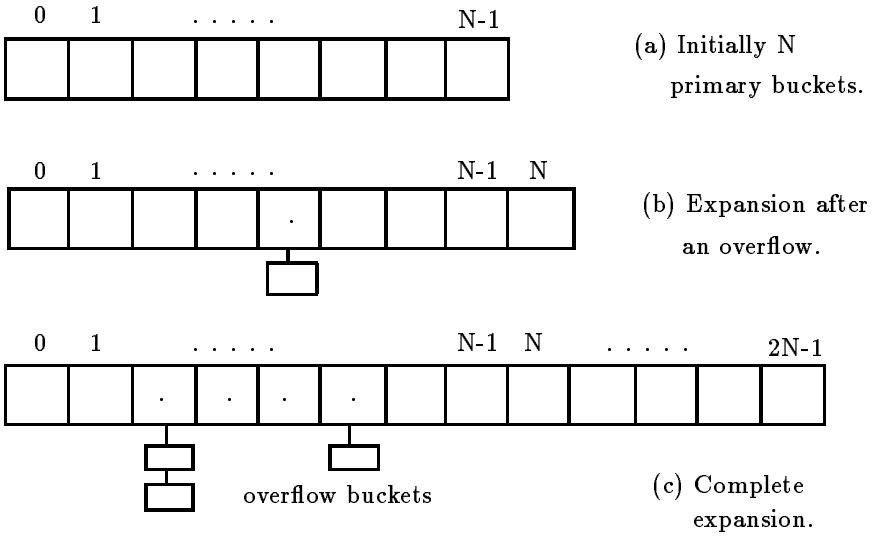

Fig. 1: Linear Hashing using control function 1.

Symbols	Meaning
N	initial number of primary buckets (and groups)
N_{bp}	number of primary buckets
N_{ov}	number of overflow buckets
n	number of inserted records
b	primary bucket size in records
c	size of the overflow bucket in records
z	file load factor
x	fraction of expanded groups
U	storage utilization
S	expected number of accesses in a successful search
$E[S]$	average value of S over a complete expansion
I	expected number of accesses in an unsuccessful search
V	expected overflow space used per record

TABLE I: Symbols used in the model and analysis.

3. Analysis of Linear Hashing

We use the same notation and formulas as Larson [7], formalizing some of them. Let y be the expected number of records per group. Then the load factor over the file is characterized by $z = y/b$. Note that z is the expected value of a random variable. An exact analysis might be possible, but this simplification gives solutions that agree very well with experimental data as shown later.

The probability that i records locate themselves in a group, given z, is modeled by a Poisson distribution, which implies a uniform probability of reaching each bucket (this is a good approximation to the exact distribution, which is Binomial). That is:

$$P(i,z) = \frac{e^{-bz}(bz)^i}{i!} \quad , \quad i = 0, 1, ...$$

The probability of having an overflow in an insertion will be given by:

$$p_0(z) = \sum_{i \geq b} P(i,z) = 1 - \sum_{i=0}^{b-1} \frac{e^{-bz}(bz)^i}{i!}$$

Since a bucket with $(k-1)c + j$ overflow records $(1 \leq j \leq c)$ needs k overflow buckets $(k \geq 1)$, then the expected number of record slots stored by each group (primary and overflow) is calculated by using:

$$t(z) = b + c \sum_{k=1}^{\infty} k \sum_{j=1}^{c} P(b + (k-1)c + j, z) \;=\; b + c \sum_{j=b+1}^{\infty} \left\lceil \frac{j-b}{c} \right\rceil P(j,z)$$

On the other hand, a successful search requires at least one access, and the expected number of accesses in a successful search, in a group with $(k-1)c + j$ overflow records $(1 \leq j \leq c \ , \ k \geq 1)$, is:

$$s(z) = 1 + \frac{1}{bz} \sum_{k=1}^{\infty} k \sum_{j=1}^{c} \left(\frac{(k-1)c}{2} + j \right) P(b + (k-1)c + j, z)$$

For unsuccessful searches the probability of accessing a given group does not depend on the number of records in the group. Then, an unsuccessful search in a group with $(k-1)c + j$ $(1 \leq j \leq c, \ k \geq 1)$ overflow records, has an expected number of accesses of:

$$i(z) = 1 + \sum_{k=1}^{\infty} k \sum_{j=1}^{c} P(b + (k-1)c + j, z) \;=\; 1 + \sum_{j=b+1}^{\infty} \left\lceil \frac{j-b}{c} \right\rceil P(j,z)$$

The storage utilization, that is the amount of space used with respect to the total space allocated is computed as follows. The expected number of records per group is bz, for the expanded groups as well as for the non

expanded ones. Recall that x is the fraction of groups that have been expanded. Then, the expected number of record slots allocated per expanded group is $2x\ t(z/2)$ and $(1-x)\ t(z)$ for the other groups. Then, the storage utilization is given by:

$$U(z,x) = \frac{bz}{2x\ t(z/2) + (1-x)\ t(z)}$$

Note that this definition is an approximation of the storage utilization because the expected value of the quotient of two random variables is not the quotient of the expected values. However, using the Kantorovich inequality [3] we know that $U(z,x)$ is a lower bound for the exact value and our experimental results show that in practice is very close to it.

Likewise, a search in a group already expanded requires $s(z/2)$ accesses on average, whereas if it is from the non expanded group, it needs $s(z)$ accesses on average. Then, the expected number of accesses for a successful search is:

$$S(z,x) = S(z,g(z)) = x\ s(z/2) + (1-x)\ s(z)$$

In order to analyze the existence of the optimal overflow bucket size, the formula of average successful search length during a complete expansion is needed. The corresponding formula is:

$$E[S] = \frac{1}{z_0} \int_{z_0}^{2z_0} S(z,g(z))\ dz$$

Similarly, the expected number of accesses for an unsuccessful search is:

$$I(z,x) = x\ i(z/2) + (1-x)\ i(z)$$

Finally, the expected amount of overflow space required (used) per record is:

$$V(z,x) = \frac{2x\ (t(z/2)-b) + (1-x)\ (t(z)-b)}{bz}$$

The formulas above depend on x and z (see Table Table I), but these variables, as already mentioned, are not independent. Each control function defines a relationship between x and z. For example, in the case of constant utilization [7] the storage utilization is always maintained as close as possible to a given value α, $0 < \alpha < 1$. In the asymptotic case this leads to constant storage utilization, that is, $U(z,x) = \alpha$ is verified. By solving this equation for x, the following relationship between x and z is obtained (in the original paper of Larson [7] there is a typo for this equation):

$$x = g(z,\alpha) = \frac{zb/\alpha - t(z)}{2t(z/2) - t(z)}$$

Considering $b = 10$, $c = 5$ and $\alpha = 0.8$ an initial load factor of $z_0 = 0.9644$ is obtained. This is an example of analysis for a control function of class II. In figure Fig. 2 (left) a graph is presented showing the relationship between z and x, considering $z_0 \leq z \leq 2z_0$ and $0 \leq x \leq 1$.

Another control function that belongs to this class is when we limit the expected number of accesses in a successful search. That is, we want to have $S(z, x) \leq 1 + \beta$ where $\beta > 0$ is a parameter defined by the user. Again, in this case there is a trade-off between storage utilization and search time as in control function 3, but here the file expansion is controlled by the search cost. However, now the relation between x and z that is reached in the limit when $S(z, x) = 1 + \beta$ is not simple as S is an infinite sum. The analysis of this case using a numerical technique can be found in Soza-Pollman [10].

4. Control Functions Based on Local Events

Let $P_E(z, x)$ be the probability that an insertion triggers an expansion when the file has load factor z and a fraction x of it has been expanded. The next theorem shows how the rate of expansion and the load factor are related to this probability.

THEOREM 1. *The relationship between the load factor and the expansion ratio of the file in the case of a class I control function is given by the equation:*

$$\frac{dx}{dz} = b\, P_E(z, x)$$

with boundary conditions $x(z_0) = 0$ *and* $x(2z_0) = 1$. *For the boundary conditions,* z_0 *is the initial load factor in a period, due to which* $x(z_0)$ *and* $x(2z_0)$ *represent the fraction of expanded buckets at the beginning and at the end of the expansion, respectively.*

PROOF. Suppose that we have a file with N groups, having n records, and with a fraction x of expanded groups. Let z be the load factor before the insertion, given by $z = \frac{n}{b\,N}$. Let z' be the load factor after the insertion, that is $z' = \frac{(n+1)}{N\,b}$. Then the growing rate of z is:

$$\triangle z = z' - z = \frac{1}{N\,b}$$

Note that the load factor is defined with respect to the initial number of groups at the beginning of the expansion process. The file expansion ratio, before the insertion is x, and afterwards it is:

$$x' = x + \frac{P_E(z, x)}{N}$$

Thus, the rate of expansion $\triangle x$ is:

$$\triangle x = x' - x = \frac{P_E(z, x)}{N}$$

Dividing $\triangle x$ by $\triangle z$ we obtain:

$$\frac{\triangle x}{\triangle z} = b\ P_E(z, x)$$

In the limit the stated differential equation is obtained. □

As an example of a first use of the equation above, we analyze a variant of the control function 5. This function (expand after every K insertions) can be approximated using the following simple randomized scheme: expand after each insertion with probability $1/K$. That is, $P_E(z, x) = 1/K$. This gives a very simple expansion rate which is linear:

$$x(z) = \frac{b}{K} z - 1$$

with $z_0 = K/b$. So the load factor grows from K/b to $2K/b$. If $K = 1$ we have the lowest possible utilization. On the other hand if K is large with respect to b, the utilization is very good but the insertion time increases. So, with just choosing K we can obtain a desired performance. For a given K and b it is possible to obtain the optimal value of c to maximize U as given in section 8 for control function 3.

5. Litwin's Control Function

In this section the original control function of Litwin is analyzed [8], which expands a file when there is an overflow. This is based on the general equation for control functions based on local events shown previously. In the case of Litwin's scheme, we have

$$P_E(z, x) = P_{ov}(z, x) = x\ p_0(z/2) + (1 - x)\ p_0(z)$$

where $p_0(z)$ is the probability of having overflow in an insertion. Applying the main theorem, we obtain

$$\frac{dx}{dz} = bx(\ p_0(z/2) - p_0(z)) + b\ p_0(z)$$

The resulting differential equation has the problem of depending on boundary conditions defined over an a priori unknown value z_0. Starting with a given z_0 (a guess), the equation is solved using the Runge-Kutta numerical method. Next, $x(z_0)$ is evaluated using the Newton method. If this value was not close enough to 0, we interpolate a new value for z_0 and we repeat the process until the desired precision is obtained. This process is done automatically via a Maple V program.

Figures Fig. 2 and Fig. 3 show the performance measures for $b = 10$ and $c = 5$, where the value of the initial load factor was $z_0 = 0.611$. It is observed that the relationship between z and x is increasing, whereas the utilization varies during the expansion, reaching a minimum of 56% almost at the end of the expansion, approximately. Table Table II shows the extreme values reached by the performance measures.

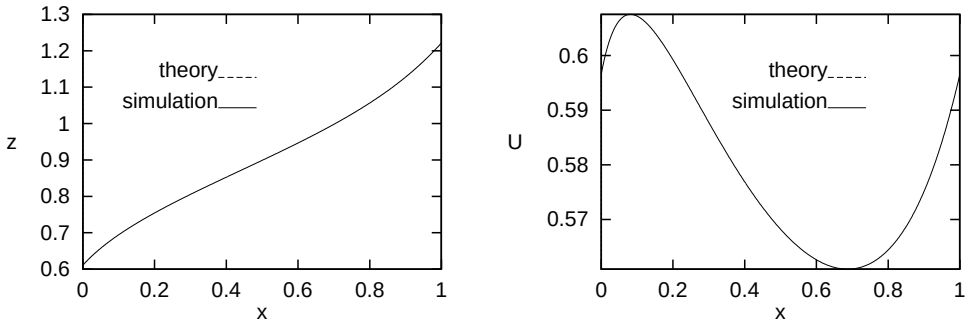

Fig. 2: Analysis and simulation of $z(x)$ (left) and $U(x)$ (right).

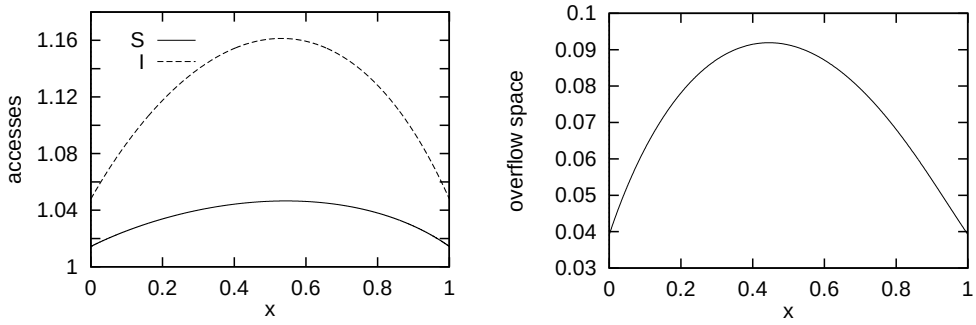

Fig. 3: Left: Number of accesses for the successful and unsuccessful search. Right: Expected overflow space used per record.

6. Hybrid Control Functions

The case of controlled expansion plus a local event is outlined here, as an example for the analysis of a technique of class III. That is, the file is expanded if the storage utilization is larger than a limit given by a parameter α $(0 < \alpha < 1)$ and an overflow also occurs. For this purpose the differential equation which relates x and z, changes to:

$$\frac{dx}{dz} = b\ P_{ov}(z,x)\ \mu(U(z,x) \geq \alpha)$$

where

$$\mu(U(z,x) \geq \alpha) = \begin{cases} 1 & \text{if } U(z,x) \geq \alpha \\ 0 & \text{if not} \end{cases}$$

From the relation $U(z,x) = \alpha$, $x = g(z,\alpha)$, is obtained, so that the equation to be solved is the following:

c	S_{max}	$E[S]$	I_{max}	V_{max}	U_{min}	U_{max}
1	1.10485	1.07331	1.39737	0.04368	0.62159	0.57365
2	1.06695	1.04772	1.24383	0.05412	0.61853	0.57075
3	1.05457	1.03938	1.19466	0.06550	0.61513	0.56776
4	1.04918	1.03578	1.17258	0.07812	0.61144	0.56451
5	1.04654	1.03402	1.16136	0.09197	0.60755	0.56101
6	1.04521	1.03312	1.15549	0.10692	0.55724	0.60358
7	1.04455	1.03266	1.15248	0.12276	0.55323	0.59963
8	1.04423	1.03243	1.15101	0.13924	0.54901	0.59576
9	1.04409	1.03231	1.15032	0.15610	0.54464	0.59110
10	1.04402	1.03226	1.15001	0.17319	0.54020	0.58837

TABLE II: Extreme values with $b = 10$ and different values of c.

$$\frac{dx}{dz} = \begin{cases} b\, P_{ov}(z,x) & \text{if } x \leq g(z,\alpha) \\ 0 & \text{if not} \end{cases}$$

where $g(z,\alpha)$ is defined in Equation 1. The condition is expressed in function of x, because it is not possible to solve it in function of z. This equation is solved using Runge-Kutta and Newton as in the previous section, and the numerical solution is checked against the boundary condition $x \leq g(z,\alpha)$, between z_0 and $2z_0$ for the given α. Figure Fig. 4 (left) shows the relationship between z and x for the values $b = 40$, $c = 5$ and $\alpha = 0.85$, in which case $z_0 = 0.898$ was obtained. Figure Fig. 4 (right) shows the storage utilization. The deviations in the space utilization are due to the fact that early expansions produce a higher variation of the storage utilization than later expansions (the bucket size is fixed while the file size grows). Nevertheless, notice that the variations around $\alpha = 0.85$ are less than 0.01. Table Table III shows the extreme values for different α with $b = 10$ and $c = 5$.

α	S_{max}	$E[S]$	I_{max}	V_{max}	U_{min}	U_{max}
0.70	1.13635	1.10008	1.44738	0.19678	0.69956	0.70023
0.75	1.19876	1.14965	1.63267	0.25331	0.74927	0.75019
0.80	1.29641	1.23072	1.90699	0.32488	0.79887	0.80014
0.85	1.47391	1.38978	2.37565	0.42204	0.84836	0.85008
0.90	1.89553	1.77849	3.50880	0.57185	0.89879	0.90005

TABLE III: Extreme values for different values of α for $b = 10$ and $c = 5$.

It can be observed that apparently the condition of having overflow does not allow to increase the utilization over α in this case, when α is close to 1. So, it is almost the same as control function 3. Figure Fig. 5 shows the

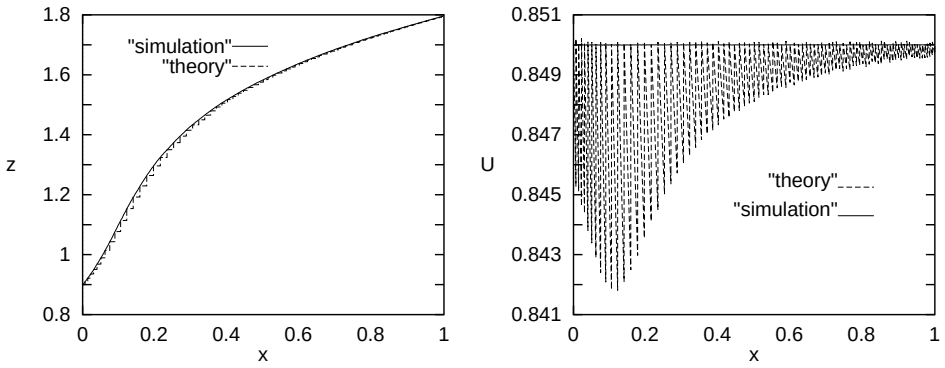

Fig. 4: Analysis and simulation of $z(x)$ (left) and $U(x)$ (right) for the hybrid case with $\alpha = 0.85$.

$\alpha \setminus b$	10	20	30	40
0.70	0.364396	0.284503	0.246017	0.225103
0.75	0.473233	0.387024	0.338540	0.310691
0.80	0.606308	0.513805	0.450785	0.408861
0.85	0.771931	0.676137	0.590809	0.522717
0.90	0.950534	0.850748	0.778011	0.658139

TABLE IV: Values of $E[P_{ov}]$ for different values of α and b.

average overflow probability during an expansion for a given α (see values in table Table IV) obtained using:

$$E[P_{ov}] = \frac{1}{z_0} \int_{z_0}^{2z_0} P_{ov}(z, g(z))\, dz$$

It may be observed here that this conditional probability rapidly tends to 1 when α increases. The effect of the overflow condition is only noticeable for small values of α, as is shown in figure Fig. 6.

7. Experimental Results

An experimental study was carried out with the following characteristics:

- The case of expansion when there is overflow in an insertion, when the storage utilization is controlled, and the hybrid case were considered (control functions 1, 2 and 3).
- The parameters used were b, c, N and α (when needed) and the initial number of records.
- The group where the record is inserted is chosen at random and the decision in which table entry the incoming key is inserted when the group is already expanded is also uniformly chosen.

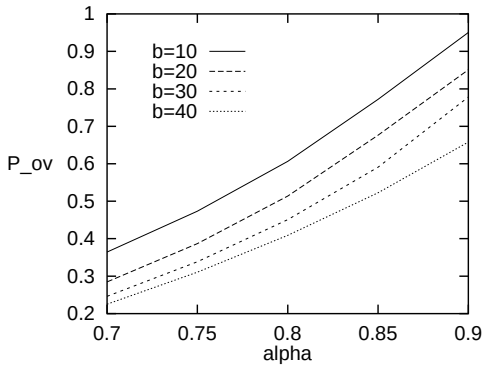

Fig. 5: $\bar{P_{ov}}$ versus α for different b.

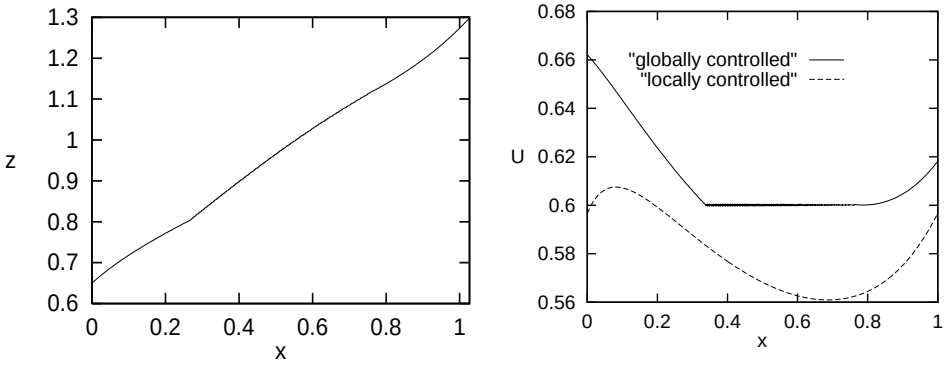

Fig. 6: Left: Load factor z versus expansion ratio x with $\alpha = 0.6$. Right: Utilization (globally controlled with $\alpha = 0.6$ and locally controlled) versus x.

- All the performance measures were obtained, among which we can mention the fraction of expanded groups, the load factor, the storage utilization, and the expected number of accesses in successful and unsuccessful searches.
- When x reaches 1, it means that the file size has been duplicated and a new expansion begins. The simulation is finished when a given amount of hash table entries is reached (2 millions which implies several millions of records). The number of groups N is constant in each expansion, setting $N = 2N$ for the next expansion.
- The results are the average of ten runs, taking the last doubling period. Starting from a given number of keys, the stable behavior is reached usually in less than ten full expansions unless we start from a very small initial file (that is, the number of keys is doubled ten times). Also, the variance is very small, which justifies our approximation for

$U(z, x)$.

With the data generated by the simulation the relationships between z and x and the storage utilization were plotted (see figure Fig. 2) for the locally controlled case and the globally controlled utilization, considering $b = 40$, $c = 10$, and $\alpha = 0.85$. For all the performance measures the simulations agreed completely with the analytical results, and there is no appreciable difference in the curves shown. These results also agree with those of Litwin's [8] and Larson's [6].

8. Optimal Size of the Overflow Bucket

An important parameter is the optimum size of the overflow bucket in order to minimize the search time or the storage utilization. When we use control function 3, the storage utilization is fixed, but we can minimize the expected successful search performance. The existence of this optimum had been indicated in Larson [7], but only partial results were given. The idea is to find a value of c which minimizes the value of $E[S]$. In the locally controlled case, $E[S]$ always decreases when c increases. In practice this is not completely realistic because the model only considers the number of accesses and not the transfer time of the buckets, which is relevant for large c.

On the other hand, in the globally controlled case, as the utilization is fixed, c cannot be too large, because the expansion activity would decrease and then $E[S]$ would increase until it would be necessary to expand the file to keep the utilization fixed. Figure Fig. 7 shows the curves obtained when plotting the average amount of accesses in a successful search $E[S]$ versus size c of the overflow bucket for the limited case, for different values of b. It can be observed that a minimum is reached in each one, as shown in table Table V. Table Table VI and figure Fig. 7 show different dependencies of $E[S]$ concerning b, c and α. Notice the sudden growth on $E[S]$ in function of the overflow bucket size c due to the decrease of the expansion activity, as the storage utilization drops faster when we add bigger overflow buckets. A similar analysis can be done for the other control functions.

$\alpha \setminus b$	10	20	30	40
0.70	5	8	11	13
0.75	4	8	11	13
0.80	4	7	10	12
0.85	3	6	9	12
0.90	3	5	7	9

TABLE V: Values of c_{opt} for different values of α and b.

$\alpha \setminus c$	10	11	12	13	14
0.70	1.045823	1.044945	1.044376	1.044060	1.089320
0.75	1.077755	1.076077	1.074979	1.074374	1.339492
0.80	1.127178	1.146798	1.123769	1.144555	1.457519
0.85	1.211537	1.208729	1.208392	1.409010	1.464783

TABLE VI: Values of $E[S]$ for different values of α and c.

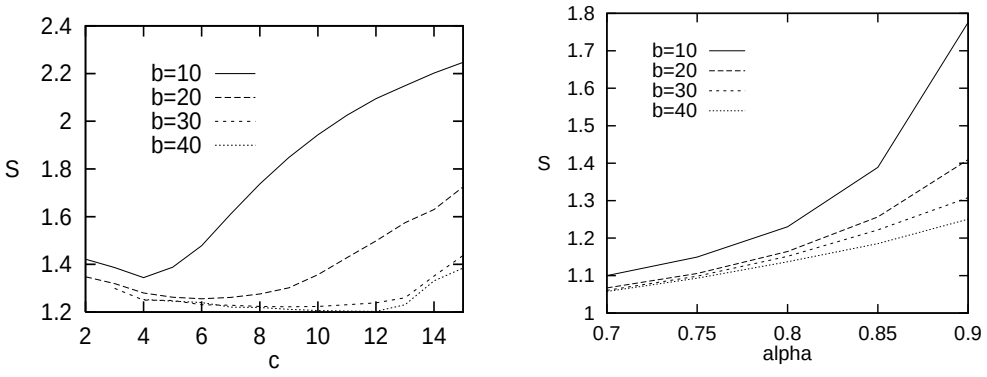

Fig. 7: Left: $E[S]$ versus c for $\alpha = 0.85$ and different values of b (globally controlled case). Right: $E[S]$ versus α for $c = 5$ and different values of b (globally controlled case).

9. Concluding Remarks and Future Work

The analyses developed in this paper are asymptotic, that is they suppose an infinite size file. However, it is known that the expected performance of a finite file is not far from the asymptotic values in more than a small percentage (our results and others done through simulation and real experiments demonstrate this fact [7]), due to which the results obtained can be considered valid for practical purposes.

When a large amount of records is intended to be inserted in a file, the overflow controlled linear hashing variant (Litwin's) is recommended since the access time is lower. By using a bucket size 10, we get an initial load factor of approximately 61%, a storage utilization of 57% and determining whether a record is in the file or not takes little more than one access.

If the efficient use of the secondary memory is critical and the utilization of the storage space is established a priori, linear hashing with limited storage utilization must be used, considering the fraction α of occupation of the proposed file. Under this control function, and considering $\alpha = 0.8$, $b = 10$ and $c = 5$ limit, an initial load factor of 96% is obtained, with the retrieval of one record in little more than one access and a low overflow space required. Access deteriorates in an unsuccessful search which reaches

almost two accesses in the worst case.

The analysis developed in this paper can also be extended to partial expansions. For the case of n_0 partial expansions, $n_0 \geq 1$, the relationship between x and z leads to a system of differential equations, with the expansion factor and event probability varying in each equation. For the $n_0 = 2$ case, the following system must be solved:

$$\frac{dx_1}{dz} = b\ P_{E1}(z,x) \quad , \qquad \frac{dx_2}{dz} = \frac{3}{2} b\ P_{E2}(z,x)$$

where $P_{E1}(z,x)$ and $P_{E2}(z,x)$ are the probabilities of the event triggering the expansion in each case. The boundary conditions are:

$$x_1(z_0) = 0 \quad , \quad x_1(3z_0/2) = 1 \quad , \quad x_2(3z_0/2) = 1 \quad , \quad x_2(2z_0) = 2$$

Finally, it would be of interest to add the bucket transfer time to the model to obtain the optimum overflow bucket size in Litwin's case and to improve the results of the globally controlled case.

Acknowledgements

We would like to thank the helpful comments of P-Å. Larson, who suggested the analysis of Litwin's original hashing scheme, and the unknown referees.

This work was partially supported by FONDECYT Project No. 1950622. Part of this paper was done while the first author was visiting the CRM, Barcelona, the first quarter of 1996.

References

[1] Bell, J., Gupta, G. K. *Implementing Linear Hashing in Main Memory.* Australian Computer Science Communications, Vol. 14(1), 1992, pp. 71-91.

[2] Char B., Geddes K., Gonnet G., et al. *MAPLE V Language Reference Manual.* Springer Verlag, 1991.

[3] Clausing, A. *Kantorovich-type Inequalities*, Amer. Math. Monthly, Vol. 89(5), 1982, pp. 314-330.

[4] Enbody, R., Du, H. C. *Dynamic Hashing Schemes.* ACM Computing Surveys, Vol. 20(2), June 1988, pp. 85-113.

[5] Gonnet, G.H., Baeza-Yates, R. *Handbook of Algorithms and Data Structures*, 2nd edition, Addison Wesley, 1991.

[6] Larson, P-Å. *Linear Hashing with Partial Expansions.* In Proceedings of 6th. Conf. V. L. Data Bases, Montreal, Canada, ACM, October 1980, pp. 224-232.

[7] Larson, P-Å. *Performance Analysis of Linear Hashing with Partial Expansions.* ACM Transactions on Database Systems, Vol. 7(4), Dec. 1982, pp. 566-587.

[8] Litwin, W. *Linear Hashing : A new tool for file and table addressing.* In Proceedings 6th. Conf. Very Large Data Bases, Montreal, Canada, ACM, 1980, pp. 212-223.

[9] Ramamohanarao K., Lloyd, J.W. *Dynamic Hashing Schemes.* The Computer Journal, Vol. 25(4), 1982, pp. 478-485.

[10] Soza, H. *Analysis of Linear Hashing.* M.Sc. Thesis in Computer Science, University of Chile, 1993.