ELSEVIER

Information Processing Letters 60 (1996) 305–311

Information Processing Letters

Partial evaluation of queries for bit-sliced signature files

Seyit Kocberber [a,1], Fazli Can [b,*]

[a] *Department of Computer Engineering and Information Science, Bilkent University, Bilkent, 06533 Ankara, Turkey*
[b] *Department of Systems Analysis, Miami University, Oxford, OH 45056, USA*

Received 29 June 1995; revised 15 July 1996
Communicated by K. Ikeda

Abstract

Our research extends the bit-sliced signature organization by introducing a partial evaluation approach for queries. The partial evaluation approach minimizes the response time by using a subset of the on-bits of the query signature. A new signature file optimization method, Partially evaluated Bit-Sliced Signature File (P-BSSF), for multi-term query environments using the partial evaluation approach is introduced. The analysis shows that, with 14% increase in space overhead, P-BSSF provides a query processing time improvement of more than 85% for multi-term query environments with respect to the best performance of the bit-sliced signature file (BSSF) method. Under the sequentiality assumption of disk blocks, P-BSSF provides a desirable response time of 1 second for a database size of one million records with a 28% space overhead. Due to partial evaluation, the desirable response time is guaranteed for queries with several terms.

Keywords: Information retrieval; Signature files

1. Introduction

Signature files provide a space efficient fast search structure by searching the record signatures instead of searching the actual records. For simplicity, an instance of any kind of data will be referred to as a *record* in the rest of this paper. A *record signature* is a bit string reflecting the essence of the record attributes. Insertions and updates in signature files require less time compared to inverted files [5].

In signature files, each term (record attribute) is hashed into a bit string of length F with S bits set to 1 (on-bit) which is called a *term signature* [1,8,9]. Record signatures are generally obtained by superimposing, i.e., bitwise ORing, the term signatures occurring in the record. In superimposed signatures $F \gg S$. These record signatures are stored in a separate file, called the *signature file.*

The query evaluation with signature files is conducted in two phases. In the first phase, the signature file is used for eliminating the irrelevant records. In this phase, first the signatures of the terms occurring in the query are superimposed to obtain the query signature, and then, this query signature is compared with the record signatures. The records whose signatures contain at least one 0 in the positions of the 1s in the query signature are eliminated, i.e., they are irrelevant to the query. Due to the hashing operation used to obtain term signatures and superimposition, the result of the first phase may contain *false drops*:

* Corresponding author. Email: fc74sanf@miamiu.acs.muohio.edu.
[1] Email: seyit@bilkerft.edu.tr.

PII S0020-0190(96)00176-7

the record signature satisfies the query although the actual record does not. Therefore, in the second phase possible false drops are resolved by accessing the actual records.

Several signature file organization methods have been proposed to obtain fast response time [1]. Storing a signature file in column-wise order is called bit-sliced signature file (BSSF) method [8]. The BSSF method requires retrieval of the bit slices corresponding to the 1s of the query signature. Consequently, most of the bit slices are eliminated for queries with a few bits set to 1 (on-bit) in their signatures [8]. This may provide further speedup in query evaluation while introducing extra processing time for insertion and updates. We will use *response time*, the time required to process the signature file, and to find the first qualified record during the false drop elimination, as a performance measure as used in [5].

We repeat the formulas to compute the number of on-bits in the query signature (query weight) and the expected number of false drops given in [8]. The false drop probability ($fd_{W(Q)_t}$) for a t term query is computed as follows,

$$fd_{W(Q)_t} = \left(1 - (1 - S/F)^D\right)^{W(Q)_t}, \tag{1}$$

$$W(Q)_t = F \cdot \left(1 - (1 - S/F)^t\right), \tag{2}$$

where D is the average number of terms per record and $W(Q)_t$ is the query weight of a t term query [8]. S and F are design parameters. Previous works show that the false drop probability becomes minimum when the *optimality condition* is satisfied, i.e., half of the bits in a record signature are on-bits [2,8]. The expected number of false drops after processing $W(Q)_t$ bit slices, $FD_{W(Q)_t}$, is proportional to the number of records in the database (N) and computed as follows,

$$FD_{W(Q)_t} = N \cdot fd_{W(Q)_t}. \tag{3}$$

In BSSF, especially for multi-term queries, the time required to complete the first phase of the query evaluation increases as the query weight increases [8].

There are previous proposals to improve the performance of BSSF. Sacks-Davis et al. [9] proposed using S bit slices in the first phase of the query evaluation of a multi-term query without providing a formal stopping condition. Lin and Faloutsos proposed adjusting the value of S for a specific number of query terms, t, such that the response time is minimized [5,6]. However, in a multi-term query environment, queries containing less than t terms will obtain many false drops. Also, since no stopping condition was defined, the queries with more than t terms will unnecessarily process many bit slices. Panagopoulos and Faloutsos defined a partial fetch policy with spooling the bit slices on a parallel machine architecture [7]. Ishikawa et al. [3] tried to find the optimum S value experimentally by measuring the response time for changing S values for a specific database instance.

We propose a new signature file optimization method, Partially evaluated Bit-Sliced Signature File (P-BSSF), which combines optimal selection of S with a partial evaluation strategy in a multi-term query environment. The partial evaluation strategy uses a subset of the on-bits of a query signature and oversees the equal contribution of each query term to the query evaluation process until it reaches the stopping condition. During selection of the optimal S value, we consider the submission probabilities of the queries with various numbers of terms.

2. Partial evaluation of queries in BSSF: P-BSSF

Our objective is to obtain the minimum response time in a multi-term query environment. Therefore, we derive the query response time estimation formulas first. The response time, $RT(i)$, can be written as a function of i, the number of bit slices used in the first phase for a t term query, as follows,

$$RT(i) = i \cdot T_{slice} + FD_i \cdot T_{resolve}$$
$$\text{where } 0 < i \leqslant W(Q)_t, \tag{4}$$

where T_{slice} is the time required to process a bit slice and $T_{resolve}$ is the time required to resolve a false drop. In the BSSF method i equals $W(Q)_t$.

To process a bit slice, the bit slice must be read and ANDed with the result of the processed bit slices. By assuming two bit slices will be stored in main memory, T_{slice} is computed as follows,

$$T_{slice} = Read\left(\left\lceil \frac{N}{8 \cdot B_{size}} \right\rceil\right) + T_{bitop} \cdot \left\lceil \frac{N}{8 \cdot W_{size}} \right\rceil, \tag{5}$$

where B_{size} and W_{size} are the size of a disk block and the size of a memory word in bytes, respectively and $\lceil\ \rceil$ indicates the ceiling function. T_{bitop} is the time required to perform a bitwise AND operation between two memory words and store the result in one of the words. $Read(b)$ incorporates the sequentiality probability, SP, to the estimation of the time required to read b logically consecutive disk blocks. SP is the probability of reading the next logically consecutive disk block without a seek operation.

$$Read(b) = (1 + (b-1)\cdot(1-SP))\cdot T_{seek} + b\cdot T_{read}, \quad (6)$$

where T_{seek} and T_{read} are average times required to position the disk head to the block to be accessed and to transfer a disk block to memory, respectively. The first disk block of each request always requires a seek operation.

To check a record, the record pointer is obtained, the record is read, and the record is scanned to test whether it matches the query. The false drop resolution time for one record, $T_{resolve}$, is computed as follows,

$$T_{resolve} = \left(1 - \frac{PB}{N}\right)\cdot Read\left(\left\lceil \frac{PB\cdot P_{size}}{B_{size}} \right\rceil\right) + Read(RB) + T_{scan}, \quad (7)$$

where T_{scan} is the time required to compare a record with the query and RB is the average number of disk blocks that must be accessed to read a record. In the above equation obtaining the record pointer can be explained as follows. PB record pointers, each occupying P_{size} bytes, are read into a buffer of $PB\cdot P_{size}$ bytes long at the database initialization stage. Since this is a one time cost, it is excluded from the cost calculations. The probability of finding a requested record pointer in the buffer is approximately equal to PB/N. For the databases with fixed length records or when all record pointers are stored in main memory, PB must be equal to N, i.e., the cost of finding the record pointers is zero.

To estimate the false drop probability in partial evaluation of the first phase, we use the *on-bit density* (op) which is the probability of a particular bit of a bit slice being an on-bit [4]. Total number of on-bits in a signature file is $N\cdot F\cdot(1-(1-S/F)^D)$. Since there are $N\cdot F$ bits in the signature file, by assuming the on-bits are uniformly distributed in a record signature and there are no interdependency among the records and among the terms, the on-bit density becomes

$$op = 1 - (1 - S/F)^D. \quad (8)$$

For partial evaluation we will use op^i instead of fd_i provided that $0 < i \leqslant W(Q)_t$ [4].

To find the i value, the number of bit slices used in the first phase for a t term query, for the minimum response time for given S, F, D, and N values, we replace FD_i with $N\cdot op^i$ in Eq. (4) and we take the derivative of $RT(i)$ with respect to i. The result is:

$$\frac{dRT(i)}{di} = T_{slice} + N\cdot T_{resolve}\cdot op^i\cdot \ln op. \quad (9)$$

To find the optimum number of evaluation steps, i, we let Eq. (9) equal 0 and solve it for i.

$$i = \ln\left(\frac{T_{slice}}{N\cdot T_{resolve}\cdot(-\ln op)}\right)/\ln op. \quad (10)$$

If reaching the stopping condition requires more on-bits than the query signature contains, i.e., $i > W(Q)_t$, i is taken as $W(Q)_t$. The on-bits used in the query evaluation are selected from the query terms using a round robin approach (the first on-bit comes from the first query term, the second on-bit comes from the second query term, and so on). This ensures that each query term contributes to the query evaluation.

To find an intuitive explanation of the stopping condition, we substitute $\ln op \cong op - 1$ [2] in Eq. (9) and we obtain

$$T_{slice} = N\cdot op^i\cdot(1-op)\cdot T_{resolve}. \quad (11)$$

In Eq. (11), $N\cdot op^i\cdot(1-op)$ gives the expected number of 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 or equal to the time required to resolve these false drops by accessing the actual records.

[2] Since $0 < op \leqslant 0.5$ holds, by taking $k = op - 1$ we can apply the linear approximation $\ln(k+1) \cong k$.

3. Minimizing the response time in P-BSSF

The stopping condition may leave unused on-bits in the query signatures. For such configurations decreasing the S value while keeping the F value unchanged decreases the on-bit density. Each step eliminates more false drops with lower on-bit density. Consequently, the stopping condition is satisfied by processing less bit slices and the response time decreases. On the other hand, the reduced S value must provide enough on-bits in the query signatures to reach the stopping condition.

Optimizing the signature file parameters according to a specific number of query terms may give poor performance in a multi-term query environment. Therefore, the submission probabilities of queries with varying number of terms must be considered in the optimization of signature file parameters. The expected response time in a multi-term query environment can be computed as follows,

$$TR = \sum_{t=1}^{t_{\max}} P_t \cdot RT(S, t), \tag{12}$$

where P_t is the probability of submission of a t term query, and $t_{\max}$ is the maximum number of terms that can be used in a query. $RT(S, t)$ is the expected response time of a t term query expressed as a function of S and t as follows,

$$RT(S, t) = i \cdot T_{slice} + \left(1 - (1 - S/F)^D\right)^i \cdot N \cdot T_{resolve}, \tag{13}$$

where i is computed with Eq. (10) and $0 < i \leqslant F \cdot (1 - (1 - S/F)^t)$ holds.

The derivative of $RT(S, t)$ with respect to S is very complicated. Since S must be an integer between 1 and $F \cdot \ln 2/D$ (upper bound corresponds to the S value which satisfies the optimality condition), the domain of S is finite and very small (note that $S \ll F$). Therefore, the optimum value of S that gives the minimum TR can be found with a linear search for given F, D, and N values as illustrated in Fig. 1.

```
MinimumResponseTime ← infinity
for NewS = 1 to ⌈F · ln 2/D⌉ do
  NewResponseTime ← Compute total query evaluation time with Eq. (12) using NewS
  if NewResponseTime < MinimumResponseTime then
    S ← NewS
    MinimumResponseTime ← NewResponseTime
  endif
endfor
```

Fig. 1. Algorithm to find the optimum S value.

4. Experimental results

To estimate the performance of P-BSSF a simulation environment is designed. The aim of the experiments is to analyze the change in the performance of the proposed method as the values of important input parameters change. Data record statistics are obtained by inspecting the MARC records of Bilkent University Library collection. MARC records are widely used to store and distribute the bibliographic information about various types of materials such as books, films, slides, videotapes, etc. A 33 MHz, 486 DX personal computer with a hard disk of 360 MB running under DOS is used to test the performance of the proposed method. We prefer to use the DOS environment since it provides exclusive control of all resources. Also, controlling the sequentiality probability is easy in the DOS environment. The values of the variables are determined experimentally and they are given in Table 1 (the UNIX case is used later). As an aside we also provide the total number of distinct terms of the database and it is 166,216.

To compare the performance of P-BSSF and BSSF in multi-term query environments, three different query cases are considered: Uniform Distribution (UD), Low Weight (LW), and High Weight (HW) queries. P_t $(1 \leqslant t \leqslant 5)$ values for these distributions are given in Table 2.

Expected response time values of the query cases obtained by simulation runs for changing F values are plotted in Fig. 2 for $SP = 1$. The (percentage) space overhead for a given F value is defined as $100 \cdot F/(8 \cdot 613)$ where the average record length of the test database is 613 bytes. In P-BSSF, the difference among the response times of the LW, UD, and

Table 1
Parameter values for the simulation runs and experiments (UNIX values, if different)

t_{max}	= 5		maximum number of terms in a query
B_{size}	= 8192		size of a disk block (bytes)
D	= 25.7		average number of terms in a record
N	= 152.850		number of records
P_{size}	= 4		size of a record pointer (bytes)
PB	= 2048		number of record pointers in the record pointer buffer
RB	= 1		average number of disk block accesses to retrieve a record
T_{bitop}	= 0.98	(0.5)	time required to perform bit operations between two memory words (microseconds)
T_{read}	= 5.77	(2.5)	time required to read a disk block (milliseconds, ms)
T_{scan}	= 4.5	(2.7)	average time required to match a record with query (ms)
T_{seek}	= 30	(17)	time required to position the read head of disk (ms)
W_{size}	= 4		size of a memory word (bytes)

HW query cases become insignificant for space overheads greater than 16%. Therefore, we include only the UD query case in Fig. 2.

In BSSF, S increases for increasing F since S is adjusted to satisfy the optimality condition for each F value. At lower space overheads, the query signature contains insufficient on-bits which produces many false drops. Since the weight of the query signature increases for increasing F value, the response time decreases rapidly until the expected number of false drops is reduced to an optimum value. Increasing the F value after reaching the optimum point just increases the response time due to processing additional bit slices without eliminating any false drops. Therefore, there is an optimum space overhead for each N value that provides minimum response time for BSSF. For smaller N values or higher t values minimum response time is obtained at lower space overheads.

In P-BSSF, S is adjusted for each F value to obtain minimum response time. At lower space overheads, the weights of the queries are insufficient to reduce the expected number of false drops to the optimum value. Therefore, both methods produce similar results until sufficient on-bits are obtained in the query signatures. For P-BSSF, unlike the BSSF method, increasing the signature size after obtaining sufficient on-bits in the query signature reduces op, that causes a decrease in the response time. For $N = 10^6$, $SP = 1$, and $F = 1400$ the LW, UD, and HW query cases obtain expected response times of 1.02, 1.00, and 0.97 seconds, respectively.

Table 2
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

Higher numbers of query terms provide more on-bits in the query signature. Therefore, for P-BSSF, S can take smaller values that provide lower op values. Consequently, the stopping condition is reached by processing fewer numbers of bit slices and the response time of P-BSSF decreases for increasing number of query terms. This property makes P-BSSF a promising method for the applications with high number of query terms, such as image databases [10].

Since the response time of BSSF increases for the F values greater than the optimum space overhead, the space overhead must be fixed at the optimum F value. We can compute the performance improvement of P-BSSF over BSSF with respect to additional space overhead incurred by selecting a higher F value for P-BSSF. For example, P-BSSF provides a query processing time improvement of 85% over the BSSF method with a 14% increase in the space overhead with respect to BSSF for the UD query case.

The simulation runs for various SP values show that similar performance improvements are achieved for smaller SP values while the response times of both methods increase for decreasing SP value.

We measure the response time of the proposed method with real data used to obtain data record statistics by using zero hit queries which is the worst

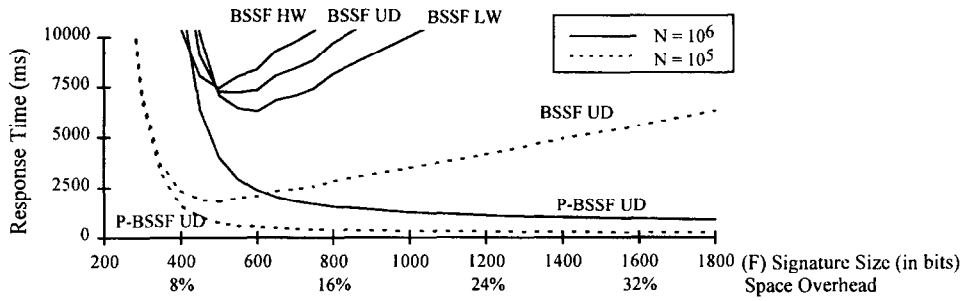

Fig. 2. Expected response time versus F in a multi-term query environment ($SP = 1$).

case. To find the first relevant record, all false drops must be eliminated for zero hit queries since there are no relevant record. To smooth the differences among the results of queries, a query set containing 1000 zero hit queries is generated randomly by considering the occurrence probabilities of number of query terms (P_t) for each query case.

The expected and measured response time values are plotted in Fig. 3. Since the expected response times of the query cases are very close, we give only the HW case which obtains the lowest response times.

The number of terms in the records of the test database varies. The largest record contains 166 terms and there are 178 records containing more than 99 terms. Although the observed and the estimated average on-bit density values are very close, signatures of large records have very high on-bit densities for small F values. These large records cause an increase in the observed number of false drops. Consequently, the observed response time is higher than the expected response time. For larger F values the on-bit densities of these large records are smaller, hence the difference between expected and observed response time values decreases considerably.

The experiments show that obtaining a response time around 0.5 seconds is possible for the test database containing 152,850 MARC records with a space overhead between 24% and 32% by using a personal computer. We repeated the same experiments in the UNIX environment by using a Sparc Server Model 10-51 (see Table 1). About 55 other users were running SQL processes using the library collection database of Bilkent University during the experiments. We obtained very promising response times in such a multi-user environment where the value of SP can be considered as zero. For example, for $F = 1400$ which corresponds a 28% space overhead, the LW, UD, and HW query cases obtain the response times of 0.62, 0.46, and 0.41 seconds, respectively. (All UNIX numbers are obtained using the "elapsed time" feature of the system.)

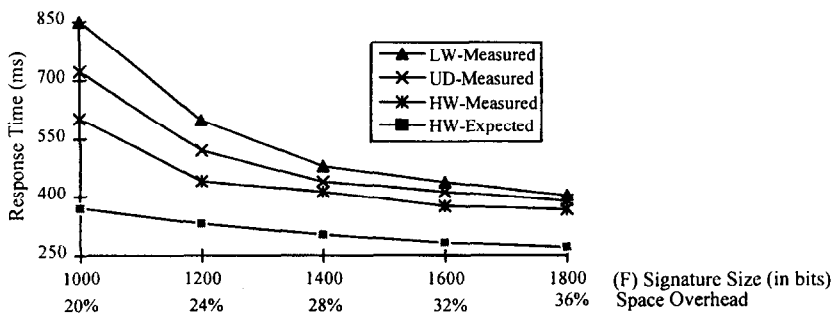

Fig. 3. Expected and measured response time versus F ($SP = 1$, $N = 152{,}850$).

5. Conclusion

Optimizing the signature file for a fixed number of query terms may give undesirable results for other queries containing different number of terms. In this study the response time is optimized by considering multi-term queries along with the probability of submission of such queries applying a partial evaluation approach. Depending on the database and query statistics' parameters, simulation results show that the proposed signature generation method may provide up to 85% performance improvement over the Bit-Sliced Signature File method.

The contributions of this paper are: A stopping condition is defined for the partial evaluation of the queries for bit-sliced signature storage model. Variable numbers of query terms are considered in the minimization of the response time with the relaxation of the optimality condition.

Our current research involves the comparison of the proposed method with other signature file organization methods. Also, for the databases with varying record lengths, using actual numbers of terms of the records in the optimization of the signature file parameters instead of using average number of terms is being investigated.

References

[1] D. Aktug and F. Can, Signature files: An integrated access method for formatted and unformatted databases, submitted to *ACM Computing Surveys* (under revision).

[2] S. Christodoulakis and C. Faloutsos, Design considerations for a message file server, *IEEE Trans. Software Engineering* **10** (2) (1984) 201–210.

[3] Y. Ishikawa, H. Kitagawa and N. Ohbo, Evaluation of signature files as set access facilities in OODBs, in: *Proc. ACM SIGMOD '93 Conf.*, Washington, DC (1993) 247–256.

[4] S. Kocberber and F. Can, Generalized vertical partitioning of signature files, Tech. Rept. BU-CEIS-9501, Dept. of Computer Engineering and Information Science, Bilkent University, 1995, ftp://ftp.cs.bilkent.edu.tr/pub/tech-reports/1995/BU-CEIS-9501.ps.z

[5] Z. Lin and C. Faloutsos, Frame-sliced signature files, *IEEE Trans. Knowledge and Data Engineering* **4** (3) (1992) 281–289.

[6] Z. Lin and C. Faloutsos, Frame-sliced signature files, Tech. Rept. CS2146 and UMIACS-TR-88-88, Computer Science Dept. University of Maryland, 1988.

[7] G. Panagopoulos and C. Faloutsos, Bit-sliced signature files for very large text databases on a parallel machine architecture, in: *Proc. EDBT'94 Conf.*, Cambridge, MA (1994) 379–392.

[8] C.S. Roberts, Partial-match retrieval via the method of superimposed codes, in: *Proc. IEEE* **67** (12) (1979) 1624–1642.

[9] R. Sacks-Davis, A. Kent and K. Ramamohanarao, Performance of multikey access method based on descriptors superimposed coding techniques, *Inform. Systems* **10** (4) (1987) 391–403.

[10] P. Zezula, F. Rabitti and P. Tiberio, Dynamic partitioning of signature files, *ACM Trans. Inform. Systems* **9** (4) (1991) 336–367.