Analysis of Signature Generation Schemes for Multiterm Queries In Linear Hashing with Superimposed Signatures

Fazli Can[*] Osman Ertugay[†]

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

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

MIAMI UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
& SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-1995-008

Analysis of Signature Generation Schemes for Multiterm Queries In Linear Hashing with Superimposed Signatures

Fazli Can and Osman N. Ertugay

School of Engineering & Applied Science | Oxford, Ohio 45056 | 513-529-5928

Analysis of Signature Generation Schemes For Multiterm Queries In Linear Hashing with Superimposed Signatures

by

Fazli Can
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Osman N. Ertugay
Dept. of Computer & Info. Sci.
University of Pennsylvania
Philadelphia, PA 19104

Working Paper #95-008

Dec. 1995

ANALYSIS OF SIGNATURE GENERATION SCHEMES FOR MULTITERM QUERIES IN LINEAR HASHING WITH SUPERIMPOSED SIGNATURES

Fazli CAN*
Dept. of Systems Analysis
Miami University
Oxford, OH 45056

Osman N. ERTUGAY
Dept. of Computer and Infor. Science
University of Pennsylvania
Philadelphia, PA 19104

Abstract - Signature files provide efficient retrieval of data by reflecting the essence of the data objects into bit patterns. 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). The first scheme (SM) allows all terms set the same number of bits whereas the second and third schemes (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 a detailed analysis of LHSS in multiterm query environments by incorporating the term discrimination values based on document and query frequencies. The approach of the study can also be extended to other signature file access methods based on partitioning. The derivation of the performance evaluation formulas, the simulation results based on these formulas for various experimental settings, and the implementation results based on INSPEC and NPL text databases are provided. Results indicate that MMM and MMS outperform SM in all cases in terms of access savings, especially when terms become more distinctive. MMM slightly outperforms MMS in high weight and low weight query cases. The performance gap among all three schemes decreases as the database size increases, and as the signature size increases the performances of MMM and MMS decrease and converge to that of the SM scheme when the hashing level is fixed.

CR Categories and Subject Descriptors: E.5 [**Data**]: Files; H.2.2 [**Database Management**]: Physical Design-*access methods*; H.3.2 [**Information Storage and Retrieval**]: Information Storage-*file organization*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.4.1 [**Information Systems Applications**]: Office Automation

General terms: Design, Performance

Additional Key Words and Phrases: Access methods, document retrieval, dynamic file, hashing, information retrieval, partial-match retrieval, signature files, signature generation, superimposed coding, term discrimination value, text retrieval

* The majority of this work has been done when the first author was on sabbatical leave from Department of Systems Analysis, Miami University, Oxford, OH 45056.
voice: (513) 529 5950, fax: (513) 529 3841, e-mail: fc74sanf@miamiu.acs.muohio.edu.

1. INTRODUCTION

Information retrieval systems (IRSs) are used to 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, FAL92, SAL89, TIB93, ZEZ91]. The concern of our study is signature files which are widely applicable 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 stores the essence of data objects in terms of superimposed bit patterns, called signatures, and acts as a filter during query processing. (Other signature generation schemes, such as word signatures and compressed bit strings, are also available in the literature [FAL84, TIB93].) 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 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.

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 generation and example queries.

Figure 1 provides an example for signature generation using Superimposed Coding where each term is hashed to a bit string where it sets a pre defined 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 false match conditions are also provided.

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 [CHR84, TIB93, 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 desirable 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 [CIA93, 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 [ZEZ89, ZEZ91]. The first scheme treats all terms in an identical way, neglecting the differences in their (object) 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 analysis for queries with any number of terms. In contrast with the study by Aktug and Can [AKT93b, AKT93c] the query weight calculation is done using a new probability function developed by Grandi [GRA94] for estimating the query signature weight in multiterm query environments. The previous study was using a complicated Markov Chain approach [MUR92]. Our method in this paper simplifies the model and calculations. Furthermore, the simulation experiments based on analytical derivations are extensive and show the system behavior for very large databases (e.g., more than 1 million records) and various signature sizes. Experiments based on the documents and queries of two common databases of the information retrieval literature, INSPEC and NPL, are used to validate the simulation results.

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. Section 5 provides the description of the experimental environment. Section 6 presents the results of the experimental analysis based on simulation and the INSPEC and NPL databases. Section 7 concludes the paper.

2. LINEAR HASHING WITH SUPERIMPOSED SIGNATURES (LHSS)

Linear hashing is an efficient way to organize partitioned dynamic files [LIT80]. A partitioned signature file scheme, which was originally introduced by Zezula, is linear hashing with superimposed signatures [ZEZ89]. In LHSS and other partitioned signature file organizations [LEE89] a portion of a record signature is used as the signature key and the records with the same key are put into the same file fragment.

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 [ZEZ89, ZEZ91].

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

where β_i is the value of the i^{th} 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 signature of object 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. Whenever the load of the file exceeds a predetermined load factor, a "split" is initiated, i.e., a new primary page is created [LIT80, THA88]. 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

β_i	: value of the i^{th} binary digit of the term signature
h	: hashing level
n	: no. of addressable pages
s_i	: signature of object i
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 key are also set in the page signature key. 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 qualifies 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. Actually, in LHSS the query resolution is performed first at signature levels by matching query and object signatures of the selected pages and then by matching the query terms and the contents of the selected objects whose signatures match the query signature.

2.2 Performance Evaluation

It has been shown that the number of pages that do not need to be accessed 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. Then

$$\text{EXPH}(W(Q),h) = \sum_{j=1}^{\min\{h, W(Q)\}} j \,.\, P(j) \qquad (2)$$

where P(j) is the probability that j bits are set in the h-bit suffix of the query. Formally, we have a random variable J which has hypergeometric probability distribution as follows.

$$P(j)= P(J= j)= \text{hyp}(j{:}F,\, W(Q),\, h)= \frac{\binom{F-h}{W(Q)-j}\binom{h}{j}}{\binom{F}{W(Q)}} \qquad (3)$$

Thus, EXPH(W(Q), h) is the expected value of the hypergeometric random variable J and written as follows.

$$\text{EXPH}(W(Q), h)= E(J)= \frac{W(Q)\,.\,h}{F} \qquad (4)$$

Next probability of access savings, P(W(Q), is 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

$$P(W(Q), h)= 1 - \frac{npa}{n} \qquad (5)$$

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

$$npa= \frac{n}{2^{\text{EXPH}(W(Q),\, h)}} \qquad (6)$$

and

$$P(W(Q), h)= 1 - \frac{1}{2^{\text{EXPH}(W(Q),\, h)}} \qquad (7)$$

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

$$R(h)= 2(n - 2^{h-1}) = 2n - 2^h \qquad (8)$$

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

$$R(h - 1) = n - R(h) = 2^h - n \qquad (9)$$

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.

$$N(n, h, W(Q)) = R(h)\ P(W(Q), h) + R(h - 1)\ P(W(Q), h - 1) \qquad (10)$$

As the measurement of performance, we will use the percentage of pages that do not need to be accessed for a given query and call this percentage savings, PERSAV, which is expressed as follows.

$$\mathrm{PERSAV} = \frac{N(n, h, W(Q))}{n} . 100 \qquad (11)$$

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

b_i	: no. of terms from S_i in a query
c_i	: no. of bit set by the i 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
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_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
r	: node number in Figure 2 ($1 \leq r \leq t$)
t	: maximum no. of terms in a query
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
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

3. REFLECTING TERM DISCRIMINATION VALUES TO RECORD SIGNATURES

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 and their document frequencies [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. (For other approaches on determining term discrimination values refer to [CAN87, OZK86, SAL89].) 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.

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

Method	Formula (equation no.)	References
Single m (SM)	$m = \frac{F \ln 2}{D}$ (12)	[FAL85]
Multiple m Based on Single Term Queries (MMS)	$m_i = \frac{F \ln 2}{D} + \frac{1}{\ln 2}\left[\ln\frac{q_i}{D_i} - \frac{\sum_{i=1}^{n_s} D_i \ln\frac{q_i}{D_i}}{D}\right]$ (13)	[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 $L_i = \ln\left[\frac{P_i(0)}{P_i(1)} D_i\right]$ (14)	[FAL87a, FAL88]

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 shows the formulas for the optimal assignment strategies for three signature generation schemes which are based on this approach. The 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 arising 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 the 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 IN MULTITERM QUERY ENVIRONMENTS

4.1 Query Weight Calculation

In order to compute the percentage of the addressable pages that do not have to be accessed, PERSAV, for a particular experimental setting, expected number of ones in the query signature, W(Q), should be known. For a single term query, 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, W, and therefore has a probability distribution.

The distribution of the query weight has been derived by Grandi [GRA94] by means of a counting method based on the principle of inclusion and exclusion [KNU73]. For a given signature size, F, we should know the number of query terms, nqt, and the number of bits set to one by each query term, c_i, in order to find the probability that W is equal to a specific value w. The probability distribution function for the query weight is defined as follows [GRA94].

$$P(W = w) = \binom{F}{w} \sum_{j=0}^{w} (-1)^j \binom{w}{j} \prod_{i=1}^{nqt} \frac{\binom{w-j}{c_i}}{\binom{F}{c_i}} \qquad (15)$$

4.2 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_i (in equation 15) 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 $(P_1 + P_2 +. .. + P_t) = 1$ is satisfied.

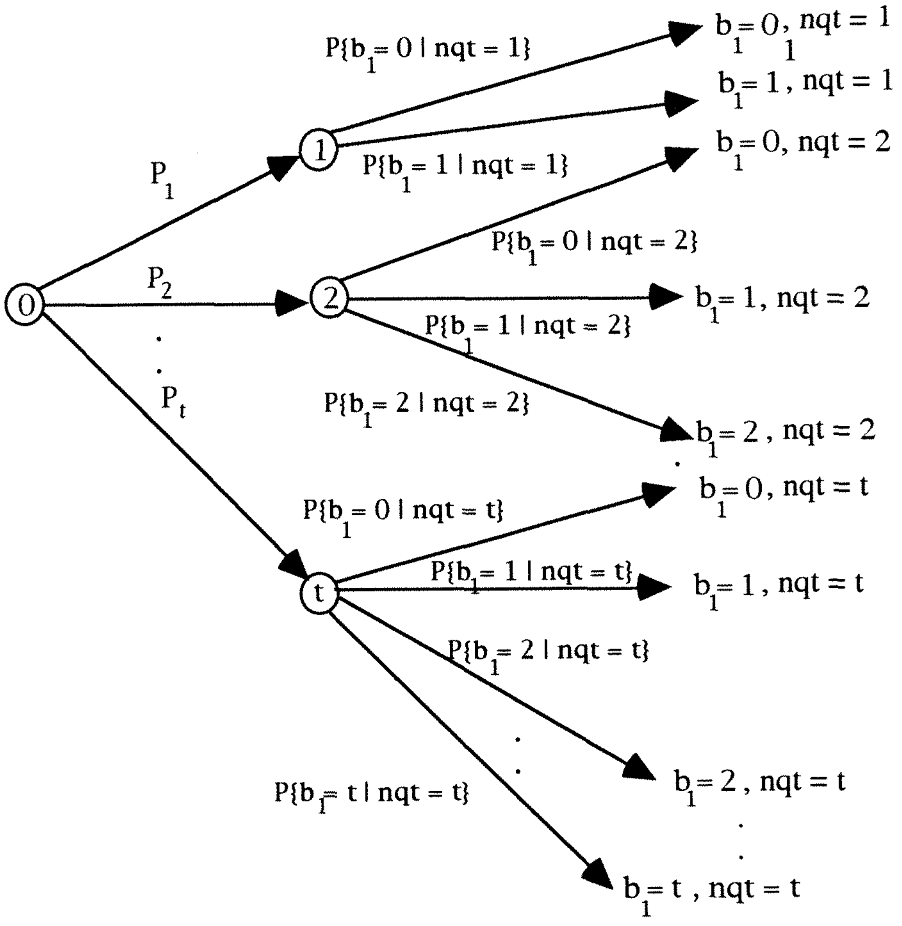


Figure 2. Tree diagram for query outcomes.

The tree diagram in Figure 2 enables us identify all different query combinations each of which is represented as a final outcome. At each outcome we know the number of query terms and how many terms are from S_1 and how many from S_2. This information enables us to compute the value of P(W = w) for each query outcome for those values of w that are realizable. Next, we can compute the value of the expected number of ones in the query signature.

So far we have clarified our reason to identify the 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 this 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 probabilities associated with the branches emanating from a particular node adds up to one. 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

$$\sum_{i=1}^{2} b_i = \text{nqt} \tag{16}$$

should be satisfied. Therefore it is sufficient to use just b_1 (or b_2) to specify a query combination, once nqt is known. In general, from any node r ($1 \leq r \leq t$), where r = nqt, (nqt + 1) branches emanate, each corresponding to one possible value for b_1 in the range 0 to nqt. The probability associated with any of these second level branches is given as

$$P\{b_1 = v \mid \text{nqt} = V\} = \binom{V}{v} q_1^{v} q_2^{(V - v)} \tag{17}$$

where $0 \leq v \leq V$ and $1 \leq V \leq t$.

The occurrence probability of a final outcome is given by the product of the probabilities associated with the first and second level branches corresponding to that outcome. The expected query weight for an outcome (nqt and b_1 are known) is given by

$$W(Q) = \sum_{w = w_{min}}^{w_{max}} P(W = w) \, . \, w \tag{18}$$

where

$$w_{max} = \sum_{i=1}^{nqt} c_i,\ w_{min} = \max\{c_1, \ldots, c_{nqt}\} \text{ and } c_i = \begin{cases} m_1 \text{ if term}_i\ \varepsilon\ S_1 \\ m_2 \text{ if term}_i\ \varepsilon\ S_2 \end{cases}$$

Typically $(c_1 + c_2 + \ldots + c_{nqt}) < F$ is guaranteed and the upper limit refers to the case when all nqt terms set different bits.

The expected value (W(Q)) of the random variable W is given in [GRA94] as follows.

$$W(Q) = E(W) = F\left[1 - \prod_{i=1}^{nqt}\left[1 - \frac{c_i}{F}\right]\right] \qquad (19)$$

where c_i is equal to m_1 if term is a member of S_1, otherwise it is equal to m_2.

Instead of using equation 18 and performing a lot of computations, we will use equation 19 for calculating W(Q) for an outcome. In a more simplified form W(Q) can be written as

$$W(Q) = F\left[1 - \left[1 - \frac{m_1}{F}\right]^{b_1}\left[1 - \frac{m_2}{F}\right]^{nqt - b_1}\right] \qquad (20)$$

Finally, by substituting W(Q) in equation 11 we find the PERSAV for that specific outcome. The overall percentage saving is then computed by summing up the product of the occurrence probability and the PERSAV value for all outcomes.

5. DESCRIPTION OF THE EXPERIMENTAL ENVIRONMENT

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, KNU75]. 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 IV. 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

The experiments cover three different query cases which are characterized by assigning different probability values (P_j) to the events of having queries with j terms. As for the P_j values, ($j = 1, 2, \ldots, 10$), three different query cases are considered: Uniform Distribution (UD), Low Weight (LW) queries and High Weight (HW) queries. The definitions of the query cases are given in Table IV.

The performance of the three signature generation schemes is evaluated by substituting the m values computed by each scheme in the performance evaluation formulas. For the SM case, the tree diagram in Figure 2 reduces to one level and calculations become simpler than those for the MMS and MMM schemes. For the MMM scheme, we need to compute the $P_i(k)$ value such that $P_i(k)$ is the probability that exactly k terms will be specified from the ith set where $1 \leq i \leq 2, 0 \leq k \leq 1$. The $P_i(k)$ probability for specific values of i and k is calculated by summing up the occurrence probabilities of the outcomes which comply with the constraint that exactly k terms are specified from set S_i.

Note that an experimental design that will achieve complete coverage of all possible combinations of the input parameters is impractical if not unnecessary. Hence the aim of the experiments, broadly, is to analyze the performance of the three schemes (SM, MMS, MMM) in three different query environments and to draw inferences on how the system performance is affected as we alter the values of some input parameters. What is meant by the performance of a signature generation scheme (SM, MMS, MMM) should be regarded, throughout the experiments, as the performance of LHSS for a signature file generated by one of the SM, MMS, or MMM schemes.

5.1 Analysis of m Values

For all signature generation schemes F and m values are directly proportional as long as other parameters are kept constant.

SM scheme:

$m = k \,.\, F$

MMS scheme:

$$m_i = k \,.\, F + d_i \qquad 1 \leq i \leq 2 \qquad d_i = \frac{D_1\left[\ln\frac{q_i}{D_i} - \ln\frac{q_1}{D_1}\right] + D_2\left[\ln\frac{q_i}{D_i} - \ln\frac{q_2}{D_2}\right]}{D \,.\, \ln 2}$$

MMM scheme:

$$m_i = k \,.\, F + e_i \qquad 1 \leq i \leq 2 \qquad e_i = \frac{D_1\left[L_1 - L_i\right] + D_2\left[L_2 - L_i\right]}{D \,.\, \ln 2}$$

where $k = \ln 2 \,/\, D$ (constant).

Generally speaking, when S_1 contains the terms with high discriminatory power, the d_1 and e_1 values are greater than 0, and d_2 and e_2 values are smaller than 0. More formally

if $q_1 / D_1 > q_2 / D_2$ then $d_1 > 0, d_2 < 0,$

and

if $L_2 > L_1$ then $e_1 > 0, e_2 < 0.$

According to our experimental observations,

$$| \ln q_1 / D_1 - \ln q_2 / D_2| < |L_1 - L_2|_{UD} < |L_1 - L_2|_{LW} < |L_1 - L_2|_{HW}.$$

The above inequalities can be interpreted as follows.

1. $m_{1,mms} < m_{1,mmm/ud} < m_{1,mmm/lw} < m_{1,mmm/hw}$
2. $m_{2,mms} > m_{2,mmm/ud} > m_{2,mmm/lw} > m_{2,mmm/hw}$

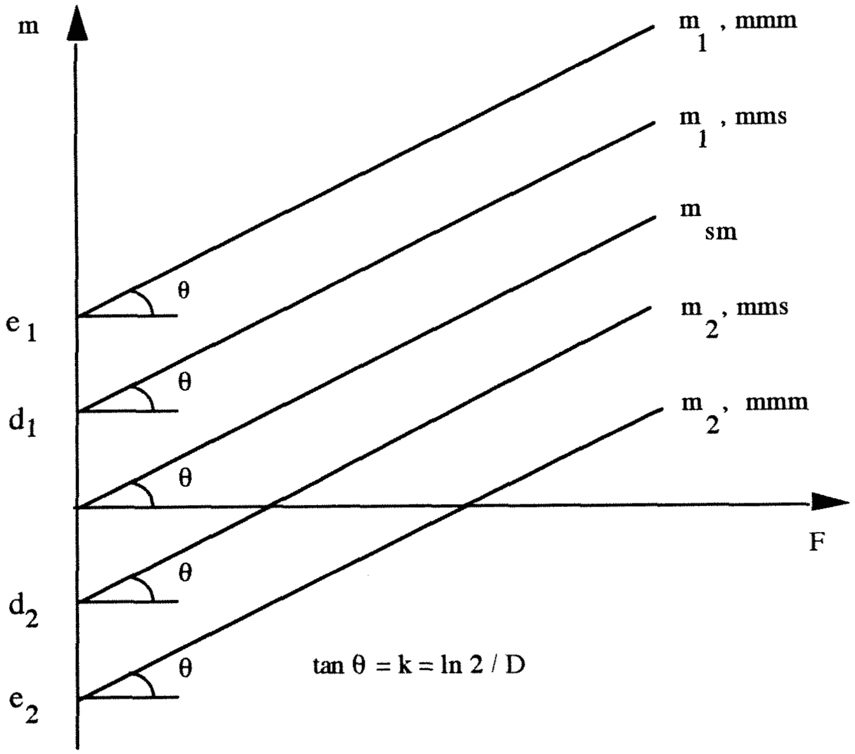


Figure 3. m values vs. signature size F (use F with positive m values).

This analysis is based on the exact **m** (not on the integer **m** values) as seen in Figure 3. Since we should use integer m values, the rounding of m values may lead to deviations from the above inequalities (1) and (2). That is, it is possible that we have all m_1s equal and m_2s equal to each other when they are rounded.

Additionally, as the discriminatory power of the terms in set S_1 increases (i.e., $| \ln q_1/D_1 - \ln q_2/D_2|$ or $|L_1 - L_2|$ increases) the magnitude of the constants d_i and e_i increases. This leads to larger m_1 and smaller m_2 values when F is kept constant.

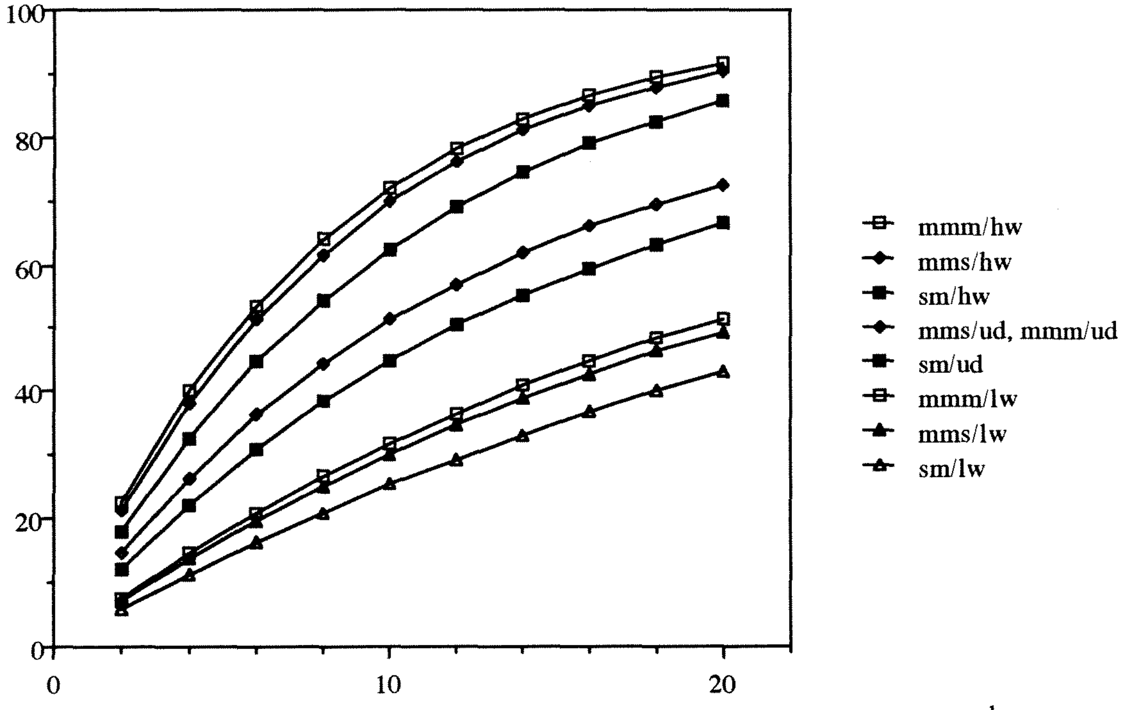
6. EXPERIMENTS

6.1 Simulations

In the simulations we kept the values of D_1 (8), D_2 (32), q_1 (0.8), and q_2 (0.2) fixed. Note that the choice of q_i values is also consistent with the 80-20 rule which approximates most of the real life cases [KNU75].

Simulation 1

In this experiment we observed the system performance for varying the values of the hashing level h. The signature size, F, is taken as 500. Figure 4 summarizes the results and shows that as the level of hashing increases, PERSAV increases with a decreasing rate for all schemes and query cases. (Note that h = 20 implies a database of more than one million data objects.) The percentage savings provided by all the schemes are highest in the HW case since we have a large suffix weight due to the large number of query terms.



m_i values (in figure legend order): (13, 8), (12, 8), 9, (12, 8), (12, 8), 9, (13, 8), (12, 8), 9; n= 2^h

Figure 4. Percentage savings (PERSAV) vs. hashing level (h).

For all query cases, MMM and MMS provide higher PERSAV values than SM does. This is due to the fact that the MMM and MMS schemes make use of more information about the system characteristics in determining the m values as compared to the SM scheme (which does not take the discriminatory power of terms into account).

The MMM scheme also outperforms the MMS scheme in the HW and LW query cases since the type and relative frequency of the queries are considered by MMM, but not by MMS (which considers single term queries only). In HW and LW, there is a deliberate non uniformity in the number of query terms and this results in higher performance of MMM which takes this fact into account. In the UD case, MMM and MMS provide the same integer m values most of the time. (Recall from the "analysis of m values" that the m values for MMS and the m values of MMM in UD queries are very close to each other. In fact in Figure 4 m_1 and m_2 values of both schemes are the same.) Thus, MMM and MMS provide the same PERSAV value when the distribution of the number of query terms is uniform.

A still more interesting observation is that as the value of h increases, the amount of extra percentage savings provided by MMM over other schemes increases until h reaches a certain value, and then the amount of extra percentage savings starts to decrease. The specific value of h at which this change (from increase to decrease) occurs depends on the value of the other input parameters; but our observation is that this value of h is always smaller for HW queries than that for UD queries, and smaller for UD queries than that for LW queries. Of course in the case of UD queries, the amount of extra savings provided by MMM over MMS is zero when they yield the same m values, but the above observation is still valid when MMM and MMS provide different m values for UD queries.

The intuition behind the above observation is that when h is high, the suffix weight of the query signature is high regardless of the type of the scheme used. The similar discussion is valid also in terms of the number of query terms (i.e., if a query contains many terms the suffix weight will be very high regardless of the type of scheme used).

Thus we can say that after some value of h, the performance of MMM, MMS, and SM schemes will get closer to each other as the value of h increases.

Note that the above discussions and observations about h are also valid in terms of n (number of primary pages) since h and n are directly related to each other.

Simulation 2.

In this simulation we observe the system performance as the signature size varies. The hashing level, h, is taken as 10. Hence $n = 2^h = 1024$.

Figure 5 shows the percentage savings that are observed when the m values provided by all the schemes are used without rounding. The reason for using exact values is to eliminate deviations from actual system behavior introduced by rounding the m values. Figure 6 shows the results obtained by using the integer m values. Thus, Figure 5 refers to the ideal case whereas Figure 6 provides us with what we should expect to see in case of an implementation.

According to Figure 5, the performance of the SM scheme is constant regardless of the F value. This is due to the fact that m/F ratio is always constant and this leads to a constant suffix weight as long as h is constant (see equation 19). However, the m_i/F ratios do not remain constant as F increases in MMM and MMS schemes. As the value of F increases, the percentage savings provided by MMM and MMS decreases with a decreasing rate since the suffix weights decrease. The reason behind the decrease in suffix weight (as F increases) follows from the fact that m_1/F ratio decreases and m_2/F ratio increases as F increases (recall the analysis of m values). Since the query weight is dominated by the bits set by terms from S_1, the suffix weight decreases in MMM and MMS schemes and, thus, the access savings decrease.

As seen in Figure 5, initially MMM provides considerable extra savings over SM for all query cases. However, as F increases, the amount of extra savings provided by MMM over SM decreases fast and reaches to a small value at F = 2,000 for all query cases. The same observation is valid also for the extra savings provided by MMS over SM and for those provided by MMM over MMS.

Thus, it is clearly seen that the performance of MMM and MMS converges to that of SM as F increases (provided that the hashing level is constant).

Finally, when we compare Figure 5 and Figure 6, we see that, for large F values, there is not so much difference between the system performances observed in both figures. But, for small F values (e.g., 100, 200), Figure 6 exhibits large deviations, especially for the SM scheme. It is due to the fact that we introduce a smaller deviation when rounding a large m value to its closest integer as compared to rounding a small m value.

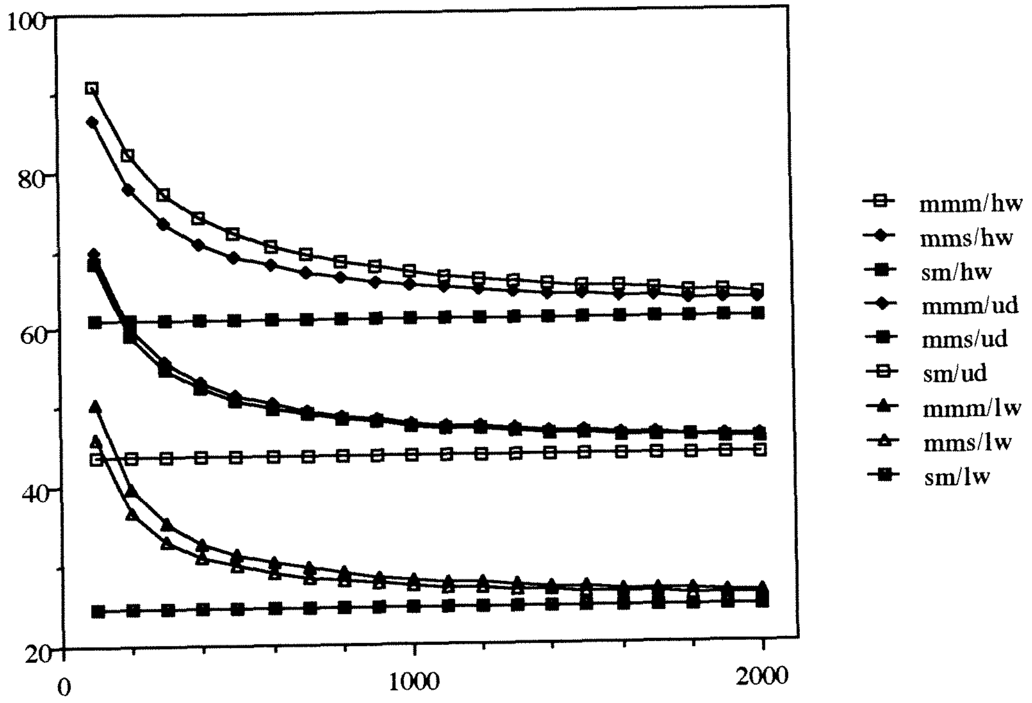


Figure 5. Percentage savings (PERSAV) vs. signature size (F) for exact m values.

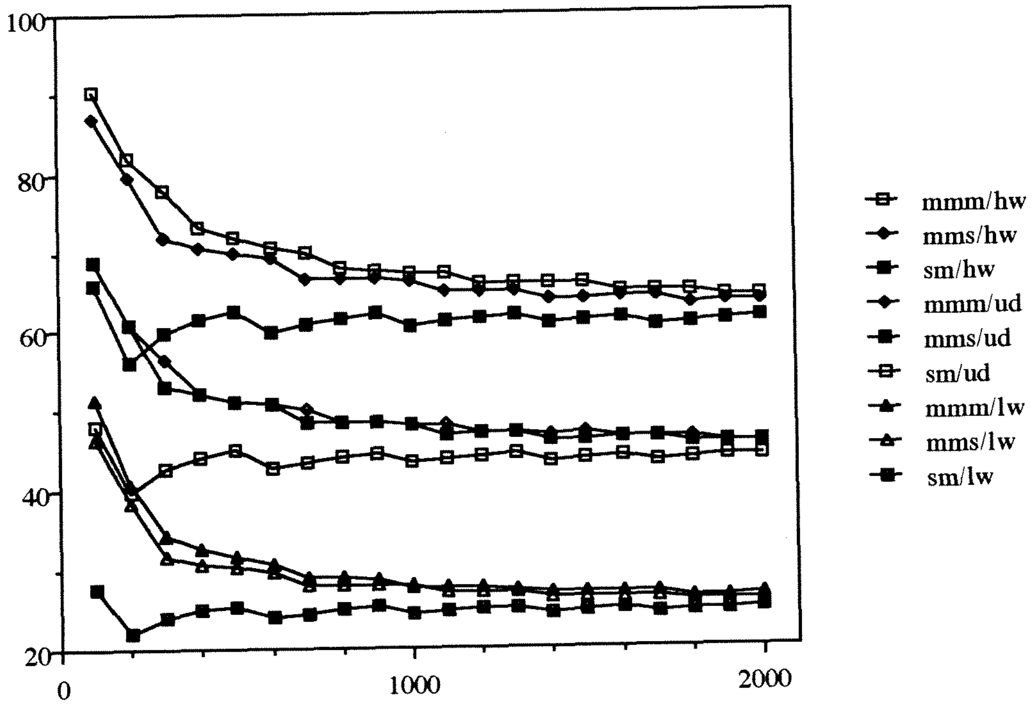


Figure 6. Percentage savings (PERSAV) vs. signature size (F) for integer m values.

6.2 Implementation for Actual Text Databases

We have implemented the SM, MMS, and MMM signature generation schemes for LHSS using two common databases (and queries) of the information retrieval literature: INSPEC and NPL. Our data related to these text databases was their document and query vectors. Database statistics are provided in Table V.

Table V. Database statistics

Database	No. of Documents	No. of Terms	Avg. Doc. Vec. Length	Std.* of Doc Vec. Length	No. of Queries	Avg. Query Vec. Length
INSPEC	12,684	14,573	32.5	14.27	77	15.8
NPL	11,429	7,491	20.0	10.84	100	7.2

(* Std.: Standard deviation)

For each of the databases, we first grouped the database terms into two sets (S_1 and S_2). After finding the query frequencies of all the terms, we sorted the terms by decreasing query frequency. For both databases, we included the top terms with query frequency greater than one in the set S_1, and the terms that occur in only one or no query at all were automatically considered in set S_2. By this approach we were able to put the terms with low document frequency into S_1, i.e., the set S_1 contains the terms with high discriminatory power.

Thereafter we calculated the q_1, q_2, D_1, and D_2 values as follows.

$$q_1 = \frac{\text{sum of query frequencies of } S_1 \text{ terms}}{\text{sum of query frequencies of all terms}} \quad \text{and} \quad q_2 = 1 - q_1$$

$$D_i = \frac{\sum_{j=1}^{\text{total no. of docs.}} D_{i,j}}{\text{total no. of documents}} \qquad (1 \leq i \leq 2)$$

where $D_{1,j}$ and $D_{2,j}$ are, respectively, the number of terms from S_1 and S_2 for the jth document (see the results in Table VI).

Table VI. Database Statistics for Term Classes

Database	D_1	D_2	q_1	q_2
INSPEC	11.897	20.605	0.711	0.289
NPL	6.155	13.802	0.691	0.309

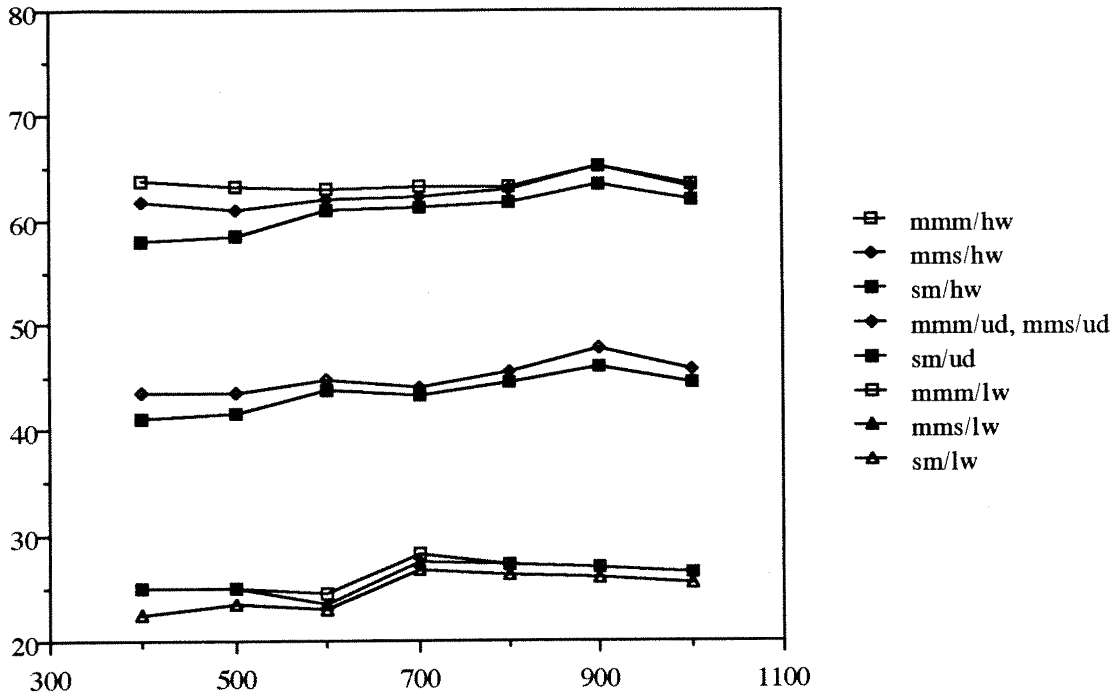


Figure 7. Percentage savings (PERSAV) vs. signature size (F) for INSPEC.

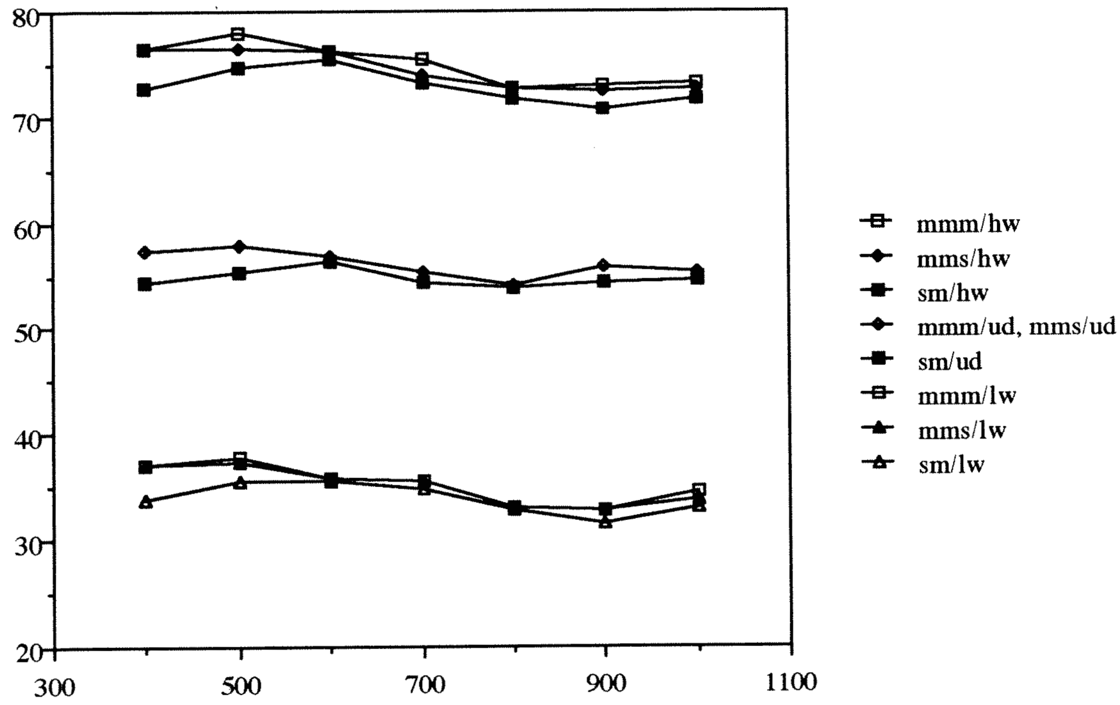


Figure 8. Percentage savings (PERSAV) vs. signature size (F) for NPL.

After the determination of q_1, q_2, D_1, and D_2 values, we generated the document signatures according to the m values calculated by each scheme (i.e., SM, MMS, $\mathrm{MMM}_{\mathrm{UD}}$, $\mathrm{MMM}_{\mathrm{LW}}$, $\mathrm{MMM}_{\mathrm{HW}}$) for various values of the signature size. Then the document signatures were hashed with a load factor of 75% and page size of 2K bytes. The document signatures stored in pages contain F bits plus an extra four bytes for the purpose of pointing to the document itself.

Parallel to the document signature generation, we also generated signatures for 1,000 queries which were produced randomly choosing terms from S_1 or S_2 according to the q_1 (q_2) values. The queries were generated for all query cases (UD, LW, HW) according to the P_j values given in Table IV (e.g., for the HW case, 300 queries contain 10 terms, 250 queries contain nine terms, and so on). After query generation, we generated query signatures by using the SM, MMS, and MMM schemes for each query set.

After signature generation, we hashed the document signatures and started the retrieval process. We counted the total number of primary and overflow pages accessed by 1,000 queries and found the access ratio by dividing this number by 1,000 times the total number of addressable (primary) and overflow pages. Finally, the percentage access savings (PERSAV) is calculated by subtracting the access ratio from one and multiplying the result by 100.

An important point about the implementations is that variance of D (average number of terms per document) for both databases is not small enough to be simply ignored (note that in the derivations of the formulas the variance is assumed as negligible [FAL85, FAL87a, FAL88]). This may lead to deficiencies in the system performance that would otherwise not be observed.

Table VII. LHSS File Statistics for the Databases (for all schemes)

	LHSS	F						
Database	Parameters	400	500	600	700	800	900	1000
INSPEC	h	9	10	10	10	10	10	11
	n	458	564	677	769	891	995	1128
NPL	h	9	9	10	10	10	10	10
	n	412	508	610	693	803	897	1016

Table VII shows the final LHSS file structure in terms of h and n. (The structure is the same in terms of h and n for all schemes.) As seen in the table, when the page size is kept fixed, an increase in F leads to an increase in n (h). Based on the results of Simulation 1, we expect the system performance to decrease due to the increase in F (except SM) and to increase due to the increase in n. Actually, these two effects almost neutralize each other such that we observe very little performance differences for various values of F as can be seen in Figures 7 and 8. We may say that percentage savings provided by each scheme in each query case are almost constant for varying F values. When we simulate these implementations by our mathematical model, we observe that the system performance improves very slightly (remains almost constant) for the simulation parameters of Table VII obtained from the implementations. Another observation is that the performance levels of the simulations are always slightly higher than the levels observed in the implementations (see Table VIII for a summary of PERSAV differences).

Table VIII. Summary of PERSAV Differences of Simulation and Implementation Results for All Observations Obtained for NPL and INSPEC Databases

	INSPEC	NPL
avg (sim_result - imp_result)	4.8	7.0
standard deviation (sim_result - imp_result)	1.5	2.5
max (sim_result - imp_result)	8.0	11.0
min (sim_resul - imp_result)	1.0	2.0

The difference between the simulation and implementation results may be based on three reasons.

1. The variance of D for both database is high,
2. The mathematical model assumes that 1s (weight bits) are uniformly and randomly spread over both query and document signatures; however, interdependencies among terms may slightly invalidate this assumption,
3. The mathematical model for the performance evaluation of LHSS ignores the overflow pages; however, in our implementations about 15 to 20% of signatures are in overflow pages.

Note also that the performance of SM is quite close to the performance of the other schemes. In most cases MMM provides the same performance as MMS and sometimes outperforms MMS with a very small amount of percentage savings.

Another interesting observation is that the percentage savings provided for the NPL database is higher than those for the INSPEC database for all query cases and signature generation schemes. This is due to the fact that signature generation schemes provide larger m values for NPL as compared to INSPEC. When F is the same for both databases, the m values of NPL are greater than those of INSPEC, since the D value of NPL is smaller than the D value of INSPEC. Note that, in this case, the average document signature weights for both databases are the same (approximately 50% of the bits set to 1), but the suffix weights of the query signatures for NPL are larger than those of INSPEC when we submit a query (e.g., 10-term query) to both databases.

7. CONCLUSIONS AND FURTHER RESEARCH POINTERS

In this study, 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. The first scheme (SM) allows all terms to 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. By means of employing MMS and MMM signature generation schemes, we have relaxed the uniform document/query frequency and single term query assumptions as applied to LHSS.

The simulations based on the mathematical model developed for performance evaluation reveal that MMM and MMS are always superior to SM since they both make use of more information about the system. However, it is hard to say that MMM is superior to MMS. MMM provides better performance than MMS when the distribution of the number of query terms is non uniform; but this performance gap between MMM and MMS is always very small when compared to the gap between MMM and SM or MMS and SM. Hence, the choice between MMM and MMS is a difficult one and necessitates a careful analysis of the typical queries that can be submitted to the system.

As the database size (i.e., the level of hashing) increases, the performance provided by all the schemes increase, and, after some point, the performance gap between the schemes starts decreasing while their performance continues to increase with a decreasing rate. Hence, for large databases the choice of a scheme does not affect the system performance that much

The results concerning the effect of signature size, F, on the system performance are also interesting. When we keep the level of hashing constant, the performances of MMS and MMM decrease as F increases, and the performance of SM remains constant. In other words, the performances of MMM and MMS approach to that of SM as the signature size is increased. Hence, if we desire high performance, i.e., high access savings, we should prefer small signature sizes. At this point, considering the fact that false drop probability is higher for smaller signature size, we should choose the signature size according to the trade-off between the false drop probability and the access savings and also the availability of system resources such as storage space.

Finally, the implementation results obtained by using INSPEC and NPL text databases validate our analytical observations with a small deviation from the simulation results based on the performance evaluation formulas. Hence, the results of our simulations can be used to describe the expected behavior of actual systems and may be helpful in determining the optimal values of some system parameters for achieving high performance.

With some modification, our derivations can be used to evaluate the performance of additional signature partitioning methods. An initial effort on this is provided in [AKT93c]. 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 acknowledge the contributions of Deniz Aktug to an earlier version of this paper. We thank Paolo Ciaccia of University of Bologna for his comments for an earlier version of this paper and Pavel Zezula of Technical University of Brno for providing us with timely results of his work before they are published in the media. The first author would like to thank Mehmet Baray of Bilkent University for the environment provided during his sabbatical leave.

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*. (March 1993), pp. 148-153.

[AKT93b] AKTUG, D., AND CAN, F. Analysis of multiterm queries in a dynamic signature file organization. In *Proceedings of the 16th Annual International ACM-SIGIR Conference* (June 1993), pp. 96-105.

[AKT93c] AKTUG, D., AND CAN, F. Analysis of signature generation schemes for multiterm queries in partitioned signature file environments. Working Paper 93-007, Dept. of Systems Analysis, Miami Univ., Oxford, Ohio, May 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.

[FAL84] FALOUTSOS, C., AND CHRISTODOULAKIS, S. Signature files: An access method for documents and its analytical performance evaluation. *ACM Transactions on Office Information Systems*. 2, 4. (October 1984), 267-288.

[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.

[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.

[GRA94] GRANDI, F. On the signature weight in "multiple" m signature files. Reprint, submitted for publication. (Author's address: C.I.O.C.-C.N.R. and Dipartimento di Elettronica, Informatica e Sistemistica Universita di Bologna, Viale Risorgimento 2, I-40136 Bologna, Italy.)

[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.

[KNU73] KNUTH, D., E. *The Art of Computer Programming.*, 2nd ed. vol. 1: *Fundamental Algorithms.* Addison-Wesley, Reading, Mass., 1973.

[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.

[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.

[TIB93] TIBERIO, P., AND ZEZULA, P. Selecting signature files for specific applications. *Information Processing and Management*. 29, 4 (1993), 487-498.

[ZEZ89] ZEZULA, P. Linear hashing for signature files. In K. Boyanov & R. Angelinov (Eds.), *Network Information Processing Systems*. Amsterdam: Elsevier Science, North-Holland, and IFIP Publishers, 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.