Use of Mean Distance Between Overflow Records to Compute Average Search Lengths in Hash Files with Open Addressing

J. BRADLEY
Computer Science Department, University of Calgary, 250 University Drive, N.W. Calgary, Alberta, Canada T2N 1N4

Average search lengths for hash files with open addressing have been computed using the well-known Poisson distribution for the number of addresses assigned x records, and a new expression for the mean distance between overflow records overflowing from a common home address. The method involves computing first the number of disk accesses required to randomly retrieve the y records overflowing from any home address, using a knowledge of the mean distance between overflow records on the disk. The Poisson distribution is then used to obtain the total disk accesses required to retrieve all records in the file, from which the average search length, as total accesses divided by total records, may be deduced. The average search length values obtained agree closely with experimental results. Because it also dispenses with the complex mathematics of existing methods, this new method can be recommended for use in practical design situations. A by-product is that values for the mean distances between overflow records for different loading factors and address capacities are also predicted.

Received February 1985

1. INTRODUCTION

The method of open addressing with hash files, also known as progressive overflow or consecutive spill, was first investigated experimentally by Peterson[6] in the 1950s. In a now classic series of experiments, Peterson showed that the average search length decreases with address capacity and increases with loading factor. The average search length is defined as average number of disk accesses required to retrieve a random record from the file.[1,4,6] The loading factor is defined as the fraction of the total disk space occupied by the records, and address capacity is defined as the number of records an address can hold, where it is an address value that is calculated by the hashing routine.[1,4,6]

Peterson's work assumed that the hashing routine distributed the records over the address space in a random fashion, that is, each address has an equal probability of being generated when a record key is hashed. It has since been realised[1,2,3,4,5] that a random distribution is the worst case that should be accepted. It is a distribution that is better than random that is desirable, that is, a distribution where the records are more evenly distributed over the addresses. Nevertheless, in practice such a more even distribution has proven very difficult to attain,[1,2,3] whereas it is relatively easy to generate a random or close to random distribution. In consequence, Peterson's early results are still very relevant in practical hash file design.

Following publication of Peterson's results, attempts were made to deduce the average search lengths theoretically using the Poisson distribution, which gives the number of addresses $F(x)$ that have been assigned x records, when the assignment is random. When r records are assigned to R addresses, the Poisson distribution expression may be shown to be:[1]

$$F(x) = R(r/R)^x (1/x!) \exp(-r/R) \qquad (1)$$

provided the number of addresses is large, which it normally is with large hash files. It turned out that the deduction of the average search length from the Poisson distribution involved remarkably difficult (or interesting) mathematics, and the problem was first solved by the mathematician Tainiter in the early 1960s.[7]

Tainiter's results agreed well with those of Peterson.[6,7] Later Lum and co-workers[4,5] undertook a comprehensive study of hash file performance, and their experimental results essentially confirmed Peterson's earlier results. Thus it may be concluded that Tainiter's mathematical analysis is essentially correct, and there the matter has apparently rested for the past decade or more.

Nevertheless, from the point of view of the designer of hash files, the current situation is not satisfactory. Some useful tools are missing with regard to open addressing. Experiments to determine average search lengths for large files are expensive and difficult, so that experimental data are limited. For example, there appear to be experimental results for only address capacities of 1, 2, 5, 10, 20 and 50. And because of their difficulty, use of methods based on Tainiter's original analysis is out of the question for most designers. To appreciate the depth of the mathematics involved, readers should reference Tainiter's paper.[7] What would be useful is a fairly simple method that could be used to generate average search lengths accurate to within a few per cent. Such a method is described in this paper, and is based on the concept of the mean distance between overflow records.

2. DERIVATION OF AVERAGE SEARCH LENGTH

We define $(G-1)$ as the mean distance or gap separating overflow records in a hash file with open addressing. If this quantity is known, then it is an easy matter to deduce an approximation for the average search length, given the distribution $F(x)$.

Suppose that y records overflow from a given home address. Assuming that the overflow records are distributed as in Fig. 1, that is, that they are $(G-1)$ addresses apart on average, then the number of disk accesses required to retrieve all y records in random order must be:

$$G(1+2+3+\ldots+y)$$

or

$$Gy(y+1)/2.$$

If b is the address capacity, the number of home addresses with y overflow records is $F(y+b)$, where $F(x)$ is the Poisson distribution given by expression (1), since

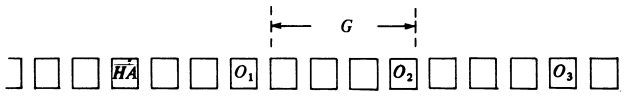

Fig. 1. Showing G with respect to a record in its home address (HA) and records overflowing from that home address in addresses O_1, O_2 and O_3. The diagram is idealized. Since $G = 4$, it takes 4 accesses to retrieve the first overflow record randomly, $2 * 4$ for the second, and $3 * 4$ for the third, and so on.

any records in excess of b will overflow. Hence the number of disk accesses required to randomly retrieve all overflow records assigned to addresses that each give rise to y overflow records is:

$$F(b+y) * Gy(y+1)/2. \quad (2)$$

In addition, there will be addresses for which only one record overflows ($y = 1$), addresses for which two records overflow ($y = 2$), and so on. Hence the number of disk accesses required to retrieve randomly all the overflow records in the file must be the summation of expression (2) for each y value, or:

$$G \sum_{y=1} F(b+y) * y(y+1)/2. \quad (3)$$

Let us refer to this quantity as GV. (Thus V approximates the average number of disk accesses required to retrieve all overflow records if the value of G is unity, that is, if $G-1$ and the gap between overflow records is zero.)

If H is the number of home address records, the number of disk accesses required to retrieve them in random order must be H, since a record in its home address can be retrieved in a single access. Hence the total number of disk accesses to retrieve randomly all the records in the file must be:

$$H+GV. \quad (4)$$

The value of H can be obtained from the Poisson distribution as the total number of records in the file (r), less the number of overflow records, that is:

$$H = r - \sum_{y=1} F(b+y). \quad (5)$$

Since the average search length by definition is the total accesses required to obtain all the records randomly, divided by the total records, the average search length (s) must be given by:

$$s = (H+GV)/r. \quad (6)$$

Both V and H may be computed using expressions (3) and (5), and the Poisson expression (1). The crucial step is the derivation of an expression for G.

When the file is loaded, a simple measure of G must be the number of addresses in the file, per empty space for a record, that is:

$$R/(bR-r). \quad (7)$$

Here $(bR-r)$ is the number of records that could still be placed in the file, at the point where the file has been fully loaded. To see the justification for expression (7), suppose that $b = 2$, $R = 1000$, and $r = 1750$, so that the file is 87.5% full – and 12.5% empty. There are 250 empty record slots in the file and these are distributed over 1000 addresses. Hence the average number of addresses per empty slot must be 1000/250 or 4 addresses. The number of addresses, on average, separating each empty slot must therefore be $4-1$, or 3. Hence the value 4 is a measure of G, as is also given by expression (7).

There are many reasons why expression (7) would not be exactly equal to the mean distance between records overflowing from any given home address. The most important is the inevitable overlapping of groups of overflow records from other home addresses close by. Thus we would expect that the value for G given by expression (7) would be too small.

Consequently, the best we could expect is that G would be given by:

$$G = kR/(bR-r) \quad (8)$$

where k is a factor, greater than unity, than that should depend in a complex manner on R, r and b. The precise theoretical determination of k could be expected to involve theory as complex as that developed originally by Tainiter.[7] However, it turns out that for all practical purposes k is a constant equal to 1.5. This has been determined from a study of all available experimental data it is a finding of certain practical utility, since it greatly simplifies calculations of average search lengths with open addressing.

Using the k value of 1.5 in expression (8) to determine G, we can then use expression (6) to compute the average search length.

Using expressions (3), (5), (6) and (8), we can also give a final composite expression for the average search length:

$$s = [(r - \sum_{y=1} F(b+y)) + (kR/(bR-r)) \sum_{y=1} F(b+y) * y(y+1)/2]/r$$

where $F(x)$ is the Poisson distribution, k is 1.5, R is the total addresses, r is the total records, and b is the address capacity.

3. RESULTS

The results of using this method to predict average search lengths versus loading factors for different address capacities are shown in Fig. 2. (Note that the loading factor expressed as a percentage is $100r/Rb$.) The curves for b equal to 1, 2, 5, 10 and 50 agree to within a few per cent with the data determined experimentally by Peterson,[6] and later by Lum.[5] There does not appear to be any experimental data for the curve for $b = 3$.

4. NUMERICAL METHODS

For the benefit of those who would like to use the above theory to compute the average search length for a specific case, assuming a random distribution, we give an example.

Suppose that we have a file with 80000 records stored in 50000 addresses, each with an address capacity of two records per address. Because of the proportionality inherent in the Poisson distribution expression (1), the average search length will be the same as in the case of 1600 records stored in 1000 addresses with address capacity of 2. It is always more convenient to work with

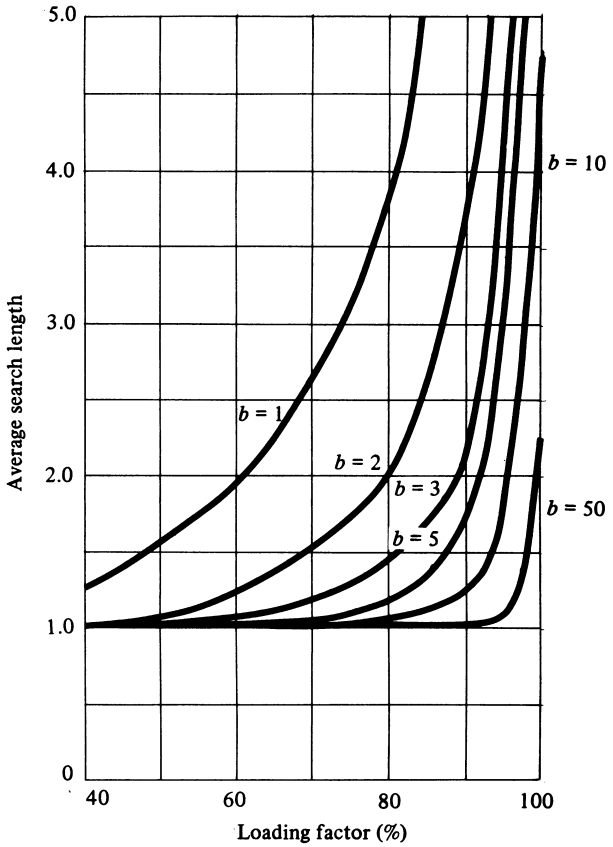

Fig. 2. Showing how the computed value for average search length varies with the loading factor and the value for *b* (the address capacity).

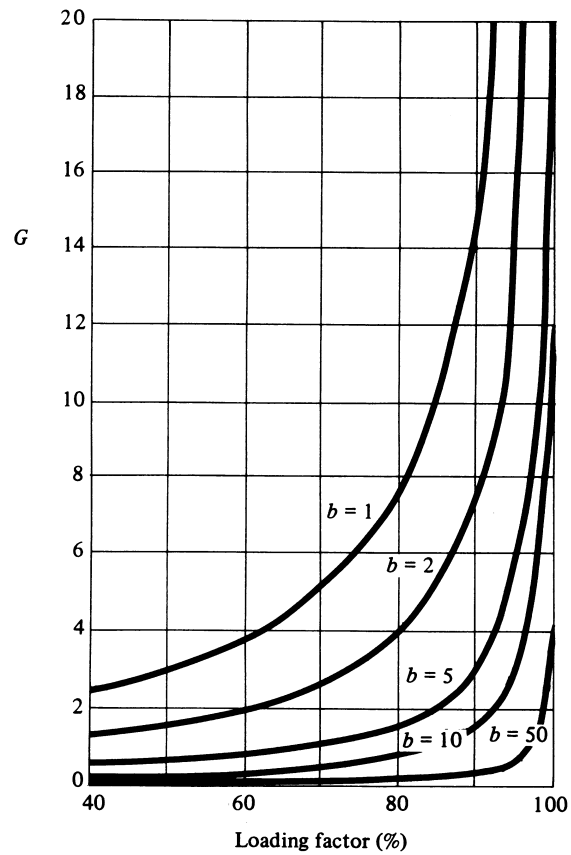

Fig. 3. Showing how computed values for *G* vary with loading factor and the value for *b*. The mean distance between overflow records is *G*−1.

1000 addresses.[1] Using $r = 1600$ and $R = 1000$ in expression (1), the records of the file will be distributed according to:

$$F(0) = 202,\ F(1) = 323,\ F(2) = 258,\ F(3) = 138$$
$$F(4) = 55,\ F(5) = 17,\ F(6) = 5,\ F(7) = 2.$$

Applying expression (3), the number of disk accesses to retrieve randomly all the overflow records must be:

$$138 * G * 1 + 55 * G * 3 + 17 * G * 6 + 5 * G * 10 + 2 * G * 15$$

or

$$485G.$$

Using expression (8), G can be computed as $1.5 * 1000\ (2000 - 1600)$ or 3.75. Using this value in expression (3), the number of accesses required to retrieve all the overflow records in random order is $485 * 3.75$ or 1881.75.

Applying expression (5) to determine the number of home address records, the number of overflow records is:

$$138 * 1 + 55 * 2 + 17 * 3 + 5 * 4 + 2 * 5 \quad \text{or } 329.$$

Hence the number of home address records is $(1600 - 329)$ or 1,271 records. Thus the number of disk accesses to retrieve all the home address records in random order is also 1,271.

The number of disk accesses to retrieve randomly all the records (home address plus overflow records) is therefore $(1881.75 + 1{,}271)$ or 3156.75 accesses, as in expression (4). Finally, applying expression (6), the average search length is 3156.75/1600 or 1.97 disk accesses per record retrieved.

The loading factor is $100r/Rb$ or $100 * 1{,}600/1000 * 2$ or 80%. Thus, referring to Fig. 2, we have the point [80, 1.97] On the curve for $b = 2$. The value of the corresponding average search length measured by Peterson was 1.92.[6] Other values obtained by this method are equally close to those measured by Peterson. This would indicate that the method has more than enough accuracy for use in practical design situations.

5. THE MEAN DISTANCE BETWEEN OVERFLOW RECORDS

Central to the method of computing average search lengths as described above is the concept of the mean distance $(G - 1)$ between records overflowing from a home address, as given by expression (8). This quantity could be measured experimentally, although as far as the author is aware no experimental data has ever been published. Measurements would involve determining the distance, in addresses, between each pair of consecutive overflow records with a common home address.

Expression (8) can be rewritten as:

$$G = k/b(1 - r/bR).$$

Since r/bR is the loading factor expressed as a fraction, expression (8) is equivalent to:

$$G = k/b(1 - L) \qquad (9)$$

where L is the loading factor.

This relationship can be expressed graphically, as illustrated in Fig. 3. As we would expect, G, and thus the

mean distance between overflow records ($G-1$), decreases with increasing address capacity, and increases with increasing loading factor. As the loading factor tends towards 100%, the value for G tends to infinity. Clearly, at 100% loading factor, although the value for G will be very large, it cannot be infinite. It must be less than R, since overflow records are within the file. Thus the formula cannot be entirely accurate for loading factors close to 100%. This is not a disadvantage, however, since hash files are rarely, if ever, designed for 100% loading. It would be interesting to see how experimental values for G compare with those predicted by expression (9) in Fig. 3.

6. SUMMARY

We can conclude that there is a relatively simple but accurate method for computing average search lengths for fully randomised hash files employing open addressing for dealing with records overflowing from home addresses. The method uses the Poisson distribution to determine the numbers of records overflowing from addresses, and the mean distance between overflow records as a measure of how the overflow records are spaced.

The mean distance between overflow records ($G-1$) is determined from an expression for G that involves an empirical quantity that unexpectedly turns out to be a constant. The method gives results that are very close to those obtained experimentally by earlier investigators. The method is simple compared with the classic but unusually complex analysis used originally by Tainiter to predict average search lengths. However, despite its accuracy and ease of use it should be remembered that the method is essentially empirical. In addition to the empirical expression (8) for G, if analysed closely, it will be seen that expression (2) is also empirical in nature, especially for values of G of unity or less. A by-product of the method is the prediction of mean distances between overflow records. These could be measured, although experimental results are not currently available.

REFERENCES

1. J. Bradley, *File and Database Techniques*. Holt, Rinehart & Winston, New York (1982).
2. J. Bradley, *Computer File Techniques*. Holt, Rinehart & Winston (In the press) (1986).
3. W. Bucholtz, File organization and addressing, *IBM Systems Journal*, no. 2, pp. 86–111 (1963).
4. J. Hanson, *Design of Computer Data Files*. Computer Science Press, Rockville, Maryland (1982).
5. V. Y. Lum *et al.*, Key-to-address transform techniques: a fundamental performance study on large formatted files. *Comm. ACM*. **14** (2), 228–239 (1971).
6. W. W. Peterson, Addressing for random access storage. *IBM Journal of Research and Development* **1** (2), 130–146 (1957).
7. M. Tainiter, Addressing for random-access storage with multiple bucket capacities. *J. ACM* **10**, 307–315 (1963).