Information Systems Vol. 14, No. 2, pp. 151–162, 1989
Printed in Great Britain. All rights reserved

0306-4379/89 $3.00 + 0.00
Copyright © 1989 Pergamon Press plc

ANALYSIS OF OVERFLOW HANDLING FOR VARIABLE LENGTH RECORDS

S. CHRISTODOULAKIS,[1] Y. MANOLOPOULOS[2] and P.-Å. LARSON[1]

[1]Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
and [2]Department of Electrical Engineering, University of Thessaloniki, Thessaloniki, Greece 54006

(*Received* 5 *April* 1988; *in revised form* 22 *December* 1988)

Abstract—Although files with variable length records are very frequent in actual databases due to variable length fields, missing attribute values, multiple values of an attribute and compression, little has been reported in the literature about file structures appropriate for variable length records. In this paper we describe and analyze several overflow handling techniques for the case when records are of variable length. We develop analytic models that take into account variable length records and study the performance in the context of indexed sequential (ISAM) files. We also propose a new overflow handling technique and show that it provides an incremental reorganization capability appropriate for handling variable length records. Analytic results demonstrate that its performance is better than the performance of previous methods. The overflow handling techniques and the analytic methods developed in this paper are also applicable to other file structures with variable length records that require overflow handling.

Keywords: File organization, file structures, index sequential files, ISAM, indexed sequential access method, overflow, overflow handling, overflow chaining, variable length records, searching, performance analysis, dynamic reorganization, clustering.

INTRODUCTION

Variable length records occur frequently in practice as a result of variable length fields, missing attribute values, multivalued attributes and compression. File organizations appropriate for handling variable length records have not been studied extensively. Some work on the effect of variable length records on the performance of various database operations has been reported, e.g. [1–9]. In this paper we analyze the effect of variable length records on the performance of indexed sequential (ISAM) files. However, the approach is of more general interest and applies to other file organizations as well. We study four different methods for handling overflow records. Three of them have been analyzed previously, but only for records of fixed length. The fourth one is new. It provides dynamic reorganization of the data in the file to improve clustering.

Larson developed a model for analyzing the performance effects of random insertions in indexed sequential files [10]. Our analysis is based on a similar model. We extend the analysis to other overflow handling techniques and variable length records. The model provides a common framework for studying the performance of these overflow handling techniques. Alternative models for analyzing the performance deterioration of a file caused by insertions and deletions have been proposed in [11–13]. Other related work on indexed sequential files is described in [14–21]. Extensive surveys appear in [9, 22].

In Section 2 of this paper we present the basic mathematical model. In Section 3 we analyze the performance of overflow handling by a simple chaining scheme, under the assumption that a single record exists in each overflow bucket. In Section 4 we study a chaining scheme which uses large overflow pages. In Section 5 we analyze the performance of a method using hashing into a fixed number of overflow chains. In Section 6 we analyze the performance of a new method which provides a dynamic reorganization capability. In Section 7 we present numerical results and compare the four schemes.

2. ASSUMPTIONS AND BASIC MODEL

The mathematical model presented in this section is an extension of the model presented in [10] for calculating the probability distribution of the number of records in a bucket after a sequence of random insertions. This distribution is needed in subsequent sections for deriving performance estimates for the various overflow handling techniques. But first a note on terminology. A bucket is merely a logical record container, which physically consists of a primary page and some number of overflow records, stored on overflow pages. A page is assumed to be the smallest unit of transfer between disk and main memory.

During initial loading of an ISAM file only primary pages are loaded with records. The highest key of each primary page is extracted and the resulting set of keys constitutes the lowest level of the index. The index is not changed until the entire file is reorganized. As a result of initial loading, each bucket is assigned a key interval that thereafter remains

fixed. We assume that the keys of the records loaded are a random sample from some underlying key distribution. It was shown in [23] that the probability mases Q_i assigned to buckets $i = 1, 2, \ldots, NB$ by this procedure are identically distributed, random variables with mean $E(Q_i) = m/(M+1)$ where m is the number of records loaded in a bucket, M the total number of records loaded and NB is the total number of buckets in the file.

The exact distribution of the probability mass assigned to a bucket and the correct distribution of the number of records per bucket after a sequence of random insertions were first derived by Batory [11]. The analysis in [10] is based on an approximate model. In the Appendix we present a different derivation which shows that the asymptotic distribution is a negative binomidal distribution. Let α, $\alpha \geqslant 0$, denote the *file expansion factor*, defined as N/M where M is the total number of records originally loaded and N is the total number of records inserted thereafter. The probability that a bucket loaded with m records contains n, $n \geqslant m$, records after the file has expanded by a factor α is then asymptotically:

$$P_m^*(n, \alpha) = \binom{m+n-1}{n} \frac{\alpha^n}{(1+\alpha)^{m+n}} \qquad (1)$$

In our experiments we found that the performance estimates obtained using the correct distribution are considerably higher than those obtained by approximating the probability of insertion in a bucket by its mean, as done in [10]. The observed difference ranged from 5 to 20%.

We are interested in the performance as a function of the file expansion factor. The *state* of a bucket is defined as the number of records x in the bucket ($x = m + n$). If the buckets of the file were not originally loaded with the same number of records, the record distribution is given by:

$$P(x, a) = \sum_{m=1}^{\max} R_m P_m^*(x - m, \alpha), \quad \text{for } x = m, m+1, \ldots \qquad (2)$$

where R_m is the fraction of buckets that received m records during loading and max is the maximum number of records loaded in a bucket.

We assume that the population of records is divided into L classes, $C_1, C_2, \ldots, C_L$, according to length. Class C_1 contains records with length l_1, class C_2 contains records with length $l_2, \ldots$, and class C_L contains records with length l_L. Without loss of generality we assume that $l_1 < l_2 < \cdots < l_L$. The probability that a record belongs to class C_i is p_i, $i = 1, 2, \ldots, L$ and is assumed to be known. Record lengths are assumed to be independent of primary key values. The assumption of a fixed number of classes is motivated by the fact that varible length records frequently are the result of missing attribute values, repeated groups or storing records of a few different types in the same file. If variable length records are the result of variable length fields or compression the length distribution is continuous and more difficult to model. In this case the range of record lengths can be subdivided into subranges and a class associated with each subrange.

Consider a set of x records assigned to a bucket. The probability that n_1 of these records have been selected from class C_1, n_2 from $C_2, \ldots, n_L$ from C_L follows a multinominal distribution and hence is:

$$q(n_1, \ldots, n_L) = \frac{x!}{n_1! \ldots n_L!} p_1^{n_1} p_2^{n_2} \cdots p_L^{n_L}. \qquad (3)$$

The sum of the record lengths is

$$\sum_{i=1}^{x} n_i l_i.$$

Assume that b out of the x records are stored in the primary page. Note that b is a random variable and depends on the distribution of record lengths. Let $Q(b, x)$ be the probability that *exactly* b out of the x records in a bucket are stored in the primary page. Furthermore, let BS denote the number of bytes available per primary page, PS the size of a pointer in bytes and KS the size of a key in bytes (fixed). These definitions are summarized in Table 1.

Table 1. Symbol definitions

Symbol	Definition
$X = X(\alpha)$	Number of records in a bucket (as a function of α)
$P(x, \alpha)$	Probability that a bucket contains x records when the file expansion factor equals α
$\bar{X}(\alpha)$	Average number of records in a bucket (as a function of α)
$b = b(\alpha)$	Number of records in the primary page of a bucket (as a function of α)
m	Number of records initially loaded in a bucket
n	Number of records inserted into a bucket
M	Total number of records initially loaded in the file
N	Total number of records inserted into the file
BS	Size of a primary page (in bytes)
BS_0	Size of an overflow page (in bytes)
KS	Key size (in bytes)
PS	Pointer size (in bytes)
L	Number of classes of record lengths
C_i	Class i
l_i	Record length of class i (in bytes)
max	Maximum number of records in a bucket
$q(n_1, \ldots, n_L)$	Probability that n_i records have length $l_1, \ldots, n_L$ records have length l_L
$Q(b, x)$	Probability that exactly b out of x records are stored in the primary page of a bucket

When computing the number of disk accesses for record retrieval we consider only the extra access required for retrieving overflow records, that is, accesses over and beyond those required for traversing the index and for retrieving the primary page. The cost of traversing the index depends on the structure of the index, which is a separate issue.

3. ANALYSIS OF SIMPLE SEPARATE CHAINING

In this section we analyze the performance of a simple separate chaining scheme. In the scheme overflow records are placed in a overflow area and chained together. Each primary page has one overflow chain which starts from the primary page, not the index. Records in the primary page and overflow records are kept sorted according to key values. The highest keys are placed on the overflow chain. We assume a single record per page in the overflow area but a primary page may contain several records. The motivation for a single record per page in the overflow area is to make better use of main memory and to reduce the page transfer time from the disk to main memory. Note that if the proportion of overflow records is small, then most of the buckets have only one overflow record. If so, it is not justified to use large overflow pages. This file implementation is similar to IBM's ISAM files. Separate changing was also analyzed in [10], but only for fixed-size records.

The probability that exactly b records are stored on the primary page when the bucket contains x records is:

$$Q(b,x) = \sum_{\substack{n_1+n_2+\cdots+n_L=b\leqslant x \\ \sum_i n_i l_i \leqslant BS-PS}} q(n_1,\ldots,n_L) \times \sum_{l_i+\sum_j n_j l_j > BS-PS} p_i \tag{4}$$

The above formula is explained as follows. Recall that record lengths are independent of primary key values. The ordering of the x records is determined by their key values and does not change. However, the length of a record is a random variable and takes the value l_1 with probability p_1, l_2 with probability p_2, l_2 with probability p_2, and so on. The first summation is over all combinations of b records from L classes where the sum of the lengths of the records is less than or equal to the size of a primary page minus the pointer size. The probability of such a combination is $q(n_1,\ldots,n_L)$, which is given by (3). This condition guarantees that the b records fit in the primary page. In order to have exactly b records in the primary page, the $(b+1)$st record must be longer than the empty space left in the page. The probability of this event is given by the second sum.

The expected number of records in the primary page when the bucket contains x records can be calculated as

$$\sum_{b=1}^{x} b\,Q(b,x).$$

The expected number of overflow records is:

$$\sum_x \sum_{b=1}^{x} [p(x)Q(b,x)(x-b)] = \bar{X} - \sum_x p(x) \sum_{b=1}^{x} bQ(b,x).$$

For large values of X (i.e. long overflow chains), the expected number of overflow records can be approximated by

$$\bar{X} - \sum_{b=1}^{\bar{X}} bQ(b,\bar{X}).$$

This equation can be used to compare the expected length of overflow chains for different record length distributions. The expected space overhead can be computed for the expected number of overflow records and the maximum record length.

We are now ready to compute the expected retrieval performance. Assuming, as in [10], that every record of the file has the same probability of being retrieved, the expected number of additional access for a successful search is given by:

$$S(\alpha) = \frac{1}{2\bar{X}(\alpha)} \sum_{x=2}^{\infty} P(x,\alpha) \sum_{k=1}^{x-1} k(k+1)\,Q(x-k,x), \tag{5}$$

where $\bar{X}(\alpha)$ is the expected number of records per bucket,

$$\bar{X}(\alpha) = \sum_{x=1}^{\infty} xP(x,\alpha). \tag{6}$$

Recall that we consider only the extra access required for retrieving overflow records. Formula (6) is derived as follows. $Q(x-k,x)$ is the probability that there are exactly $x-k$ records in the primary page and therefore exactly k overflow records. Assuming a single record per overflow page, the expected number of additional accesses required to find a record is $k(k+1)/(2x)$. The probability of accessing a bucket which contains x records is $xP(x,\alpha)/\bar{X}(\alpha)$. Formula (6) follows after summation over k and x.

The expected number of additional accesses for an unsuccessful search is given by:

$$U(\alpha) = \frac{1}{2\sum_{j=1}^{\max} jR_j} \sum_{m=1}^{\max} mR_m \sum_{x=2}^{1} \frac{P_m^*(x,\alpha)}{x+1} \times \sum_{k=1}^{x-1} k(k+3)\,Q(x-k,x). \tag{7}$$

The formula is explained as follows. The x keys of a bucket divide the key space of the bucket into $x+1$ subintervals. We assume that a specified search key belongs to any of these subintervals with the same probability [23, 10]. A search for a key in the jth subinterval is terminated when reaching the jth record, $j = 1, 2, \ldots, x$. A search for a key in the last subinterval is terminated when the xth record has been inspected. Thus, if there are k overflow records, the expected number of additional accesses is $k(k+3)/2(x+1)$. This number of additional accesses has to be weighted by the probability $Q(x-k, x)$ of having k overflow records. The probability of an unsuccessful search hitting a bucket is proportional to the number of records originally loaded into the bucket, since this number determines the probability mass from the underlying key distribution that was assigned to the bucket. Thus the probability of hitting the bucket which was loaded with m records originally is

$$P_m^*(x, \alpha)\ mR_m / \sum_{j=1}^{\max} jR_j.$$

Formula (5) follows.

The additional expected cost required for accessing all the records of a bucket during a sequential scan or range search is given by:

$$R(\alpha) = \sum_{x=2}^{\infty} P(x, \alpha) \sum_{k=1}^{x-1} kQ(x-k, x). \qquad (8)$$

The second sum in the above formula gives the expected number of overflow pages, each one holding a single record. This is also the expected number of additional accesses required when the bucket contains x records. It has to be weighted by the probability that the bucket contains x records.

The overflow space required per bucket is given by:

$$SP(\alpha) = BS_o \sum_{x=2}^{\infty} P(x, \alpha) \sum_{k=1}^{x-1} kQ(x-k, x), \qquad (9)$$

where $BS_o = l_L + PS$ is the size of an overflow page. Each overflow page is of the same size and must be large enough to hold a record of minimum length. The second sum in the formula gives the expected number of overflow pages required.

4. ANALYSIS OF CHANGING USING LARGE OVERFLOW PAGES

In this section we analyze the performance when an overflow area with large pages is used. Overflow records are placed on an overflow chain as in the previous case, but now several overflow records from different or the same bucket may be stored in the same overflow page. We assume that an overflow page that is examined remains in main memory so that the same page does not have to be retrieved more than once during a search. Compared with the previous case, the search cost is thus reduced. In addition, since overflow pages are large, variable length records may be packed better within a page, which improves the storage utilization and reduces the search cost further. This file implementation is well-suited for systems which use preformatted disks (disks with fixed block size).

The probability distribution of the lengths of records in an overflow page is not the same as in a primary page. The reason is that longer records have a higher probability of being intercepted by the primary page boundary than shorter records. Hence longer records are more likely to be found in the overflow area. Let $P_i^a(t)$ be the probability of all arrangements of records where a record of type C_i is intercepted by the boundary of a primary page. Then:

$$P_i^a(\alpha) = \sum_{x=2}^{\max} P(x, \alpha) \sum_{\substack{n_1+n_2+\cdots+n_L<x \\ \sum_{j=1}^{L} n_j l_j \le BS-PS<l_i+\sum_{j=1}^{L} n_j l_j}} \times q(n_1, \ldots, n_L)^* P_i \quad \text{for} \quad i = 1, \ldots, L.$$

This formula is derived in the same manner as formula (4). It takes into account the fact that the record with the next key value may not fit in the primary page and therefore may have to move into the overflow area. Any record with a higher key value will then be in the overflow area as well.

The probability that a record intercepted by a page boundary is of type C_i is:

$$P_i^{\text{int}}(\alpha) = \frac{P_i^a(\alpha)}{\sum_{j=1}^{L} P_j^a(\alpha)}. \qquad (10)$$

To find the length distribution of records in the overflow area we must consider both the intercepted records and the remaining records on the overflow chain (which follow the probability distribution P_i $i = 1, \ldots, L$). The fraction of intercepted records to the total number of records in the overflow area is given by:

$$\frac{NB \sum_{x=2}^{\infty} P(x, \alpha) \sum_{b<\alpha} Q(b, x)}{NB \sum_{x=2}^{\infty} P(x, \alpha) \sum_{b<x} (x-b)\, Q(b, x)}. \qquad (11)$$

The enumerator is the expected number of buckets having overflow chains (i.e. the expected number of intercepted records), while the denominator is the expected number of overflow records. The proportion of all other overflow records is then:

$$1 - \frac{\sum_{x=2}^{\infty} P(x, \alpha) \sum_{b<x} Q(b, x)}{\sum_{x=2}^{\infty} P(x, \alpha) \sum_{b<x} (x-b)\, Q(b, x)} \qquad (12)$$

Hence, the probability that the length of an overflow record is l_i is given by:

$$P_i^{ov}(\alpha) = (P_i^{int}(\alpha) - P_i \times \frac{\sum_{x=2}^{\infty} P(x,\alpha) \sum_{b<z} Q(b,x)}{\sum_{x=2}^{\infty} P(x,\alpha) \sum_{b<x} (x-b)\, Q(b,x)} + P_i, \quad i = 1, \ldots, L. \quad (13)$$

Since $P_i^{int}(\alpha)$ is greater than P_i for long records, $P_i^{ov}(\alpha)$ will be greater than P_i for long records. The reverse is true for short records. For large values of α, the first term in the sum tends to zero and $P_i^{ov}(\alpha)$ asymptotically approaches P_i. Figure 1 shows the probability distribution $P_i^{ov}(a)$ for two record classes.

Given the length distribution of overflow records, the number of overflow pages can be estimated. The expected number of overflow records per bucket is:

$$N_o(\alpha) = NB \sum_{x=2}^{\infty} P(x,\alpha) \sum_{k=1}^{x-1} kQ(x-k,x). \quad (14)$$

The expected number of pages in the overflow area is then given by:

$$NB_o(\alpha) = \frac{N_0(\alpha) * \bar{l}^{ov}(\alpha)}{U} \quad (15)$$

where

$$\bar{l}^{ov}(\alpha) = \sum_{i=1}^{L} p_i^{ov}(\alpha) l_i$$

and $\bar{U}$ is the average number of bytes used in an overflow page. $\bar{U}$ depends on how effectively the overflow space is used. Assuming that a good algorithm is used for packing records into overflow pages we can approximate $\bar{U}$ by:

$$\bar{U} = BS_o - \frac{l_1}{2}. \quad (16)$$

where l_1 is the length of the smallest record.

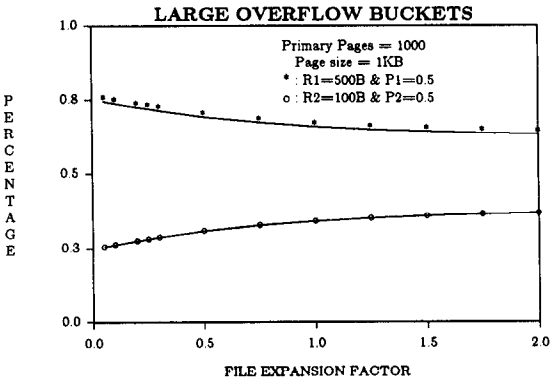


Fig. 1. Overflow distribution of records belonging to two classes.

The expected number of recors per overflow page is then:

$$\bar{b} = \frac{N_o(\alpha)}{NB_o(\alpha)} = \frac{\bar{U}}{\bar{l}^{ov}(\alpha)}. \quad (17)$$

The expected number of overflow pages containing k overflow records can then be approximated by [2]:

$$B(k) \approx NB_o(\alpha) * \left(1 - \frac{C_k^{N_o(\alpha)-\bar{b}}}{C_k^{N_o(\alpha)}}\right) \quad (18)$$

where C_k^i is the binomial coefficient, i choose k. Note, however, that this formula overestimates the expected page accesses [2].

Next we derive an estimate of the expected cost of successful and unsuccessful searches when a simple search algorithm is used for finding the qualifying record. The algorithm simply follows the pointers of the overflow chain until it finds the record or decides that no record with the given key exists in the file. Overflow pages retrieved from secondary storage are assumed to remain in main memory so that a page containing several overflow records from the same bucket does not have to be retrieved more than once.

The expected number of accesses for finding one of the k overflow records is:

$$L(k) = \frac{1}{k} \sum_{r=1}^{k} B(r). \quad (19)$$

The expected number of accesses for an unsuccessful search when the length of the overflow chain is k records is:

$$L'(k) = \frac{1}{k+1} \left(\sum_{r=1}^{k} B(r) + B(k) \right). \quad (20)$$

The expected number of additional page access for a successful search when using this overflow handling technique is:

$$S(\alpha) = \frac{1}{\bar{X}(\alpha)} \sum_{x=2}^{\infty} P(x,\alpha) \sum_{k=1}^{x-1} k\, L(k)\, Q(x-k,x) \quad (21)$$

and the expected number of additional page access for an unsuccessful search is:

$$U(\alpha) = \frac{1}{\sum_{j=1}^{\max} jR_j} \sum_{m=1}^{\max} mR_m \sum_{x=2}^{\infty} \frac{P_m^*(x,\alpha)}{x+1} \times \sum_{k=1}^{x-1} (k+1)\, L'(k)\, Q(x-k,\alpha). \quad (22)$$

The expected number of additional accesses per bucket for a range search or a sequential scan is given by:

$$R(\alpha) = \sum_{x=2}^{\infty} P(x,\alpha) \sum_{k=1}^{x-1} B(k)\, Q(x-k,x). \quad (23)$$

The space per bucket required for overflow records is given by:

$$SP(\alpha) = \bar{I}^{ov}(\alpha)\frac{BS_0}{U}\sum_{x=2}^{\infty} \times P(x,\alpha)\sum_{k=1}^{x-1} kQ(x-k,x) \quad (24)$$

There are more efficient search algorithms (in terms of page accesses) than those discussed above. For example, if a key value between the values of the ith and the rth record is found when the page containing the ith record is retrieved, then the search can be continued by following the chain emanating from that record. An analysis of this type of algorithm is considerably more complex. The expected cost of the algorithms analyzed in this section is an upper bound of the cost of this algorithm.

5. ANALYSIS OF CHAINING WITH MULTIPLE CHAINS

In this section we analyze a chaining scheme which uses several outflow chains per bucket. Denote the number of chains per bucket by c, $c > 1$. Hashing is used to determine to which one of the chains an overflow record belongs. As in the case of separate chaining, each overflow page is assumed to store a single record. Within each overflow chain records are kept sorted according to key value.

The probability $Q(b, x)$ that b records are stored in the primary page when the bucket contains x records can be calculated using equations (6) and (7). However, in equation (7), the constraints should take into account the space requirements of the c pointers in the primary page. That is, the space available in the primary page for record storage is $BS-cPS$ instead of $BS-PS$.

The expected number of additional accesses for successful search can be computed as:

$$S(\alpha) = \frac{c}{2X(\alpha)}\sum_{x=2}^{\infty} P(x,\alpha)\sum_{k=1}^{x-1} Q(x-k,x) \times \sum_{i=1}^{k}\binom{k}{i}\left(\frac{1}{c}\right)^i\left(1-\frac{1}{c}\right)^{k-i} i(i+1).$$

This formula is derived in a similar way as formula (5), except that the last summation takes into account the fact that the k overflow records are distributed among c chains. Assuming a uniform hashing function, each chain has a probability of $1/c$ of receiving an overflow record. The probability that there are i records on a particular chain is then

$$\binom{k}{i}\left(\frac{1}{c}\right)^i\left(1-\frac{1}{c}\right)^{k-i}.$$

The average cost of accessing these records is $i(i+1)/2$. The above formula follows by summing over all values of i and multiplying by c (because there are c chains). Using the binomial theorem [24], the formula can be simplified to:

$$S(\alpha) = \frac{1}{2X(\alpha)}\sum_{x=2}^{\infty} P(x,\alpha)\sum_{k=1}^{x-1} k \times\left(\frac{k-1}{c}+2\right)Q(x-k,x). \quad (25)$$

This formula is an approximation which does not take into account the dependencies between chain lengths. An expensive to compute, but exact formula, can be derived using the multinomial distribution. Note that assuming that each chain has the same number of records, (k/c), would underestimate the expected cost. For $c = 1$ the above formula reduces to formula (5) which gives the expected cost when using a single chain.

The expected number of additional accesses for an unsuccessful search is given by:

$$U(\alpha) = \frac{c}{2\sum_{j=1}^{\max} jR_j}\sum_{m=1}^{\max} mR_m\sum_{x=2}^{\infty}\frac{P_m^*(x,\alpha)}{x+1} \times\sum_{k=1}^{x-1} Q(x-k,x)\sum_{i=1}^{k}\binom{k}{i}\left(\frac{1}{c}\right)^i \times\left(1-\frac{1}{c}\right)^{k-i}\cdot(i+3)$$

$$= \frac{1}{2\sum_{j=1}^{\max} jR_j}\sum_{m=1}^{\max} mR_m\sum_{x=2}^{\infty}\frac{P_m^*(x,\alpha)}{x+1} \times\sum_{k=1}^{x-1} k\left(\frac{k-1}{c}+4\right)Q(x-k,x). \quad (26)$$

This formula is derived in the same way as formula (7). The last summation determines the total expected cost of unsuccessful searches in a chain. This formula reduces to formula (7) for $c = 1$.

The additional expected cost for accessing all records in a bucket is given by:

$$R(\alpha) = \sum_{x=2}^{\infty} P(x,\alpha)\sum_{k=1}^{x-1} kQ(x-k,x). \quad (27)$$

This is the same formula as (8). The only difference is that the records must be sorted before being returned to the user.

The additional space required per bucket is given by:

$$Sp(\alpha) = BS_0\sum_{x=2}^{\infty} P(x,\alpha)\sum_{k=1}^{x-1} kQ(x-k,x). \quad (28)$$

This formula again is similar to (9) keeping in mind that $Q(x-k,x)$ is somewhat different, as discussed in the begining of this section.

6. ANALYSIS OF A METHOD USING A LOCAL OVERFLOW DIRECTORY

In this section we analyze the performance of a new overflow handling scheme. The basic idea is to have on each primary page a small overflow directory which stored the keys, record lengths and location of all overflow records of the bucket. If a record is not found in the primary page, the overflow directory is searched to determine whether the record exists as an overflow record. At most one additional access is required to find any overflow record and no additional accesses for an unsuccessful search. The overflow records do not necessarily have higher key values than the records in the primary page. Instead, which records become overflow records is determined by the record length. The smallest records are stored in the primary page and larger records in the overflow area. By keeping the smallest records in the primary page more records will fit on the page. The fraction of overflow records will be smaller and retrieval, on average, faster. To achieve this, the class to which the record belongs (or its length) must be indicated in the overflow directory.

It may be necessary to move a record from the primary page to the overflow area to make room for expanding the overflow directory. An entry in the overflow directory of a primary page would typically be much smaller than an actual record, even as much as an order of magnitude smaller Thus it is highly unlikely that a primary page would not have enough space for the overflow directory. If this occurs the problem can be solved either by reorganizing the file or by using a flag indicating that the bucket is full and a pointer to a second "primary page". In the analysis we ignore this case and assume that reorganization takes place.

In this scheme the probability $Q(b, x)$ that b records are stored in the primary page when the bucket contains x records is different from (5) because the length of records in the primary page is not random. Instead, the smallest b records are placed there. Let x records be selected, the b shortest of them from the first j classes $C_1, \ldots, C_j$. Let $n_1, \ldots, n_j$, where $n_j \neq 0$ and

$$\sum_{i=1}^{j} n_j = b,$$

be such a combination of b records. The remaining $x-b$ records have a length greater than or equal to l_j. Define $C(j, b, x)$ to be the sum of the probabilities of selecting x records where the b smallest are from the first j classes and the $x-b$ records from the classes $C_j, C_{j+1}, \ldots, C_L$. Furthermore, exactly b records must fit in the primary page. Then

$$Q(b, x) = \sum_{j=1}^{L} C(j, b, x).$$

The function $C(j, b, x)$ is given by:

$$C(j, b, x) = \sum_{\substack{n_1 + \cdots + n_j \\ n_j \neq 0}} \frac{b!}{n_1! \ldots n_j!} p_1'^{n_1} \ldots p_j'^{n_j} \times \binom{x}{b} \left(\sum_{i=1}^{j} p_i \right)^b \left(\sum_{i=r}^{L} p_i \right)^{x-b}, \quad (29)$$

where the summation is over all combinations such that

$$\sum_{i=1}^{j} n_i l_i \leqslant BS - (x-b)(KS + PS + LI)$$

and where $r > j$ is the minimum value that satisfies

$$l_r - KS - PS - LI + \sum_{i=2}^{j} n_i l_i > BS - (x-b-1)(KS + PS + LI).$$

LI is the size of the record length indicator.

The above formula can be explained as follows. The first b records are selected from classes C_1 to C_j while the last $x-b$ records are selected from classes C_r to C_L where $r > j$. This occurs with probability

$$\binom{x}{b} \left(\sum_{i=1}^{j} p_i \right)^b \left(\sum_{i=r}^{L} p_i \right)^{x-b}.$$

The b smallest records are distributed over the j classes with probability

$$\frac{b!}{n_1! \ldots n_j!} p_1'^{n_1} \ldots p_j'^{n_j}.$$

In this formula

$$p_i' = \frac{p_i}{\sum_{k=1}^{j} p_k}.$$

The conditions stated guarantee that the next smallest record, which is taken from class r, $r > j$, does not fit into the primary page.

If $r = j$ (i.e. n_j records from class r are inside the primary page and l records are outside) the function $C(j, b, x)$ is given by:

$$C(j, b, x) = \sum_{l=0}^{x-b} \sum_{\substack{n_1 + \cdots + n_j + l = b + l \\ n_j \neq 0}} \times \frac{(b+l)!}{n_1! \cdots (n_j + l)!} p_1'^{n_1} \ldots p_j'^{n_j + l} \times \binom{x}{b+l} \left(\sum_{i=1}^{j} p_i \right)^{b+l} \left(\sum_{i=j+1}^{L} p_i \right)^{x-b-l}, \quad (30)$$

where the summation is over all combinations such that

$$\sum_{i=1}^{j} n_i l_i < BS - (x-b)(KS + PS + LI)$$

and

$$l_r - KS - PS - LI + \sum_{i=1}^{j} n_i l_i > BS - (x-b-1)(KS + PS + LI).$$

Given the function $Q(b, x)$ the performance measures can be expressed as follows:

$$S(\alpha) = \frac{1}{X(\alpha)} \sum_{x=2}^{\infty} P(x, \alpha) \sum_{k=1}^{x-1} kQ(x-k, x), \quad (31)$$

$$U(\alpha) = 0, \quad (32)$$

$$R(\alpha) = \sum_{x=2}^{\infty} P(x, \alpha) \sum_{k=1}^{x-1} kQ(x-k, x), \quad (33)$$

$$SP(\alpha) = BS_0 \sum_{x=2}^{\infty} P(x, \alpha) \sum_{k=1}^{x-1} kQ(x-k, x). \quad (34)$$

7. NUMERICAL RESULTS AND COMPARISON

In this section we present numerical results computed using the formulae derived in the previous sections. We use these results to compare the performance of the four methods considered, and to analyze the performance effects of variable length records.

Figure 2 shows the expected number of disk accesses for a successful search as a function of the file expansion factor. The primary page size for this experiment was 4 kbytes. The records belonged to two classes. The record size of the first class was kept fixed at 400 bytes. The record size of the second class was 400, 300 and 201 in three different experiments. As expected, for the case of equiprobable classes, the cost decreases as the size of the records of the second class decreases. However, the cost reduction is not linear. It is apparent from that figure that the cost decreases more when the record length of the second class changes from 300 to 201 than when it changes from 400 to 300 bytes.

The above observation is explained as follows. There are two factors affecting the length of overflow chains (and therefore the expected cost). The first is the average record length. The smaller the average record length, the more records are needed to fill a primary page and therefore the lower is the expected number of records on the overflow chain. The second factor is the empty space left at the end of the primary page (this is not always true but holds in general, e.g. [9]). Intuitively, the larger the record size (relative to the page size), the more space will be wasted at the end of a primary page.

More wasted space implies longer overflow chains, on average. If there were no empty space left at the end of primary pages, then the distance between the three curves in Fig. 2 would be the same. Since the lowest average record length also implies less wasted space at the end of primary pages, the distance of the curves for $R2 = 201$ and 300 is greater than the distance for $R2 = 400$ and 300. In addition, as can be seen from the figures, when the probability of the class with the smallest record size increases the expected cost decreases. This is expected since both a larger number of records can be packed in a primary page and, at the same time, the average wasted space in the primary page is reduced because of smaller records.

Figure 3 shows a different experiment where the average record length was kept constant at 250 bytes for a block size of 500 bytes. As the variance of the record lengths changed, the expected wasted space at the end of the primary page changed, affecting the length of overflow chains. The expected length of an overflow chain for a large number of records per bucket was derived in Section 2. Base on this analysis it is easy to verify that the expected cost will be the highest for $R2 = 300$, followed by $R2 = 350$, 400 and 250. As the page size increases the wasted space at the end of a primary page becomes small relative to the page size. Thus the variance of the record length does not significantly affect the performance. This is shown in experimental results reported in [25].

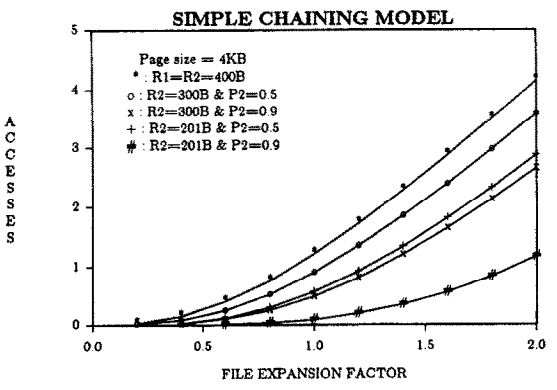


Fig. 2. Expected length of successful searches.

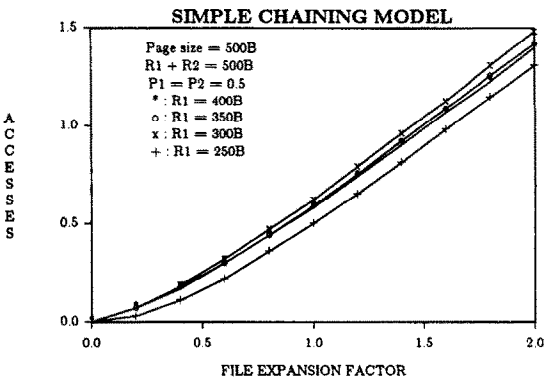


Fig. 3. Expected length of successful searches.

The performance figures for unsuccessful searches are similar to those for successful searches. They are not presented here but can be found in [25]. The cost of range searches (or sequential scans) and the space overhead increase linearly with the file expansion factor [25]. Figure 4 shows the expected space overhead as a function of the file expansion factor. Figure 5 shows the expected space overhead for the case of records with $R1 + R2 = 500$ and $BS = 500$. In this case the expected cost is highest for $R2 = 400$, followed by $R2 = 350$, 300 and 250. This order is different than the order based on the cost of successful searches. This is explained by the fact that the storage cost depends on the maximum record length, in addition to the length of the overflow chain, as was analyzed in Section 1. This dependence is not eliminated as the size of the primary page increases contrary to the cost of successful and unsuccessful searches.

The performance figures for successful, unsuccessful and range search when using chaining with large overflow pages are similar to those of simple chaining [25]. The most significant advantage of this method when records are of variable length is the reduced amount of storage space required for overflow records. Figure 6 shows the expected storage overhead for this method.

The expected cost of successful searches is much smaller when multiple chains are used, as shown in Fig. 7. The expected cost of an unsuccessful search is also much smaller than when using a single overflow chain [25]. The overflow space requirements and the range search cost are approximately the same.

Figure 8 shows the expected cost of a successful search for the method using local overflow directories. The expected cost is much lower than for the other techniques, especially for high values of the file expansion factor. This is due to the fact that at most one overflow page has to be read to find a record. A second observation from Fig. 8 is that, for equiprobable classes, the difference between the curves for $R2 = 400$ and 300 is approximately the same as the difference between the curves for $R2 = 300$ and 201. The reason is that, for large values of the

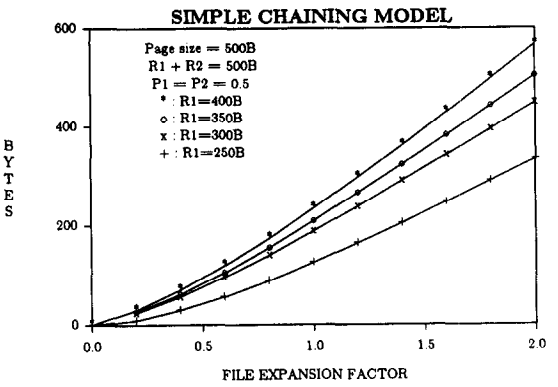


Fig. 5. Overflow space per bucket.

file expansion factor, a primary page is almost always filled with the smallest records. This implies little (and independent of the class) average wasted space per page. On the other hand, the expected cost of a successful search decreases as the probability of the class with the smallest records increases. The expected cost remains lower than for the other methods.

For all the other methods analyzed the cost of a succesful search is a convex function becoming asymptotically linear for large number of overflows. For the method using local overflow directories this is not true. The first part of the function is convex but the last part is concave. This is better shown in Fig. 9. Asymptotically, the curve approaches an additional cost equal to one. The cost of an unsucessful search is always zero. The expected cost for retrieving all the records in a bucket is shown in Fig. 10.

It is interesting to compare the storage overhead of this method (Fig. 11) with that of chaining (Fig. 4). In the case of a single record length, chaining has less expected storage overhead than the overflow directory method. However, when there are several record classes, the local overflow directory method may use less storage space. The reason is that, since the smallest records are stored in the primary page, it utilizes better the space of overflow pages with longer records. This also affects the range search cost since all the pages of a particular bucket have to be

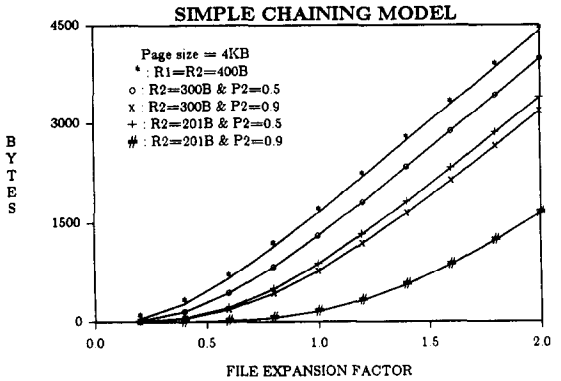


Fig. 4. Overflow space per bucket.

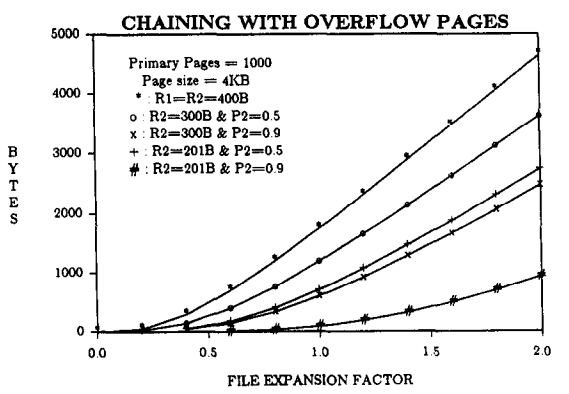


Fig. 6. Overflow space per bucket.

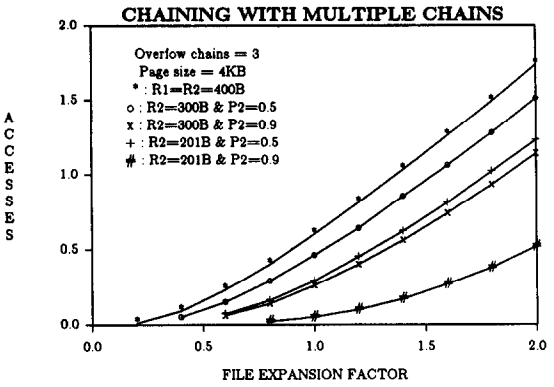


Fig. 7. Expected length of successful searches.

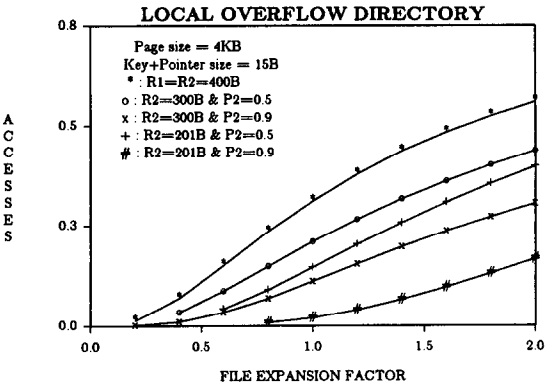


Fig. 8. Expected length of successful searches.

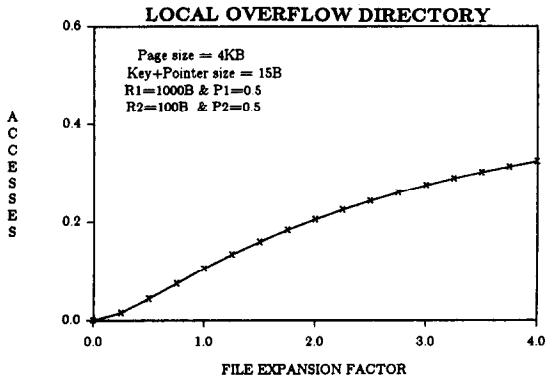


Fig. 9. Expected length of successful search.

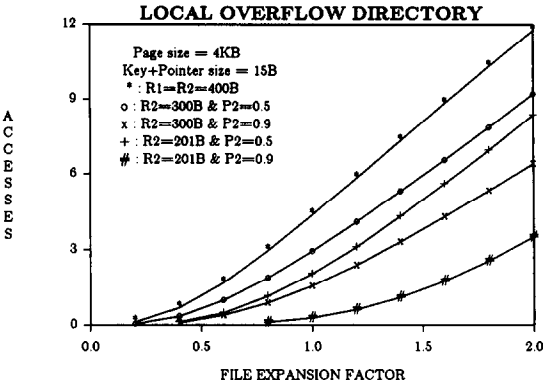


Fig. 10. Expected cost of retrieving all records of a bucket.

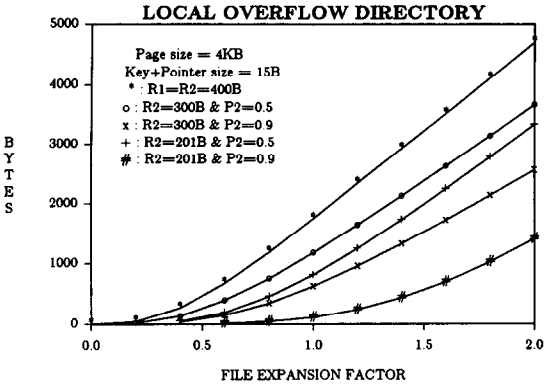


Fig. 11. Overflow space per bucket.

accessed (Fig. 10). The difference increases as the variance between the record lengths increases.

As a final conclusion, when the variation in record sizes is high, it appears that the overall performance of the overflow directory method is superior to the other methods analyzed, with the possible exception of changing with larger overflow pages which may require less space. A hybrid scheme combining the overflow directory approach with the use of large overflow pages may be the most cost effective solution. In addition to the reduced space requirements of the overflow directory method, it may also reduce the cost of range searches by clustering overflow records into larger pages.

8. CONCLUDING REMARKS

In this paper we have studied several overflow-handling techniques for variable length records and derived analytic estimates of their performance. Our formulae take into account the record length distribution of the underlying population of records. We found that a new overflow handling method, the overflow directory method, offers several advantages over other methods. It allows for a gradual reorganization of the file so that the smallest records are kept in the primary page, thus achieving better clustering. The net result is a reduction in the expected search lengths.

Future research involves the analysis of alternative placement and search algorithms for variable length records in an overflow area with large pages and analysis of more efficient ways of searching overflow chains. In addition, some approximation of the cost equations presented is desirable to reduce the cost of numerical computation.

REFERENCES

[1] S. Christodoulakis. Estimating block transfers and join sizes. *Proc. ACM–SIGMOD-83 Conf.* San Jose, pp. 40–54 (1983).

[2] S. Christodoulakis. Implications of certain assumptions in data base performance evaluation. *ACM Trans. Database Systems* 9(2), 163–186 (1984).

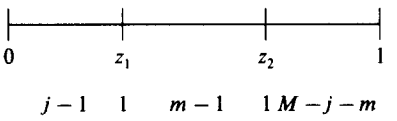
[3] G. Diehr and B. Faaland. Optimal pagination of B-trees with variable-length items. *Commun. ACM* **27**(3), 241–247 (1984).
[4] J. Hakola and A. Heiskanen. On the distribution of wasted space at the end of file blocks. *BIT* **20**(2), 145–156 (1980).
[5] L. L. Larmore and D. S. Hirschberg. Efficient optimal pagination of scrolls. *Commun. ACM* **28**(8), 854–856 (1985).
[6] E. M. McCreight. Pagination of B-trees with variable-length records. *Commun. ACM* **20**(9), 670–674 (1977).
[7] T. Montgomery. Designing magnetic tape files for variable length records. *Mgmt Informatics* **3**(6), 271–276 (1974).
[8] J. L. Szwarcfiter. Optimal multiway search trees for variable size keys. *Acta Informatica* **21**(1), 47–60 (1984).
[9] G. Wienderhold. *Database Design*, 2nd edn. McGraw-Hill, New York (1983).
[10] P. Å. Larson. Analysis of index-sequential files with overflow chaining. *ACM Trans. Database Systems* **6**(4), 671–680 (1981).
[11] D. Batory. Optimal file designs and reorganization points. *ACM Trans. Database Systems* **7**(2), 60–81 (1982).
[12] R. B. Cooper and M. K. Solomon. The average time until bucket overflows. *ACM Trans. Database Systems* **9**(3), 392–408 (1984).
[13] D. P. Heyman. Mathematical models of database degradation. *ACM Trans. Database Systems* **7**(4), 615–631 (1982).
[14] J. A. Behymer, R. A. Ogilvie and A. G. Merten. Analysis of indexed sequential and direct access file organizations. *Proc. ACM–SIGMOD Workshop on Data Description, Access and Control*, pp. 389–417 (1974).
[15] B. K. Gairola and V. Rajaraman. A distributed index sequential access method. *Inform. Process Letts* **5**(1) 1–5 (1976).
[16] T. Leipala. On the design of one-level indexed sequential files. *Int. J. Comput. Inform. Sci.* **10**(3), 177–186 (1981).
[17] T. Leipala. On optimal multilevel indexed sequential files. *Inform. Process. Letts* **15**(5), 191–195 (1982).
[18] Y. Manolopoulos. Batched search of index sequential files. *Inform. Process. Letts* **22**(5), 267–272 (1986).
[19] K. Maruyama and S. E. Smith. Optimal reorganization of distributed space disk files. *Commun. ACM* **19**(11), 634–642 (1976).
[20] J. K. Mullin. An improved index sequential access method using hashed overflow. *Commun. ACM* **15**(5), 301–307 (1972).
[21] K. F. Wong and J. C. Strauss. An analysis of ISAM performance improvement options. *Mgmt Datamatics* **4**(3), 95–107 (1975).
[22] T. J. Teorey and J. P. Fry. *Design of Database Structures*. Prentice Hall, Englewood Cliffs, New Jersey (1982).
[23] D. G. Keen and J. O. Lacy. VSAM data design parameters. *IBM Systems J.* **13**(3), 186–212 (1974).
[24] D. E. Knuth. *The Art of Computer Programming. Vol. 1: Fundamental Algorithms*, 2nd edn. Addison-Wesley, Reading, Mass. (1973).
[25] S. Christodoulakis, Y. Manolopoulos and P. Å. Larson. File organizations for variable length records. *Computer Science Department, University of Waterloo*, (1987).
[26] W. Feller. *An Introduction to Probability and its Applications*, Vol. 1, 3rd edn. Wiley, New York (1968).
[27] T. Nakamura and T. Mizoguchi. An analysis of storage utilization in block split data structuring scheme. *Proc. VLDB-78 Conf.* Berlin pp. 489–495 (1978).

APPENDIX

Probability Mass Assigned to a Bucket

Consider an arbitrary but fixed bucket immediately after initial loading. It is assumed that the keys loaded are a random sample from the underlying key distribution. Denote the cumulative key distribution by $F(x)$. Let the keys assigned to the bucket under consideration be K_{j+1}, $K_{j+2}, \ldots, K_{j+m}$. The variables $y_i = F(K_i)$, $i = 1, 2, \ldots, M$, are mutually independent and uniformly distributed in [0, 1]. The total probability mass, that is, the probability that a randomly selected key will be inserted into the bucket, assigned to the bucket as a result of initial loading is then $y_{j+m} - y_j = F(K_{j+m}) - F(K_j)$. We therefore need the distribution of the random variable $x = y_{j+m} - y_j$, where y_1 is the lowest value obtained in sample of size M from the uniform distribution, y_2 is the next higher value, ..., and y_M is the highest value obtained.

$$\binom{M-2}{j-1} M \binom{M-j-1}{m-1} M-1$$



We first need the joint probability $P(y_j = z_1, y_{j+m} = z_2)$ where $z_1, z_2 \in [0, 1)$ and $z_1 \leqslant z_2$. To have $y_j = z_1$ and $y_{j+m} = z_2$, we must in the random sample have $j-1$ values less than z_1, one value equal to z_1, $m-1$ values between z_1 and z_2, one value equal to z_2, and the rest $(M-j-m)$ greater than z_2 (see the above). The variable with value z_1 can be chosen in M ways, the one having value z_2, in $M-1$ ways, the $j-1$ variables less than z_1 in

$$\binom{M-2}{j-1}$$

ways, and the $m-1$ variables between z_1 and z_2, in

$$\binom{M-j-1}{m-1}$$

different ways. This gives the following probability:

$$P(y_j = z_1, y_{j+m} = z_2)$$
$$= M(M-1)\binom{M-2}{j-1} \times \binom{M-j-1}{m-1} z_1^{j-1} (z_2 - z_1)^{m-1} (1 - z_2)^{M-j-m}.$$

Let $K = M(M-1)\binom{M-2}{j-1}\binom{M-j-1}{m-1}$.

The probability that $y_{j+m} - y_j = x$ is then obtained as:

$$P(y_{j+m} - y_j = x) = \int_0^{1-x} P(y_j = z_1, y_{j+m} = z_1 + x)\, dz_1$$
$$= K x^{m-1} \int_0^{1-x} z_1^{j-1} (1 - z_1 - x)^{M-j-m}\, dz_1.$$

Now perform the variable substitution $z_1 = t(1-x)$ which yields:

$$= K x^{m-1}(1-x)^{M-m}\int_0^1 t^{j-1}(1-t)^{M-j-m}\,dt$$

$$= \frac{M(M-1)(M-2)!}{(j-1)!(M-j-1)!}\cdot\frac{(M-j-1)!}{(m-1)!(M-j-m)!}$$

$$\times\frac{(j-1)!(M-j-m)!\,x^{m-1}(1-x)^{M-m}}{(M-m)!}.$$

After simplification we finally obtain:

$$P(y_{j+m}-y_j=x) = M\binom{M-1}{m-1}x^{m-1}(1-x)^{M-m}. \quad (1)$$

Record distribution after insertions

The probability that n of the N records inserted hit the bucket, given that it has been assigned a probability mass of x, has a binomial distribution:

$$P(n|x) = \binom{N}{n}x^n(1-x)^{N-n}.$$

The probability that the bucket receives n out of the N records is then:

$$P_m(n) = \int_0^1 P(n|x)P(x)\,dx$$

$$= M\binom{M-1}{m-1}\binom{N}{n}$$

$$\times\int_0^1 x^n(1-x)^{N-n}x^{m-1}(1-x)^{M-m}\,dx$$

$$= M\binom{M-1}{m-1}\binom{N}{n}\frac{(n+m-1)!(M+N-m-n)!}{(M+N)!}$$

$$P_m(n) = \frac{m}{m+n}\frac{\binom{M}{m}\binom{N}{n}}{\binom{M+N}{m+n}}.$$

We can find the asymptotic distribution by letting M, $N\to\infty$, keeping $M/N=\alpha$ constant. From (2) we have:

$$P_m(n) = \frac{m}{m+n}\frac{(m+n)!}{m!\,n!}$$

$$\times\frac{M(M-1)\ldots(M-m+1)N(N-1)\ldots(N-n+1)}{(M+N)(M+N-1)\ldots(M+N-m-n+1)}.$$

Divide both the nominator and denominator by M^{m+n}, noting that $M+N=M(1+\alpha)$. This gives:

$$= \frac{m}{m+n}\frac{(m+n)!}{m!\,n!}$$

$$\frac{1\left(1-\frac{1}{M}\right)\ldots\left(1-\frac{m-1}{M}\right)\alpha\left(\alpha-\frac{1}{M}\right)\ldots\left(\alpha-\frac{n-1}{M}\right)}{(1+\alpha)\left(1+\alpha-\frac{1}{M}\right)\ldots\left(1+\alpha-\frac{m+n-1}{M}\right)}.$$

Letting $m\to\infty$ we finally obtain:

$$P_m^*(n\alpha) = \frac{(m+n-1)!}{(m-1)!\,n!}\frac{\alpha^n}{(1+\alpha)^{m+n}}. \quad (3)$$

The asymptotic distribution is simply a negative binomial distribution with parameter $p = 1/(1+\alpha)$ [26]. The expected number of records per bucket and its variance are therefore: $\mu = m+m\alpha = m(1+\alpha)$, $\sigma^2 = m(1+\alpha)\alpha$.

The fact that the asymptotic distribution is a negative binomial gives an alternative way of looking at the problem. Assume that we have an infinite supply of balls (records), of which a fraction $p = 1/(1+\alpha)$ are labelled "old" and the rest are labelled "new". We randomly select balls until we have m balls with label "old". Then the probability of having n balls with label "new", for a total of $m+n$ balls, is exactly $P_m^*(n,\alpha)$ above.

The probability distribution in [10] was derived using the expected value of the insertion probability mass assigned to a bucket and ignoring the fact that it is a random variable. This simplification leads to a poor approximation which underestimates the number of overflow records. The correct distribution was first derived in [11]. The first part of the derivation above, up to equation (2), is based on the same idea as the derivation in [11]. The asymptotic distribution given by equation (3) has not previously been available.