Analysis of Signature Generation Schemes for Multiterm Queries In Partitioned Signature File Environments

Deniz Aktug[*] Fazli Can[†]

[*]Miami University, commons-admin@lib.muohio.edu

[†]Miami University, commons-admin@lib.muohio.edu

TECHNICAL REPORT: MU-SEAS-CSA-1993-007

Analysis of Signature Generation Schemes for Multiterm Queries In Partitioned Signature File Environments

Deniz Aktug and Fazli Can

ANALYSIS OF SIGNATURE GENERATION SCHEMES FOR MULTITERM QUERIES IN PARTITIONED SIGNATURE FILE ENVIRONMENTS[*]

by

Deniz AKTUG Fazli CAN

Systems Analysis Department
Miami University
Oxford, OH 45056

Working Paper #93-007 May, 1993

[*] This paper has been submitted for outside publication and will be copyrighted if accepted. It is being published in technical report form to expedite the dissemination of its contents. Its distribution should therefore be limited to peer communication and specific requests.

ANALYSIS OF SIGNATURE GENERATION SCHEMES FOR MULTITERM QUERIES IN PARTITIONED SIGNATURE FILE ENVIRONMENTS

Deniz AKTUG **Fazli CAN**[*]

Department of Systems Analysis
Miami University
Oxford, OH 45056

April 28, 1993

Abstract

Our analysis explores the performance of three superimposed signature generation schemes as they are applied to a dynamic signature file organization based on linear hashing: Linear Hashing with Superimposed Signatures (LHSS). First scheme (SM) allows all terms set the same number of bits whereas the second and third methods (MMS and MMM) emphasize the terms with high discriminatory power. In addition, MMM considers the probability distribution of the number of query terms. The main contribution of the study is the combination of signature generation and signature file organization concepts together with the relaxation of the single term query and uniform frequency assumptions. The derivation of the performance evaluation formulas are provided as well as the analysis of various experimental settings. Results indicate that MMM outperforms the others as terms become more distinctive in their discriminatory power. MMM accomplishes the highest savings in retrieval efficiency for the high query weight case. We also discuss the applicability of the derivations to other partitioned signature organizations providing a detailed analysis of Fixed Prefix Partitioning (FPP) as an example. Finally, an approximate performance evaluation formula that works for both FPP and LHSS is modified to account for the multiterm case.

Key Words and Phrases: *Access methods, descriptors, document retrieval, dynamic file, file design, hashing, information retrieval, multimedia data, office automation, signature files, superimposed coding, term discrimination power, text retrieval*

[*] To whom all correspondence should be addressed
voice: (513) 529-5950, fax: (513) 529-3841, e-mail: fc74sanf@miamiu.bitnet

1. INTRODUCTION

Information retrieval systems (IRSs) find the data items that are relevant to the submitted user queries. A multimedia database can consist of formatted stored objects as well as the unformatted ones like text, voice or image. Full text/object scanning, inverted indexes, clustering and signature files are some of the basic IR techniques [CAN90, CHR84, FAL85, SAL89, TIB91, ZEZ91]. The concern of our study is signature files which are widely used in formatted and unformatted databases for multiattribute query processing.

Throughout the paper, data items stored in the database (formatted, unformatted or combined) will be referred to as "records" or "objects." A signature file is created when contents of the objects are represented as bit strings which are stored in a separate file. Upon a retrieval request, the signature of the query is created and compared against the entries of the signature file to find the qualifying signatures whose corresponding objects are to be retrieved as a response to the submitted query. Then only those objects with the qualifying signatures are retrieved. However, due to the information loss that takes place during signature generation, some signatures seem to qualify the query although the corresponding objects do not. This situation, known as a false drop or a false match, leads to unnecessary disk accesses since it cannot be detected until the original data objects are accessed.

Figure 1 provides an example for signature generation using Superimposed Coding where each term is hashed to a bit string where it sets a predefined number of bits to 1. The individual term signatures are then superimposed to form the object signature. Example queries that result in no match, true match and a false match conditions are also provided.

object	signature	generation

terms	term signatures		
object	1000	1000	
signature	0010	0100	
generation	1000	1000	
	============		
	1010	1100	<= object signature

query	query signature		result
database	1100	0000	no match
generation	1000	1000	true match
information	1010	0000	false match

Figure 1. Signature extraction.

The main virtue of a signature file is to act as a filter to eliminate a large number of non-qualifying records and reduce the number of disk accesses required to process a particular query. The size of a signature file is typically ten to twenty percent of the actual size of the source database from which signatures are extracted. This enables the signature files to provide retrieval efficiency during query processing without generating much of a storage overhead [CHR84, ZEZ91]. Signatures are also flexible enough to efficiently handle object insertions and deletions which happen frequently in dynamic environments [ZEZ91]. The use of appropriate signature generation schemes that will minimize the occurrence of false drops has been discussed in [AKT93a, AKT93b, FAL85, FAL87a, FAL87b, FAL87c, FAL88, FAL92, LEN92]. Also many signature file organizations that will provide faster response times even for very large databases without generating too much storage overhead and extra difficulty in update operations have been proposed. Examples include bit and frame sliced structures [LIN92], two-level organizations [CHA89, CHA92, SAC85, SAC87], multiorganizational schemes [KEN90], S-Tree [DEP86], indexed descriptor files [PFA80], signature trees [THA88], and partitioned organizations [GRA92, LEE89, ZEZ91].

This study concerns the comparison of three signature generation schemes as they are applied to a dynamic signature file organization structure known as Linear Hashing with Superimposed Signatures (LHSS) or Quick Filter [ZEZ88, ZEZ91]. The first scheme treats all terms in an identical way, neglecting the differences in their occurrence and query frequencies, whereas the second and third schemes actually make use of such differences in generating the term signatures. The second scheme uses an optimal signature generation strategy which is based on the assumption that only single term queries are submitted whereas the third scheme considers multiterm queries as well [FAL85, FAL87a, FAL88].

The main contribution of the paper is the derivation of the performance evaluation formulas to compute the retrieval efficiency for the selected file organization for each of the above schemes in an environment where both single and multiterm queries are submitted. We not only relax the uniform frequency and single term query assumptions, but also present the application of the suggested schemes in the dynamic file structure LHSS together with an analytical analysis for queries with any number of terms. Furthermore, we discuss the applicability of our derivations to other partitioned signature file organizations using Fixed Prefix Partitioning (FPP) as an example to illustrate our point. We also propose a modified approximate formula for more practical applications in multiterm query environments where an easier way to estimate the performance of FPP and LHSS methods is required.

Section 2 explains LHSS and presents the performance evaluation formulas. Section 3 defines the three signature generation schemes. The derivation of the performance evaluation formulas for these schemes is given in Section 4, whereas their application to our specific test cases is explained in Section 5. Section 6 presents the results of the experimental analysis. Section 7 discusses how our derivations can be used to evaluate the performance of other partitioned schemes in multiterm query environments with a special emphasis on FPP and provides a modified approximate formula applicable to both methods. The efficiency relevancy considerations for LHSS, FPP and other partitioned environments in general are discussed in Section 8. The guidelines for the application of the findings to real life cases are provided in Section 9, conclusion and some future research pointers are presented in Section 10.

2. LINEAR HASHING WITH SUPERIMPOSED SIGNATURES (LHSS)

Linear hashing is an efficient way to organize partitioned dynamic files [LIT80]. A derived method, which is originally introduced by Zezula is linear hashing with superimposed signatures [ZEZ88].

2.1 The Method

LHSS provides a method for mapping signatures to storage pages and processing the queries to find qualifying signatures. The primary component of LHSS is a split function which converts the key of each signature into an integer in the address space {0, 1, . . . , n-1} where $2^{h-1} < n \leq 2^h$ is satisfied for some integer h. The hashing function is defined as follows [ZEZ88, ZEZ91].

$$g(s_i, h, n) = \begin{cases} \sum_{r=0}^{h-1} b_{F-r} 2^r & \text{if } \sum_{r=o}^{h-1} b_{F-r} 2^r < n \\ \sum_{r=0}^{h-2} b_{F-r} 2^r & \text{otherwise} \end{cases} \tag{1}$$

where b_i is the value of the ith binary digit of the object signature, F is the signature size, h is the hashing level, n is the number of addressable (primary) pages and s_i is the object signature i. (For easy reference, the definition of the important symbols of this section is provided in Table I.)

For the initial condition, h=0, n=1, and $g(s_i, 0, 1)$ is defined as 0. In simple terms, the hashing function, g, uses the last h or (h-1) bits of a signature to determine the number of the addressable page where signature s_i is to be stored. If the storage limit of a primary page is exceeded, an overflow page is created, linked to the primary page and the last signature that has caused the overflow is placed in the overflow page and, a "split" is initiated, i.e., a new primary page is created. A split pointer, SP (with an initial value of 0), keeps track of the next primary page to be split. Whenever a split takes place, all signatures on the page pointed to by SP, together with those in the associated overflow page(s) are rehashed. The nature of the hashing function guarantees that the rehashed signatures either remain in the same page or are transferred to the page that has just been created. The hashing level is increased by one just before page zero is split, and following each split process the new value of SP is computed as SP = (SP + 1) mod 2^{h-1}. Note that at a given time in the signature file it is possible to have pages which are hashed at levels h and (h-1). Note also that linear hashing is space efficient and does not lead to many overflows [LIT80].

Table I. Definition of Important Symbols for Section 2

b_i	: value of the i^{th} binary digit of the term signature
h	: hashing level
k	: no. of bits set to 1 in the h-bit suffix of the final query signature
n	: no. of addressable pages
s_i	: i^{th} object signature
EXPH(W(Q),h]	: expected number of bits set in the h-bit suffix of a signature whose weight is W(Q)
F	: size of a signature in bits
N(n, h, W(Q))	: no. of pages that do not need to be accessed
P(j)	: probability that j bits are set in the h-bit suffix of the query
P(W(Q), h)	: probability of access savings
R(h)	: no. of pages hashed at level h
W(Q)	: query weight, i.e., the no. of 1s in query signature

During query processing a page qualifies if all bit positions that are set in the query signature are also set in the page signature. For simplicity, if we assume that $n = 2^h$ and if there is a query signature with k 1s in its h-bit suffix, then it is necessary to access 2^{h-k} primary pages (and the associated overflow pages). More number of 1s in the last h-bit suffix of a query makes the query processing faster. Note that even if a signature in the selected page seems to qualify the query, the associated data object might not contain all query terms. Hence a false drop resolution is required using the original query before the qualifying objects are returned to the user.

2.2 Performance Evaluation

It has been shown that the number of page savings can be computed as a function of the number of addressable pages (n), the hashing level (h), and the number of 1s in query signature, i.e., the query weight (W(Q)) provided that the signature size is kept fixed at F [ZEZ91].

Let EXPH(W(Q), h) be the expected number of bits set in the h-bit suffix of the query signature.

$$EXPH(W(Q),h) = \sum_{j=1}^{\min\{h,W(Q)\}} j * P(j) \tag{2}$$

where P(j) is the probability that j bits are set in the h-bit suffix of the query and can be written as follows.

$$P(j) = \frac{\binom{F-h}{W(Q)-j}\binom{h}{j}}{\binom{F}{W(Q)}} \tag{3}$$

Next probability of access savings, P(W(Q), h), can be defined as the proportion of the number of pages that do not need to be accessed (while processing a particular query) to the total number of addressable pages. Hence

$$P(W(Q), h) = 1 - \frac{npa}{n} \tag{4}$$

where npa is the number of pages accessed, n is the number of addressable pages in the signature file. Note that only $2^{h-EXPH(W(Q),\, h)}$ number of pages need to be accessed. So when $2^h = n$

$$npa = \frac{n}{2^{EXPH(W(Q),h)}} \tag{5}$$

and

$$P(W(Q),h) = 1 - \frac{n / 2^{EXPH(W(Q),h)}}{n}$$

$$= 1 - \frac{1}{2^{EXPH(W(Q),h)}} \tag{6}$$

When $n = 2^h$, SP = 0 and all pages are hashed at level h. As soon as a page split takes place, the value of h is increased by 1 and both page 0 and the new page are rehashed at this level. Since each split results in the rehashing of two pages, number of addressable pages hashed with level h, R(h) can be defined as

$$R(h) = 2(n - 2^{h-1}) = 2n - 2^h$$

where 2^{h-1} is the number of addressable pages when all pages are hashed at level (h-1). The difference between n and 2^{h-1} indicates the number of page splits that have taken place since then. Each split results in the rehashing of two pages, so the multiplication of the number of splits by two gives the number of pages hashed at level h.

It follows that

$$R(h - 1) = n - R(h) = 2^h - n$$

Finally, the total number of page savings, N(n, h, W(Q)), is defined as the number of pages that need not be accessed for a given query and can be expressed as follows.

$$N(n, h, W(Q)) = R(h)\ P(W(Q), h) + R(h - 1)\ P(W(Q), h - 1) \qquad (7)$$

Table II. Definition of Important Symbols for Sections 3-6

b_i	: no. of terms from S_i in a query
c_j	: no. of bit set by the j th query term
D	: expected no. of distinct terms in a record
D_i	: expected no. of distinct terms of S_i in a record
h	: hashing level
k	: possible values for the additional no. of bits set to 1 after the first stage
m	: no. of bits a term sets to 1 (when each term sets the same number of bits)
m_i	: no. of bits set by a term from S_i
n	: node number in Figure 2 ($0 \leq n \leq t$)
n_s	: number of disjoint sets
nqt	: no. of terms in a query ($nqt \leq t$)
q_i	: probability that query term is from S_i, given a term signature
t	: maximum no. of terms in a query
EXPH	: expected no. of bits set in the h-bit suffix of the query for a specifically identified outcome
EXTSAV	: extra percent savings provided by MMM over SM or MMS
F	: size of a signature in bits
HW	: case in which queries with high weights are frequent
LW	: case in which queries with low weights are frequent
OEXPH	: expected no. of bits set in the h-bit suffix of the query considering all possible query outcomes
P_i (k)	: probability that exactly k terms will be specified from S_i
P_j	: probability that j terms are specified in a query
PERSAV	: percentage of the addressable pages that do not have to be accessed
S_i	: set i of the terms with similar discriminatory power ($1 \leq i \leq n_s$)
UD	: case in which the probability distribution of the no. of query terms is uniform, i.e., the P_i (k) values are equal
Y_s	: weight of a query signature after s term signatures are superimposed

3. LHSS BASED ON TERM CHARACTERISTICS FOR SINGLE AND MULTITERM QUERIES

Faloutsos and Christodoulakis have suggested grouping all terms in the database into n_s number of disjoint sets ($S_1, S_2, \ldots, S_{n_s}$) based on the frequency with which they are specified in the queries [FAL85]. All terms in a given S_i ($1 \leq i \leq n_s$) set the same number of bits in generating their signatures. The optimal number of bits set by the terms in set i

(S_i), m_i, is computed by taking the query and occurrence frequency of the terms into account. (For quick reference, the definition of the symbols for Sections 3 to 6 are provided in Table II.)

The approach is based on the observation that the terms with lower database occurrence frequency are specified more frequently in the queries. Such terms are said to have high discriminatory power in the sense that they efficiently determine those documents that are most relevant to the query. Since terms with high discriminatory power are more important, they should be given the privilege to set relatively more number of bits in their associated term signatures. Unlike some other studies [LEN92, FAL87b], this approach can be used to account for multiterm queries and eliminates the need for a lookup table. The purpose is to minimize the false drop probability by using the differences between the term discriminatory power values.

The query frequency is represented by q_i where q_i is the probability that a query term is from S_i, and $(q_1 + q_2 + \ldots + q_{n_s}) = 1$. The occurrence frequency, on the other hand, is reflected in the D_i values where D_i is the average number of terms in a record that are from S_i, and $D = (D_1 + D_2 + \ldots D_{n_s})$, and D is the average number of terms in a record.

TABLE III. Optimum Weight Assignment Formulas for SM, MMS and MMM Cases

Method	Formula (equation no.)	References
Single m (SM)	$m = \frac{F\ln 2}{D}$ (8)	[FAL85]
Multiple m Based on Single Term Queries (MMS)	$m_i = \frac{F\ln 2}{D} + \frac{1}{\ln 2}\left[\ln\frac{q_i}{D_i} - \frac{\sum_{i=1}^{n_s} D_i \ln\frac{q_i}{D_i}}{D}\right]$ (9)	[FAL85]
Multiple m Based on Multiple Term Queries (MMM)	$m_i = \frac{F\ln 2}{D} + \frac{\sum_{i=1}^{n_s} D_i L_i}{D\ln 2} - \frac{L_i}{\ln 2}$ where $L_i = \ln\left[\frac{P_i(0)}{P_i(1)} D_i\right]$ (10)	[FAL87a, FAL88]

Table III shows the formulas for the optimal assignment strategies for three signature generation schemes which are based on this approach. Single m (SM) case refers to the method in which all terms are assumed to have the same occurrence and query frequencies and hence set the same number of bits regardless of their discriminatory power. This is a crude way to generate term signatures but the results can serve as a reference point against which the performance of other more sophisticated signature creation schemes can be evaluated. The derivation of the formula for the Multiple m based on Single queries (MMS) case is based on the occurrence of single term queries only and hence the resulting m_i values tend to be sub optimal when they are applied to the environments where multiterm queries are also possible. Multiple m based on Multiterm queries (MMM) case not only treats terms differently based on their discriminatory power, but also takes multiterm queries into account. Hence it is expected to give the largest savings in retrieval for our experimental settings. Yet there is additional practical overhead incurred in finding the optimal m_i values with this method rising from the need to estimate the $P_i(k)$ values where $P_i(k)$ is the probability that exactly k terms will be specified from S_i. In fact, this is one valid reason to consider the performance of MMS: It might be plausible to be content with the output provided by MMS if we are convinced that MMS provides satisfactory amount of savings. The m_i formula for MMM case is a good approximation of a complex method which gives the exact solution. The formula in Table III gives better results for large m_i values when $P_i(0) \neq 0$, $P_i(1) \neq 0$ and they are of the same order of magnitude.

4. PERFORMANCE EVALUATION FOR SM, MMS and MMM CASES

4.1 *Finding the Distribution of Query Weight*

In order to compute the value of PERSAV (percentage of the addressable pages that do not have to be accessed) for a particular experimental setting, expected number of 1s in the h-bit suffix of the query signature should be known. This expected value is a function of the query weight (see equation 2). For single term queries, query weight is a known constant and equals to the number of bits set by the only query term. For multiterm queries, however, query weight is a random variable and therefore has a probability distribution.

When nqt terms are used in a query, the query signature can be generated by superimposing the nqt individual term signatures. Let c_j be the number of bits set by the j^{th} query term (where $2 \leq j \leq nqt$) and F be the size of the signature. Then the possible values for the query weight range from $\max\{c_j\}$ $(1 \leq j \leq nqt)$ to $\min\{(c_1 + c_2 + \ldots + c_{nqt}), F\}$. The lower limit is associated with the case where all (nqt-1) terms set the same bits that have already been set by term nqt* where $c_{nqt^*} = \max\{c_j\}$. Typically, $c_1 + c_2 + \ldots + c_{nqt} < F$ is quaranteed and the upper limit refers to the case where all nqt terms set different bits.

The process of superimposing nqt term signatures can be viewed as an algorithm consisting of nqt stages. At the initial stage, c_1 of the F bits are set by the first term. If we let Y_j indicate the number of bits set to 1 after the jth stage, $Y_1 = c_1$ holds by definition. Notice that $c_1 = Y_1 \leq Y_2 \leq \ldots \leq Y_{nqt} = W(Q)$ and our concern is to find the probability distribution of Y_{nqt}, which is the number of 1s in the query signature at the nqtth stage, i.e., after nqt term signatures are superimposed. The study reported in [MUR92] has indicated that $P\{Y_s = u \mid Y_{s-1} = y, Y_{s-2} = z, \ldots, Y_1 = c_1\} = P\{Y_s = u \mid Y_{s-1} = y\}$ which means that the random variables $Y_1, Y_2, \ldots, Y_{nqt}$ form a Markov Chain [FEL68]. Using the concepts of one-step-transition probabilities together with some matrix manipulation techniques from Linear Algebra, [MUR92] has come up with an expression for the probability distribution of the query weight, conditioned on the number of 1s set by the first query term. That is

$P(W(Q) = k+c_1 \mid c_1) =$

$$\sum_{i=0}^{k} \binom{F-c_1}{i}\binom{F-c_1-i}{k-i}(-1)^{k+i} \prod_{r=2}^{nqt} \frac{\binom{c_1+i}{c_r}}{\binom{F}{c_r}} \qquad (11)$$

In the above expression, k stands for the possible values for the number of additional bit positions that are set after the first stage and F is the signature size as usual.

Recall that our aim is to find a way to compute the expected number of 1s set in the h-bit suffix of the query signature which is dependent on the query weight. The query weight, however, is no longer a constant and has a probability distribution which depends on the number of term signatures that are superimposed, nqt, together with the number of bits set to 1 by each term, c_j, and is conditioned on the number of bits set by the first query term, c_1. Hence, we not only should differentiate among the queries based on the number of terms they have, but also on the c_j values of these terms and the number of bits set by the first term, c_1.

4.2 A Closer Look at the Query Structure

Assume that the terms in the database can be grouped into two sets, S_1 and S_2, where S_1 contains the ones with high discriminatory power. The terms from S_i set m_i $(1 \leq i \leq 2)$ number of bits and therefore c_j equals to m_1 or m_2. Let t be the maximum number of terms that can be used in a query and let P_j indicate the occurrence probability of a query with j terms where $(P_1 + P_2 + \ldots + P_t) = 1$ is satisfied.

The tree diagram in Figure 2 enables us identify all different query combinations based on the criteria described Section 4.1. At any such combination, which is represented as a final outcome, we know the answers to these three questions:

1. How many terms are there in the query?
2. How many terms are from S_1 and how many are from S_2, i.e., how many of the query terms set m_1 bits and how many of them set m_2 bits?
3. Which set does the first query term belong to, i.e., how many bits does the first term set?

This information enables us to compute the value of $P(W(Q) = k+c_1 \mid c_1)$ for each query outcome for those values of k that are realizable. Next, we can compute the value of the expected number of 1s in the h-bit suffix of the query for every such outcome. (The derivations for this computation will be provided in the next section.)

So far we have clarified our reason to identify different query outcomes, now we can concentrate on the way the tree diagram is constructed: If the tree is traced from left to right, the correspondence between the branching procedure and the logical sequence of the events can be seen. Starting from the leftmost node, numbered as 0, we encounter t possibilities, each corresponding to a query with 'nqt' terms, where nqt stands for the number of query terms and ranges from 1 to t. Each of the t branches symbolize one of these t events (i.e., specification of a query with nqt terms) and the probability associated with each event is indicated on its corresponding branch. Note that the sum of the probabilities associated with the branches emanating from a particular node adds up to 1. The submission of a single term query takes us to node 1 at which we have two possibilities: The term is either from S_1 or S_2. Let b_i be the number of terms from S_i ($1 \leq i \leq 2$) in a query. Then

$$\sum_{i=1}^{2} b_i = nqt$$

should be satisfied. Therefore, it is sufficient to use just b_1 (or b_2) to specify a query combination, once nqt is known. For a single term query, the possible values for b_i are 0 and 1 where $P\{b_1 = 1 \mid nqt = 1\} = q_1$ and $P\{b_1 = 0 \mid nqt = 1\} = q_2$.

These two conditions take us to two final outcomes which can not be split up any further. From any node n ($2 \leq n \leq t$), where n = nqt, (nqt+1) branches emanate, each corresponding to one possible value for b_1 in the range 0 to nqt.

$$P\{b_1 = v \mid nqt = V\} = \binom{V}{v} q_1^v q_2^{(V-v)} \qquad (12)$$

where $0 \leq v \leq V$ and $2 \leq V \leq t$.

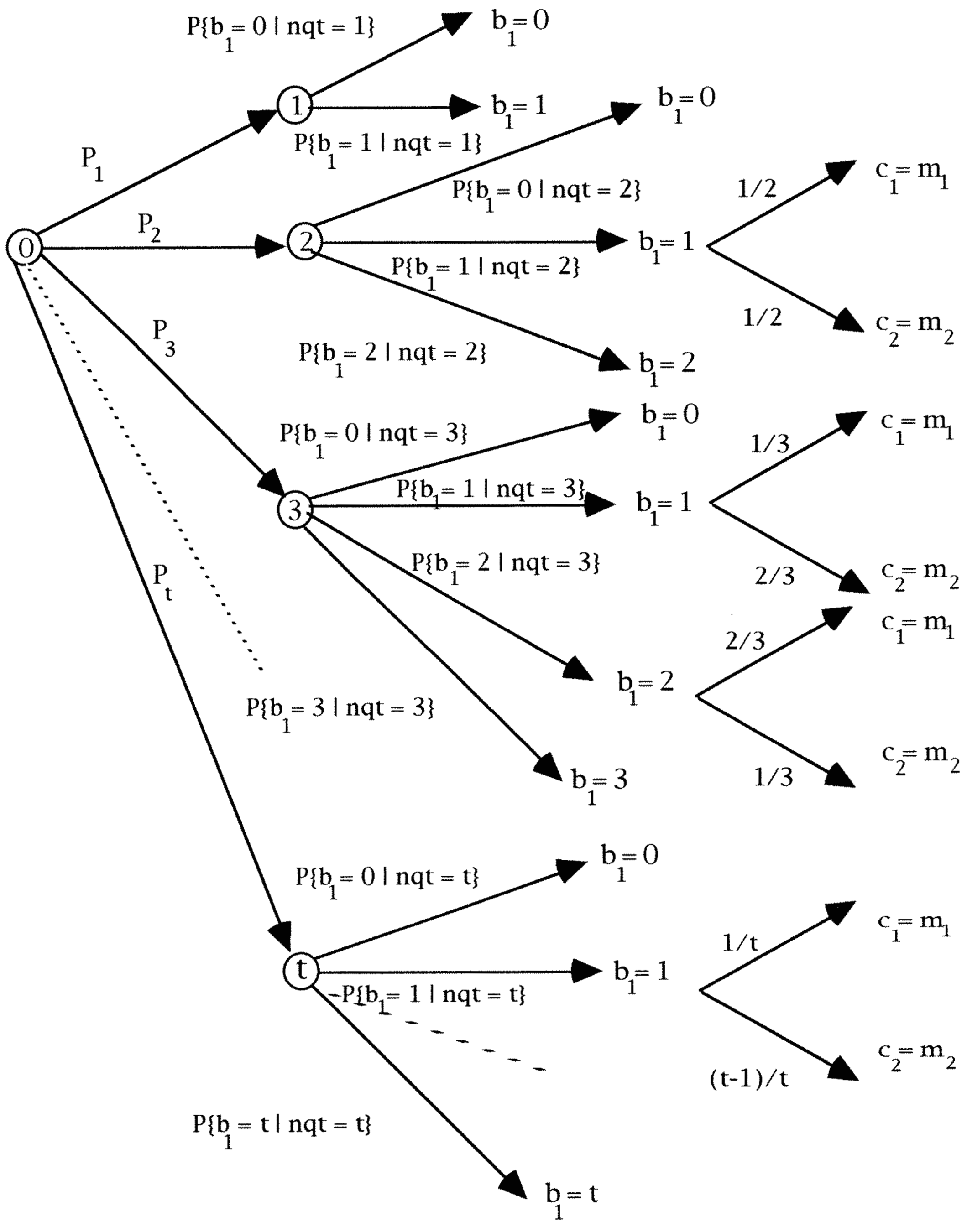
P_1
P_2
P_3
P_t
$P\{b_1 = 0 \mid nqt = 1\}$
$P\{b_1 = 1 \mid nqt = 1\}$
$P\{b_1 = 0 \mid nqt = 2\}$
$P\{b_1 = 1 \mid nqt = 2\}$
$P\{b_1 = 2 \mid nqt = 2\}$
$P\{b_1 = 0 \mid nqt = 3\}$
$P\{b_1 = 1 \mid nqt = 3\}$
$P\{b_1 = 2 \mid nqt = 3\}$
$P\{b_1 = 3 \mid nqt = 3\}$
$P\{b_1 = 0 \mid nqt = t\}$
$P\{b_1 = 1 \mid nqt = t\}$
$P\{b_1 = t \mid nqt = t\}$
$b_1 = 0$
$b_1 = 1$
$b_1 = 2$
$b_1 = 3$
$b_1 = t$
1/2
1/3
2/3
1/t
(t-1)/t
$c_1 = m_1$
$c_2 = m_2$

Figure 2. Tree diagram for query outcomes.

Starting from node n, we can end up in any one of the (n+1) outcomes. However, some of them can further be split up so that we will have the information to answer the three questions that are listed at the beginning of this section. At each of these n+1 outcomes, the number of query terms and the number of 1s set by each term are known. For the first and $(n+1)^{st}$ outcomes, the number of bits set by the first term is also known since these outcomes correspond to the cases where all query terms come from a single set. For the remaining (n-1) outcomes, we need further simplification depending on whether the first query term is from S_1 or from S_2. For each such case, let $P\{FT \in S_i\}$ be the probability that first term is from S_i ($1 \leq i \leq 2$). Then

$$P\{FT \in S_1\} = \frac{b_1}{nqt} \text{ and } P\{FT \in S_2\} = \frac{b_2}{nqt}$$

and hence $P\{FT \in S_1\} + P\{FT \in S_2\} = 1$.

At this point, it might be useful to look at a numeric example for clarification. Lets concentrate on the case where we have three query terms. Then nqt = 3 and we are at node 3 from which four branches originate. The probability associated with each branch can be computed using equation (12). Of the four outcomes, two are final. When $b_1 = 3$, all three terms are from set 1, each term setting m_1 bits and when $b_1 = 0$, all terms are from set 2, each setting m_2 bits.

For the case where $b_1 = 2$, we know that 2 out of 3 query terms are from S_1 but we still do not know the number of bits set by the first term. We compute $P\{FT \in S_1\}$ as 2/3 and $P\{FT \in S_2\}$as 1/3. For the case where $b_1 = 1$, these values are 1/3 and 2/3, respectively. Hence for the queries with three terms, we have six query outcomes that we need to treat separately.

In general, for t term queries, there are (t+1) branches and hence (t+1) outcomes. Two of these are final, the remaining (t-1) split into two. Hence we have (2 + 2(t-1)) , i.e., 2t final outcomes.

4.3 Derivation of an Expression for EXPH for Each Outcome

Recall that $P(W(Q) = k+c_1 \mid c_1)$ can be computed for each query outcome identified in Figure 2. Since

$$EXPH = \sum_{j=1}^{\min\{h,W(Q)\}} j * \frac{\binom{F-h}{W(Q)-j}\binom{h}{j}}{\binom{F}{W(Q)}}$$

substituting $k+c_1$ for W(Q) gives

$$EXPH = \sum_{j=1}^{\min\{h,k+c_1\}} j * \frac{\binom{F-h}{k+c_1-j}\binom{h}{j}}{\binom{F}{k+c_1}}$$

where EXPH is the expected number of 1s in the h-bit suffix of a particular query when k additional bits are set after stage 1. The range of values for k must be specified for each final outcome, since it depends on the number of terms and the number of bits set by each term. Then the EXPH value for an outcome with nqt number of terms can be written as

$$\sum_{k=0}^{k\,max} \left[\sum_{i=0}^{k} \binom{F-c_1}{i} \binom{F-c_1-i}{k-i} (-1)^{k+1} \prod_{r=2}^{nqt} \frac{\binom{c_1+i}{c_r}}{\binom{F}{c_r}} \right] \sum_{j=1}^{\min\{h,k+c_1\}} \frac{\binom{F-h}{k+c_1-j}\binom{h}{j}}{\binom{F}{k+c_1}} * j \tag{13}$$

where

$$kmax = \sum_{i=2}^{nqt} c_i \quad \text{assuming that} \sum_{i=1}^{nqt} c_i < F$$

Note that for the two outcomes corresponding to the single term queries, we only need to insert the value for c_1 for W(Q) and compute EXPH using equation (2).

Since the probability of occurrence of an outcome is the product of all probabilities of the branches from node 0 to the outcome, the overall EXPH (OEXPH) value can be computed by multiplying the probability of occurrence of each outcome with the EXPH value associated with it and summing them up.

This OEXPH value can then be substituted in equation (6) to compute probability of access savings, which in turn will be used to compute number of pages that need not to be accessed. Finally, PERSAV can be obtained as a function of the number of pages, n, the hashing level, h, and the overall expected number of bits set in the last h-bit suffix, OEXPH.

$$PERSAV = \frac{N(n,h,OEXPH)}{n} * 100 \tag{14}$$

5. APPLICATION OF PERFORMANCE EVALUATION FORMULAS TO THE TEST CASES

In order to evaluate the performance of the three test cases, we need only to substitute the m_i values computed by each method in the performance evaluation formulas. For the SM case, we just set $m_1 = m_2 = m$, where m is computed using equation (8). The m_i values are computed using equations (9) and (10) for MMS and MMM cases, respectively. For MMM, we need to compute $P_i(k)$ values such that $P_i(k)$ is the probability that exactly k terms will be specified from the ith set for $(1 \leq i \leq 2, 0 \leq k \leq 1)$.

In our experiments, we allow a maximum of 10 terms to be specified in a query and hence set t = 10. Recall from Figure 2 that (n+1) branches emanate from node n. Since we have 10 nodes, we end up with

$$\sum_{n=1}^{10} (n+1) = \frac{(10)\,(11)}{2} + 10 = 65 \text{ outcomes.}$$

Although some of these outcomes are not final, we will not go on any further since this much splitting is sufficient to compute the $P_i(k)$ values. Assuming we give a sequence number to each of these 65 outcomes, ranging from 1 to 65, ($b_1 = 0 \mid nqt = 1$) being the first and ($b_1 = t \mid nqt = t$) being the 65th, we now need to specify the sequence numbers (see Table IV) of the outcomes for each of the following four conditions: $b_1 = 0$, $b_1 = 1$, $b_2 = 0$, $b_2 = 1$.

Table IV. Sequence Numbers of Outcomes

$P_1(0)$	$b_1 = 0$	1	3	6	10	15	21	28	36	45	55
$P_1(1)$	$b_1 = 1$	2	4	7	11	16	22	29	37	46	56
$P_2(0)$	$b_2 = 0$	2	5	9	14	20	27	35	44	54	65
$P_2(1)$	$b_2 = 1$	1	4	8	13	19	26	34	43	53	64

Given the outcome sequence number, its occurrence probability can be computed by simply taking the product of the probabilities of the branches originating from node 0 and ending at the outcome. Summation of all occurrence probabilities of the outcomes in one row of Table IV will give us the associated $P_i(k)$ value.

Note that although Table IV, is designed for the case where t = 10, it can be easily extended using the pattern in which the sequence numbers appear in a row.

6. EXPERIMENTAL ANALYSIS

All experiments are based on the assumption that terms in the database are grouped into two sets, S_1 and S_2. This assumption not only enables us to emphasize the points of interest without going into unnecessary complexity but also represents many real life cases [FAL85]. We specify the maximum number of query terms that can appear in a query as 10, which we believe is appropriate to simulate many real life applications and is consistent with the choice of the values of the other input parameters.

Table V. Definition of the Query Cases

Query Case	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
Uniform Distribution (UD)	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
Low Weight (LW)	0.30	0.25	0.20	0.15	0.10	0.00	0.00	0.00	0.00	0.00
High Weight (HW)	0.00	0.00	0.00	0.00	0.00	0.10	0.15	0.20	0.25	0.30

As for the P_j values, ($j = 1, 2, . . . , 10$), three different query cases (QC) are considered: Uniform Distribution (UD), Low Weight (LW) queries and High Weight (HW) queries. The definition of each of the query cases are given in Table V.

Different values for the parameters F, q_i ($1 \leq i \leq 2$) will be used in the experiments. Our main purpose is to show how PERSAV is affected by the amount of change in the values of the input parameters for three different probability distributions (UD, LW, HW) for each of the three schemes (SM, MMS, MMM). Note that an experimental design that will achieve complete coverage of all possible combinations of the input parameters is impractical if not unnecessary. Hence we attempt to derive inferences about the change in the behavior of the system (represented as the change in the PERSAV) as a result of the alteration of the values of the input parameters. We then can use these observations to come up with generalized statements on the performance of the system under certain conditions.

Experiment 1.

Purpose

The purpose of the experiment is to compare the performance of SM, MMS and MMM cases in three different environments represented by the three different probability distributions (UD, HW and LW) specified above.

Parameters

The signature size (F) equals to 100, the values for D_1 and D_2 are 15 and 25, and those for q_1 and q_2 are 0.80 and 0.20, respectively. Note that the choice of the q_i values is also consistent with the 80-20 rule which approximates most of the real life cases [KNU75].

Results

Figure 3 summarizes the results of the experiment for the UD case. Here PERSAV provided by MMM is above the amounts provided by MMS and SM. Figures 4 and 5 show the results for LW and HW cases, respectively. These two cases correspond to two extreme situations where a deliberate non uniformity in the probability distribution for the number of query terms has been created. We expect MMM to provide excessive savings over MMS and SM in these cases since it makes use of more information about the system characteristics in determining the optimal assignment strategy. More specifically, MMM considers the type and relative frequency of queries with any number of terms that can be submitted to the system, whereas MMS computes the optimal m_i values as if only single term queries are submitted to the system.

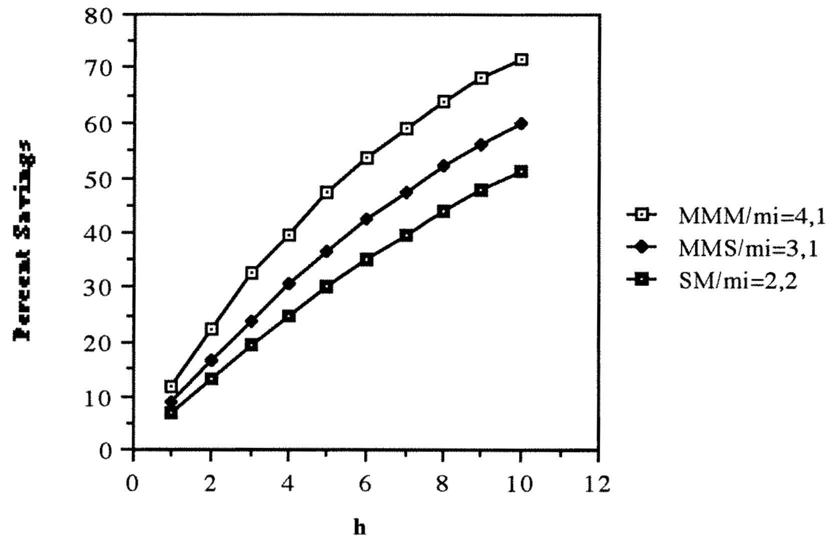

Figure 3. PERSAV in UD.

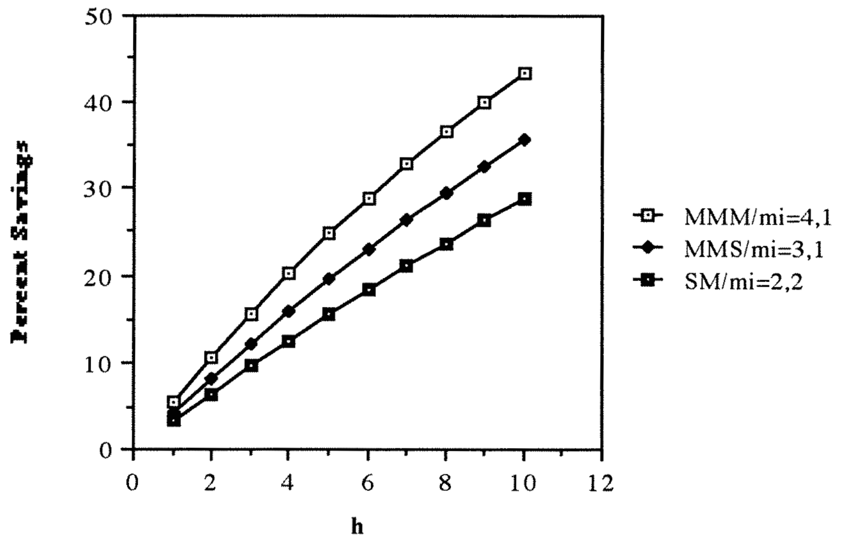

Figure 4. PERSAV in LW.

Experimental results turn out to be consistent with intuition: MMM outperforms MMS for both LW and HW cases. Note that MMM emphasizes the terms from S_1 even more by letting them set 4 bits whereas optimal number of bits set by terms from S_1 is 3 for MMS. The comparison of Figures 3, 4 and 5 shows that PERSAV provided by all three schemes are higher in the HW case compared to those in LW. This is because as the probability of having queries with high weight increases, the OEXPH value also increases, resulting in larger percent savings. For MMM, the gain is even more, since the number of bits set by the terms from set 1 is larger than the values for the two other cases. As the frequency of the queries with many such terms increase, the OEXPH value increases even more, further increasing the amount of PERSAV. Figure 6 provides a closer look at the results from a new standpoint.

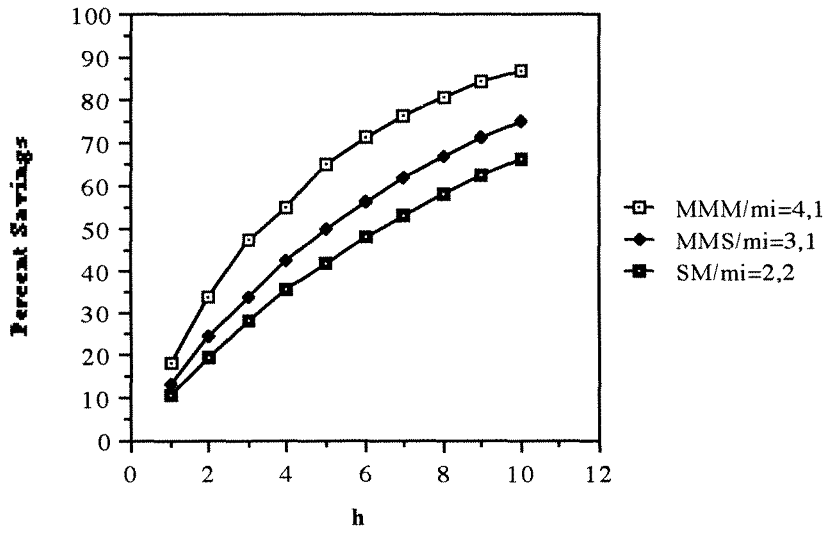

Figure 5. PERSAV in HW.

Let

$$\text{EXTSAV} = \frac{\text{PERSAV(MMM)} - \text{PERSAV(other)}}{\text{PERSAV(other)}} * 100 \qquad (15)$$

where

PERSAV(MMM) : percent savings provided by MMM;

PERSAV(other) : percent savings provided by any other scheme, SM or MMS, and;

EXTSAV is the extra percent savings provided by MMM over SM or MMS.

Naturally, EXTSAV over SM is the highest in all three cases: UD, LW, HW. However, the amount of EXTSAV decline for larger h values. For both of the HW cases, EXTSAV is the highest of all other cases at the beginning, but decline sharply with increasing values of h. This is due to the fact that OEXPH has a large value at HW regardless of the type of the scheme used. The increase in retrieval efficiency of all three methods become more effective as more bits of the signature suffix are taken into account. Hence the extra contribution introduced by MMM becomes less significant. For the LW cases, on the other hand, the savings are somewhat moderate but remain almost unaffected from the increase in h values, i.e., the database size. Although EXTSAV over MMS is smaller than that over SM, the decrease in the first one with larger h values is slower.

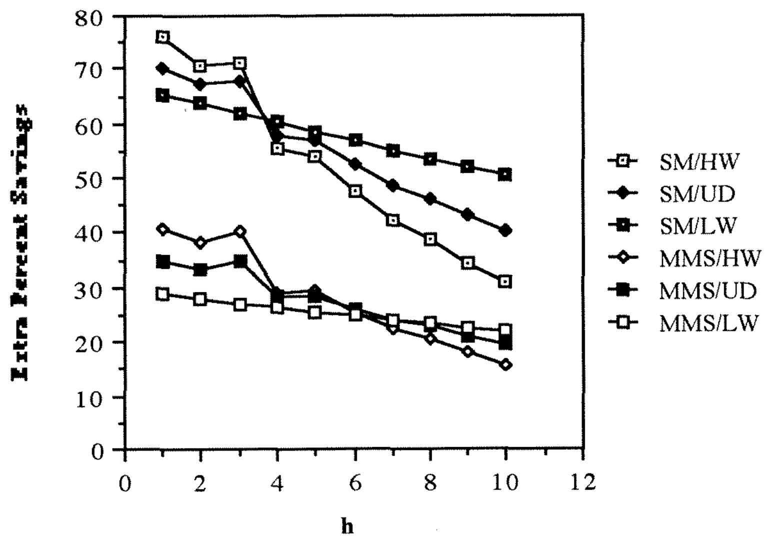

Figure 6. EXTSAV for UD, LW and HW.

The fact that the amount of savings is substantially high for our experiments does not guarantee that such savings can be expected all the time. For those cases where the m_i values are considerably small relative to the signature size, the expected query weight will be lower and hence the resulting savings will be less significant.

Experiment 2.

Purpose

The purpose of the experiment is to compare the optimal assignments (i.e., the suggested m_i values) for SM, MMS and MMM for larger signature sizes at three different settings, each corresponding to one probability distribution.

Parameters

The values for D_1 and D_2 are 15 and 25, respectively. Three values are used for the signature size: 400, 500, 600 and two values for the (q_1, q_2) pair are selected: (0.60, 0.40) and (0.80, 0.20). Since D_i ($1 \leq i \leq 2$) values are kept constant, the first pair of the q_i

values corresponds to the case where terms from both sets are close in discriminatory power whereas the second pair represents the case where terms are more distinctive.

Results

The (m_1, m_2) values at different settings for SM, MMS and MMM are summarized in Table VI.A, VI.B and VI.C , respectively. The values for the (q_1, q_2) pairs are given on the top of each table. Note that the optimal assignment in SM case depends only on the values of F and D and hence is insensitive to changes in the q_i values and the probability distribution of the query weight. MMS, on the other hand, considers q_i and D_i values but its assignment is independent of the nature of the probability distribution. Hence for a specified F and (q_1, q_2) pair, the m_i values remain constant for different probability distributions. Using all available information in the input parameters, MMM considers the values of D_i, q_i and F together with the nature of the probability distribution.

Table VI. (m_1, m_2) Values for SM, MMS, MMM

A. (m_1, m_2) Values for SM Case

query case	F = 400	F = 500	F = 600
UD	(3, 3)	(3, 3)	(4, 4)
LW	(3, 3)	(3, 3)	(4, 4)
HW	(3, 3)	(3, 3)	(4, 4)

B. (m_1, m_2) Values for MMS Case

$(q_1=0.60, q_2=0.40)$

query case	F = 400	F = 500	F = 600
UD	(3, 2)	(4, 3)	(5,4)
LW	(3, 2)	(4, 3)	(5,4)
HW	(3, 2)	(4, 3)	(5,4)

$(q_1=0.80, q_2=0.20)$

query case	F = 400	F = 500	F = 600
UD	(4, 2)	(5, 2)	(6, 3)
LW	(4, 2)	(5, 2)	(6, 3)
HW	(4, 2)	(5, 2)	(6, 3)

C. (m_1, m_2) Values for MMM Case

$(q_1=0.60, q_2=0.40)$

query case	F = 400	F = 500	F = 600
UD	(3, 2)	(4, 3)	(5, 4)
LW	(4, 2)	(4, 3)	(5, 4)
HW	(4, 2)	(4, 3)	(5, 4)

$(q_1=0.80, q_2=0.20)$

query case	F = 400	F = 500	F = 600
UD	(5, 2)	(5, 2)	(6, 3)
LW	(5, 1)	(6, 2)	(6, 3)
HW	(5, 1)	(6, 2)	(7, 2)

The insensitivity of the SM scheme to the changes in the values of the various input parameters leads to its inferior performance. The comparison of the two other methods shows that as the gap between the discriminatory power of the terms increases, the MMM assignment emphasizes the terms with higher power by letting them set more bits. Hence the gap between the m_i values is the largest for the MMM case. This observation suggests that using MMM will be especially beneficial if the terms are highly distinctive based on their discriminatory power.

Increasing the signature size has two effects working in the opposite directions: On one side, higher values of F attempt to raise the m_i values (see equations 8, 9, 10). However, since these equations compute real m_i values which need to be converted to integer numbers for practical applications, an increase in F increments the integer m_i only if it is substantially large. Assuming that such a change occurs, larger m_i values cause the query weight to increase resulting in higher OEXPH values, which in turn cause higher PERSAV. At the same time, however, as the signature size is increased, there are more bits to choose from when a term signature is generated and consequently the OEXPH value tends to drop. The resulting effect of increasing the value of F depends on the relative power of these two opposite changes.

In most cases when the increase in F is large enough to cause an increase in the integer m_i value(s), the reverse effect is suppressed and PERSAV values are raised. The amount of increase in PERSAV is substantial for the HW case in particular, since in HW queries with many terms are more frequent and hence larger m_i values have a more significant effect on the result. For the LW case, the increase in savings is moderate, if any. In some cases, even though the integer m_i value increases, savings decline. This can be attributed to the fact that higher weight queries are not frequent enough to supplement the positive effect caused by the increase in m_i and hence the reverse effect overrules.

7. APPLICABILITY OF THE FINDINGS TO OTHER PARTITIONED SIGNATURE FILE ORGANIZATIONS

We have already mentioned that the difficulty associated with the analysis of the multiterm query case compared to the single term situation arises from the fact that the query weight is no longer a constant but a random variable. We have also shown, using LHSS, that once the distribution of the query weight is determined for each possible final query outcome, all remains to the inclusion of this knowledge in the performance evaluation formulas of the organization of concern. This requires the consideration of all possible values of the query weight together with the associated probabilities as we have shown in equation 13.

Our derivation on the distribution of the query weight and the idea depicted by the tree diagram in Figure 2 are both independent of the signature file organization and hence can directly be used to evaluate the performance of other schemes. Since the query weight is a variable in the performance evaluation formulas, accounting for all possible values of the query weight and the associated probabilities for each outcome is sufficient to compute the value of the performance measure (like EXPH in LHSS) for each query outcome. Then the tree diagram is traced backwards and the probabilities of the outcomes are taken into account to compute an overall value for the performance measure (like OEXPH in LHSS.) Below we first demonstrate how this idea can be applied to Fixed Prefix Partitioning (FPP) and then discuss the modification of an approximate performance evaluation formula that has been shown to hold for both FPP and LHSS. (Table VII. provides the list of the symbols for the following sections.)

Table VII. Definition of Important Symbols for Sections 7-8

d	: weight of the partition key (i.e., the number of bits set to 1 in the partition key)
h	: key length
n	: no. of partitions
F	: size of a signature in bits
$OPAR_{FPP}$	: overall partition activation ratio for FPP (based on the distribution of the query weight)
PAR_{FPP}	: the partition activation ratio (the ratio of the number of partitions activated to the total number of partitions) for FPP
PAR_{LHSS}	: the partition activation ratio (the ratio of the number of partitions activated to the total number of partitions) for LHSS
PAR	: the estimated partition activation ratio for LHSS and FPP
PAR_{mod}	: the modified estimated partition activation ratio for LHSS and FPP (based on the distribution of the query weight)
W(Q)	: query weight, i.e., the no. of 1s in query signature

7.1 Application of Results to FPP

Fixed Prefix Partitioning is one of the three methods suggested by Lee and Leng for signature mapping [LEE89]. It uses superimposed coding technique to generate the signatures and also assumes that all signatures consist of a key portion as well as a nonkey part. It is this key part that determines which partition the signature will be stored in. Key portions of all signatures in a partition are the same and constitute the partition key. Similarly, the query signatures have these two parts. The key of a query signature is extracted in the same way as the keys for the partitions and only those partitions whose keys include the query key are accessed. Hence if the key portions of the query signature and the i th partition, P_i, are shown as K_Q and K_{Pi}, respectively, then the partition P_i is accessed if $(K_Q \cap K_{Pi}) = K_Q$.

Fixed Prefix Partitioning (FPP), takes the first k bits of the signature as the key whereas the other two methods provide different ways to extract keys from the signatures. Its performance is evaluated based on two criteria; the resulting reduction in the search space and the uniformity of the workload of the processors, assuming a parallel architecture. Signature reduction ratio, which is the ratio of the number of signatures searched to the total number of signatures and the partition activation ratio, which is the ratio of the number of partitions searched to the total number of partitions, are two possible measures of the first criterion. Partition activation probability, P_a, is defined as the probability that a partition will be searched for a query and the equality of the activation probabilities is accepted as an indicator of the uniformity of the workload, when a processor is assigned to a partition and the partitions have the same size. Below, we will concentrate on the partition activation ratio and show how our derivations can be used to determine its value for the SM, MMS and MMM cases.

The partition activation ratio for FPP (PAR_{FPP}) is shown to be computed as a function of the signature size, F, the key length, h, the weight of the partition key, d ($0 \leq d \leq h$), and the query weight, W(Q) [LEE89].

$$PAR_{FPP} = \sum_{d=0}^{h} \binom{h}{d} \frac{\binom{F-h+d}{W(Q)}}{\binom{F}{W(Q)}} * 2^{-h} \qquad (16)$$

When the number of partitions, n, is equal to 2^h, this equation can be justified intuitively: The probability that a partition whose key has weight d will be activated can be computed as

$$\frac{\binom{F-h+d}{W(Q)}}{\binom{F}{W(Q)}}$$

and since there are $\binom{h}{d}$ partitions whose keys have weight d, the product of these two terms give the expected number of partitions with weight d that are activated for a query with weight W(Q). Summing this up over all possible key weights, d, we find the total number of partitions activated. Dividing the total number of activated partitions to the total number of partitions (2^h), we obtain the partition activation ratio.

Equation 16 can easily be modified to evaluate the performance of SM, MMS and MMM methods in a multiterm environment as they are applied to FPP, by taking the distribution of the query weight into account. For each final query outcome depicted in Figure 2, PAR_{FPP} can be computed as

$$PAR_{FPP} = \sum_{k=0}^{kmax} \left[\sum_{i=0}^{k} \binom{F-c_1}{i} \binom{F-c_1-i}{k-i} (-1)^{k+1} \prod_{r=2}^{nqt.} \frac{\binom{c_1+i}{c_r}}{\binom{F}{c_r}} \right] \sum_{d=0}^{h} \binom{h}{d} \frac{\binom{F-h+d}{k+c_1}}{\binom{F}{k+c_1}} * 2^{-h}$$

(17)

and then the overall value for the partition activation ratio ($OPAR_{FPP}$) can be found by tracing the tree back and multiplying the individual PAR_{FPP} values with the associated probabilities as described in Section 4.3.

7.2 Approximation of PAR for LHSS and FPP

The above analysis shows how the performance of the partitioned signature file organizations can be evaluated. However, the required computations are tedious and the derived formulas look very complicated. In search for an easy way to determine the partition activation ratio, Ciaccia and Zezula have started with the exact formulas for FPP (equation 16) and LHSS where

$$PAR_{LHSS} = \frac{npa}{n} = \frac{n/2^{EXPH[W(Q),h]}}{n} = 2^{-EXPH[W(Q),h]} \qquad (18)$$

using equation 5 in Section 2.2. By using basic probability theory, they have proven that these two equations actually coincide although they look very different [CIA93]. The authors have further worked on this formulation and generated a closed formula that approximates the partition activation ratio, PAR for both methods where

$$PAR = \left(1 - \frac{W(Q)}{2F}\right)^{h} \qquad (19)$$

Though this approximation results in slightly higher estimates of the actual PAR value, it is very simple to compute and use. Besides, the expected error, which increases as the h/F ratio increases never exceeds 1.2%, which is acceptable for performance prediction computations. This formula can also be adjusted to be applicable in our case by the inclusion of the distribution of the query weight. Then the modified formulation, PAR_{mod}, can be written as follows

$$PAR_{mod} = \sum_{k=0}^{kmax} \left[\sum_{i=0}^{k} \binom{F-c_1}{i} \binom{F-c_1-i}{k-i} (-1)^{k+1} \prod_{r=2}^{nqt} \frac{\binom{c_1+i}{c_r}}{\binom{F}{c_r}} \right] \left[1 - \frac{k+c_1}{2F} \right]^{h} \qquad (20)$$

Note that the applicability of the same formula to both methods does not mean that they behave in the same way. In fact FPP and LHSS differ in the way they control the partition and the key size and their strategies to generate the partitions. For instance, the hashing level in LHSS can be adjusted dynamically whereas the key length is fixed in FPP which might lead to some problems: Starting with a long key length can result in poor utilization of space whereas the choice of a shorter key length can lead to many overflows. Hence LHSS provides a more flexible structure that allows expansion and shrinkage when necessary, whereas FPP can only accomplish a relatively static structure.

8. EFFICIENCY RELEVANCY CONSIDERATIONS

When SM is used together with LHSS, FPP or any other partitioning scheme, all terms are treated equally regardless of their occurrence and query frequencies and set the same number of bits (m). When a single term query is specified in a query, the query weight is constant and equals m. Hence the expected number of bits in the last h-bit suffix (or the first h-bit prefix) of the query signature is the same regardless of the term discriminatory power values. Therefore, the number of page accesses is also the same for all terms. When a term with a low discriminatory power is specified in a query, a long list of documents will be returned. (Notice that terms with low discriminatory power are the ones that appear in many documents.) Yet a large portion of the returned documents will not be of interest to the user. Hence the resulting relevancy will be very low. In contrast, when a term with high discriminatory power is used in the query, only a few documents, most of which will be relevant, are returned to the user, and the relevancy level will be significantly high [AKT93a].

A similar condition prevails for the multiterm case since when SM is used, the distribution for the query weight for a query with a certain number of terms is the same regardless of the discriminatory power of the query terms. Hence two different queries where the first one consists of more selective terms and the other is made up of less selective ones can achieve the same efficiency whereas the resulting relevancy levels will be different. This situation which is typical in the SM case indicates an obvious imbalance between efficiency and relevancy. For the same number of page accesses (i.e. for the same level of efficiency), it is possible to end up with low or high values of relevancy depending on the frequency characteristics of the query term(s). The more significant the difference between the discriminatory power of the terms, the more severe is the imbalance described above.

When MMS or MMM is used, the terms with high discriminatory power set more bits than those with low discriminatory power. Hence, the number of page accesses required for these two cases will differ in the first place. Consequently, the terms with high discriminatory power provide relatively more page savings which will be consistent with the high level of the resulting relevancy. On the other hand, terms with low discriminatory power will somehow be penalized because now they will be setting fewer bits. The resulting page savings will be low together with the undesirably low relevancy level. The way to achieve high efficiency coupled with high relevancy is to increase the query weight. This can be accomplished by using terms with high discriminatory power in the queries or by constructing term phrases from non-discriminatory terms. In an IRS, the former can be supported by an on-line thesaurus providing group of related specific terms under more general, higher level class indicators; the latter can be implemented by automatic phrase construction [CAN87, SAL89].

9. APPLICATION OF FINDINGS TO REAL LIFE CASES

Prior to selecting one of the three methods (SM, MMS, MMM) for a practical application, the database (formatted, unformatted or combined) characteristics of the system of concern should be explored (to create the non trivial term list [CAN87, OZK86, SAL89] and to determine the occurrence frequencies) together with the query characteristics (like query contents and frequencies). The findings about the occurrence and query frequencies based on sufficiently large sample sizes can then be used to determine the number of sets to be used. In most cases, two sets might be good enough unless terms can apparently be classified in more than two groups based on their occurrence and query frequencies [FAL85].

Next, each record is examined to determine the D_i values and the sample queries are reconsidered to come up with the q_i values together with a tentative frequency distribution for the number of terms in a query. This frequency distribution can enable simulating the system behavior within an acceptable margin of error provided that the query sample is of appropriate size and is selected in a proper manner. Then a value for the signature size, F, is selected based on the values of these input parameters and the amount of tolerable false drop [FAL87a].

Next, one of the three methods (SM, MMS, MMM), is selected considering the database and query characteristics of the system which should have been revealed up to this step. If, for instance, there exists a huge difference among the term discriminatory power values, using MMM will provide extra savings over the other two methods, especially when the frequency distribution is not uniform. Further computations for the sake of comparison can be carried out to find out the m_i values provided by each method. If still undecided as to which method to choose, the expected savings for each method can be determined using the sample queries at hand and our derived formulas. However, updating will be necessary if either the contents of the database or the nature of the submitted queries change significantly. Hence the intensity and frequency of substantial change in system properties are also characteristics of the system and must be considered in selecting a signature generation scheme.

10. CONCLUSION AND FUTURE RESEARCH POINTERS

The main contribution of our study is the combination of the concerns of signature extraction and signature file organization which have usually been treated as separate issues. We also relax the uniform frequency and single term query assumptions and provide a comprehensive analysis for multiterm query environments where terms can be classified based on their query and database occurrence frequencies. We present the performance evaluation analysis of three specific signature generation schemes (SM, MMS, MMM) as they are applied to LHSS in a single and multiterm query environment. First scheme (SM) allows all terms set the same number of bits regardless of their discriminatory power whereas the second and third methods (MMS and MMM) emphasize the terms with high query frequency and low occurrence frequency. Of these three schemes, only MMM takes the probability distribution of the number of query terms into account in finding the optimal mapping strategy. We point out that the complexity in evaluating the performance of the cited schemes evolves from the challenging task of finding the distribution of the query weight for each of the query outcomes identified.

We have shown that both MMS and MMM are clearly superior to SM in various query environments. Our results also indicate that the extra savings provided by MMM over MMS increase as the gap among the discriminatory power values of the terms get larger and the probability distribution of the number of terms in the query depicts a non uniform pattern. This is because MMM considers the nature of the probability distribution of the number of query terms in determining the optimal assignment strategy and emphasizes the terms with high discriminatory power in particular.

Our study shows how different system input parameters interact in the overall working mechanism of the signature generation schemes and the LHSS organization. Once this point is clear, one can make a knowledgeable selection on the values of the input parameters and predict the expected savings. We also show how our derivations can be applied to the FPP method and provide the modified version of an approximate formula that is used to estimate the performance of both LHSS and FPP methods. A discussion on the relevancy efficiency balance (which is improved by using MMM and MMS instead of SM) is also provided. With some modification, our derivations can be used to evaluate the performance of additional partitioning methods. Ways to derive approximate performance evaluation formulas (similar to PAR for LHSS and FPP) can also be explored to simplify the computations.

Future research can also deal with comparing the actual performance of a real life system organized using a particular signature scheme against the estimated performance level which is computed by using the derived equations. Simulation experiments (which can be of similar nature to those presented for LHSS) can be designed for other organizations to observe whether the experimental results obtained for LHSS about the performance of SM, MMS and MMM are applicable to other organizations. Finally, our integrated approach that combines the concepts of signature generation and signature file organization can further be pursued to analyze the applicability of various signature generation schemes to different organizations.

ACKNOWLEDGMENTS

We thank Pavel Zezula of Technical University of Brno for providing us with timely results of his work before they are published in the media.

REFERENCES

[AKT93a] AKTUG, D., AND CAN, F. Signature file hashing using term occurrence and query frequencies. In *Proceedings of the 12th Annual IEEE International Phoenix Conference on Computers and Communications*. (Phoenix, March 1993), pp. 148-153.

[AKT93b] AKTUG, D., AND CAN, F. Analysis of multiterm queries in a dynamic signature file organization. To appear in the *Proceedings of the 16th Annual International ACM-SIGIR Conference* (June 1993).

[CAN87] CAN, F., AND OZKARAHAN, E. A. Computation of term/document discrimination values by use of the cover coefficient concept. *Journal of the American Society for Information Science*. 38, 3 (May 1987), 171-183.

[CAN90] CAN, F., AND OZKARAHAN, E. A. Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. *ACM Transactions on Database Systems*. 15, 4 (Dec. 1990), 483-517.

[CHA89] CHANG, J. W., LEE, J. H. , AND LEE, Y. J. Multikey access methods based on term discrimination and signature clustering. In *Proceedings of the 12th Annual International ACM-SIGIR Conference* (September 1989), ACM, New York, pp. 176-185.

[CHA92] CHANG, J. W., YOO, J. S., AND LEE, Y. J. Performance comparison of signature-based multikey access methods. *Microprocessing and Microprogramming*. 35, (1992), 345-352.

[CHR84] CHRISTODOULAKIS, S., AND FALOUTSOS, C. Signature files: An access method for documents and its analytical performance evaluation. *ACM Transactions on Office Information Systems*. 2, 4 (October 1984), 267-288.

[CIA93] CIACCIA, P., ZEZULA, P. Estimating accesses in partitioned signature file organizations. *ACM Transactions on Information Systems*. 11, 2 (April 1993), 133-142.

[DEP86] DEPPISH, U. S-tree: A Dynamic balanced signature index for office retrieval. In *Proceedings of the 9th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval* (September 1986), ACM, N.Y., 1986, pp. 77-87.

[FAL85] FALOUTSOS, C., AND CHRISTODOULAKIS, S. Design of a signature file method that accounts for non-uniform occurrence and query frequencies. In *Proceedings of the 11th International Conference on VLDB* (August 1985). VLDB Endowment, 1985, pp. 165-170.

[FAL87a] FALOUTSOS, C. Signature files: An integrated access method for text and attributes, suitable for optical disk storage. In *University of Maryland Computer Science Technical Report Series,* June 1987.

[FAL87b] FALOUTSOS, C., AND CHRISTODOULAKIS, S. Optimal signature extraction and information loss. *ACM Transactions on Database Systems.* 12, 3 (September 1987), 395-428.

[FAL87c] FALOUTSOS, C., AND CHRISTODOULAKIS, S. Description and performance analysis of signature file methods for office filing. *ACM Transactions on Office Information Systems.* 5, 3. (July 1987), 237-257.

[FAL88] FALOUTSOS, C., Signature Files: An integrated access method for text and attributes, suitable for optical disk storage. *BIT* 28, 4, 1988, 736-754.

[FAL92] FALOUTSOS, C. Signature files In *Information Retrieval Data Structures and Algorithms,* edited by W. B. Frakes, R. Baeza-Yates, Prentice Hall, Englewood Cliffs, N.J., 1992, pp 44-65.

[FEL68] FELLER, W. *An Introduction to Probability Theory and its Applications, 3rd ed.* John Wiley & Sons, New York, 1968.

[GRA92] GRANDI, F., TIBERIO, P., AND ZEZULA, P. Frame-sliced partitioned parallel signature files. In the Proceedings of *15th Annual International ACM-SIGIR Conference* (June 1992), ACM New York, pp. 286-297.

[KEN90] KENT, A., SACKS-DAVIS R., AND RAMAMOHANARAO, K. A Signature file scheme based on multiple organizations for indexing very large text databases. *Journal of American Society for Information Science.* 41, 7 (Oct. 1990), 508-534.

[KNU75] KNUTH, D., E. *The Art of Computer Programming.* vol. 3: *Sorting and Searching.* Addison-Wesley, Reading, Mass., 1975.

[LEE89] LEE, D. L., AND LENG, C.-W. Partitioned signature files: Design issues and performance evaluation. *ACM Transactions on Information Systems.* 7, 2 (Apr. 1989), 158-180.

[LEN92] LENG, C.-W R., LEE, D. L. Optimal weight assignment for signature generation. *ACM Transactions on Database Systems.* 17, 2 (June 1992), 346-373.

[LIN92] LIN, Z., AND FALOUTSOS, C. Frame-sliced signature files. *IEEE Transactions on Knowledge and Data Engineering.* 4, 3 (June 1992), 281-289.

[LIT80] LITWIN, W. Linear hashing: A new tool for files and tables addressing. In *Proceedings of the 6th International Conference on VLDB* (Montreal, Oct. 1980), pp. 212-223.

[MUR92] MURPHREE, E., AND AKTUG, D. Derivation of probability distribution of the weight of the query signature. (Preprint. 1st author's address: Department of Mathematics and Statistics, Miami University, Oxford, OH 45056, USA.)

[OZK86] OZKARAHAN, E. A., AND CAN, F. An automatic and tunable document indexing system. In *Proceedings of the 9th Annual International ACM-SIGIR Conference* (September 1986), ACM, New York, pp. 234-243.

[PFA80] PFALTZ, J. L., BERMAN, W. J. , AND CAGLEY, E. M. Partial-match retrieval using indexed descriptor files. *Communications of the ACM.* 23, 9 (September 1980), 522-528.

[SAC85] SACKS-DAVIS, R. Performance of a multi-key access method based on descriptors and superimposed coding techniques. *Information Systems.* 10, 4, 391-403.

[SAC87] SACKS-DAVIS, AND R., RAMAMOHANARAO, K. Multikey access methods based on superimposed coding techniques. *ACM Transactions on Database Systems.* 12, 4 (December 1987), 655-696.

[SAC90] SACKS-DAVIS, WALLIS P., AND WILKINSON R. Using syntactic analysis in a document retrieval system that uses signature files. In *Proceedings of the 13th Annual International ACM-SIGIR Conference* (September 1990), ACM, New York, pp. 179-191.

[SAL89] SALTON, G. *Automatic Text Processing: The Transformation Analysis, and Retrieval of Information by Computer.* Addison Wesley, Reading, Mass., 1989.

[THA88] THARP, A. L. *File Organization and Processing.* John Wiley and Sons, New York, N.Y., 1988.

[TIB91] TIBERIO, P., AND ZEZULA, P. Selecting signature files for specific applications. In *Proceedings of Advanced Computer Technology, Reliable Systems and Applications, 5th Annual European Conference.* (Bologna, Italy, May 1991) IEEE, 1991, pp. 718-725.

[ZEZ88] ZEZULA, P. Linear hashing for signature files. In *Proceedings of the IFIP TC6 and TC8 Open Symposium on Network Information Processing Systems.* (Sofia, Bulgaria, May 1988), pp. 243-250.

[ZEZ91] ZEZULA, P., RABITTI, AND F., TIBERIO, P. Dynamic partitioning of signature files. *ACM Transactions on Information Systems.* 9, 4 (Oct. 1991), 336-367.