Composite Bit-Sliced Signature File: An Efficient Access Method for Set-Valued Object Retrieval

Hiroyuki KITAGAWA [1] and Kohei FUKUSHIMA [2]

[1] Institute of Information Sciences and Electronics, University of Tsukuba
[2] College of Information Sciences, University of Tsukuba
Tennohdai, Tsukuba, Ibaraki 305, Japan
email:{kitagawa, kohei}@dblab.is.tsukuba.ac.jp

Abstract

Advanced database systems have to support complex object structures. Efficient processing of set-valued object retrieval (simply, set retrieval) based on the set containment conditions is important for such systems. In our previous work, we have shown the potential capability of Bit-Sliced Signature File (BSSF) in handling "has-subset" and "is-subset" queries. When parameter values are properly tuned, BSSF attains more efficient query processing. However, "has-subset" and "is-subset" queries present somewhat conflicting requirements for desirable parameter values. Therefore, it is not an easy task to tune the BSSF organization to both "has-subset" and "is-subset" queries. In this paper, we present a new signature file organization named Composite Bit-Sliced Signature File (C-BSSF) to alleviate the problem, and analyze its basic properties. According to the analysis result, proper C-BSSF outperforms BSSF in environments where both "has-subset" and "is-subset" queries are probable.

1 Introduction

Advanced database systems have to support storage and retrieval of a variety of data objects. They are required to manipulate text, image, video and audio data as well as conventional fixed length records. In another dimension, they need to handle complex object structures built with constructs such as tuples, sets, and lists. Efficient access methods are indispensable for those systems to attain proper performance. The *signature file* is one of promising access methods to facilitate object manipulation in advanced database systems [2-4, 6, 17, 18, 20]. The signature file is a collection of *signatures*, each of which is a bit vector formed for a data object representing its property. The signature file was originally designed for text retrieval, and a lot of study has been done in the context [2-7]. However, its basic concept can be straightforwardly applicable to other kinds of data objects. Its applications to non-text object retrieval have been reported, such as to the record retrieval [9, 16], to the Prolog clause retrieval [15], to the image data retrieval [14], to the set-valued object retrieval [8, 10, 11], and to the object navigation in the object-oriented databases [12, 19].

We have proposed application of superimposed coded signature files to set-valued object retrieval (shortly, *set retrieval*) and showed its potential applicability [8,10,11]. The set retrieval means retrieval of data objects based on the set containment conditions (*e.g.*, $\supseteq, \subseteq$) from a large number of set-valued objects stored in the database. The concept of set retrieval includes the conventional uses of signature files as its instances. For example, the traditional text retrieval, namely retrieval of text objects which contain a given set of key words, is a kind of "has-subset" query. In advanced database systems such as nested relational database systems and object-oriented database systems, "is-subset" queries have to be supported as well as "has-subset" queries [1]. As shown in our previous work [8, 10, 11], it is a feature of signature files that they can symmetrically support both "has-subset" and "is-subset" queries. Although indexes based on B-tree can support "has-subset" queries pretty well[†], they are very poor in dealing with "is-subset" queries [8].

It is known that, in retrieval with signature files, the page access required to look up signatures is a dominant cost factor. To reduce the page access cost, a number of signature file organizations have been proposed [3, 18, 20]. *Bit-Sliced Signature File (BSSF)* [16] is one of well-known signature file organizations. Let the signature length be F. In BSSF, the collection of signatures of F bit length is vertically decomposed into F bit-slices, and each bit-slice is stored in a set of allocated disk pages. To process retrieval requests, we only have to scan some of the F bit-slices, which brings about the reduction of retrieval cost. This cost reduction equally applies to processing both "has-subset" and "is-subset" queries. In processing "has-subset" queries, the number of bit-slices which needs to be scanned decreases as the *weight* of the signature, namely the number of bits set to "1," becomes smaller. This generally leads to the reduction of retrieval cost. In contrast to this, in processing "is-

[†] Compression-based efficient inverted files are proposed in [21,22].

FACTORY		
OID	FNAME	PRODUCTS
		P N A M E
oid1	Tsukuba	pen pencil cutter
oid2	Tokyo	pen cutter eraser scissors

Figure 1: Objects Having Set Attribute Values

subset" queries, the cost reduction becomes generally larger as the weight of the signature becomes bigger. Because of this contradicting property, it is actually not an easy task to tune the BSSF organization to both "has-subset" and "is-subset" queries.

In this paper, we present a new signature file organization method, named *Composite Bit-Sliced Signature File (C-BSSF)*, to alleviate the problem. To compromise the above conflicting requirements of the "has-subset" and "is-subset" queries, each signature in C-BSSF consists of two sub-signatures whose design parameters can be determined independently of each other. Therefore, one can be tuned to the "has-subset" query and the other to the "is-subset" query. C-BSSF is a generalization of BSSF and includes it as a special case. We develop formulas to estimate the cost of set retrieval with C-BSSF. Then, the processing cost of C-BSSF is intensively analyzed, and compared with that of BSSF with the same storage overhead.

Several signature extraction methods have been studied up to the present. Although simple signature extraction schemes as shown in Subsection 2.1 are often used, there are some sophisticated methods [5, 7, 9, 13]. Most of them are to handle skew in occurrence and query frequencies of document terms [5, 7, 13]. Kent and others proposed the multiple block descriptor file [9], which uses multiple signatures. However, their main objective is to attain high selectivity with multiple filters and is definitely different from ours.

The remaining part of this paper is organized as follows. In Section 2, we give an overview of set retrieval with signature files, in particular with BSSF. Section 3 presents C-BSSF and its query processing scheme. In Section 4 , we derive formulas to estimate the cost involved in query and update processing with C-BSSF. In Section 5, the processing cost of C-BSSF is analyzed based on the formulas derived in Section 4, and it is compared with that of BSSF. Section 6 concludes this paper.

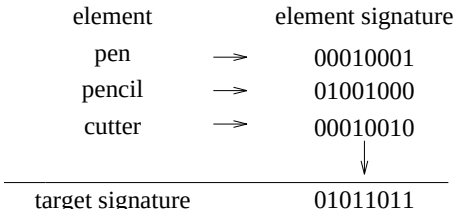

Figure 2: Generation of Target Signature

2 Set Retrieval with Signature Files

2.1 Basic Concepts

Fig. 1 is a sample nested relation representing factory objects. Each object is tagged with the object identifier (OID) and has attributes FNAME and PRODUCTS. FNAME represents the factory name. PRODUCTS is a set attribute and represents the set of products of each factory. Object retrieval based on the PRODUCTS attribute value, for instance PRODUCTS $\supseteq$ {"pen","cutter"}, is an example set retrieval considered in this paper. In this case, each PRODUCTS attribute value is called a *target set*, and {"pen","cutter"} is called a *query set*. The query is called a $T \supseteq Q$ or "has-subset" query. When the set comparison operator $\subseteq$ is used instead of $\supseteq$, the query is called a $T \subseteq Q$ or "is-subset" query.

As mentioned in Section 1, a signature is a bit vector formed for each object. For set retrieval, a *set signature* is generated for each target set and stored in the signature file. The typical procedure used in generating set signatures is as follows. First, an *element signature* is generated for each set element by hashing. Every element signature has F-bit length, and m bits are set to "1." F and m are design parameters and are determined based on the performance requirement and the allowable storage overhead. Then, a set signature is obtained by bit-wise OR-ing (*superimposed coding*) element signatures of all the elements in the target set and stored in the signature file with the corresponding OID. The set signatures generated for target sets are called *target signatures*. Fig. 2 shows the generation of the target signature for set {"pen","pencil","cutter"}. The number of "1"'s in the signature is called the *weight*.

When a query is given, the *query signature* is generated from the query set in the same way as target signatures. Then, each target signature in the signature file is examined over the query signature for potential match. If the target signature satisfies the following condition, the corresponding object becomes a *drop* which may satisfy the query condition:

$$T \supseteq Q : \text{query signature} \wedge \text{target signature} = \text{query signature}$$
$$T \subseteq Q : \text{query signature} \wedge \text{target signature} = \text{target signature},$$

where $\wedge$ stands for the bit-wise AND operation. The last step of query processing is called *false drop resolution*, and each drop is read and examined as to whether it actually satisfies the query condition. Drops that

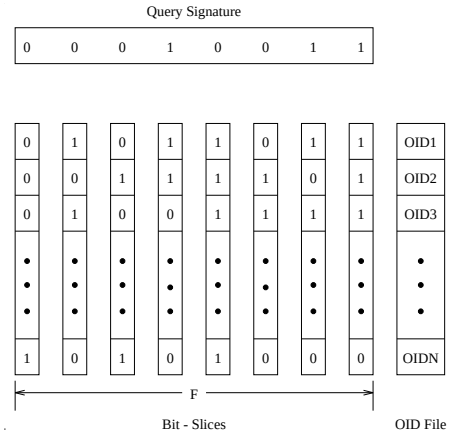

Figure 3: BSSF

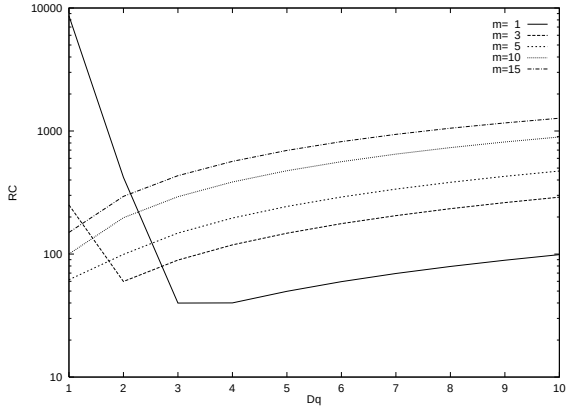

Figure 4: BSSF Retrieval Cost for $T \supseteq Q$

fail the test are called *false drops*, while qualified objects are called *actual drops*.

2.2 Set Retrieval with Bit-Sliced Signature Files

It is known that the page access required to look up signatures is a dominant cost factor in retrieval with signature files. *Bit-Sliced Signature File (BSSF)* [16] is one of well-known signature file organizations to alleviate this problem. Fig. 3 shows organization of BSSF. BSSF stores signatures in a column-wise manner using F files (called *bit-slices*), one per each bit position in the set signature. In retrieval, only some of the bit-slices have to be scanned in BSSF, which attains the reduction of retrieval cost. For example, given the query signature shown in Fig. 3, we only have to look up the three (five) bit-slices corresponding to bit positions of "1" ("0") in the query signature to process the $T \supseteq Q$ ($T \subseteq Q$) query.

In our previous work [8, 10], we analyzed the cost of set retrieval with BSSF. In the analysis, we made the following assumptions.

- The cardinality of every target set is the same value D_t.
- The elements of a target set are randomly selected from a certain domain.

Here, we briefly review the retrieval cost of BSSF. To simplify the discussion, we assume that there is no actual drop.

Fig. 4 shows the retrieval cost for $T \supseteq Q$. In this example, the signature size $F = 400$, the number of objects $N = 320000$, and $D_t = 10$. The x-axis shows the cardinality D_q of the query set, and the y-axis shows the total retrieval cost $RC_{\{T \supseteq Q\}}$ (including the cost for the false drop resolution) measured in the number of page accesses. For small m-values, $RC_{\{T \supseteq Q\}}$ is rather high for small D_q-values because of many false drops. As D_q increases, $RC_{\{T \supseteq Q\}}$ first decreases drastically. However, after taking the minimum value $RC^{min}_{\{T \supseteq Q\}}$, $RC_{\{T \supseteq Q\}}$ starts increasing. The reason is that the number of bit-slices to be scanned monotonically increases as a function of D_q and that it becomes a dominant cost factor. For larger m-values, $RC_{\{T \supseteq Q\}}$ takes the minimum value $RC^{min}_{\{T \supseteq Q\}}$ for $D_q = 1$ and monotonically increases. As exemplified here, the weight m of the element signature has much effect on the retrieval cost. Generally, smaller m-values give lower $RC_{\{T \supseteq Q\}}$ except for very small D_q-values.

In [8], we proposed an efficient query processing scheme for BSSF, named *smart retrieval*. In the object retrieval with signature files, it is finally decided at the false drop resolution step whether a candidate object actually satisfies the query condition or not. Therefore, we can obtain correct answers even if we omit scanning some bit-slices. Of course, when we skip some bit-slices, the false drops may increase to some extent. However, the reduction of the bit-slice scan cost often compensates for the increase of the false drops. Suppose $RC_{\{T \supseteq Q\}}$ takes $RC^{min}_{\{T \supseteq Q\}}$ for $D_q = {D_q}^{min}_{\{T \supseteq Q\}}$ in some BSSF configuration. Then, the smart retrieval is done as follows:

- When $D_q \leq {D_q}^{min}_{\{T \supseteq Q\}}$, the bit-slices are scanned in an ordinal way.
- When $D_q > {D_q}^{min}_{\{T \supseteq Q\}}$, the query signature is created from any ${D_q}^{min}_{\{T \supseteq Q\}}$ elements in the query set and the bit-slices are scanned as if $D_q = {D_q}^{min}_{\{T \supseteq Q\}}$.

Let the expected weight of the query signature for $D_q = {D_q}^{min}_{\{T \supseteq Q\}}$ be $f_{\{T \supseteq Q\}}$. Then, the smart retrieval scheme is actually implemented by scanning $f_{\{T \supseteq Q\}}$ bit-slices for $D_q > {D_q}^{min}_{\{T \supseteq Q\}}$. Fig. 5 shows the smart retrieval cost for $T \supseteq Q$. Under the smart retrieval scheme, $RC_{\{T \supseteq Q\}}$ is constant for $D_q \geq {D_q}^{min}_{\{T \supseteq Q\}}$. Note that smaller m-values generally give lower $RC_{\{T \supseteq Q\}}$ even under the smart retrieval scheme.

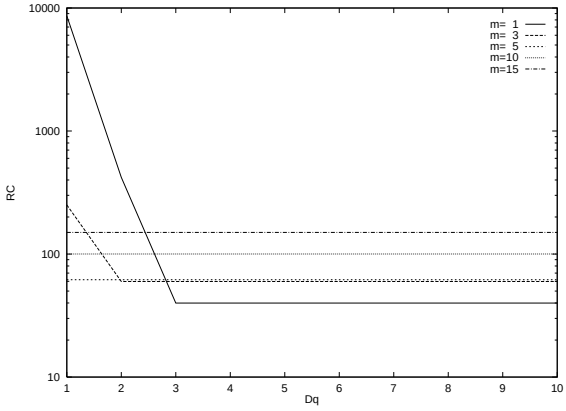

Figure 5: BSSF Smart Retrieval Cost for $T \supseteq Q$

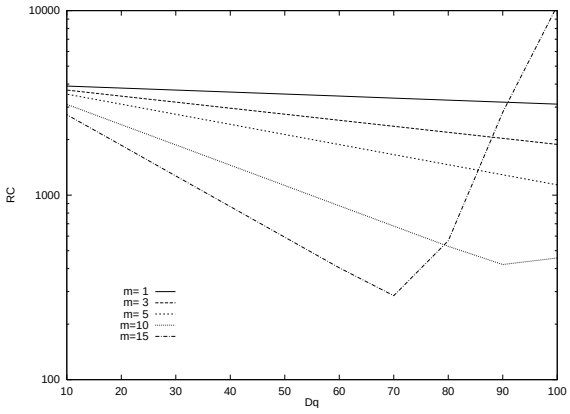

Figure 6: BSSF Retrieval Cost for $T \subseteq Q$

Fig. 6 shows the total retrieval cost $RC_{\{T \subseteq Q\}}$ of the same set of BSSFs for $T \subseteq Q$. The behavior of $RC_{\{T \subseteq Q\}}$ is in some sense "symmetric" to that of $RC_{\{T \supseteq Q\}}$. $RC_{\{T \subseteq Q\}}$ is generally high for small D_q-values due to the bit-slice scan cost. The increase of $RC_{\{T \subseteq Q\}}$ of BSSFs with $m = 10$ and $m = 15$ for large D_q-values is due to the increase of false drops. The smart retrieval scheme is also applicable to process $T \subseteq Q$ queries. In the case of $T \subseteq Q$, bit-slices corresponding to "0" bit positions in the query signature are scanned. Let the expected number of such bit-slices for $D_q = D_q{}^{min}_{\{T \subseteq Q\}}$ be $f_{\{T \subseteq Q\}}$. Then, the smart retrieval scheme is implemented by scanning $f_{\{T \subseteq Q\}}$ such bit-slices. Fig. 7 shows $RC_{\{T \subseteq Q\}}$ under the smart retrieval scheme, which is constant for $D_q \leq D_q{}^{min}_{\{T \subseteq Q\}}$. In contrast to the case of $T \supseteq Q$, note that larger m-values generally give lower $RC_{\{T \subseteq Q\}}$ for $T \subseteq Q$.

3 Composite Bit-Sliced Signature File (C-BSSF)

As exemplified in Section 2, efficiency of BSSF is affected by the weight m of the element signature. $T \supseteq Q$ queries generally prefer smaller m, while $T \subseteq Q$ queries prefer the opposite. Therefore, when we need to process both $T \supseteq Q$ and $T \subseteq Q$ queries, we have to compromise the conflicting requirements.

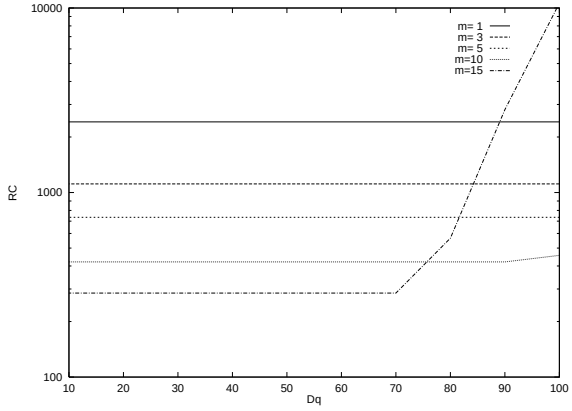

Figure 7: BSSF Smart Retrieval Cost for $T \subseteq Q$

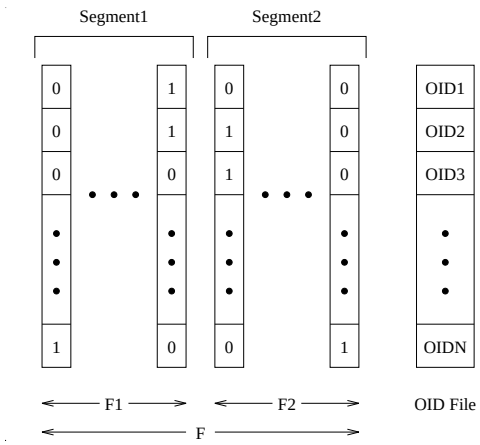

Figure 8: C-BSSF

As an approach to the problem, we propose a new signature file organization named *Composite Bit-Sliced Signature File* (*C-BSSF*). As shown in Section 5, C-BSSF with appropriate parameter values can outperform BSSF for mixtures of $T \supseteq Q$ and $T \subseteq Q$ queries. Each signature S in C-BSSF is a concatenation of two sub-signatures S_1 and S_2, whose design parameters are determined independently of each other. F- and m-values used for sub-signature S_i are denoted by F_i and m_i, respectively. C-BSSF stores the signature in a column-wise manner as BSSF. Fig. 8 illustrates organization of C-BSSF. The collection of bit-slices storing sub-signatures $S_1(S_2)$ is called *segment* 1 (2). C-BSSF can be regarded as a concatenation of two BSSFs with different design parameters sharing the same OID file. Note that C-BSSF becomes BSSF when $F_1 = 0$ or $F_2 = 0$. Thus, C-BSSF is a generalization of BSSF.

Query processing with C-BSSF is almost the same as with BSSF except that the query signature consists of two sub-signatures. The smart retrieval scheme is also applicable to C-BSSF. Let us consider the case

of $T \supseteq Q$. The essential point of the smart retrieval scheme presented in Subsection 2.2 is that only some X ($X \leq D_q$) elements are used to generate the query signature instead of all the D_q elements given in the query set. When we apply the smart retrieval scheme to C-BSSF, we can take different X values (e.g., X_1 and X_2) for the two segments of C-BSSF. For example, $X_1 \neq 0$ and $X_2 = 0$ implies that only some of the bit-slices in the segment 1 are scanned and that the segment 2 is not used. Of course, we can scan bit-slices both in the segments 1 and 2, too. The choice depends on the estimated retrieval cost. The same remark applies to the smart retrieval scheme for $T \subseteq Q$.

Table 1: Symbols and Their Meanings

Symbol	Meaning
D_t	cardinality of a target set
D_q	cardinality of a query set
N	number of objects
P	disk page size in bytes
b	number of bits per byte
oid	OID size in bytes
P_o	number of page accesses to fetch an object
F_d	false drop probability
F	signature size in bits
F_1, F_2	sub-signature size in bits
m_1, m_2	number of "1"'s in an element sub-signature
m_{t_1}, m_{t_2}	expected number of "1"'s in a target sub-signature
m_{q_1}, m_{q_2}	expected number of "1"'s in a query sub-signature

4 Cost Model

In this section, formulas to estimate the retrieval cost and the update cost of C-BSSF are derived. They are measured in the number of page accesses. We make the same assumptions as stated in Subsection 2.2. We also assume that there is no actual drop. Symbols and their meanings are summarized in Table 1.

4.1 Retrieval Cost Model

The retrieval cost RC is given by Eq. (1):

$$RC = LC_{sig} + LC_{OID} + P_o \cdot Fd \cdot N. \tag{1}$$

Here, LC_{sig} is the cost to scan bit-slices, LC_{OID} is the access cost for the OID file, and $P_o \cdot Fd \cdot N$ is the cost for false drop resolution. The false drop probability Fd is defined as follows:

$$Fd = \frac{false\ drops}{N - actual\ drops}. \tag{2}$$

Following the discussion on the false drop probability for BSSF [8, 10], Fd for C-BSSF can be given by Eqs. (3) and (4):

$$Fd_{\{T \supseteq Q\}} = \left(\frac{m_{t_1} C_{m_1}}{F_1 C_{m_1}}\right)^{D_q} \left(\frac{m_{t_2} C_{m_2}}{F_2 C_{m_2}}\right)^{D_q} \tag{3}$$

$$Fd_{\{T \subseteq Q\}} = \left(\frac{m_{q_1} C_{m_1}}{F_1 C_{m_1}}\right)^{D_t} \left(\frac{m_{q_2} C_{m_2}}{F_2 C_{m_2}}\right)^{D_t}, \tag{4}$$

where

$$m_{t_i} = F_i \left[1 - \left(1 - \frac{m_i}{F_i}\right)^{D_t}\right] \tag{5}$$

$$m_{q_i} = F_i \left[1 - \left(1 - \frac{m_i}{F_i}\right)^{D_q}\right]. \tag{6}$$

LC_{sig} is given by Eqs. (7) and (8):

$$LC_{sig\{T \supseteq Q\}} = (m_{q_1} + m_{q_2}) \left\lceil \frac{N}{Pb} \right\rceil \tag{7}$$

$$LC_{sig\{T \subseteq Q\}} = (F_1 + F_2 - m_{q_1} - m_{q_2}) \left\lceil \frac{N}{Pb} \right\rceil. \tag{8}$$

Note that $m_{q_1} + m_{q_2}$ and $F_1 + F_2 - m_{q_1} - m_{q_2}$ represent the numbers of scanned bit-slices for $T \supseteq Q$ and $T \subseteq Q$, respectively. LC_{OID} is simply estimated as follows:

$$LC_{OID} = SC_{OID} \cdot min\left(Fd \left\lfloor \frac{P}{oid} \right\rfloor, 1\right), \tag{9}$$

where $SC_{OID} = \left\lceil \frac{N}{\lfloor \frac{P}{oid} \rfloor} \right\rceil$ is the storage cost of the OID file.

The smart retrieval scheme has not been considered in deriving the above formulas. To derive the retrieval cost under the smart retrieval scheme, we need to modify Eqs. (3) and (6) as follows:

$$Fd_{\{T \supseteq Q\}} = \left(\frac{m_{t_1} C_{m_1}}{F_1 C_{m_1}}\right)^{X_1} \left(\frac{m_{t_2} C_{m_2}}{F_2 C_{m_2}}\right)^{X_2} \tag{3'}$$

$$m_{q_i} = F_i \left[1 - \left(1 - \frac{m_i}{F_i}\right)^{X_i}\right], \tag{6'}$$

where explanation for X_1 and X_2 has been given in Section 3. Then, the smart retrieval cost RC is expressed by Eqs. (10) and (11):

$$RC_{\{T \supseteq Q\}} = min_{X_1, X_2 \leq D_q} RC(X_1, X_2) \tag{10}$$

$$RC_{\{T \subseteq Q\}} = min_{X_1, X_2 \geq D_q} RC(X_1, X_2). \tag{11}$$

Here, $RC(X_1, X_2)$ is given by Eq. (1), in which Eqs. (3') and (6') are used instead of Eqs. (3) and (6).

4.2 Update Cost Model

We assume that a new object is always appended at the end of the bit-slices and the OID file and that newly allocated disk pages are initially zero cleared. Deletion is assumed to be done by setting a delete flag in the OID file. Then, the insertion cost IC and the deletion cost DC are given as follows:

$$IC = 2(m_{t_1} + m_{t_2} + 1) \tag{12}$$

$$DC = \frac{SC_{OID}}{2} + 1, \tag{13}$$

where m_{t_1} and m_{t_2} are given by Eq. (5) and $SC_{OID} = \left\lceil \frac{N}{\lfloor \frac{P}{oid} \rfloor} \right\rceil$.

Table 2: Parameter Values

Symbol	Value
D_t	cardinality of a target set (= 10,100)
N	number of objects (= 320,000)
P	size of a disk page (= 4096 bytes)
b	number of bits per byte (= 8)
oid	size of an OID (= 8 bytes)
P_o	number of page accesses to fetch an object (= 1)
F	signature size in bits (= 400,800,1000)

5 Cost Analysis

In this section, the processing cost of C-BSSF is analyzed based on the formulas derived in Section 4. The costs of C-BSSF and BSSF are compared. The retrieval and update costs of BSSF are given by setting $F_1 = F$ and $F_2 = 0$ in the cost formulas for C-BSSF. Parameter values used in the following analysis are listed in Table 2.

5.1 Retrieval Cost

The retrieval cost of C-BSSF for $T \supseteq Q$ ($D_t = 10, F = 400$) is shown in Fig. 9. To compare C-BSSF with BSSF, the retrieval cost of BSSF is also included. In Fig. 9, the smart retrieval is not used. Fig. 10 gives the retrieval cost under the smart retrieval scheme. The retrieval cost for $T \subseteq Q$ ($D_t = 10, F = 400$) is shown in Fig. 11. Fig. 12 gives the retrieval cost under the smart retrieval scheme. Although BSSF with m=3 is efficient for $T \supseteq Q$, it is poor for $T \subseteq Q$. BSSF with m=10 has the opposite property. In contrast to this, C-BSSF with $F_1 = F_2 = 200$, $m_1 = 3, m_2 = 10$ supports both $T \supseteq Q$ and $T \subseteq Q$ queries fairly well in this case.

To evaluate the effectiveness of C-BSSF for mixtures of $T \supseteq Q$ and $T \subseteq Q$ queries, we have derived the cost optimal C-BSSF configuration for given D_t, F, F_1 values. The retrieval cost for a mixture of $T \supseteq Q$ and $T \subseteq Q$ queries is estimated as follows:

$$RC = \alpha \cdot RC^{SR}_{\{T \supseteq Q\}} + (1 - \alpha) \cdot RC^{SR}_{\{T \subseteq Q\}}. \qquad (14)$$

Here, α stands for the probability that a given query is a $T \supseteq Q$ query, and $RC^{SR}_{\{T \supseteq Q\}}$ and $RC^{SR}_{\{T \subseteq Q\}}$ are retrieval costs for $T \supseteq Q$ and $T \subseteq Q$ queries, respectively, under the smart retrieval scheme. $RC^{SR}_{\{T \supseteq Q\}}$ is estimated as $\min_{X_1, X_2 \leq D^L_q} RC(X_1, X_2)$ for a given value of D^L_q. It means that the retrieval cost for $T \supseteq Q$ is at most $RC^{SR}_{\{T \supseteq Q\}}$ under the smart retrieval scheme when $D_q \geq D^L_q$. Similarly, $RC^{SR}_{\{T \subseteq Q\}}$ is estimated as $\min_{X'_1, X'_2 \geq D^H_q} RC(X'_1, X'_2)$ for a given value of D^H_q. We assume $(D^L_q, D^H_q) = (2, 45)$ for $D_t = 10$ and $(D^L_q, D^H_q) = (10, 500)$ for $D_t = 100$.

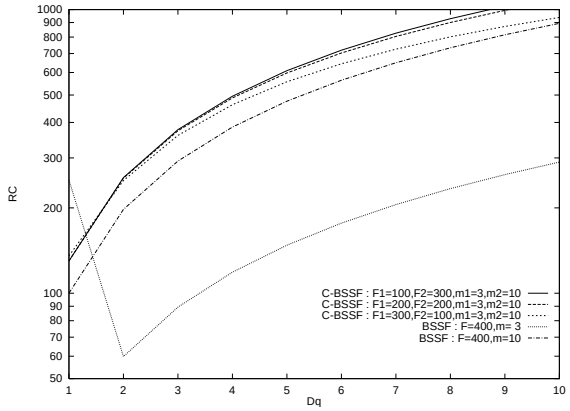

Figure 9: C-BSSF Retrieval Cost for $T \supseteq Q$

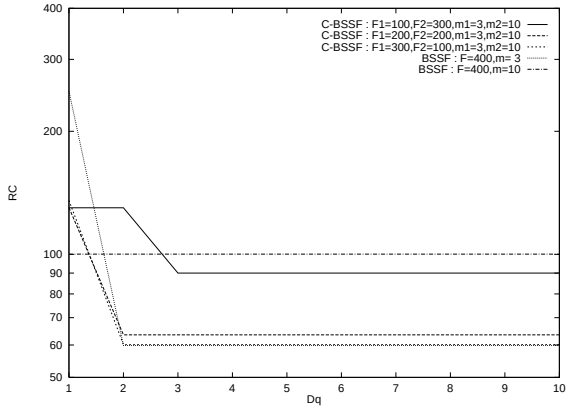

Figure 10: C-BSSF Smart Retrieval Cost for $T \supseteq Q$

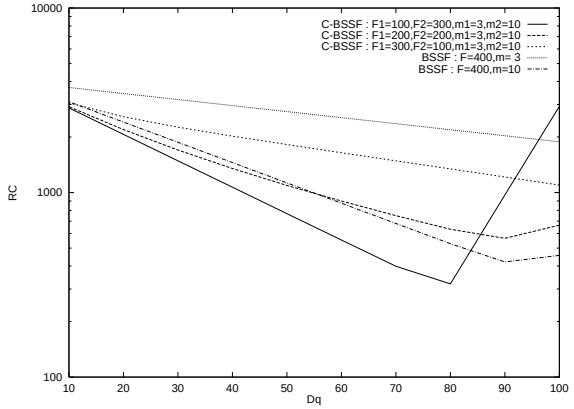

Figure 11: C-BSSF Retrieval Cost for $T \subseteq Q$

Table 3: C-BSSF optimal Retrieval Cost

	α	F_1/F 1/8	1/4	3/8	1/2	1
Dt=10 F=400	1.00	45 (1,2,0,2,-,-)	47 (1,1,2,2,-,-)	45 (1,1,2,2,-,-)	44 (1,1,2,2,-,-)	44 (2,-,2,-,-,-)
	0.75	155 (2,9,0,1,45,105)	125 (3,2,0,2,46,1012)	107 (3,14,2,0,∞,46)	101 (3,10,2,0,∞,45)	176 (11,-,1,-,88,-)
	0.50	193 (4,18,0,1,55,56)	160 (5,22,2,0,57,45)	136 (3,14,2,0,∞,46)	138 (3,11,2,0,∞,45)	209 (20,-,1,-,58,-)
	0.25	181 (4,24,0,1,52,45)	161 (6,22,2,0,50,45)	160 (13,17,0,1,45,45)	161 (14,15,1,0,45,45)	188 (29,-,1,-,45,-)
	0.00	157 (3,26,-,-,48,45)	155 (7,22,-,-,45,45)	154 (12,18,-,-,45,45)	155 (14,15,-,-,45,45)	154 (29,-,-,-,45,-)
Dt=10 F=800	1.00	40 (1,2,0,2,-,-)	39 (1,1,1,2,-,-)	39 (1,1,1,2,-,-)	40 (1,1,1,2,-,-)	40 (2,-,2,-,-,-)
	0.75	120 (3,2,0,2,46,2367)	82 (3,50,2,0,∞,45)	73 (2,39,2,0,∞,45)	71 (2,29,2,0,∞,45)	246 (16,-,1,-,141,-)
	0.50	140 (5,61,2,0,∞,45)	100 (3,50,2,0,∞,45)	97 (2,39,2,0,∞,45)	99 (2,29,2,0,∞,45)	292 (28,-,1,-,94,-)
	0.25	135 (5,61,2,0,∞,45)	110 (3,50,2,0,∞,45)	121 (2,39,2,0,∞,45)	127 (2,29,2,0,∞,45)	259 (51,-,1,-,59,-)
	0.00	125 (11,63,-,-,45,45)	125 (20,54,-,-,46,45)	126 (30,44,-,-,45,45)	126 (36,37,-,-,45,46)	126 (73,-,-,-,45,-)
Dt=100 F=1000	1.00	60 (1,1,0,5,-,-)	62 (1,1,0,6,-,-)	67 (1,1,0,6,-,-)	74 (1,1,0,7,-,-)	55 (1,-,5,-,-,-)
	0.75	147 (1,3,0,3,532,1042)	115 (1,1,0,6,587,5077)	104 (2,1,0,6,553,3866)	105 (1,3,7,0,3370,566)	159 (3,-,3,-,1144,-)
	0.50	169 (1,6,4,2,506,612)	129 (1,6,10,0,∞,504)	132 (1,5,8,0,1770,502)	135 (1,3,7,0,3370,566)	174 (7,-,2,-,596,-)
	0.25	158 (1,8,10,1,550,503)	138 (1,6,10,0,∞,504)	152 (3,6,6,0,503,510)	157 (4,5,5,0,506,506)	162 (8,-,2,-,545,-)
	0.00	142 (1,8,-,-,550,503)	142 (3,6,-,-,503,510)	138 (3,6,-,-,503,510)	141 (4,5,-,-,506,506)	139 (9,-,-,-,503,-)

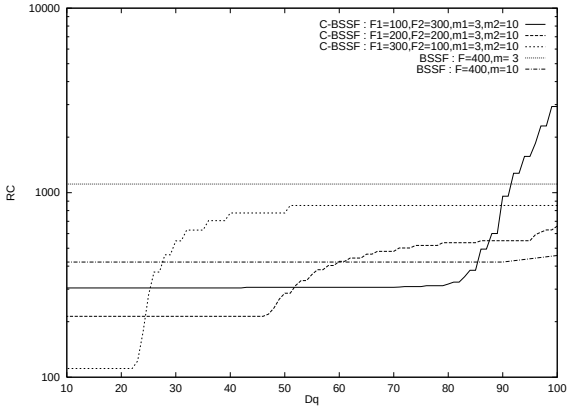

Figure 12: C-BSSF Smart Retrieval Cost for $T \subseteq Q$

Table 4: C-BSSF Retrieval Cost
(Bit-Slices as Units of Performance Tuning)

	α	F_1/F 1/8	1/4	3/8	1/2	1
Dt=10 F=400	1.00	46	46	44	44	44
	0.75	141	126	110	104	152
	0.50	155	143	140	144	169
	0.25	164	161	162	163	163
	0.00	161	157	157	158	157
Dt=10 F=800	1.00	40	39	38	40	40
	0.75	126	83	75	72	152
	0.50	129	102	99	100	169
	0.25	130	120	123	129	154
	0.00	126	126	127	127	127
Dt=100 F=1000	1.00	59	62	67	74	55
	0.75	148	116	104	104	152
	0.50	167	129	134	133	169
	0.25	159	139	153	156	160
	0.00	143	136	139	143	141

Table 3 summarizes the experimental results for $(D_t, F) = (10, 400), (10, 800), (100, 1000)$. Each value gives the optimal retrieval cost of C-BSSF for given D_t, F, F_1, α values derived by the randomized method. The list under the value gives $(m_1, m_2, X_1, X_2, X_1^{'}, X_2^{'})$ for the optimal configuration. Calculation has been done for $F_1/F = 1/8, 1/4, 3/8, 1/2, 1$. Note that C-BSSF becomes BSSF when $F_1 = F$, namely $F_1/F = 1$. As shown in the table, the optimal retrieval cost of C-BSSF is fairly lower than that of BSSF for mixtures of $T \supseteq Q$ and $T \subseteq Q$ queries, namely in cases of $\alpha = 0.25, 0.5, 0.75$. For example, when $D_t = 10, F = 800, \alpha = 0.5$, the optimal C-BSSF configuration is represented by $F_1 = 300, F_2 = 500, m_1 = 2, m_2 = 39$ and its retrieval cost is 97. In contrast to this, $m = 28$ gives the optimal BSSF configuration whose retrieval cost is 292.

Elements in the query set have been considered as basic units of performance tuning in the smart retrieval scheme. Actually, we can finely tune the performance by deciding the number of scanned bit-slices. Because of space limitation, we omit details of analysis when we take bit-slices as basic units of performance tuning. Table 4 summarizes the estimation of the optimal retrieval cost of C-BSSF under the scheme. Although the ratio of the retrieval cost of the optimal BSSF to that of the optimal C-BSSF is reduced to some extent, C-BSSF with proper parameter values still outperforms BSSF.

5.2 Update Cost

We have calculated the insertion and deletion costs of the retrieval cost optimal C-BSSF and BSSF for $(D_t, F) = (10, 400), (10, 800), (100, 1000)$ and $\alpha = 0.5$ (Table 5). They all have the same deletion cost. The insertion cost is determined by the sum of m_1 and m_2. In Subsection 4.2, we have assumed that new disk pages are zero cleared. If we introduce an option that new disk pages are initialized either all "0"'s or all "1"'s, the insertion cost IC is given as follows:

$$IC = 2(min(m_{t_1}, F_1 - m_{t_1}) + min(m_{t_2}, F_2 - m_{t_2}) + 1). \quad (15)$$

Then, the insertion costs of the C-BSSFs and BSSFs

are given in Table 6. As shown in Tables 5 and 6, the update costs of C-BSSF and BSSF are almost comparable.

Table 5: Update Cost

Organization		IC	DC
C-BSSF	(F_1 = 150, F_2 =250, m_1 =3, m_2 =14)	276	313.5
BSSF	(F =400, m =20)	324	313.5
C-BSSF	(F_1 = 300, F_2 =500, m_1 =2, m_2 =39)	596	313.5
BSSF	(F =800, m =28)	482	313.5
C-BSSF	(F_1 = 250, F_2 =750, m_1 =1, m_2 =6)	996	313.5
BSSF	(F =1000, m =7)	1012	313.5

Table 6: Update Cost

Organization		IC
C-BSSF	(F_1 = 150, F_2 =250, m_1 =3, m_2 =14)	276
BSSF	(F =400, m =20)	324
C-BSSF	(F_1 = 300, F_2 =500, m_1 =2, m_2 =39)	484
BSSF	(F =800, m =28)	482
C-BSSF	(F_1 = 250, F_2 =750, m_1 =1, m_2 =6)	840
BSSF	(F =1000, m =7)	992

6 Conclusion

In this paper, we have considered set retrieval with signature files. Bit-Sliced Signature File (BSSF) is a well-known signature file organization, and can be used for set retrieval. However, it cannot cope well with the conflicting requirements of $T \supseteq Q$ and $T \subseteq Q$ queries in tuning parameter values to attain high performance. To alleviate the problem, we have proposed *Composite Bit-Sliced Signature File (C-BSSF)* and analyzed its basic properties. C-BSSF is a generalization of BSSF and has more freedom in parameter setting. Through the analysis, we have demonstrated that C-BSSF with proper parameter values outperforms BSSF in environments where both $T \supseteq Q$ and $T \subseteq Q$ queries are probable.

When the number of objects increases, the retrieval performance of signature files generally degrades because of their sequential property. The vertical decomposition scheme is proposed to alleviate the problem [11,18,20]. For large-scale databases, the vertical decomposition scheme can be incorporated into C-BSSF.

Acknowledgement

This work was supported in part by the Ministry of Education, Science, Sports and Culture, Japan through the Grand-in-Aid for Scientific Research on Priority Areas ♯08244101.

References

[1] R. G. G. Catell, editor, *Object Database Standard: ODMG-93.* Morgan Kaufmann, San Francisco, California, 1996. Rel. 1.2.

[2] C. Faloutsos, "Access Methods for Text," *ACM Comput. Surv.*, 17(1):49-74, 1985.

[3] C. Faloutsos, "Signature-Based Text Retrieval Methods: A Survey," *IEEE Database Eng.*, 13(1):25-32, 1990.

[4] C. Faloutsos and S. Christodoulakis, "Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation," *ACM Trans. Off. Inf. Syst.*, 2(4):267-288, 1984.

[5] C. Faloutsos and S. Christodoulakis, "Design of a Signature File Method that Accounts for Non-Uniform Occurrence and Query Frequencies," *Proc. 11th VLDB*, pages 165-170, Stockholm, 1985.

[6] C. Faloutsos and S. Christodoulakis, "Description and Performance Analysis of Signature File Methods for Office Filing," *ACM Trans. Off. Inf. Syst.*, 5(3):237-257, 1987.

[7] C. Faloutsos and S. Christodoulakis, "Optimal Signature Extraction and Information Loss," *ACM Trans. Database Syst.*, 12(3):395-428, 1987.

[8] Y. Ishikawa, H. Kitagawa, and N. Ohbo, "Evaluation of Signature Files as Set Access Facilities in OODBs," *Proc. ACM SIGMOD*, pages 247-256, Washington, D.C., 1993.

[9] A. Kent, R. Sacks-Davis, and K. Ramamohanarao, "A Superimposed Coding Scheme Based on Multiple Block Descriptor Files for Indexing Very Large Data Bases," *Proc. 14th VLDB*, pages 351-359, Los Angeles, 1988.

[10] H. Kitagawa, Y. Fukushima, Y. Ishikawa, and N. Ohbo, "Estimation of False Drops in Set-valued Object Retrieval with Signature Files," *Proc. 4th Intl. Conf. on Foundations of Data Organization and Algorithms (FODO'93)*, pages 146-163, Evanston, 1993.

[11] H. Kitagawa, N. Watanabe, and Y. Ishikawa, "Design and Evaluation of Signature File Organization Incorporating Vertical and Horizontal Decomposition Schemes," *Proc. 7th Intl. Conf. on Database and Expert Systems Applications (DEXA'96)*, pages 875-888, Zurich, 1996.

[12] W. Lee and D. L. Lee, "Signature File Methods for Indexing Object-Oriented Database Systems," *Proc. Intl. Computer Science Conf. (ICSC)*, pages 616-622, Hong Kong, 1992.

[13] C. W. R. Leng and D. L. Lee, "Optimal Weight Assignment for Signature Generation," *ACM Trans. Database Syst.*, 17(2):346-373, 1992.

[14] F. Rabitti and P. Savino, "Image Query Processing Based on Multi-level Signatures," *Proc. 14th Annual Intl. ACM/SIGIR Conf. on Research and Development in Information Retrieval*, pages 305-314, Chicago, 1991.

[15] K. Ramamohanarao and J. A. Shepherd, "A Superimposed Codeword Indexing Scheme for Very Large Prolog Databases," *Proc. 3rd Intl. Conf. on Logic Programming*, pages 569-576, London, 1986.

[16] C. S. Roberts, "Partial-Match Retrieval via the Method of Superimposed Codes," *Proc. IEEE*, 67(12):1624-1642, 1979.

[17] R. Sacks-Davis, A. Kent, and K. Kamamohanarao, "Multikey Access Methods Based on Superimposed Coding Techniques," *ACM Trans. Database Syst.*, 12(4):655-696, 1987.

[18] P. Tiberio and P. Zezula, "Storage and Retrieval: Signature File Access," In A. Kent and J. G. Williams, editors, *Encyclopedia of Microcomputers*, volume 16, pages 377-403, Marcel Dekker, Inc., New York, 1995.

[19] H. S. Yong, S. Lee, and H. J. Kim, "Applying Signatures for Forward Traversal Query Processing in Object-Oriented Databases," *Proc. 10th Intl. Conf. on Data Engineering*, pages 518-525, Houston, 1994.

[20] P. Zezula, F. Rabitti, and P. Tiberio, "Dynamic Partitioning of Signature Files," *ACM Trans. Inf. Syst.*, 9(4):336-369, 1991.

[21] J. Zobel, A. Moffat, and R. Sacks-Davis, "An Efficient Indexing Technique for Full-Text Database Systems," *Proc. 18th VLDB*, pages 352-362, 1992.

[22] J. Zobel, A. Moffat, and R. Sacks-Davis, "Searching Large Lexicons for Partially Specified Terms Using Compressed Inverted Files," *Proc. 19th VLDB*, pages 290-301, 1993.