

---

# Optimization of Signature File Parameters for Databases with Varying Record Lengths

SEYIT KOÇBERBER<sup>1</sup>, FAZLI CAN<sup>2</sup> AND JON M. PATTON<sup>3</sup>

<sup>1</sup>*Department of Computer Engineering and Information Science, Bilkent University, Bilkent, 06533 Ankara, Turkey*

<sup>2</sup>*Department of Systems Analysis, Miami University, Oxford, OH 45056, USA*

<sup>3</sup>*Department of Applied Technologies, Miami University, Oxford, OH 45056, USA*

*Email: seyit@bilkent.edu.tr*

---

**For signature files we propose a new false drop estimation method for databases with varying record lengths. Our approach provides more accurate estimation of the number of false drops by considering the lengths of individual records instead of using the average number of terms per record. In signature file processing, accurate estimation of the number of false drops is essential to obtain a more accurate signature file and therefore to obtain a better (query) response time. With a formal proof we show that under certain conditions the number of false drops estimated by considering the average record length is less than or equal to the precise ‘expected’ estimation which is based on the individual record lengths. The experiments with real data show that the proposed method accurately estimates the number of false drops and the actual response time. Depending on the space overhead, our approach obtains up to 33% and 20% response time improvements for the conventional sequential and new efficient multiframe signature file methods, respectively.**

*Received January 10, 1997; revised February 16, 1999*

---

## 1. INTRODUCTION

The growing capacities of data storage devices enable the storage of very large multimedia databases containing formatted and unformatted data, such as text, voice and image. The queries for such multimedia databases contain many search conditions based on various media stored in the database [1, 2] which causes an increase in the number of query terms. Since signature files provide compact representation for today’s large databases [3] they are used to access both formatted and unformatted data via search queries. In particular, conjunctive Boolean queries with many terms can be evaluated efficiently by using signature files [4–11]. We note that today’s databases require multiterm queries since queries with one or an inadequate number of terms do not provide the necessary record selectivity.

In signature files, the content of a *record* (the term ‘record’ is used to refer to any kind of data) is encoded into a bit string called a *record signature*. These record signatures are stored in a separate file called the *signature file*. Several signature generation and signature file methods have been proposed to obtain a desirable response time and space overhead. A survey of these signature file methods can be found in Faloutsos [5] and Aktug and Can [7]. In this study, we disregard other signature generation schemes such as word signatures [12] and consider only widely used superimposed

signatures and conjunctive queries. In the superimposed signature file method, each attribute of a record (*term*) which describes the record is hashed into a bit string of size  $F$  by setting  $S$  bits to ‘1’ (on-bit) where  $S \ll F$ . Record signatures are obtained by superimposing (bitwise *ORing*) the signatures of record terms.

To process a query with signature files, a signature is produced in the same way as document signatures. This query signature is compared with the record signatures. If a record contains all of the query terms, i.e. if the record matches the query, its signature will have on-bits in the corresponding bit positions of all on-bits of the query signature. Therefore, the records whose signatures contain at least one ‘0’ bit (off-bit) in the corresponding positions of on-bits of the query signature definitely do not match the query.

In bit-sliced signature files (BSSF), to retrieve the record signature bits corresponding to a bit position without retrieving other bits, the signature file is vertically partitioned and the bits of a vertical partition (a bit slice) are stored sequentially [13]. For query evaluation, only the bit slices corresponding to the on-bits of the query signature are retrieved and bitwise *ANDed*. Thus, BSSF improves performance by reducing the amount of data to be read and processed.

Since signatures are approximate representations, some records may pass the signature file processing phase

although they do not match the query. Such records are called *false drop* records and they must be accessed and eliminated by using the actual query terms. Therefore, the performance of a signature file method is affected by the number of false drop records ( $FD$ ). If  $FD$  can be estimated accurately, signature file parameters, like  $F$  and  $S$ , can be optimized to obtain a better response time [6, 10, 11, 14].

For a given signature size ( $F$ ) and the number of distinct terms in a record ( $D$ ) the false drop probability ( $fd$ ) and hence  $FD$ , is minimized when half of a record signature bits are on-bits (we call this the optimality condition) [13, 15]. (In Christodoulakis and Faloutsos [15] the trade-off between two mutually competing issues: (a) false drop minimization, and (b) storage utilization efficiency maximization were inspected in detail.) However, generally, databases used in information retrieval (IR) contain records with *varying lengths* (we will use the phrase '*varying record length*' to mean that records may contain different numbers of distinct terms). If the same  $F$  and  $S$  values are used for all records of a database, the signatures of the records with many terms will contain more '1's than the optimality condition requires. This increases  $FD$  and consequently reduces the performance.

Christodoulakis and Faloutsos suggest dividing a record into blocks that contain equal numbers of distinct terms and producing a separate signature for each block [15]. However, the numbers of '1's in block signatures expose a normal distribution and there may be block signatures containing non-optimal numbers of '1's. Leng and Lee call this the fixed size block (FSB) method and they propose the fixed weight block (FWB) method as an alternative [16]. In FWB, instead of controlling the number of terms in a block, the numbers of '1's in a block signature are controlled [16].

FSB and FWB obtain lower false drop probabilities than the sequential signature files (SSF) that involve no blocking [16, 17]. However, with blocking record level search and retrieval operations become complex. For example, the terms of a record that matches a multiterm conjunctive query may be distributed into more than one block. Therefore, for a multiterm query, to determine the relevance of a record the matching status of all of its blocks must be considered and this involves additional costs.

The FSB and FWB methods are extensions of SSF and their use in practice involves similar difficulties if the query terms are distributed into more than one block of a matching record. Similar block assumption related problems exist in various signature file methods. To alleviate these problems, usually, block level matching or single term queries are considered in false drop analysis and performance estimations for signature files. This creates an unrealistic environment since the records of an unformatted database contain varying numbers of terms (i.e. require different numbers of blocks) and user queries usually involve more than one term in real IR applications.

To address the problems mentioned here we propose a new false drop estimation approach, the individual false drop estimation method (IFD), which considers databases with *varying record lengths* and multiterm queries without

dividing a record into blocks. In IFD, each record has a false drop probability which is computed by considering the number of distinct terms in the record. The expected number of false drops for the database,  $FD$ , is computed by adding the individual false drop probabilities of all records.

In signature files, the estimated  $FD$  value is used in the optimization of the signature file parameters. Therefore, accurate estimation of  $FD$  provides better estimation of signature file parameters which in turn provides superior performance that can be achieved in real applications. The IFD concept is general purpose and can be used in association with various signature file methods. In this paper we provide two example cases for this purpose and study the performance of IFD in the conventional sequential signature file (SSF) method and a new vertical partitioning environment, the multiframe signature file (MFSF) method, that we introduced in our recent study [10, 11, 18]. For this purpose we developed a test environment and implemented the SSF and MFSF methods. We extended these methods to use IFD and tested their performance with real data. The experiments show that IFD improves the performance of the inspected methods by reducing the observed  $FD$  and the (query) response time. (Further experiments with similar results involving a generalized frame sliced signature file approach are reported in Kocberber [18].)

The organization of the paper is as follows. In Section 2, the conventional  $FD$  estimation method and the proposed  $FD$  estimation method, IFD, are explained. Section 3 explains the test environment used in the experiments. In Sections 4 and 5, we apply IFD to the SSF and MFSF methods, respectively, and measure the performance improvements obtained by IFD experimentally with real data. Section 6 provides the conclusion. In the Appendix we provide a formal proof which shows that under certain conditions the number of false drop records ( $FD$ ) estimated by considering the average number of terms in the records is less than or equal to the  $FD$  estimated by considering individual  $D$  values of the records.

## 2. ESTIMATING THE NUMBER OF FALSE DROPS

Due to hashing and superimposition operations used in obtaining signatures, the signature of a record that does not satisfy all query terms may match the query signature. All matching records, including *false drops*, must be accessed and compared with the query after processing the signature file to make sure that they really contain the query terms. Consequently, to estimate the response time, we need to estimate  $FD$  accurately.

To better illustrate signature extraction, query processing with superimposed signatures and the false drop (match) concept, an example is provided in Figure 1. An intuitive observation in Figure 1 is that the false match probability of R2 is greater than the false match probability of R1 for the same set of queries since the signature of R2 contains more on-bits than the signature of R1. The false match probability of an on-bit of a query signature increases as the number of on-bits in a record signature increases. Note that a record

Record 1 (R1) = { computer, information}			Record 2 (R2) = { computer, information, signature}		
<b>Term</b>	<b>Term Signature</b>		<b>Term</b>	<b>Term Signature</b>	
computer	0 1 0 0 0 1 0 0 1 0		computer	0 1 0 0 0 1 0 0 1 0	
information	0 0 0 0 1 0 0 1 0 1		information	0 0 0 0 1 0 0 1 0 1	
<hr/>			<hr/>		
<b>Signature of R1</b>	0 1 0 0 1 1 0 1 1 1		<b>Signature of R2</b>	1 1 0 0 1 1 1 1 1 1	
<b>Query</b>	<b>Query Signature</b>	<b>Result</b>	<b>Query</b>	<b>Query Signature</b>	<b>Result</b>
access	0 1 0 0 0 1 0 0 0 1	False match	access	0 1 0 0 0 1 0 0 0 1	False match
information	0 0 0 0 1 0 0 1 0 1	True match	information	0 0 0 0 1 0 0 1 0 1	True match
retrieval	1 0 0 0 1 0 1 0 0 0	No match	retrieval	1 0 0 0 1 0 1 0 0 0	False match

FIGURE 1. Signature extraction and query evaluation with superimposed signatures ( $F = 10$ ,  $S = 3$ ).

signature with only on-bits matches all queries irrespective of the query terms. The ratio of on-bits to the number of total bits in a record signature is called *on-bit density* ( $op$ ). Parameters affecting the on-bit density are the length of the record signature ( $F$ ), the number of distinct terms in the record ( $D$ ) and the number of bits set to '1' by each term ( $S$ ). Note that in our case the 1s of a term signature do not overlap. There exists yet one more SSF (FSB) model that allows the overlapping of the 1s at the term signature level [15, 19].

Smaller  $S$  values provide lower on-bit densities in the record signatures. On the other hand, reducing  $S$  also reduces the number of on-bits in a query signature. A query signature with fewer on-bits matches more record signatures accidentally. As a result, while the on-bit density decreases for decreasing  $S$  value, the false match probability of a record signature and query signature (*false drop probability*) may increase. (For easy reference the definitions of the important symbols used in this paper and the meanings of the frequently used method acronyms are provided in Table 1.)

### 2.1. Using average number of terms per record in estimating $FD$

A record signature qualifies a query accidentally if the record does not contain some query terms and all on-bits of the query signature were also set by the terms of the record. Since there will be more on-bits in the query signatures of queries with more terms, the false drop probability will decrease for such queries. An exact formula was derived in Roberts [13] to compute the false drop probability of a particular record with  $D$  terms for a  $t$  ( $t > 0$ ) term query. However, the following approximate formula can be used to estimate the false drop probability of a record with  $D$  distinct terms due to its simplicity [13].

$$fd(F, S, D, t) = (1 - (1 - S/F)^D)^{W(Q)_t} \quad (1)$$

where  $W(Q)_t$  is the expected number of on-bits in the signature of a  $t$  term query (*query weight*) and it is computed as follows:

$$W(Q)_t = F \cdot (1 - (1 - S/F)^t). \quad (2)$$

These approximations are valid for large  $F$  values and relatively small values of  $S$ ,  $D$  and  $t$  and they give close results to the exact formula [13].

The false drop probability,  $fd$ , for the whole database is defined as

$$\text{false drop probability} = \frac{\text{number of false matches (drops)}}{N - \text{number of true matches}}.$$

By assuming the number of true matches will be negligible,  $FD$  is computed by multiplying the false match probability of a record by the number of records in the database ( $N$ ) as follows [13]:

$$FD = N \cdot fd. \quad (3)$$

Since  $fd$  is computed for a specific  $D$  value, Equation (3) can be used safely for the databases whose records contain exactly  $D$  terms. In databases containing records with varying numbers of distinct terms, an average  $fd$  value is obtained by using the average number of distinct terms per record,  $D_{\text{avg}}$ , instead of  $D$  in Equation (1) [4, 6, 10, 11, 13, 14]. We call this approach the *average false drop estimation method* (AFD).

The FSB [17] and FWB [18] methods solve the problem caused by the variation in  $D$  values of the records by dividing long records into blocks.

### 2.2. Proposed false drop estimation method: IFD

For databases with records having varying numbers of distinct terms, each record may have a different  $D$  and consequently a different  $fd$  value. Therefore, the 'expected' number of false drops for a database with  $N$  records can be computed precisely by adding the individual false drop probabilities of the records as follows:

$$FD = \sum_{r=1}^N fd \text{ of record } r = \sum_{r=1}^N \left( 1 - \left( 1 - \frac{S}{F} \right)^{D_r} \right)^{W(Q)_t} \quad (4)$$

where  $D_r$  is the number of distinct terms in the  $r$ th record. The records containing equal numbers of distinct terms, i.e.

**TABLE 1.** Definition of abbreviations.

## A. Definition of important symbols (related method).

Symbol	Definition
$f$	Number of frames (MFSF)
$fd$	Expected false drop probability
$op$	Ratio of on-bits to the number of total bits in a record signature
$p$	Number of partitions
$t$	Number of query terms ( $1 \leq t \leq t_{\max}$ )
$t_{\max}$	Maximum number of terms that can be observed in a query
$C_D$	Number of records containing $D$ distinct terms
$D$	Number of distinct terms in a record
$D_{\text{avg}}$	Average number of distinct terms in a record
$D_{\text{max}}$	Maximum number of distinct terms in a record
$D_{\text{min}}$	Minimum number of distinct terms in a record
$D_r$	Number of distinct terms in $r$ th record
$F$	Size of a signature (bits)
$F_i$	Size of $i$ th frame (bits) (MFSF)
$FD$	Number of false drops
$FD_i$	Expected number of false drops in $i$ th partition
$IFD_{(i,D)}$	Expected number of false drops for the records containing $D$ distinct terms after processing $i$ bit slices
$IP$	Improvement percentage
$N$	Number of records in database
$P_t$	Probability of submission of a $t$ term query
$P_{\text{size}}$	Size of a record pointer (bytes)
$RT$	Query response time
$S$	Number of bits set to 1 for each term
$S_i$	Number of bits set to 1 by each term in $i$ th frame (MFSF)
$STD$	Standard deviation
$TFD_i$	Expected number of false drops after processing $i$ bit slices (MFSF)
$W(Q)_t$	Expected number of on-bits in signature of a $t$ term query (query weight)

## B. Meanings of frequently used method acronyms.

Acronym	Meaning
AFD-MFSF	Average false drop estimated multiframe signature file
AFD-SSF	Average false drop estimated sequential signature file
BSSF	Bit sliced signature file
IFD-MFSF	Individual false drop estimated multiframe signature file
IFD-SSF	Individual false drop estimated sequential signature file
MFSF	Multiframe signature file
SSF	Sequential signature file

with the same  $D$  value, have the same  $fd$  value. Therefore, Equation (4) is rewritten as follows:

$$FD = \sum_{d=1}^{D_{\max}} C_d \cdot \left( 1 - \left( 1 - \frac{S}{F} \right)^d \right)^{W(Q)_t} \quad (5)$$

where  $C_d$  is the number of records containing  $d$  distinct terms. Since the number of non-zero  $C_d$  values is always less than or equal to  $N$ , Equation (5) is simpler and more efficient than Equation (4).

Since individual  $fd$  values of the records are used in computing  $FD$ , we call this method the *individual false drop computation method* (IFD). In Equation (5) we assume the same  $S$  and  $F$  values are used for all records. If  $F$  and  $S$

can be adjusted according to the  $D_r$  values of the records, a lower false drop probability may be obtained [18]. In such an environment, each partition will have its own  $F$  and  $S$  value and the query processor must compute a different query signature for each partition corresponding to a different  $D_r$  value.

AFD and IFD are extreme cases for  $FD$  estimation. IFD requires more information about the database instance than AFD, but provides a more accurate estimation of  $FD$  than AFD. AFD can also be seen as the use of FSB with 'average document size' blocks. The use of FWB in a similar context is impossible, since IFD contradicts the definition of FWB [16].

In PFD, the database is divided into  $p$  disjoint conceptual

Case I: $N = 2, F = 200, D_1 = 25, D_2 = 35, D_{\text{avg}} = 30, S = 5, t = 1$ , Standard deviation of $D = 5$		
Case II: $N = 2, F = 200, D_1 = 20, D_2 = 40, D_{\text{avg}} = 30, S = 5, t = 1$ , Standard deviation of $D = 10$		
AFD*	IFD: Case I	IFD: Case II
$FD = 2 \cdot (1 - (1 - \frac{5}{200})^{30})^5$ $FD = 2 \cdot 0.04266$ $FD = 0.0853$	$FD = (1 - (1 - \frac{5}{200})^{25})^5 +$ $(1 - (1 - \frac{5}{200})^{35})^5$ $FD = 0.0227 + 0.0701$ $FD = 0.0928$	$FD = (1 - (1 - \frac{5}{200})^{20})^5 +$ $(1 - (1 - \frac{5}{200})^{40})^5$ $FD = 0.0099 + 0.1047$ $FD = 0.1146$
$Deviation \text{ for Case I} = 100 \cdot \frac{0.0928 - 0.0853}{0.0853} = 8.79\%$		
$Deviation \text{ for Case II} = 100 \cdot \frac{0.1146 - 0.0853}{0.0853} = 34.35\%$		

\* AFD computation is the same for both cases

FIGURE 2. Example of  $FD$  computations by using average  $D$  and individual  $D$  values.

partitions according to the number of distinct terms in the records. Each partition is considered as a separate signature file and the average number of distinct terms in the partition is used to estimate  $FD$  in this partition. If there is only one partition, i.e.  $p = 1$ , PFD converges to AFD. If there are  $D_{\text{max}}$  partitions, i.e.  $p = D_{\text{max}}$ , PFD converges to IFD. In other words, PFD provides a generalized model that ranges from AFD to IFD.

Example  $FD$  computations are provided in Figure 2. In this example, the  $FD$  value computed by AFD is less than the  $FD$  value computed by IFD. The formal proof provided in the Appendix shows that under certain conditions the  $FD$  value computed with AFD is less than or equal to the  $FD$  value computed with IFD. These conditions, see the Appendix for details, are (1)  $S = F \ln 2 / D_{\text{avg}}$  and (2)  $W(Q)_r > 2^{\max(x_1, x_2, \dots, x_N)}$  where  $x_r = D_r / D_{\text{avg}}$  for ( $1 \leq r \leq N$ ). In practice  $S$  should be rounded to an integer. The experimental evidence provided in Sections 4.2 and 5.2 shows that in practical cases the distinction between IFD and AFD remains valid even without satisfying the conditions stated earlier. In Figure 2 although  $D_{\text{avg}}$  values are the same for both case I and II, different  $FD$  values are obtained for IFD. The difference between the  $FD$  values of AFD and IFD increases for increasing (standard) deviation of  $D$  values.

### 3. EXPERIMENTAL ENVIRONMENT AND DESIGN

As shown in the Appendix, under certain conditions the  $FD$  value estimated by AFD is less than or equal to the  $FD$  value estimated by IFD for the same database instance and signature file parameters. In Sections 4 and 5, we test the performance of the IFD-based SSF and MFSF signature file methods with real data to investigate the real life impacts of IFD.

#### 3.1. Experimental environment

In our experiments we used MARC records of the Bilkent University library collection. The test database, BLISS-1,

TABLE 2. Parameter values for the experiments.

$t_{\text{max}} = 5$	maximum number of terms in a query
$B_{\text{size}} = 8192$	size of a disk block (bytes)
$D_{\text{avg}} = 25.7$	average number of distinct terms in a record
$N = 152,850$	number of records in database
$P_{\text{size}} = 4$	size of a record pointer (bytes)
$PB = 2048$	number of record pointers in the record pointer buffer
$T_{\text{read}} = 5.77$	time required to read a disk block (milliseconds, ms)
$T_{\text{scan}} = 4.5$	average time required to match a record with query (ms)
$T_{\text{seek}} = 30$	time required to position the read head of disk to desired disk block (ms)

contains 152,850 MARC records with varying lengths which is suitable to compare AFD and IFD. The smallest and largest records after stop word elimination contain 1 and 166 distinct terms, respectively. Our stop word list contains all single letters and about 100 additional entries ([18], Appendix D). The average number and standard deviation of distinct terms per record are, respectively, 25.7 and 11.12; and the database contains 166,216 unique terms. We provide the record length distribution of the test database BLISS-1 in Figure 3 where the last bar represents the number of records with more than 62 unique terms.

A personal computer is used in the experiments. The test environment provides exclusive control of all resources including the physical layout of the file on the disk medium. Non-interrupting execution of user programs provides an accurate measure of the response time and produces consistent and reproducible results [20]. The experimental environment is defined in more detail in Table 2.

The physical layout of a signature file on the disk affects the time required to process the signature file. To obtain consistent and reproducible response times, the record file and the signature files are allocated fully contiguously on

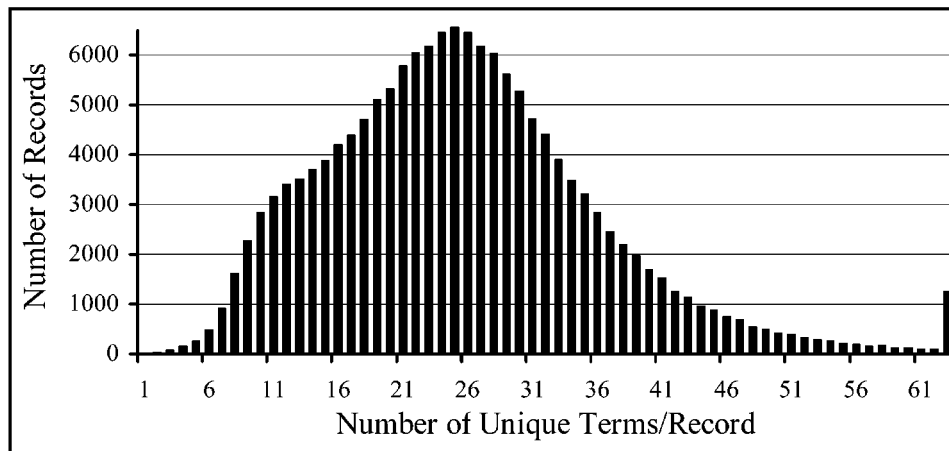


FIGURE 3. Distribution of record lengths of test database BLISS-1.

the disk. For MFSF, to obtain the positions of the records for given record numbers (signature file processing phase of MFSF produces a list of record numbers as the result), a record pointer file is used. Each record pointer occupies  $P_{size}$  bytes and only  $PB$  record pointers are kept in the main memory. In SSF each record signature is followed by its record pointer.

### 3.2. Experimental design

The performance of the inspected signature file methods is measured in terms of observed *response time* and observed *FD*. The *response time* is defined as the time required to:

- process the signature file;
- access all records shortlisted as candidates to qualify for the query, following the signature file processing phase, if any;
- retrieve the qualifying records.

Different response time values may be obtained for the queries containing the same number of query terms, since queries may have different numbers of matching records. In a given experimental environment, if different file access methods use the same storage structure for the record pointers and the records, all methods will require the same amount of time to retrieve the true matches after obtaining their record numbers. Independent of the number of matching records, the signature file processing and the false drop resolution phases must be performed, i.e. the response time is biased with the time required to perform these operations. Zero hit queries are the queries with no matching records. If the response time of zero hit queries, i.e. the time required to process the signature file plus the time required to resolve all false drops, is minimized; the response time of the queries containing the same number of terms with matching records will also be minimized. Therefore, we used zero hit queries in the experiments.

Search queries in real information retrieval systems contain a varying number of terms and the number of query

TABLE 3.  $P_t$  values for LW, UD and HW query cases.

Query case	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$
Low weight (LW)	0.30	0.25	0.20	0.15	0.10
Uniform distribution (UD)	0.20	0.20	0.20	0.20	0.20
High weight (HW)	0.10	0.15	0.20	0.25	0.30

terms tends to increase for large databases. Therefore, to obtain realistic results we tested the inspected methods in multiterm query environments. We considered three different query cases: low weight (LW), uniform distribution (UD) and high weight (HW) queries. Each query may contain up to five terms ( $t_{max}$ ). The values of  $P_t$  ( $1 \leq t \leq 5$ ), where  $P_t$  denotes the probability of submitting a  $t$  term query, for these query cases are given in Table 3. For the queries containing more than five terms we can assume  $t = 5$ . Note that this is a pessimistic assumption since *FD* decreases for increasing numbers of query terms.

We generated a query set containing 1000 zero hit queries randomly by considering the occurrence probabilities of the number of query terms for each query case. For example, since the occurrence probability of a one term query is 0.10 in the HW query case, the HW query set contains 100 ( $0.10 \cdot 1000$ ) one term queries. The observed *FD* and response time values are obtained by taking the average of the *FD* and response time values obtained by each query in the query sets.

## 4. USING IFD IN SEQUENTIAL SIGNATURE FILES

### 4.1. Concepts of sequential signature files (SSF): AFD-SSF

The sequential signature file (SSF) method requires retrieving the whole signature file for each query [15]. To minimize the number of seek operations to access the

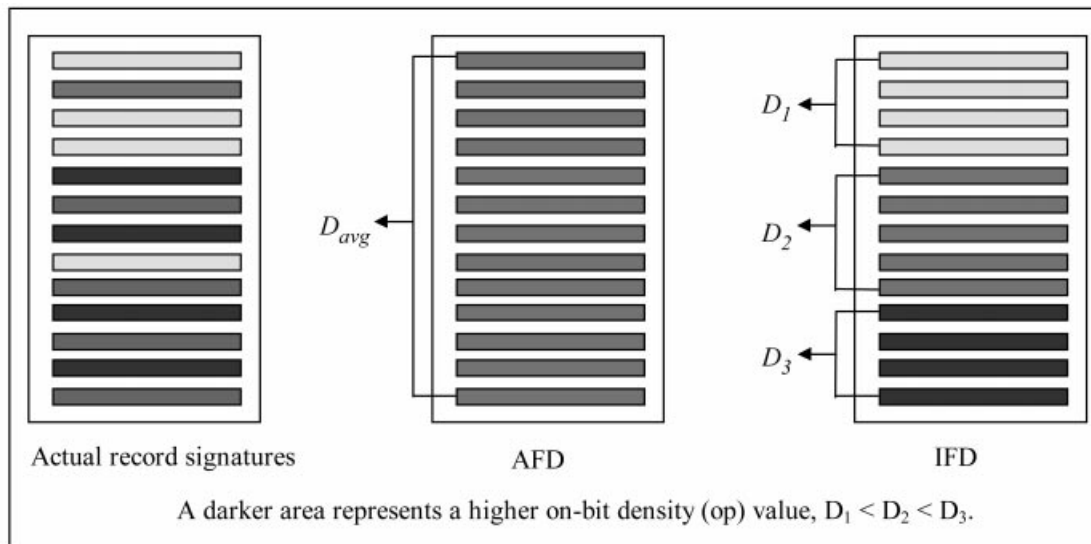


FIGURE 4. Graphical representations of estimating  $FD$  with AFD and IFD for SSF.

actual records, the record pointers are stored along with the signatures. Each record pointer holds the position of the corresponding actual record and it occupies four bytes ( $P_{size}$ ).

Graphical representations of estimating  $FD$  with AFD and IFD for SSF are illustrated in Figure 4. A darker area indicates a record with a higher  $D$  value (we assume that both methods use the same  $S$  value for all records).

The false match (drop) probability of a record signature and a query signature becomes minimal and the storage utilization efficiency becomes maximal when half of the signature bits of a record are on-bits. For given  $F$  and  $D$  values, the value of  $S$  that satisfies this optimality condition is computed as follows ([15], Equation A8):

$$S = F \cdot \ln 2 / D. \quad (6)$$

For databases with varying record lengths, the value of  $S$  in Equation (6) is determined by using  $D_{avg}$  instead of individual  $D$  values of the records. Note that the signatures of the records with  $D_r > D_{avg}$  ( $1 \leq r \leq N$ ) would contain more on-bits than off-bits, i.e. the optimality condition may not hold for all record signatures. (Similarly  $D_r < D_{avg}$  would lead to less on-bits than off-bits.) We will refer to this  $D_{avg}$ -based method as the AFD sequential signature file (AFD-SSF) method.

For a given  $F$  value (space overhead), since the whole signature file must be retrieved and processed for query evaluation, the time required to process the SSF will be the same for all  $S$  values. Therefore, minimizing the observed  $FD$  will also minimize the response time for SSF.

#### 4.2. IFD-based SSF and performance evaluation

In the IFD version of SSF, IFD-SSF, the  $S$  value, which provides the minimum  $FD$ , is determined with Equation (5)

by a linear search in the domain of possible  $S$  values. The lower and upper bounds of the search space are  $F \cdot \ln 2 / D_{max}$  and  $F \cdot \ln 2 / D_{min}$ , respectively. Since the value of  $S$  must be an integer, the number of possible  $S$  values will be small.

Note that IFD-SSF tries to find the  $S$  value that will give the best performance by paying attention to the individual  $D$  values; however, AFD-SSF uses the  $D$  value that will give the best performance for the average record. Note that, in IFD-SSF there is only one  $S$  value which is used for the whole signature file, i.e. for all partitions. The same is true for the value of  $F$ .

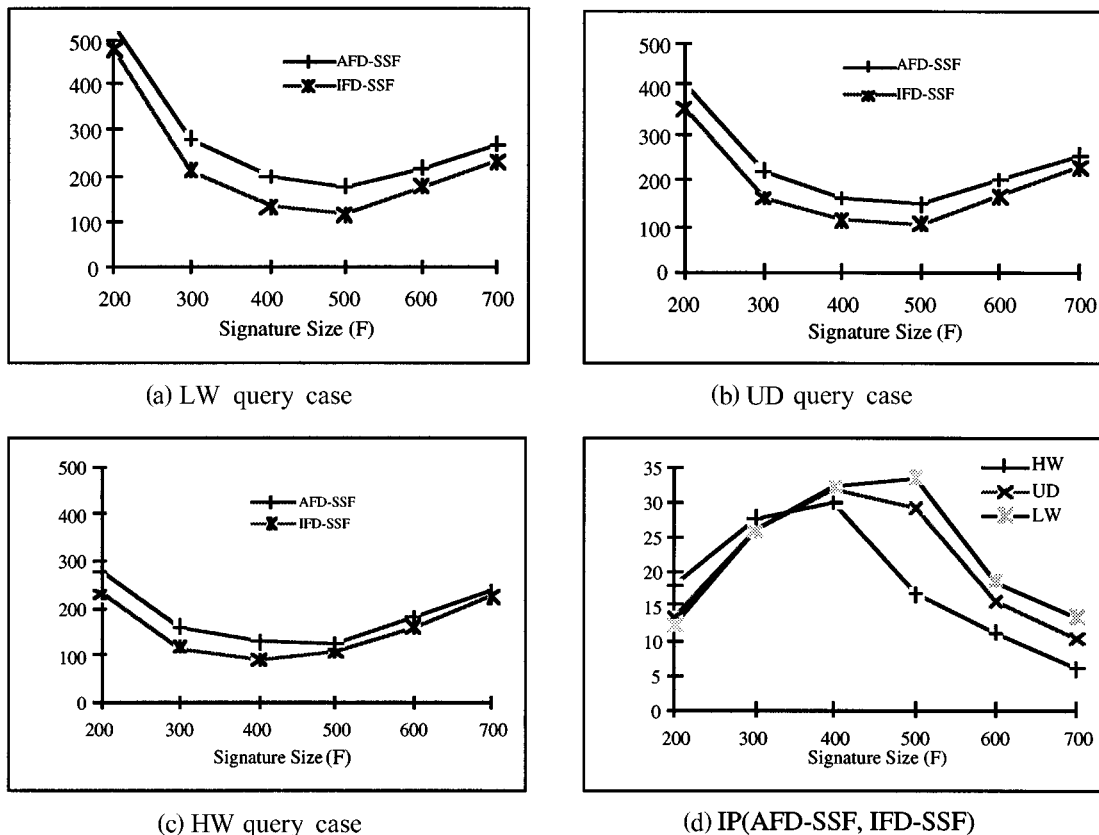
Although the signature file occupies less space than the original records, for large databases the response time of SSF is still very high. However, the SSF approach may efficiently search small databases or small subsets of a database. For example, the two-level access method [4] partitions a signature file horizontally such that the signatures of each partition fit into a disk block and the signatures are stored sequentially in the disk blocks. For query evaluation each qualifying disk block is searched sequentially. Similar approaches also apply to other horizontal signature partitioning methods such as linear hashing with superimposed signatures (also known as Quick Filter) [1, 21]. Therefore, we tested the performance of AFD-SSF and IFD-SSF on a small database with 1000 records of BLISS-1. The expected (denoted by Exp) and the observed (denoted by Obs) average false drop values of both methods for various  $F$  values are given in Table 4.

Table 4 shows that the expected  $FD$  values of AFD are always less than the observed  $FD$  values of this method. Another important result is that the observed  $FD$  values of IFD-SSF are always less than the observed  $FD$  values of AFD-SSF. Finally, the expected and observed average  $FD$  values for IFD-SSF are very close for all query cases. This shows that IFD estimates  $FD$  more accurately than AFD.

**TABLE 4.** Expected and observed average false drop (*FD*) values for AFD-SSF and IFD-SSF\* ( $N = 1000$ ).

<i>F</i>	LW query case						UD query case						HW query case					
	AFD-SSF			IFD-SSF			AFD-SSF			IFD-SSF			AFD-SSF			IFD-SSF		
<i>S</i>	Exp	Obs	<i>S</i>	Exp	Obs	<i>S</i>	Exp	Obs	<i>S</i>	Exp	Obs	<i>S</i>	Exp	Obs	<i>S</i>	Exp	Obs	
200	5	8.10	12.76	4	11.83	11.98	5	5.42	8.62	4	8.61	8.13	5	2.75	4.76	4	4.73	4.33
300	8	1.29	4.07	5	3.38	3.28	8	0.86	2.80	5	2.31	2.19	8	0.43	1.62	5	1.24	1.18
400	10	0.21	1.59	6	1.11	1.04	10	0.14	1.13	6	0.76	0.65	10	0.07	0.67	6	0.40	0.34
500	13	0.03	0.96	7	0.44	0.43	13	0.02	0.64	7	0.30	0.28	13	0.01	0.43	6	0.15	0.13
600	15	0.01	0.54	7	0.19	0.18	15	0.00	0.38	7	0.13	0.11	15	0.00	0.23	7	0.07	0.05
700	18	0.00	0.40	7	0.09	0.09	18	0.00	0.27	7	0.06	0.05	18	0.00	0.16	7	0.03	0.03

\* An expected value of 0.00 is due to rounding, actual values are greater than 0.

**FIGURE 5.** Observed response time against  $F$  for AFD-SSF and IFD-SSF ( $N = 1000$ ). (For the definition of query cases refer to Table 3.)

The observed response time values against  $F$  are plotted in Figure 5. In Figure 5d the improvement percentage obtained by IFD-SSF over AFD-SSF in terms of observed response time is plotted. The improvement percentage,  $IP$ , for any two methods, say  $A$  and  $B$ , is defined as

$$IP(A, B) = 100 \cdot (TR(A) - TR(B)) / TR(A)$$

where  $TR(A)$  and  $TR(B)$  are the response times obtained by  $A$  and  $B$ , respectively. In other words,  $IP(A, B)$  indicates the improvement obtained by  $B$  with respect to  $A$ .

The observed  $FD$  values, hence the time required to resolve the false drop records, decrease for increasing  $F$  values. Since the size of the signature file increases

for increasing  $F$  values, the time required to process the signature file also increases. The decrease in  $FD$  becomes negligible after a certain  $F$  value while the increase in the time required to process the signature file increases almost linearly. Therefore, the response time decreases as  $F$  increases for small  $F$  values and starts to increase after a certain  $F$  value (for  $F$  values 400, 500 and 500 for HW, UD and LW query cases, respectively). We call this point optimum  $F$  (space overhead) for a database instance [18]. Since the observed  $FD$  diminishes more rapidly for increasing query weights, the optimum  $F$  value decreases as the query weight increases.

In text database environments using SSF with an  $F$  value

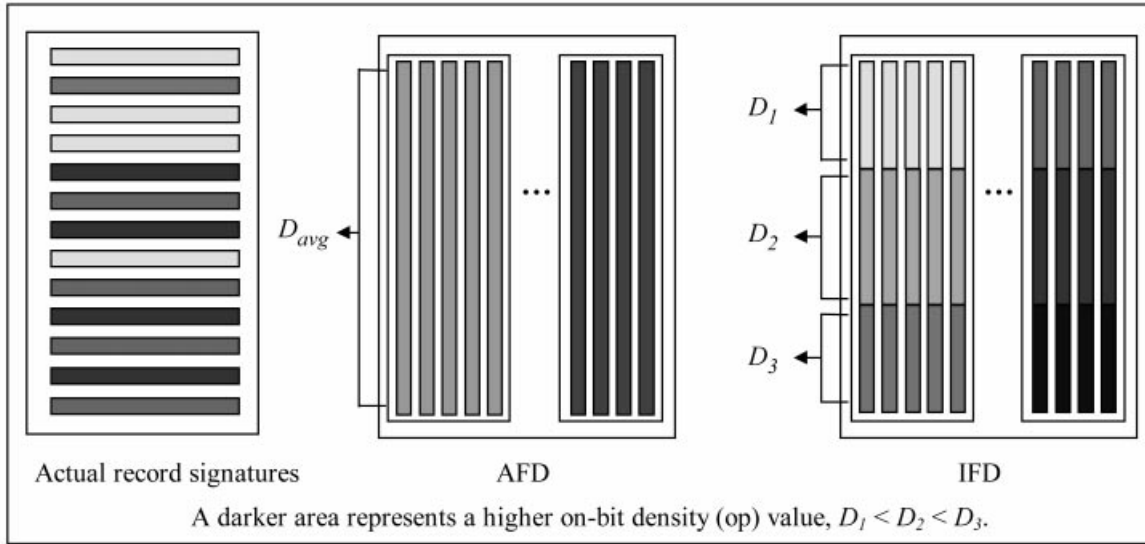


FIGURE 6. Graphical representations of estimating  $FD$  with AFD and IFD for MFSF.

greater than the optimum,  $F$  value is meaningless. Also, the response time values for small  $F$  values are very high. Therefore, we can assume the IP values observed around the optimum  $F$  values as the performance increased were obtained by using IFD for SSF. The IP (AFD-SSF, IFD-SSF) values obtained at optimum  $F$  values are 30%, 29% and 33% for HW, UD and LW query cases, respectively. These results indicate that our IFD approach outperforms the conventional AFD approach in SSF.

## 5. USING IFD IN MULTI-FRAME SIGNATURE FILES

### 5.1. Concepts of multiframe signature files (MFSF): AFD-MFSF

MFSF is a new signature file method that outperforms signature files with good performance (such as generalized frame sliced signature files) [11, 18]. It is designed for very large databases and considerably faster than SSF. As discussed in Kocberber and Can [11] its performance is competitive with inverted files.

In MFSF a signature file is conceptually divided into  $f$  sub-signature files. Each sub-signature file is a bit-sliced signature file (BSSF) [13] with its own  $F$  and  $S$  parameters. The bits of a signature file are distributed among the  $f$  sub-signature files, (vertical) frames, such that  $F = F_1 + F_2 + \dots + F_f$  ( $1 \leq f \leq F$ ). Each term sets  $S_r$  bits in the  $r$ th frame and  $S = S_1 + S_2 + \dots + S_f$  ( $1 \leq S_r \leq F_r$ ,  $1 \leq r \leq f$ ). Consequently, each frame may have a different on-bit density ( $op$  value). Graphical representations of estimating  $FD$  with AFD and IFD for MFSF are illustrated in Figure 6 (in MFSF the frames are ordered in increasing  $op$  values). A

darker area indicates more on-bits in that part of the record signature.

In MFSF, the response time is minimized in a multiterm query environment by employing a partial evaluation strategy and by considering the submission probabilities of the queries with different numbers of terms [18]. The technique employs a stopping condition that tries to perform signature file processing without using all on-bits of the query signature, i.e. by partial evaluation.

The aim of the stopping condition is to reduce the number of expected false drops to an optimum level that will also provide the lowest response time within the framework of MFSF [18]. The signature file processing continues as long as processing the signature file decreases the response time by decreasing the expected number of false drops. The stopping condition of MFSF provided in Kocberber [18] is as follows:

$$T_{\text{slice}} \leq (TFD_i - TFD_{i+1}) \cdot T_{\text{resolve}} \quad (7)$$

where  $T_{\text{slice}}$  is the time required to read and process a bit slice,  $TFD_i$  is the number of expected false drops after processing  $i$  bit slices and  $T_{\text{resolve}}$  is the time required to resolve a false drop record by accessing the actual record. In formula (7),  $(TFD_i - TFD_{i+1})$  gives the number of expected false drops which will be eliminated if we process the  $(i + 1)$ st bit slice after processing  $i$  bit slices. At the stopping step the time required to process a bit slice becomes greater than the time required to resolve these false drops by accessing the actual records, i.e. the inequality is no longer satisfied. Therefore, the signature file processing stops at this step (Kocberber [18], Chapter 5; [11]).

In MFSF,  $FD$  is computed incrementally. In this method all the records are initially assumed as false drops, i.e.  $TFD_0 = N$ . For  $(i + 1)$ st ( $i \geq 0$ ) bit slice processing, the

**TABLE 5.** Expected (Exp) and observed (Obs) average false drop ( $FD$ ) values for AFD-MFSF and IFD-MFSF ( $N = 152,850$ , complete BLISS-1 database).

$F$	LW query case				UD query case				HW query case			
	AFD-MFSF		IFD-MFSF		AFD-MFSF		IFD-MFSF		AFD-MFSF		IFD-MFSF	
	Exp	Obs	Exp	Obs	Exp	Obs	Exp	Obs	Exp	Obs	Exp	Obs
800	0.80	12.48	8.90	8.78	0.89	10.45	6.75	6.25	0.61	7.53	4.32	3.50
1000	0.60	5.37	3.43	3.40	0.64	4.82	2.79	2.45	0.47	3.11	2.26	2.59
1200	0.54	2.92	2.00	1.92	0.43	2.57	1.64	1.71	0.29	1.37	1.31	1.22
1400	0.43	2.15	1.02	1.19	0.30	1.50	1.07	1.03	0.35	1.45	0.72	0.55
1600	0.43	1.51	0.87	1.17	0.35	1.26	0.68	0.75	0.27	0.97	0.62	0.56
1800	0.37	1.31	0.66	0.72	0.30	1.02	0.64	0.72	0.21	0.60	0.49	0.53

expected number of false drops is computed as follows [18]:

$$TFD_{i+1} = TFD_i \cdot (1 - (1 - S_r/F_r)^{D_{\text{avg}}}) \quad (8)$$

where  $r$  is the number of the frame that the  $(i + 1)$ st bit slice is selected from. Since  $FD$  is computed by using  $D_{\text{avg}}$  this method is called AFD-MFSF.

## 5.2. IFD-based MFSF and performance evaluation

In the IFD version of MFSF, IFD-MFSF, we consider individual  $D$  values of the records in  $FD$  estimation. The initial number of false drops and  $FD$  after processing the  $(i + 1)$ st ( $i \geq 0$ ) bit slice are computed by conceptually grouping the records containing the same number of distinct terms together as follows:

$$\begin{aligned}
 IFD_{(0,D)} &= C_D \quad \text{for } 1 \leq D \leq D_{\text{max}} \\
 TFD_0 &= \sum_{D=1}^{D_{\text{max}}} IFD_{(0,D)} \\
 IFD_{(i+1,D)} &= IFD_{(i,D)} \cdot (1 - (1 - S_r/F_r)^D) \quad \text{for } i \geq 0 \\
 TFD_{i+1} &= \sum_{D=1}^{D_{\text{max}}} IFD_{(i+1,D)} \quad \text{for } i \geq 0
 \end{aligned} \quad (9)$$

where  $IFD_{(i,D)}$  is the expected number of false drops for the records containing  $D$  distinct terms after processing  $i$  bit slices and  $r$  is the number of the frame the  $(i + 1)$ st ( $i \geq 0$ ) bit slice is selected from. In IFD-based MFSF each group of records with the same  $D$  value is considered as an interdependent separate MFSF file. The interdependence of the partitions implies that the same  $f$ ,  $S_i$  and  $F_i$  ( $1 \leq i \leq f$ ) values are used in all groups of records.

Since each term sets bit(s) in each frame, queries with more terms have more query signature on-bits in the bit locations corresponding to the lower on-bit density (low  $op$  value) record signature frames. Lower  $op$  values eliminate false drops more rapidly during query processing and the stopping condition is reached in fewer evaluation steps. This property provides better response times for increasing numbers of query terms [11].

In MFSF-based query evaluation (both in AFD and IFD versions) the query on-bits of the initial (lower on-bit density) record signature frames are used first. The IFD-MFSF approach provides lower on-bit densities in the record signature frames that are first used in query processing and therefore it has the potential of being better than AFD-MFSF. As will be shown, this is the case indeed.

We used the heuristic search algorithm given in Kocberber [18] and Kocberber and Can [11] to search the optimum IFD-MFSF configuration, i.e. the values of  $f$  ( $1 \leq f \leq F$ ),  $F_r$ , and  $S_r$  ( $1 \leq r \leq f$ ), by using Equation (9) for  $FD$  estimation. The algorithm starts with a random configuration, i.e. with random  $f$  ( $1 \leq f \leq F$ ),  $F_i$  and  $S_i$  ( $1 \leq i \leq f$ ) values. A candidate configuration is obtained by randomly selecting a frame among  $f$  frames and changing the  $S_i$  value and the  $F_i$  value of this frame. If a smaller response time is obtained in one of the candidate configurations, all frames are considered as untested. The search for minimum response time continues until all frames are tested for a candidate configuration without obtaining a smaller response time. The same configuration, i.e. the same  $F_i$  and  $S_i$  ( $1 \leq i \leq f$ ) values, is used in all partitions.

The expected (Exp) and the observed (Obs) false drop,  $FD$ , values for the BLISS-1 database ( $N = 152,850$ ) and various  $F$  values are given in Table 5. Since  $FD$  values are estimated differently, the stopping conditions of AFD-MFSF and IFD-MFSF may require processing of different numbers of bit slices. Consequently, signature file processing times may be different. Therefore, we also provide the corresponding observed response time values in Figure 7.

Like IFD-SSF, the observed  $FD$  values of IFD-MFSF are always less than the observed  $FD$  values of AFD-MFSF. Additionally, IFD-MFSF estimates  $FD$  more precisely and provides smaller observed  $FD$  values with shorter response times. Therefore, the IFD-MFSF method outperforms the AFD-MFSF method. As shown in Figure 7d, depending on the space overhead, IFD-MFSF provides up to 20% response time improvements over AFD-MFSF (since the observed  $FD$  values are very high, we considered the space overheads with  $F < 1000$  as practically unusable and ignored the higher response time improvements obtained for them).

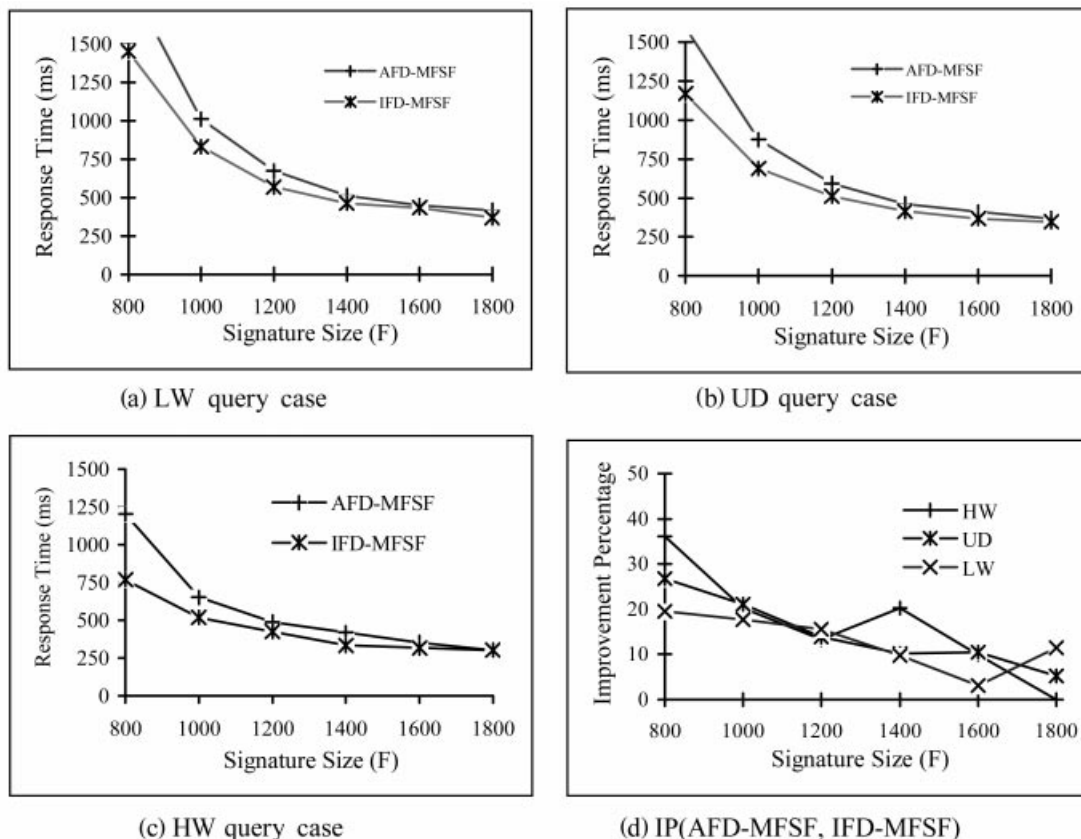


FIGURE 7. Observed response time against  $F$  for AFD-MFSF and IFD-MFSF ( $N = 152,850$ ). (For the definition of query cases refer to Table 3.)

## 6. CONCLUSION

For signature files a new method, called the individual false drop estimation method (IFD), is presented. The new method provides an accurate estimation of the number of false drops by considering individual numbers of distinct terms of the records, instead of using the average number of distinct record terms. In signature files the estimated number of false drops is used in the optimization of the signature file parameters that will give the minimum response time in the specified context. The accurate estimation of false drops provides the design of a signature file with a superior performance.

Our approach can be used in multiterm query environments and with different signature file organization methods. For this purpose we extended the conventional sequential signature file (SSF) and multiframe signature file (MFSF) methods to estimate the number of false drops with IFD in various query environments. The experiments with real data show that the proposed method estimates the number of false drops accurately and improves the performance of both methods significantly.

Interesting research possibilities include an analytical solution for the optimal value of  $S$  in IFD, implementation and evaluation of various signature file methods by using the concepts of IFD and IFD-based applications in memory resident databases.

## ACKNOWLEDGEMENTS

We greatly appreciate the constructive criticism provided by the anonymous referees. Their suggestions have greatly improved both the contents and presentation of the paper.

## REFERENCES

- [1] Zezula, P., Rabitti, F. and Tiberio, P. (1991) Dynamic partitioning of signature files. *ACM Trans. Inform. Syst.*, **9**, 336–367.
- [2] Lee, S.-Y., Yang, M.-C. and Chen, J.-W. (1992) Signature file as a spatial filter for iconic image database. *Visual Languages Comput.*, **3**, 373–397.
- [3] Witten, I. H., Moffat, A. and Bell, T. C. (1994) *Managing Gigabytes: Compression and Indexing Documents and Images*. Van Nostrand Reinhold, New York.
- [4] Sacks-Davis, R., Kent, A. and Ramamohanarao, K. (1987) Performance of multikey access method based on descriptors superimposed coding techniques. *Inform. Syst.*, **10**, 391–403.
- [5] Faloutsos, C. (1992) Signature files. In Frakes, W. B. and Baeza-Yates, R. (eds), *Information Retrieval Data Structures and Algorithms*, pp. 44–65. Prentice Hall, Englewood Cliffs, NJ.
- [6] Lin, Z. and Faloutsos, C. (1992) Frame-sliced signature files. *IEEE Trans. Knowledge Data Eng.*, **4**, 281–289.

- [7] Aktug, D. and Can, F. (1993) Signature files: An integrated access method for formatted and unformatted databases. *Working Paper 93-006*, Dept. of Systems Analysis, Miami University, Oxford, OH.
- [8] Dervos, D., Linardis P. and Manolopoulos, Y. (1994) Binary ranking for the signature file method. *Inform. Software Technol.*, **36**, 131–139.
- [9] Ciaccia, P., Tiberio, P. and Zezula, P. (1996) Declustering of key-based partitioned signature files. *ACM Trans. Database Syst.*, **21**, 295–338.
- [10] Kocberber, S. and Can, F. (1996) Partial evaluation of queries for bit-sliced signature files. *Inform. Process. Lett.*, **60**, 305–311.
- [11] Kocberber, S. and Can, F. (1997) Vertical framing of superimposed signature files using partial evaluation of queries. *Inform. Process. Management*, **33**, 353–376.
- [12] Faloutsos, C. (1985) Access method for text. *ACM Comput. Surveys*, **17**, 49–74.
- [13] Roberts, C.S. (1979) Partial-match retrieval via the method of superimposed codes. In *Proc. IEEE*, **67**, 1624–1642.
- [14] Lin, Z. and Faloutsos, C. (1988) *Frame-sliced Signature Files*. Technical Report CS2146 and UMIACS-TR-88-88, Computer Science Dept., University of Maryland.
- [15] Christodoulakis, S. and Faloutsos, C. (1984) Design considerations for a message file server. *IEEE Trans. Software Eng.*, **10**, 201–210.
- [16] Leng, C. W. R. and Lee, D. L. (1992) Optimal weight assignment for signature generation. *ACM Trans. Database Syst.*, **17**, 346–373.
- [17] Faloutsos, C. (1988) Signature files: An integrated access method for text and attributes, suitable for optical disk storage. *BIT*, **28**, 736–754.
- [18] Kocberber, S. (1996) *Partial Query Evaluation for Vertically Partitioned Signature Files in Very Large Unformatted Databases*. Ph.D. Thesis, Dept. of Computer Eng. and Information Science, Bilkent University, Ankara, Turkey. Available on <http://www.cs.bilkent.edu.tr/theses.html>.
- [19] Dervos, D., Manolopoulos, Y. and Linardis, P. (1998) Comparison of signature file models with superimposed coding. *Inform. Process. Lett.*, **65**, 101–106.
- [20] Salzberg, B. J. (1988) *File Structures: An Analytical Approach*. Prentice Hall, Englewood Cliffs, NJ.
- [21] Aktug, D. and Can, F. (1993) Analysis of multiterm queries in a dynamic signature file organization. In *Proc. 16th Ann. Int. ACM-SIGIR Conf.*, ACM, New York, pp. 96–105.
- [22] Winston, W. L. (1994) *Operations Research, Applications and Algorithms* (3rd edn). Duxbury Press, Belmont, CA.

## APPENDIX

**THEOREM A.1.** *The number of false drop records (FD) estimated by considering the average number of terms ( $D_{\text{avg}}$ ) in the records is less than or equal to the FD estimated by considering individual  $D$  values of the records ( $D_r$ , where  $1 \leq r \leq N$ ), for the value of  $S$  that satisfies the optimality condition*

$$S = \frac{F \ln 2}{D_{\text{avg}}}$$

under conditions

$$W(Q)_t > 2^{\max(x_1, x_2, \dots, x_N)}$$

where

$$x_r = \frac{D_r}{D_{\text{avg}}} \quad r = 1, 2, \dots, N.$$

Note that the condition on  $W(Q)_t$  may be quite restrictive for databases having a large range of distinct terms per record.

*Proof.* We want to show for all positive integers  $N$  that

$$\begin{aligned} & \sum_{r=1}^N \left[ 1 - \left[ 1 - \frac{S}{F} \right]^{D_r} \right]^{W(Q)_t} \\ & \geq N \left[ 1 - \left( 1 - \frac{S}{F} \right)^{D_{\text{avg}}} \right]^{W(Q)_t} \end{aligned} \quad (\text{A.1})$$

where

$$D_{\text{avg}} = \sum_{r=1}^N \frac{D_r}{N} \quad (\text{A.2})$$

$$S = \frac{F \ln 2}{D_{\text{avg}}} \quad (\text{A.3})$$

$$W(Q)_t > 2^{\max(x_1, x_2, \dots, x_N)} \quad (\text{A.4})$$

$$x_r = \frac{D_r}{D_{\text{avg}}} \quad (\text{A.5})$$

By substituting the approximation

$$1 - \left( 1 - \frac{S}{F} \right)^D \approx 1 - e^{-SD/F}$$

in (A.1) we obtain

$$\sum_{r=1}^N \left[ 1 - e^{-S \cdot D_r / F} \right]^{W(Q)_t} \geq N \left[ 1 - e^{-S \cdot D_{\text{avg}} / F} \right]^{W(Q)_t}.$$

Also (A.3) gives the ratio

$$\frac{S}{F} = \frac{\ln 2}{D_{\text{avg}}}.$$

Thus, we wish to show

$$\sum_{r=1}^N \left[ 1 - e^{-\ln 2 \cdot D_r / D_{\text{avg}}} \right]^{W(Q)_t} \geq N \left[ 1 - e^{-\ln 2} \right]^{W(Q)_t}$$

or

$$\sum_{r=1}^N \left[ 1 - \left( \frac{1}{2} \right)^{D_r / D_{\text{avg}}} \right]^{W(Q)_t} \geq \frac{N}{2^{W(Q)_t}}.$$

To simplify notation we set  $W(Q)_t = W$ . From (A.5)

$$\sum_{r=1}^N x_r = \sum_{r=1}^N \frac{D_r}{D_{\text{avg}}} = N.$$

Thus, we can prove this result by showing that the minimum of

$$\sum_{r=1}^N \left[ 1 - \left( \frac{1}{2} \right)^{x_r} \right]^W \quad (\text{A.6})$$

subjected to the constraint

$$\sum_{r=1}^N x_r = N \quad (\text{A.7})$$

is

$$\frac{N}{2^W}.$$

By showing that the function

$$f(x_1, x_2, \dots, x_r) = \sum_{r=1}^N \left[ 1 - \left( \frac{1}{2} \right)^{x_r} \right]^W$$

is convex and noting that the constraint (A.7) is linear in  $x_r$ , any point  $(x_1, x_2, \dots, x_N, \lambda)$  satisfying

$$\frac{\partial L}{\partial x_1} = 0, \quad \frac{\partial L}{\partial x_2} = 0, \dots, \frac{\partial L}{\partial x_N} = 0, \quad \frac{\partial L}{\partial \lambda} = 0 \quad (\text{A.8})$$

where  $L$  is the Lagrangian

$$\begin{aligned} L(x_1, x_2, \dots, x_N, \lambda) \\ = \sum_{r=1}^N \left[ 1 - \left( \frac{1}{2} \right)^{x_r} \right]^W - \lambda \left( \sum_{r=1}^N x_r - N \right) \end{aligned}$$

will have  $(x_1, x_2, \dots, x_N)$  yielding an optimal solution to (A.6) [22].

To show that  $f$  is convex, we will show that its Hessian matrix is positive definite. Taking the first and second partial derivatives of  $f$  with respect to  $x_r$ ,

$$\frac{\partial f}{\partial x_r} = W(\ln 2) \left[ 1 - \left( \frac{1}{2} \right)^{x_r} \right]^{W-1} \left( \frac{1}{2} \right)^{x_r}$$

$$\begin{aligned} \frac{\partial^2 f}{\partial x_r^2} &= -W(\ln 2)^2 \left( \frac{1}{2} \right)^{x_r} \left[ 1 - \left( \frac{1}{2} \right)^{x_r} \right]^{W-1} \\ &\quad + W(W-1)(\ln 2)^2 \left( \frac{1}{2} \right)^{2x_r} \left[ 1 - \left( \frac{1}{2} \right)^{x_r} \right]^{W-2} \\ &= W(\ln 2)^2 \left( \frac{1}{2} \right)^{x_r} \left[ 1 - \left( \frac{1}{2} \right)^{x_r} \right]^{W-2} \left[ W \left( \frac{1}{2} \right)^{x_r} - 1 \right]. \end{aligned}$$

Since  $x_r > 0$  for  $r = 1, 2, \dots, N$

$$1 - \left( \frac{1}{2} \right)^{x_r} > 0 \quad r = 1, 2, \dots, N.$$

From (A.4), we have for each  $r = 1, 2, \dots, N$

$$W \left( \frac{1}{2} \right)^{x_r} \geq \frac{W}{2^{\max(x_1, x_2, \dots, x_N)}} > 1.$$

Thus

$$W \left( \frac{1}{2} \right)^{x_r} - 1 > 0 \quad r = 1, 2, \dots, N.$$

Thus, for  $r = 1, 2, \dots, N$

$$\frac{\partial^2 f}{\partial x_r^2} > 0.$$

Also for  $i \neq j$

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = 0 \quad i, j = 1, 2, \dots, N.$$

Therefore, the Hessian matrix of  $f$  is a diagonal matrix with positive entries. This matrix is then positive definite which makes  $f$  convex.

To find a point satisfying the system of equations (A.8), the first partials of the Lagrangian with respect to  $x_r$  are computed and set to zero

$$\begin{aligned} \frac{\partial L}{\partial x_r} &= \frac{\partial f}{\partial x_r} - \lambda \frac{\partial}{\partial x_r} \left( \sum_{r=1}^N x_r - N \right) \\ &= W \cdot \ln 2 \left[ 1 - \left( \frac{1}{2} \right)^{x_r} \right]^{W-1} \left( \frac{1}{2} \right)^{x_r} - \lambda \end{aligned}$$

for  $r = 1, 2, \dots, N$ . Then

$$\begin{aligned} \frac{\partial L}{\partial x_r} = 0 \rightarrow \lambda &= W \cdot \ln 2 \left[ 1 - \left( \frac{1}{2} \right)^{x_r} \right]^{W-1} \left( \frac{1}{2} \right)^{x_r} \\ r &= 1, 2, \dots, N. \end{aligned}$$

Also

$$\frac{\partial L}{\partial \lambda} = 0 \rightarrow \sum_{r=1}^N x_r = N.$$

Since  $f$  is symmetric with respect to its arguments, a point  $(x_1, x_2, \dots, x_N, \lambda)$  satisfying these  $(N+1)$  equations would be  $x_1 = x_2 = \dots = x_N = 1$  and

$$\lambda = W \cdot \ln 2 \left[ 1 - \frac{1}{2} \right]^{W-1} \left( \frac{1}{2} \right) = \frac{W \cdot \ln 2}{2^W}.$$

Thus, the function

$$\sum_{r=1}^N \left[ 1 - \left( \frac{1}{2} \right)^{D_r/D_{\text{avg}}} \right]^W$$

achieves its minimum value when

$$\frac{D_r}{D_{\text{avg}}} = x_r = 1$$

for  $r = 1, 2, \dots, N$ . Then its minimum value would be

$$\sum_{r=1}^N \left[ 1 - \frac{1}{2} \right]^W = \sum_{r=1}^N \left( \frac{1}{2} \right)^W = \frac{N}{2^W}.$$

Hence

$$\sum_{r=1}^N \left[ 1 - \left( \frac{1}{2} \right)^{D_r/D_{\text{avg}}} \right]^W \geq \frac{N}{2^W}$$

whenever conditions (A.3) and (A.4) hold. This completes the proof.  $\square$