Efficient Signature File Methods for Text Retrieval

Dik Lun Lee, *Member, IEEE and IEEE Computer Society*, Young Man Kim, and Gaurav Patel

***Abstract*—Signature files have been studied extensively as an access method for textual databases. Many approaches have been proposed for searching signatures files efficiently. However, different methods make different assumptions and use different performance measures, making it difficult to compare their performance. In this paper, we study three basic methods proposed in the literature, namely, the *indexed descriptor file*, the *two-level superimposed coding scheme*, and the *partitioned signature file approach*. The contribution of this paper is two-fold. First, we present a uniform analytical performance model so that the methods can be compared fairly and consistently. The analysis shows that the two-level superimposed coding scheme, if stored in a transposed file, has the best performance. Second, we extend the two-level superimposed coding method into a *multilevel superimposed coding* method, we obtain the optimal number of levels for the multilevel method and show that for databases with reasonable size the optimal value is much larger than 2, which is assumed in the two-level method. The accuracy of the analytical formula is demonstrated by simulation.**

***Index Terms*—Access methods, text retrieval, performance analysis, superimposed coding.**

I. INTRODUCTION

SIGNATURE FILES have been studied extensively as an access method for textual databases. They have been used in a variety of applications, ranging from textual databases such as news databases [20] to multimedia office filing [3] to chemical databases for DNA matching [15].

Research on signature files can be roughly classified into two categories. The first category focuses on new signature schemes for reducing the false drop probability without increasing the storage overheads. Numerous methods have been proposed and evaluated in the literature [9], [10], [14]. The second category of research is motivated by the fact that the search time on a signature file is directly proportional to the size of the text file, resulting in an unacceptable performance when the database is large. To alleviate the problem, many efficient search methods have been proposed in the literature, including the indexed descriptor file method [16] and its variant S-tree [4], the two-level superimposed coding method [1], [19] and the partitioned signature file method [12], [13]. These methods in general organize the signature file in a way such that only a small number of the signatures are accessed in response to a query. Methods utilizing special hardware processors have also been proposed [11], [21].

The research reported in this paper falls into the second category. It is prompted by the differences in performance measures and assumptions used by these various search methods. For instance, different coding schemes were used for generating the signatures—disjoint coding was used in the indexed descriptor file for generating the block signatures while superimposed coding was used in other methods. Further, different techniques were employed to improve the search time (e.g., the partitioned signature file is based on hashing while the other methods are based on tree structures). The performance measures used in these methods were also different—the number of disk accesses was used in some methods while signature reduction ratio was used in others. These wide differences make it difficult to compare the performance of different methods in a consistent manner.

In this paper, a uniform framework is used to analyze these methods in the context of text retrieval. The analysis is performed using the same signature coding technique and the same performance measures. The analytical results reveal several unexpected characteristics of the methods. The reasons for these phenomena are discussed. Based on these results, we propose a new method called the *multilevel superimposed coding*, which is a generalization of the two-level method. For the same storage overhead as a single-level method, the multilevel method yields the same false drop probability for unsuccessful searches and an excellent signature reduction ratio. The optimal number of levels for the multilevel superimposed coding method is obtained. It is found that for databases of reasonable size the optimal value is much larger than 2, which is assumed in the two-level method.

The rest of this paper is organized as follows. In Section II, superimposed coding is reviewed to provide the readers with the basic ideas and terminology used in this paper. A common framework, the symbols adopted in our analysis, and the analytic performance models of the methods are presented in Section III. The results of the analysis and a discussion are given in Section IV. Section V presents the multilevel superimposed coding method and its performance analysis. Finally, Section VI concludes our study and gives a look at some research issues for further study.

II. SUPERIMPOSED CODING

Signatures can be obtained in a number of ways [5], [7]. Superimposed coding is perhaps the most common method used. In superimposed coding, a text is divided into text blocks containing the same number of unique, nontrivial, words. Each word in a text block is hashed into a word signature. A block signature is generated by superimposing all word signatures generated from the block. In a query, the query terms are hashed and superimposed into a query signature in a similar way. Then the query signature is matched against each signature in the signature file. Fig. 1 is an example showing the generation of the block signature from a text block.

Manuscript received July 13, 1992; revised Apr. 5, 1993.
D.L. Lee and G. Patel are with the Department of Computer and Information Science, Ohio State University, Columbus, OH 43210-1277.
Y.M. Kim is with the Mitsubishi Materials Corp., Omiya City, Japan.
IEEECS Log Number K95038.

text block	··· **text** ······ **database** ···
word signatures:	
text	001 000 110 010
database	000 010 101 001
block signature ($\vee$)	001 010 111 011

Queries	**Query Signatures**	**Results**
1) **retrieval**	010 001 000 011	← **no match**
2) **database**	000 010 101 001	← **match**
3) **database** $\wedge$ **text**	001 010 111 011	← **match**
4) **information**	001 000 111 000	← **false drop**

Fig. 1. Signature generation and comparison based on superimposed coding.

The signature file is a filtering mechanism which will eliminate most, but not all, of the text blocks which will not match the query. The first case shown in Fig. 1 illustrates this point. The query signature doesn't "match" with the text signature in that some of the bits in the text signature are zero while the corresponding bits in the query signature are set to one. If the query term "retrieval" is indeed in the text, the query signature would be one of the word signatures forming the text signature, thus every bit set in the query signature will be set in the text signature. The second and third cases show a match between the query signature and the block signature when, for each bit in the query signature set to one, the corresponding bit in the block signature is also set to one. The third case shows that a conjunctive query of more than one query term can be matched in one comparison. The fourth case is a false drop. False drops are text blocks which the signature file identifies as containing the query terms (i.e., a match) but indeed they don't. They can be eliminated by further comparing the query terms with the text blocks, but the performance will be degraded. False drops are unique in the signature file approach, and much work has been done on minimizing the false drop probability [9], [14]. Intuitively, for the same number of distinct keywords in a text block, when the length of the signatures increases, the "density" of ones in the signatures decreases, and the chance of getting false drops will decrease correspondingly. However, it will increase the storage overhead (i.e., more bits are "unused"). It has been shown that in order to minimize false drop probability, the expected number of zeros and ones in a signature must be the same [2].

The advantage of the signature file method over the conventional inverted file method is its moderate and controllable storage overhead—10–20% for signature files compared to over 100% for inverted files. The retrieval speed of the signature file method is much faster than full-text scanning but slower than an inverted file. In other words, it is a compromise between inverted file and full-text scanning methods.

III. Analytical Models

A. Basic Configurations and Assumptions

A.1. Methods to be Analyzed

The indexed descriptor file method [16] and the S-tree method [4] are essentially the same in that superimposed index nodes are used and that an index signature of a nonleaf node in the tree is obtained from superimposing all signatures of its descendent nodes. Thus, only the *indexed descriptor method* is considered. The *two-level superimposed coding scheme* [18], [19] is the second method to be analyzed. The *partitioned signature file* [12], [13] method has three variants according to the ways that the keys are selected. According to Lee and Leng [12], the *extended prefix* technique has a variable key length, and its performance is not as good as the other two. Therefore, we only analyze the *fixed-prefix* and the *floating-key* partitioned signature file methods. To avoid the confusion with different terminologies used in different methods, *indexes* in the indexed descriptor file and *segment signatures* in the two-level superimposed coding method are simply called *signatures*, and the root in a tree structure is at level 1.

A.2. Signature Coding Method

In these four methods, the first method uses disjoint coding and the other methods use superimposed coding. Superimposed coding generates a block signature by *superimposing* word signatures together. On the other hand, disjoint coding forms a block signature by *concatenating* word signatures together. To facilitate comparison, superimposed coding is used as the standard signature coding method. This assumption won't affect the basic superimposition mechanism used in the index descriptor method for reducing the search space.

A.3. Hash Function

It is assumed that a hash function provides a uniform distribution of ones in the signature and that the number of ones in a word signature is much less than the length of the word signature. This assumption is required when the false drop probability is minimized.

A.4. Performance Measures

Two performance measures are studied: 1) the amount of storage required for each method (i.e., the storage overhead), and 2) the total number of block signatures and index signatures which require comparison to the query signature. The latter is normalized by the number of signatures in the original signature file and becomes the *signature reduction ratio* [12]. However, for the two-level and multilevel superimposed coding schemes, the cost of searching a signature at the higher level is more expensive than that at the lower level due to the difference in signature lengths. Since the signature reduction ratio cannot reflect this difference, the *computation reduction ratio* is introduced to measure the ratio of the number of bit comparisons *actually required* in a method to that of a signature file with only one level (hereafter referred to as the *single-level method*). The computation reduction ratio reflects the amount of CPU processing (i.e., comparison) involved. The bits actually requiring comparison in a signature are those specified in the query signature [11]. The reduction ratios are preferred over the actual number of disk accesses as a performance measure, because the reductions ratios give an implementation-independent measure of the ability of a method to reduce the search space, and hence the I/O cost, without considering the low-level implementation, whereas the number of disk accesses depends on exactly how the signature file is organized (e.g., as a sequential file or in bit slices).

A.5. Signature Parameters

For a fair comparison of the performance, the same set of parameters (e.g., the length of a block signature and the number of bits set in a word signature) must be used. These parameters can be obtained by minimizing the false drop probability [19] or the disk access cost function [2]. However, the resulting equations for determining the parameter values are similar. In this paper, the false drop probability is used as a target for minimization [19].

Since the false drop probability affects the storage overhead (thus the amount of processing required), the methods are evaluated based on the same false drop probability so that they can be compared fairly. For instance, if a method is faster for searching than another method but it also has a higher false drop probability, then no conclusion can be made on the relative superiority of the two methods. The false drop probability is the probability that a signature may seem to qualify in a query when the corresponding text does not actually satisfy the query. However, for a signature file structure with more than one level of signatures, this definition must be extended. Thus, we define the *local false drop probability* at the *i*th level as the probability at which a signature at the *i*th level may seem to satisfy a query although the corresponding descendent text blocks at the last (leaf) level do not contain a qualified signature, and the *global false drop probability* (or simply false drop probability when no ambiguity arises) as the probability at which a text's signatures at all levels seem to qualify a query but the text itself actually does not. In this paper, different methods are compared based on the same global false drop probability, because it is the one which affects the number of false drops seen by the user.

A.6. Dimensions

Several variables may affect the performance measures: the total number of block signatures, the false drop probability, the number of keywords per block, the number of query terms, the number of blocks containing the query terms, the *packing factor* (the number of index or block signatures per index node) in the indexed descriptor file method, the *segment size* in the two-level method, and the *partition size* in the partition methods. We note that the number of blocks containing the query terms (true drops) was largely ignored in previous analysis. However, as we shall see in our analysis, it intensely affects the performance of the two-level and the multilevel methods. Thus, this parameter is selected as one of the variables in our performance study. In the following sections, the general equations for each method are derived. The performance is obtained for different number of keywords per block and different number of blocks containing the query words.

B. Symbols

A	number of signatures searched (at all levels).
A_F	number of signatures searched (at all levels) due to false drop.
A_T	number of signatures searched (at all levels) due to true drop.
b	packing factor (number of child signatures per index node).
C	ratio of the number of searched index and block signatures to the total number of block signatures (i.e., signature reduction ratio).
D	ratio of the number of bits compared in the searched block and index signatures to the number of bits compared in the original signature file method using a transposed signature access method (i.e., computation reduction ratio).
h	height of the index tree, $h = \log_b n$.
k	key length in the partitioned signature method.
m	length (in bits) of a signature.
M	total number of bits required (i.e., storage requirement).
n	number of blocks in a text file.
n_1	number of segment signatures in the two-level method.
n_2	number of blocks per segment in the two-level method.
P^f	global false drop ratio.
$P(a1,a2)$	probability that a particular set of $a2$ bits is set to 1 in a signature superimposed from $a1$ word signatures.
q	number of distinct keywords per query.
s	number of distinct keywords per text block.
t	number of blocks containing the query words (true drops).
w	number of ones in a word signature.
w_q	number of ones in a query signature.
w_s	number of ones in a block signature.

C. Basic Equations

In this section, we describe some basic equations obtained from false drop probability minimization. These equations are common to all the methods to be studied and have been proved elsewhere [17], [19]. Let m and w be the length and weight of a word signature, respectively. It is assumed that bits are set randomly in the signatures and each of the $\binom{m}{w}$ possible word signatures is equally likely to be chosen when a word signature is generated. The probability, $P(a1, a2)$, that $a2$ bit positions are set to one in a signature superimposed from $a1$ word signatures with m and w is:

$$P(a1,a2) = \sum_{j=0}^{a2} (-1)^j \binom{a2}{j} \binom{m-j}{w}^{a1} \binom{m}{w}^{-a1}$$

If $a1$ is sufficiently large and $w \ll m$, then the following approximation can be obtained:

$$P(a1,a2) \approx \left[1-(1-w/m)^{a1}\right]^{a2} \qquad (1)$$

If we assume that w_x is the weight of a signature superimposed from x word signatures, then the following equation shows the relationship between $P(a1, a2)$ and w_x:

$$w_x = mP(x,1) = m\,(1-(1-w/m)^x).$$

The false drop probability of a signature file, P^f, can be represented by $P(s,w)$.[1] Note that P^f, which denotes the global false drop probability, is used since for a single-level signature file the global false drop probability is the same as the local false drop probability.

$$P^f = P(s,w)$$

By minimizing the false drop probability, the following relationship can be obtained:

1. The probability is actually the false drop probability for an unsuccessful search of a single-term query. However, it approximates accurately the false drop probability for a successful search [8].

$$(1 - w/m)^s = 0.5$$
$$P^f = 0.5^w$$

Given P^f, s, and the above relationship, w and m have the following values:

$$w = (1/\log_e 2)\log_e(1/P^f) \tag{2}$$

$$m = (1/\log_e 2)^2 s \log_e(1/P^f) \tag{3}$$

Equations (1), (2), and (3) are used extensively in the analysis in the rest of this paper. In the single-level method, given the number of blocks in a text file, n, the required storage in bits, M, is: $M = mn$.

D. Indexed Descriptor File

An indexed descriptor file structure is a tree with h levels in which the lowest level consists of block signatures which are superimposed codes obtained from the text blocks. A group of b signatures at the ith level is superimposed together to form a signature at the $(i-1)$th level. Thus, a signature at the ith level is indirectly obtained from superimposing all its descendent signatures, including signatures at the last (leaf) level. We can also observe that every signature has the same length in all levels. The structure of the indexed descriptor3 file is shown in Fig. 2.

When there are n text blocks, the relationship between n and h is: $n = b^h$

During a retrieval, every signature in the first level is tested. Then, for each signature matched with the query signature, its child signatures in the next lower level are tested and so on. Since signatures at the higher levels are a superimposition of signatures at the lower levels, the global false drop probability is exactly the same as the local false drop probability at the last level. w, m, w_s, and w_q can be formulated as follows [19]:

$$w = (1/\log_e 2)\log_e(1/P^f) \tag{4}$$

$$m = (1/\log_e 2)^2 s \log_e(1/P^f) \tag{5}$$

$$w_s = 0.5m$$

$$w_q = m\left(1 - 0.5^{q/s}\right)$$

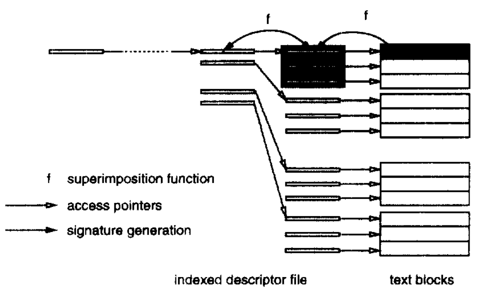


Fig. 2. Indexed descriptor file.

Since an indexed descriptor file consists of $\sum_{i=1}^{h} b^i$ signatures, each of which is of length m, the total storage (in bits) required is:

$$\begin{aligned} M &= m\sum_{i=1}^{h} b^i \\ &= m\frac{b}{b-1}(n-1) \\ &\approx \frac{b}{b-1}mn \end{aligned} \tag{6}$$

Since mn is the storage overhead for a single-level signature file at the same false drop probability, P^f, the indexed descriptor file takes $b/(b-1)$ times the storage of a single-level signature file to yield the same false drop probability.

The total number of signatures searched at all levels, A, consists of two components: searches resulting from true drops and those from false drops. In the following equations, $a_{T,i}$ and $a_{F,i}$ are the probabilities that a node at the ith level is searched due to true drop and false drop, respectively. At the first level (root), all signatures must be searched. Thus, $a_{F,1} = a_{T,1} = 1$. At the ith level, where $i > 1$,

$$\begin{aligned} a_{F,i} &= \prod_{j=1}^{i-1} P\left(b^{h-j}s, w_q\right) \\ &= \prod_{j=1}^{i-1}\left(1 - 0.5^{b^{h-j}}\right)^{w_q} \end{aligned}$$

$$a_{T,i} = \left(1 - \left(1 - 1/b^{i-1}\right)^t\right)$$

$$A = \sum_{i=1}^{h} b^i\left(a_{F,i} + a_{T,i} - a_{F,i}\,a_{T,i}\right) \tag{7}$$

Since the ith level has b^i nodes, $b^i a_{F,i}$ and $b^i a_{T,i}$, represent the number of signatures searched resulting from false drops and true drops at the ith level, respectively. Thus, na_F^h and na_T^h are the respective number of signatures searched at the leaf level.

An informal proof of the above equations is given below. At the root node of the index tree, there are b signatures, and all of them must be compared. At the next level, there are b sets of signatures, each in turn containing b signatures. A signature in the root node is obtained by superimposing its corresponding child signatures at the second level. The probability for a node at the first level to be selected due to false drop is $P(b^{h-1}s, w_q)$ (i.e., $a_{F,2}$), where $(b^{h-1}s)$ is the number of words superimposed into a signature at the root node and w_q is the number of bits to be tested. The probability for a node at the first level to be selected because it actually contains the query terms can be similarly derived as $(1-(1-1/b)^t)$ (i.e., $a_{T,2}$), which is the probability for a particular signature at the root level to have a block signature in its subtree satisfying the query. Let's explain more about this result. There are t blocks containing the query terms. Since these blocks are uniformly distributed into b subtrees, the probability for a subtree to contain a specific block is $1/b$. The probability for a subtree not to contain a specific block is $(1-1/b)$. The probability for a subtree not to contain any of the t blocks is

$(1-1/b)^t$. Thus, the probability for a subtree to contain one or more of the t blocks is $(1-(1-1/b)^t)$. Since there are b sets, each containing b signatures in the second level, the total number of signatures searched at the second level due to false drop and true drop becomes $b^2P(b^{h-1}s, w_q)=b^2a_{F,2}$ and $b^2(1-(1-1/b)^t)=b^2a_{T,2}$, respectively. The derivation for other levels is similar to that of the second level. Finally, we note that there are overlaps between the nodes searched due to false drop and those due to true drop. Since the distributions of false drops and true drops are independent from each other, the region that is counted twice at the ith level can be derived as $b^i a_{F,i} a_{T,i}$, which is reflected in (7).

The signature reduction ratio, C, and the computation reduction ratio, D, can be derived accordingly:

$$C = A/n \tag{8}$$

$$D = (Aw_q)/(nw_q) = C$$

E. Two-Level Superimposed Coding Method

The two-level superimposed coding scheme consists of two levels of signatures (see Fig. 3). Like the indexed descriptor method, the signatures at the lower level (block signatures) are superimposed codes generated from the text blocks. The difference between the indexed descriptor file method and the two-level superimposed coding scheme is in the way that signatures at the higher level (segment signatures) are constructed. In the two-level superimposed coding method, a signature at the higher level is a superimposed code generated *directly* from a group of n_2 text blocks, instead of superimposing the n_2 block signatures. In other words, a signature in the higher level can be considered as being generated from a very long text block consisting of approximately n_2s words. Therefore, the optimal signature length and weight at the higher level are different from those at the lower level.

In the following equations, subscript 1 and 2 indicate, respectively, the parameter at the segment level and the block level. We use n_1 and n_2, respectively, to denote the number of signatures in the first (root) level and the number of signatures in the second (leaf) level indexed by a signature at the higher level. Since a block signature is searched only when its parent signature satisfies the query, the relationship between the global false drop probability, P^f, and the local false drop probability for the first and second level, P_1^f and P_2^f, respectively, can be expressed as: $P^f = P_1^f P_2^f$

Note that the relationship among the false drop probabilities is different from that of the indexed descriptor file method. This is because a signature at the higher level is generated independent of the block signatures at the lower level, which is not the case for the indexed descriptor method.

For unsuccessful search (i.e., $t=0$), the following equations can be derived [19]:

$$n = n_1 n_2$$
$$s_1 = n_2 s$$
$$s_2 = s$$
$$w_1 = (1/\log_e 2)\log_e\left(\frac{1}{P_1^f}\right)$$
$$w_2 = (1/\log_e 2)\log_e\left(\frac{1}{P_2^f}\right)$$
$$m_1 = (1/\log_e 2)^2 s_1 \log_e\left(\frac{1}{P_1^f}\right)$$
$$m_2 = (1/\log_e 2)^2 s_2 \log_e\left(\frac{1}{P_2^f}\right)$$
$$w_{q,1} = b_1\left(1-(1-w_1/m_1)^q\right)$$
$$w_{q,2} = b_2\left(1-(1-w_2/m_2)^q\right)$$

Using the same approach as in the indexed descriptor file method, we can find the effective value of w and m for a single-level signature file having the same false drop probability as that of the two-level method:

$$w = (1/\log_e 2)\log_e\left(\frac{1}{P^f}\right) = w_1 + w_2$$
$$m = (1/\log_e 2)^2 s \log_e\left(\frac{1}{P^f}\right) = m_1 n_2 + m_2$$

Since the number of signatures in the higher level is n_1, and in the lower level, $n_1 n_2 = n$, the total storage required is:

$$\begin{aligned} M &= n_1 m_1 + n_1 n_2 m_2 \\ &= n_1 n_2 (1/\log_e 2)^2 s \log_e\left(\frac{1}{P_1^f P_2^f}\right) \\ &= n(1/\log_e 2)^2 s \log_e\left(\frac{1}{P^f}\right) \\ &= nm \end{aligned} \tag{9}$$

Thus, if a two-level signature file has a global false drop probability equal to the false drop probability of a single-level signature file, then the above equation reveals a very important fact, i.e., the storage required by both methods is the same. This

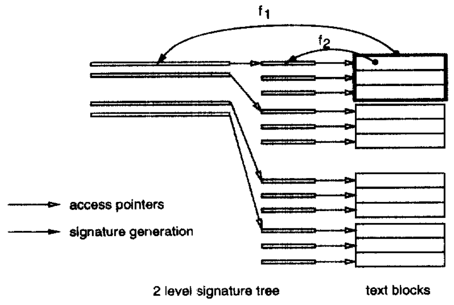


Fig. 3. Two-level superimposed coding.

may be surprising since intuitively the two-level method will have more signature than a single-level method. However, since the two levels are generated independently, a false drop introduced at the first level may be eliminated by the second level and vice versa. Hence, the local false drop probabilities in a two-level method could be made smaller than that of a single-level method, resulting in shorter signatures in the two-level method. As noted in Section III.C, the equations used to obtain this result is based on the false drop probability of unsuccessful searches, so it is fair to emphasize that for successful searches (t>0) the storage overhead of the two-level method (as well as the multilevel method discussed in Section V) is greater than a single-level method for the same false drop probability.

The cost of searching the two-level signature file is A_1+A_2, which is the cost of searching the two levels. A_1 and A_2 are derived in Section V.A, and is not repeated here. In general, the two-level signature file is a special case of the multilevel signature file described in Section V.

F. Partitioned Signature File Method (Fixed-Prefix)

In a partitioned signature file, a segment of the signature is used as the key of the signature and signatures with the same key are grouped into one partition. The common key of the signatures in a partition forms the key of the partition and is stored in a key table [13]. In the fixed-prefix method, a fixed-length prefix of length k from the signature serves as the key of the signature. Fig. 4 illustrates a possible organization for the fixed-prefix method. The key table is small enough to fit into main memory, so it doesn't contribute to the search cost in the analysis. The following equations are true for all partitioned methods [19]:

$$w=\left(1/\log_e 2\right)\log_e\left(1/P^f\right)$$

$$m=\left(1/\log_e 2\right)^2 s\log_e\left(1/P^f\right)$$

$$w_s=m\left(1-\left(1-w/m\right)^s\right)=0.5m$$

$$w_q=m\left(1-\left(1-w/m\right)^q\right)=m\left(1-0.5^{q/s}\right)$$

Unlike the previous two methods, the number of true drops won't have any dramatic effect on the performance of this

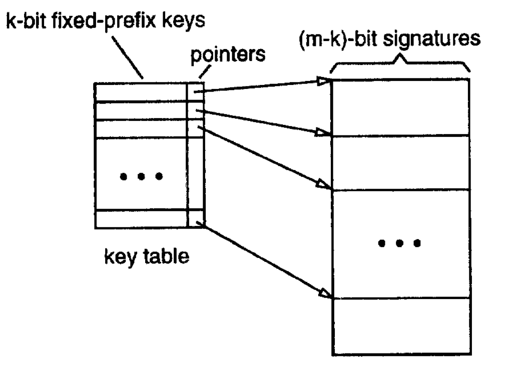


Fig. 4. Fixed-prefix partitioned signature file.

method. This is because the existence of a true drop in a partition only causes that partition to be searched, whereas in the indexed descriptor method (as well as the mutlilevel superimposed coding method described later) the true drop will cause all its ancestor nodes to be searched as well.

The following equations are applicable exclusively to the fixed-prefix method. In the equations, A_1 and A_2 represent, respectively, the number of keys and block signatures searched resulting from false drops. Since all keys in the key table have to be examined, A_1 is the number of keys in the key table. The number of signatures searched in the partitions can be calculated based on the number of partitions searched and the expected size of each partition without distinguishing between true drops or false drops [12]. Hence, the subscripts t and f in A are dropped.

A_2 is obtained directly from [12] while the other parameters are derived in a way similar to the previous model equations.

$$A_1=2^k$$

$$A_2=n\sum_{i=0}^{k}\binom{k}{i}\binom{m-k+i}{w_q}\binom{m}{w_q}^{-1}2^{-k}$$

$$A=A_1+A_2$$

$$C=A/n$$

$$D=\left(\frac{k}{m}w_qA_1+\frac{(m+k)}{m}w_qA_2\right)\Big/\left(nw_q\right)$$

$$=\left(kA_1+(m-k)A_2\right)/(mn)$$

The storage requirement of the fixed-prefix method consists of two parts: storage for the keys and storage for the block signatures. Since there are at most 2^k keys of length k and the keys of the signatures need not be stored explicitly, the storage overhead is: $M=2^k k+(m-k)n$

G. Partitioned Signature File Method (Floating-Key)

The floating-key method is similar to the fixed-prefix method, except that the method for obtaining the keys is more complicated. In the floating-key method, every consecutive, but nonoverlapping, k-substrings of the signature is examined and the substring with the least number of ones is chosen as the key. Thus, a key in this case consists of a k-bit string and the starting position of the key in the signature. Fig. 5 depicts an organization for the floating-key method, where the key table keeps both the key segments and the starting positions of the key segments in the respective signatures.

The equations describing the parameters m, w, w_s, and w_q are the same as in the fixed-prefix method. In the following, the equation describing A_2 and its related parameters, $P(i,j)$ and $\chi(i,j)$, are obtained from [12]. The others are derived in a way similar to the previous model parameters. A_1 is the size of the key table, each entry of which consists of a k-bit field for the key and a $\lfloor m/k \rfloor$-bit field for the starting position of the key.

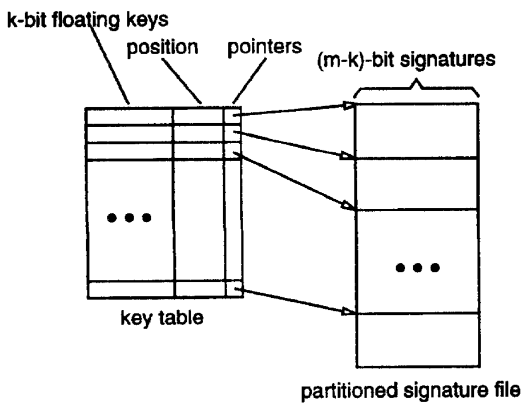


Fig. 5. Floating-key partitioned signature file.

$$P(i,j)=\left(2^k-\sum_{y=0}^{i}\binom{k}{y}\right)^j\left(2^k-\sum_{y=0}^{i}\binom{k}{y}\right)^{\lfloor m/k\rfloor-j-1}2^{m-k\lfloor m/k\rfloor}$$

$$\mathcal{X}(i,j)=\begin{cases}1 & \text{if } P(i,j)n\geq 1\\ P(i,j)n & otherwise\end{cases}$$

$$A_1=2^{k+\log_2\lfloor m/k\rfloor}$$

$$A_2=\frac{n\sum_{i=0}^{k-1}\binom{m-k+i}{w_q}\binom{m}{w_q}^{-1}\binom{k}{i}\sum_{j=0}^{\lfloor m/k\rfloor-1}P(i,j)\chi(i,j)}{\sum_{i=0}^{k-1}\binom{k}{i}\sum_{j=0}^{\lfloor m/k\rfloor-1}P(i,j)\chi(i,j)}$$

$$A=A_1+A_2$$

The signature reduction ratio and the computation reduction ratio can be derived as in the fixed-prefix method, except that there are $\lfloor m/k\rfloor$ key tables (one for each possible key position) and all of them must be searched.

$$C=A/n$$

$$D=(kA_1+(m-k)A_2)/(mn)$$

The storage requirement increases slightly in comparison with that of the fixed-prefix method: $M=\lfloor m/k\rfloor 2^k k+(m-k)n$

IV. Performance

There are many factors affecting the performance. In this paper, two important parameters are studied, and their effects on the performance are examined: the number of distinct keywords per block, s, and the number of blocks containing the query terms, t. The performance measures we obtain are C, D, and M. We fix the values for the other parameters as follows.

$$n=2^{24}$$
$$q=1$$
$$n_1=n_2=2^{12}=4{,}096$$
$$k=15$$
$$b=4$$
$$P^f=n^{-1}=2^{-24}$$
$$P_1^f=P_2^f=\frac{1}{\sqrt{n}}=2^{-12}$$

The following sets of data are used for evaluating the model equations:

	t	s
Set 1	1	5, 10, 20, 40
Set 2	1, 2, 4, 8, 16, 32, 64, 128, 256,	20
	512, 1,024, 2,048, 4,096	

Figs. 6 and 7 are derived from Set 1 and Figs. 8 and 9 from Set 2. The rest of this section discusses the observations we made from the results.

A. Storage Overhead

The storage requirement of each method is evaluated against the storage required by a single-level signature file having the same false drop probability. The partitioned methods require the least amount of storage, since the key of a signature need not be stored explicitly. Neglecting the key table, the fixed-prefix method can reduce the storage by a factor of $(m-k)/m$. For $10 \leq k \leq 20$ and $500 \leq m \leq 2{,}000$, it saves up to 4% of storage compared to the original signature file. The floating-key method will take slightly more storage since the key table is larger than that of the fixed-prefix method.

Since the original storage required by the signature file method is mn bits, we can obtain from (6) the extra storage required by the indexed descriptor method compared to a single-level signature file:

b	overhead
2	100%
4	33%
10	11%
20	5%

We can see that the indexed descriptor method requires a storage space at most twice as much as a single-level signature file.[2] From (9), the two-level method requires no extra storage. Therefore, we conclude that the storage overheads for the partitioned method and the two-level method are about the same, but the indexed-descriptor file is less efficient in storage than the other two methods.

B. Optimal Blocking Factor in the Indexed Descriptor Method

As can be seen from (7) and (8), the signature reduction ratio is dependent upon b, w_q, t, q and indirectly on n. If we fix t and q

2. We only count the space taken up by the signatures and indexes, and ignore the pointers required to link the nodes.

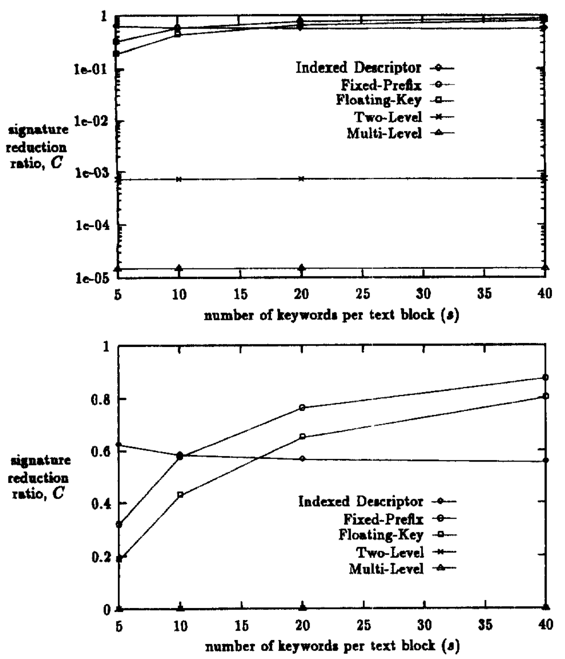


Fig. 6. Signature reduction ratio, *C*, vs. the number of keywords per text block, *s*, in log and linear scales.

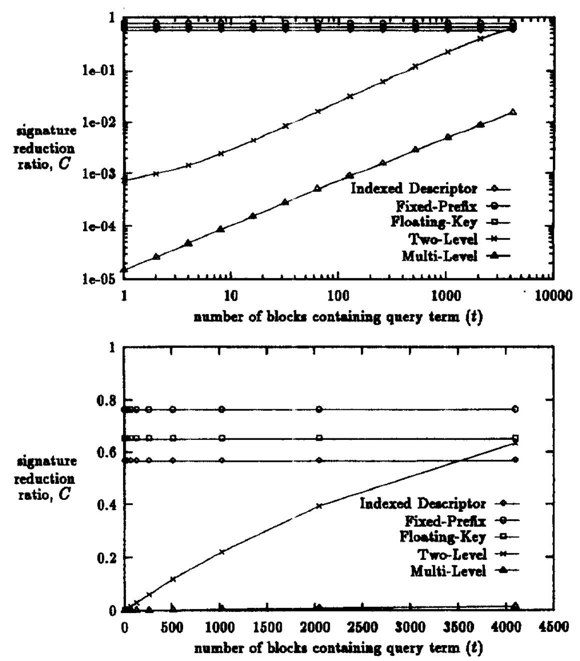


Fig. 8. Signature reduction ratio, *C*, vs. the number of true drops, *t*, in log and linear scales.

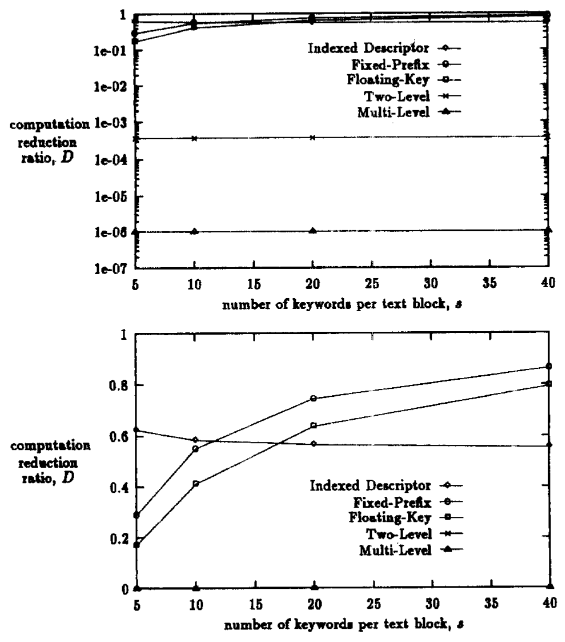


Fig. 7. Computation reduction ratio, *D*, vs. the number of keywords per text block, *s*, in log and linear scales.

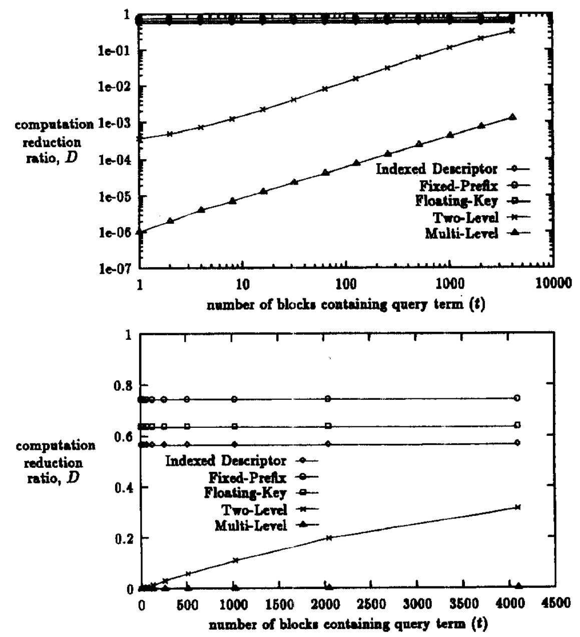


Fig. 9. Computation reduction ratio, *D*, vs. the number of true drops, *t*, in log and linear scales.

to 1, then w_q becomes w. From (4), we know that w is dependent on the false drop probability. In practice, P^f is set to a very small value to minimize the cost of unnecessary retrieval. If we set P^f to $1/n$ (i.e., one false drop is expected in a signature file of n signatures) and n varies from 10^3 to 10^{15}, w would vary from 10 to 50. Fig. 10 shows the signature reduction ratio versus b for w ranging from 10 (when $n = 10^3$) to 50 (when $n = 10^{15}$). The figure suggests that b=3 is optimal for w<30 (when n<10^9) and b=4 is optimal for w≥30 (when n≥10^9). The minimum signature reduction ratio shown in the graph is 0.373 at b=4 and w=50. From the table in the last section, this corresponds to a storage overhead of 33%, which seems rather profitable from a performance point of view.

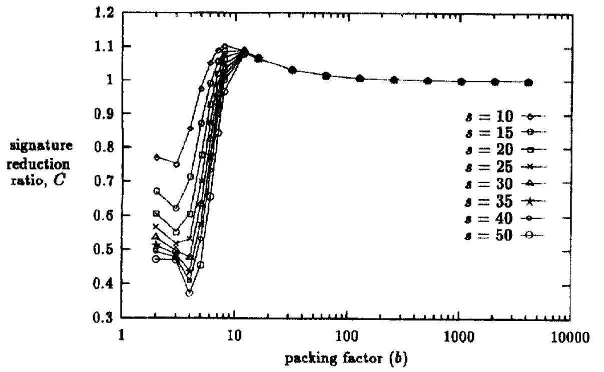


Fig. 10. Signature reduction ratio (C) for various packing factors and numbers of keywords per text block, (s).

C. Optimal Signature Reduction Ratio of the Other Methods

The signature reduction ratio of the two-level method is dependent on t, q, n_1, and P_1^f. If we set P_1^f and P_2^f to be inversely proportional to n_1 and n_2 such that $P_1^f = \sqrt{c}/n_1$, $P_2^f = \sqrt{c}/n_2$, and $P^f = c/n$ with a constant c, then the factors reduce to t, q, and n_1. If we further fix t and q, then the optimal case can be derived by differentiating the signature reduction ratio with respect to n_1. For the case of unsuccessful search (i.e., $t=0$) and $q=1$, the signature reduction ratio, C, becomes

$$n_1\left(1 + n_2 P(s_1, w_1)\right)/n = \left(n_1 + \sqrt{c}n/n_1\right)/n .$$

By differentiating the expression with respect to n_1, the optimal value of n_1 for minimizing the signature reduction ratio is $n_1 = \sqrt{\sqrt{c}n}$. For the parameters we used in our analysis, $c=1$ and $n_1 = n_2 = \sqrt{n} = 2^{12}$.

In the partitioned method, the signature reduction ratio is dependent upon the partition key length, k. Since the number of partitions increases exponentially with k, there is a practical limit on the value of k.

D. Discussion

Some unexpected results can be observed from our analysis. First, the two-level method shows a much better reduction ratio than the other methods. The indexed descriptor method uses a similar structure, but the problem with the way that higher-level signatures are generated is apparent. For instance, if m is 500, then about 250 bits is set in a block signature. Let's say $b=4$. Then, each index node in the next higher level is obtained from superimposing four different block signatures, resulting in setting 468 bits, or 93.8%, of the bits in the a signature at the next higher level. This percentage will increase as the index level goes up. Thus, the signatures at higher levels provide little or no filtering effect.

The fixed-prefix method also has a similar problem. Since k must be kept small in order not to generate too many partitions, the chance that a query signature would specify a bit in the key is small, depending on the ratio between k and m. The floating-key method achieves better performance by minimizing the number of ones in the partition keys. However, for single-term queries, the performance gain seems to be in the neighborhood of 20–30% (see Figs. 7 and 8, and [12]). The main advantage of the partitioned method is its simple file structure, low storage overhead, and that partitions can be accessed directly without the need of following many levels of pointers. Also, its performance would not be drastically affected by the selectivity of the query.

The main feature of the two-level superimposed coding method is that, unlike the indexed descriptor method, signatures in the segment level is obtained directly from the text blocks. In our analytical results, the signature reduction ratio, C, with $t=1$ is approximately $2/n_1=0.00049$ or 0.049%. However, the signature length in the segment level is very large compared to the signatures in the bottom level. Thus, even though there are only 4,096 signatures in the first level in contrast to 2^{24} signatures in the second level, the first level still occupies one-half of the storage taken by the whole signature tree (in (9)), the first term in the RHS accounts for the size of the index signatures). However, if we use a transposed organization for the signatures [11], the computation reduction ratio, D, with $t=1$, becomes 0.00037 or 0.037% which is in the same order as that of the signature reduction ratio.

Second, the performance of the two-level method is dependent on t. In the analysis, the segment size is 4,096. Thus if there are 4,096 independent blocks containing the query words, the performance degenerates to the level of (or even worse than) the other methods. The extreme case where t is very large is analyzed in the next section. For the parameters used in our analysis, the minimum gain in terms of computation reduction ratio is still 50% compared to a single-level signature file. On the other hand, the indexed-descriptor method is less vulnerable to the value of t, because b is small and thus t has to be very large to produce the degenerated case. The partitioned method is best in terms of the independency of the signature reduction ratio on t, since all signatures containing the same words tend to cluster in the same set of partitions.

Third, the number of distinct keywords per text block, s, affects the reduction ratio of the partitioned methods significantly, while the other methods are hardly affected. The reason is that when s decreases, w/m, the probability of having a bit set to one in a word signature, increases according to (4) and (5). This will increase the query signature's weight and thus the chance of having some bits specified in the query signature's key. For example, with $k=15$ in the fixed-prefix method, the ratio kw/m equals to 0.52 and 1.04 for $s=20$ and 10, respectively. In other words, the chance of having a one specified in the query signature's key is doubled in the latter case. From Fig. 7, it can be seen that the reduction ratio improves (i.e., decreases) as s decreases.

Finally, the performance of the indexed descriptor method and the partitioned methods are at the same level, and the floating key scheme is better than the fixed-prefix method.

V. Multilevel Superimposed Coding Method

Observing the excellent performance of the two-level method, we propose a multilevel superimposed coding method which performs even better than the two-level method and show that the two-level method is indeed suboptimal.

The multilevel superimposed coding method is an extension of the two-level superimposed coding method. That is, it can have more than two levels. A signature at a nonleaf node is formed, by superimposed coding, from all the text blocks indexed by the subtree of which the signature is the root. The file structure is depicted in Fig. 11. Note that a word has to generate different word signatures of different lengths for different levels.

Multilevel signature file has been investigated in the literature [1]. There are two problems associated with multilevel methods. First, as more text blocks are included in the generation of higher-level signatures, the bit density of the signatures will increase if special precaution is not taken. In the method proposed below, this problem doesn't exist, since signatures at higher levels have correspondingly longer lengths. Second, combinatorial errors will occur, if a query asks for text blocks containing a conjunction of query terms. Higher-level signatures will satisfy the query even if the text blocks indexed by them don't contain the query terms in the *same* block. This problem is more severe for relational databases (or any record-based databases) since conjunctive query conditions must be evaluated on the same tuple. However, the problem is less critical for text retrieval since a document typically generates a large number of signatures, and, a user is typically interested in whether or not the query terms exist in the documents regardless of the exact occurrence of the terms in the documents. In other words, the conjunctive condition is evaluated over the entire document instead of a single text block. To reduce the combinatorial problem further, a method was proposed to generate signatures using combinations of keywords [1]. This technique doesn't affect our analysis, since it only increases the effective number of distinct keywords in a block (i.e., a combination of keyword is considered as a new unique keyword).

A. Model Equations

A multilevel signature file is a forest of b-nary trees with every node, except leaf nodes, in the structure having b child nodes. The number of levels in the structure is h. To simplify the analysis, it is assumed that the trees are complete b-ary trees of level h. The relationships between n, b, and h are: $n = b^h$.

We denote a local parameter representing the value of some global parameter p at level i as p_i. For instance, at the ith level,

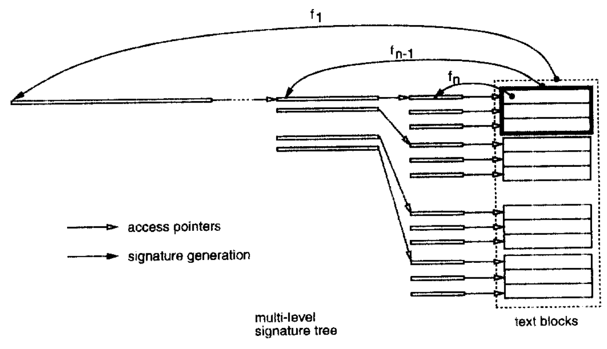


Fig. 11. Multilevel signature file.

the local false drop probability is represented by P_i^f. To further simplify the analysis, it is assumed that the local false drop probability is the same for every level. The relationship between the global and local false drop probabilities is:

$$P_i^f = P_j^f \qquad \forall i,j \tag{10}$$

$$P^f = \frac{\prod_{i=1}^{h}\left(bP_i^f\right)}{n} = \prod_{i=1}^{h} P_i^f \tag{11}$$

$$P_i^f = \left(P^f\right)^{1/h} \tag{12}$$

Equation (12) can be derived from (10) and (11). In the following equations, w, m, s, w_s, and w_q are the parameters for a single-level method. The approximate expressions for w_q and $w_{q,i}$ are meaningful in the range of $(wq)/m \le 1$ and $(w_i q)/m_i \le 1$, respectively. This approximation is reasonable because w and w_i are much less than m and m_i, respectively.

$$w = (1/\log_e 2)\log_e\left(1/P^f\right)$$
$$m = (1/\log_e 2)^2 s \log_e\left(1/P^f\right)$$
$$w_s = 0.5m$$
$$w_q = m\left(1-(1-w/m)^q\right) \simeq wq$$
$$s_i = sb^{h-i}$$
$$w_i = (1/\log_e 2)\ \log_e\left(1/P_i^f\right) = w/h$$
$$m_i = (1/\log_e 2)^2 s_i\ \log_e\left(1/P_i^f\right) = mb^{h-i}/h$$
$$w_{s,i} = 0.5m_i$$
$$w_{q,i} = m_i\left(1-(1-w_i/m_i)^q\right) \simeq w_i q = w_q/h$$
$$w_{q=1,i} = w_i = w/h$$

The required storage (in bits) of the mulitlevel method, M, is:

$$M = \sum_{i=1}^{h} b^i m_i = nm$$

That is, if the global false drop probabilities of the single-level and the multilevel method are the same, the required storage is the same for both methods. The same conclusion has been observed in the two-level method, and the same discussion applies to the multilevel method.

The search performance of the multilevel method can be derived as follows. Let

t = number of blocks containing the query terms,
b = packing factor = number of signatures at the first level,
$w_{q,i}$ = number of bits set in the query signature at level i.

We want to obtain A, the total number of signatures searched in the multilevel signature tree in order to answer a query. At the first level, all the b signatures have to be searched. Let

A_i = number of signatures searched at level i,

$A_{T,i}$ = number of signatures matched at level i due to true drops,
$A_{F,i}$ = number of signatures matched at level i due to false drops.

We have,

$$A_i = \begin{cases} b & i = 1 \\ (A_{T,i-1} + A_{F,i-1})b & i > 1 \end{cases}$$

$$A_{T,i-1} = b^i\left(1 - \left(1 - \frac{1}{b^i}\right)^t\right)$$

$$A_{F,i} = \begin{cases} b(1 - 1/b)^t (0.5)^{w_{q,i}} & i = 1 \\ (A_i - A_{T,i})(0.5)^{w_{q,i}} & i > 1 \end{cases}$$

where $(0.5)^{w_{q,i}}$ is the probability of a match due to false drops at level i. Note that for optimality, about half of the bits in a block signature should be 1s.

A = total number of signatures searched at all levels

$$= \sum_{i=1}^{h} A_i$$

The other measures, C and D can be obtained as follows:

$$C = \frac{A}{n}$$

$$D = \frac{A(w_q/h)}{w_q n} = \frac{C}{h}$$

$$D_{t=n} = h^{-1}$$

where w_q is the number of bits set in a query signature in the single-level method.

B. Optimal Performance of the Multilevel Method

To derive the optimal number of levels, we assume for simplicity that $t = 0$. Then,

$$A_{T,i} = 0$$

$$A_{F,i} = \begin{cases} b(0.5)^{w_{q,i}} & i = 1 \\ A_i(0.5)^{w_{q,i}} & i > 1 \end{cases}$$

$$= A_i(0.5)^{w_{q,i}}$$

$$A_i = \begin{cases} b & i = 1 \\ A_{F,i-1}b & i > 1 \end{cases}$$

The following closed-form formulas for $A_{F,i}$ and A can be obtained as shown below:

$$A_{F,i} = b^i\left(0.5^{w_{q,i}}\right)^i$$

$$= \left(0.5^{w_{q,i}}b\right)^i$$

$$A = \sum_{i=1}^{h} A_i$$

$$= \sum_{i=1}^{h} b\left(0.5^{w_{q,i}}b\right)^{i-1}$$

$$= b\sum_{i=0}^{h-1}\left(0.5^{w_{q,i}}b\right)^i$$

$$= b\frac{1 - \left(0.5^{w_{q,i}}b\right)^h}{1 - \left(0.5^{w_{q,i}}b\right)}$$

Since $b^h = n$, $h = \log n / \log b$, and $w_q = (w_{q,i})h$,

$$A = b\frac{1 - 0.5^{w_{q,i}}n}{1 - 0.5^{w_{q,i}}b}$$

$$= (1 - zn)\frac{b}{1 - b^{1+1/\log_z n}} \text{, where } z = 0.5^{w_q}$$

It can be shown easily that A is monotonically increasing with respect to b. Thus, the value of b should be as small as possible in order to minimize A, which means that b=2 and that the optimal number of levels is $\log_2 n$.

C. Performance

We evaluate the performance model of the multilevel method with the following parameter values. The number of levels, h, is set using the optimal value b=2:

$$n = 2^{24} \textit{ blocks}$$
$$b = 2$$
$$h = 24$$
$$P^f = 1/n$$

The model equations are evaluated with the same set of data as given in Section IV. The results are displayed in Figs. 6–9. We can see that the optimal multilevel superimposed coding method shows a superior performance than the two-level and all other methods.

A simulation is performed to verify the analytical results of the multilevel method. A text file containing random words is generated and multiple-level signature trees are generated with packing factors of two and four. For practical reasons, the number of signatures used in the simulation is 2^{14}. Then random queries are generated. Figs. 12 and 13 show the computation reduction ratios for successful searches (t=1; q=1,2,3,4,5; b=2,4) and unsuccessful searches (t=0; q=1,2,3,4,5; b=2,4), respectively. In both cases, the analytical results are almost identical with the simulation results. It is shown that the performance for b=2, which is optimal, is better than that for b=4.

D. Other Issues

D.1. Storage Overhead and False Drop Probability

If the same false drop probability is used, the multilevel method requires the same storage space as the single-level

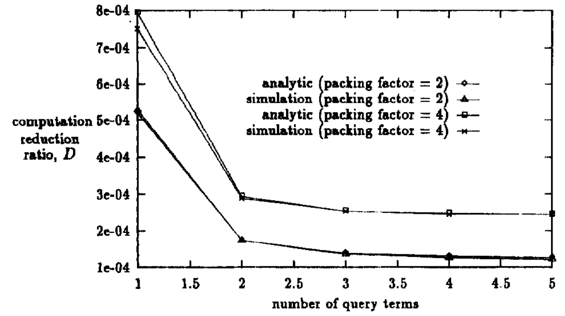


Fig. 12. Computation reduction ratio, D, vs. number of query terms in a multilevel signature file (number of true drops = 1).

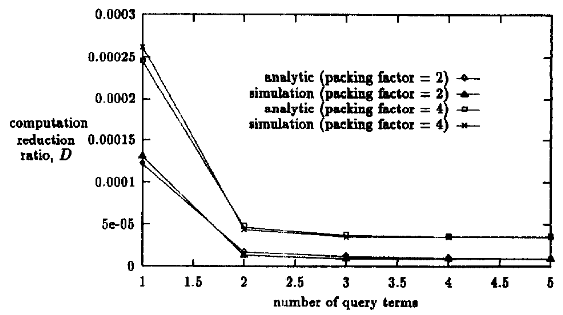


Fig. 13. Computation reduction ratio, D, vs. number of query terms in a multilevel signature file (number of true drops = 0).

method. For signatures at the ith level, the required storage is $b^i m_i = mn/h$. In other words, each level consumes the same amount of storage. We can observe that for each level up the hierarchy, the length of a signature increases b times and the number of signatures in a level decreases $1/b$ times. Thus, the required storage for each level remains constant.

The extreme case in which all text blocks contain a query word shows that the number of bits compared at all levels is inversely proportional to the number of levels of the structure, h. For example, with two levels (h=2,$b \gg 1$), $D_{t=n}$=0.5, or one-half of the original total search space. With h=5 and $b \gg 1$ (this is a reasonable assumption with large data base), it reduces to one-fifth of the total search space. Thus, we can see that the two-level method is not optimal.

Next we consider the effect of the false drop probability P^f on the storage required. Consider two cases:

1) $P^f = 1/n = 1/b^h$, and
2) $P^{f'} = 1/b^x$.

The ratio between the storage requirements for these two cases is $nm'/nm = log(b^x)/log(b^h) = x/h$. That is, if case 2 is to take half as much storage as case 1, x has to be $0.5h$. This means that the false drop probability for case 2 would be $b^{0.5h} = \sqrt{n}$, resulting in an intolerable number of false drops. We can see from this example that the false drop probability is not a significant factor affecting the storage. Thus, it is not advisable to compromise false drop probability for lower storage overhead.

D.2. Insertions and Deletions

When a text block is inserted in the multilevel method, h different signatures are generated, one for each level, and superimposed into the corresponding index signatures. The superimposition requires only ($h-1$) simple updates to the index hierarchy (i.e., the nonleaf nodes). Note that, when a new record is inserted, the total number of bits set by the new record on all levels is the same as that for the single-level method. However, since the bits are spread on all levels, the number of disk access is higher for the multilevel method.

If a text block is deleted, the h index signatures along the path from the deleted block to the root node must be *regenerated*. This requires $b(b^h-1)/(b-1)$ signature generations, which is unacceptable for large databases. To alleviate the problem, when a block is deleted, only the signature at the leaf level is removed. This simplification increases the signature reduction ratio C by $hb/n = b(b/n)\log_b n$, when the words inside the removed block are used as query words. Since the increase is small, this simple method is a reasonable compromise.

D.3. Combinatorial Errors

It has been shown that combinatorial errors result in false drops when a number of records are combined to form a block or a segment signature in a two-level signature file method [18]. The false drops due to combinatorial errors are more of a problem for queries with many search terms. The same applies to the multilevel method. Our model does not take the effect of combinatorial errors into account for calculating the number of signatures searched due to false drops. As a result, our method underestimates the number of signatures searched due to false drops for multiterm queries. However, since the blocking factor plays an important role in combinatorial errors, our method, where the optimal blocking factor is two, minimizes the effects of combinatorial errors. This can also be observed from Fig. 13, where the simulation results closely match the analytical results for queries with multiple query terms. Moreover, these false drops are concentrated over the first few levels of the signature hierarchy where the number of constituent records for a signature is large. At higher (leaf) levels, signatures are generated from fewer blocks, hence lowering the effect of combinatorial errors.

VI. Conclusion

In this paper, we compare the performance of three basic search methods for signature files using a uniform performance model. We further propose a multilevel superimposed coding method which shows superior performance over the other methods while consuming the same amount of storage. In the analysis of the methods, we introduce the notion of global and local false drop probabilities and distinguish between searches due to false drops and true drops. For the multilevel signature file, we obtain the optimal number of levels and show that the optimal point is when the degree (packing factor) of the tree is two. Thus, for a signature file with n signatures, the optimal number of levels is $\log_2 n$. Since the number of levels grows with the database size, the multilevel level is particularly suitable for indexing large databases. The analysis performed on these methods also reveals other important characteristics of the methods, which are otherwise difficult to observe. For instance, the same storage overhead is incurred for the single-level, two-level, and multilevel method to yield the same false drop probability for unsuccessful searched, and, if the false drop probability is set to $1/n$, as is in our analysis, the number of bits set by each keyword in a signature is one.

The partitioned signature files are not as competitive as the other methods. However, they are attractive because of their simple file structures, low storage overheads, and the direct access to the partitions (assuming the key table is small enough to fit into main memory). Hence, it can be combined with the multilevel superimposed coding method easily (or for that matter any other methods). That is, the partitions themselves can be organized using a multilevel structure. The performance of this "hybrid" method is an interesting topic for further research. Another interesting research problem is the relaxation of (10). It is quite possible that the search performance of the multilevel method would be different if local false drop probabilities are not the same for every level. In this case, further investigation is needed to find out the best way of assigning false drop probabilities to each level in the tree. Deletion in the multilevel method is problematic. Schemes for regenerating the signatures only for a small subtree containing the deleted text block and an analysis incorporating the cost of periodic reorganization of the tree structure are interesting research problems.

Acknowledgments

The authors would like to thank the reviewers for their careful reviews. Their comments greatly improved the clarity and presentation of the paper.

References

[1] W.W. Chang and H.J. Schek, "A signature access method for the Starburst database system," *Proc. 15th Int'l Conf. Very Large Data Bases*, Amsterdam, The Netherlands, Aug. 1989, pp. 145–153.

[2] S. Christodoulakis and C. Faloutsos, "Design considerations for a message file server," *IEEE Trans. Software Eng.*, vol 10, no. 2, pp. 201–210, Mar. 1984.

[3] S. Christodoulakis, M. Theodoridou, F. Ho, M. Papa, and A. Pathria, "Multimedia document presentation, information extraction and document formation in MINOS: A model and a system," *ACM Trans. Office Information Systems*, vol. 4, no. 4, pp. 345–383, Oct. 1986.

[4] U. Deppisch, "S-Tree: A dynamic balanced signature index for office retrieval," *Proc. ACM Conf. Research and Development Information Retrieval*, Pisa, Italy, pp. 77–87, Sept. 1986.

[5] C. Faloutsos, "Access methods for text," *ACM Computing Surveys*, vol. 17, no. 1, pp. 49–74 Mar. 1985.

[6] C. Faloutsos, "Signature-based text retrieval methods: A survey," *Data Engineering Bulletin*, vol. 13, no. 1, pp. 25–32, Mar. 1990.

[7] C. Faloutsos, "Signature files: Design and performance comparison of some signature extraction methods," *Proc. 1985 SIGMOD Conf.*, Austin, Tex., pp. 63–82, May, 1985.

[8] C. Faloutsos and S. Christodoulakis, "Signature files: An access method for documents and its analytical performance evaluation," *ACM Trans. Office Information Systems*, vol. 2, no. 4, pp. 267–288, Oct. 1984.

[9] C. Faloutsos and S. Christodoulakis, "Description and performance analysis of signature file methods for office filing," *ACM Trans. Office Information Systems*, vol. 5, no. 3, pp. 237–257, July 1987.

[10] C. Faloutsos and S. Christodoulakis, "Optimal signature extraction and information loss," *ACM Trans. Database Systems*, vol. 12, no. 3, pp. 395–428, Sept. 1987.

[11] D.L. Lee, "A word-parallel, bit-serial signature processor for superimposed coding," *Proc. Second Int'l Conf. Data Engineering*, Los Angeles, pp. 352–359, Feb. 1986.

[12] D.L. Lee and C.-W. Leng, "Partitioned signature files: Design issues and performance evaluation," *ACM Trans Information Systems*, vol. 7, no. 2, pp. 158–180, Apr. 1989.

[13] D.L. Lee and C.-W. Leng, "A partitioned signature file structure for multiattribute and text retrieval," *Proc. Sixth Int'l Conf. Data Engineering*, pp. 389–397, Los Angeles, Feb. 1990.

[14] C.W. Leng and D.L. Lee, "Optimal weight assignment for signature generation," *ACM Trans Database Systems*, vol. 17, no. 2, pp. 346–373, June, 1992.

[15] D.J. Lipman and W.R. Pearson, "Rapid and sensitive protein similarity searches," *Science*, vol. 227, pp. 1,435–1,441, Mar., 1985.

[16] J.L. Pfaltz, W.J. Berman, and E.M. Cagley, "Partial match retrieval using indexed descriptor files," *Comm. ACM*, vol. 23, no. 9, pp. 522–528, Sept., 1980.

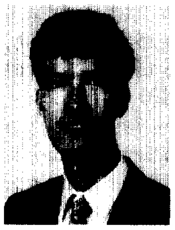
[17] C.S. Roberts, "Partial-match retrieval via method of superimposed codes," *Proc. IEEE*, vol. 67, no. 12, pp. 1,624–1,642, Dec. 1979.

[18] R. Sacks-Davis, A. Kent, and K. Ramamohanarao, "Multikey access methods based on superimposed coding techniques," *ACM Trans. Database Systems*, vol. 12, no. 4, pp. 655–696, Dec. 1987.

[19] R. Sacks-Davis, and K. Ramamohanarao, "A two-level superimposed coding scheme for partial match retrieval," *Information Systems*, vol. 8, no. 4, pp. 273–280, 1983.

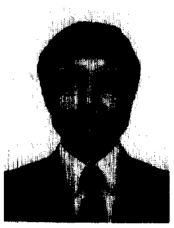
[20] C. Stanfill and B. Kahle, "Parallel free-text search on the connection machine system," *Comm. ACM*, vol. 29, no. 12, pp. 1,229–1,239, Dec. 1986.

[21] B. Yuwono and D.L. Lee, "A backend text retrieval machine for signature-based document ranking," *Proc. Int'l Conf. Computing and Information (ICCI '91)*, pp. 288–297, Ottawa, Canada, May 1991.



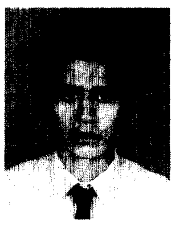
Dik Lun Lee received his BSc in electronics from the Chinese University of Hong Kong in 1979, and his MSc and PhD degrees in computer science from University of Toronto in 1982 and 1985, respectively. An associate professor of computer and information science at Ohio State University, his current research is in document retrieval and management, knowledge base systems, and object-oriented and heterogeneous database systems.

A member of the ACM, the IEEE, and the IEEE Computer Society, Dr. Lee was the guest editor of the special issue on document processing for *IEEE Data Engineering Bulletin* in 1990 and the co-guest-editor of the special track on artificial intelligence techniques for text-based information systems for *IEEE Expert*. He served on the Program Committee of the International Conference on Data Engineering from 1989 to 1994 and was an ACM lecturer from 1991 to 1992.



Young Man Kim received the BS degree in mechanical engineering from Seoul National University, Seoul, South Korea, in 1980 and the MS degree in mechanical engineering from the Korea Advanced Institute of Science and Technology, Seoul, in 1982. After five years of work experience, he entered the Department of Computer and Information Science at Ohio State University and received the MS and PhD degrees in 1988 and 1992, respectively.

Dr. Kim works at Mitsubishi Materials Corp., Japan. His research interests include signature files, image processing, parallel processing, distributed systems, communication networks and protocols, and programming languages.



Gaurav Patel is pursuing a PhD degree in computer and information science at Ohio State University. He received his BE in computer engineering from Gujarat University in India in 1989 and his MS in computer and information science from Ohio State University in 1991. His current research interests include document and information retrieval systems, heterogeneous database systems, and object-orientation.