Design and Evaluation of Signature File Organization Incorporating Vertical and Horizontal Decomposition Schemes

Hiroyuki Kitagawa[1], Noriyasu Watanabe[2*] and Yoshiharu Ishikawa[3]

[1] Institute of Information Sciences and Electronics, University of Tsukuba
[2] Doctoral Degree Program in Engineering, University of Tsukuba
[3] Graduate Institute of Information Science
Nara Institute of Science and Technology (NAIST)

Abstract. Signature files are known as promising facilities to speed up accesses to large information repositories in database and information retrieval systems. This paper presents a new signature file organization method, named *Partitioned Frame-Sliced Signature File (P-FSSF)*, and studies its performance. P-FSSF incorporates both vertical and horizontal decomposition schemes to reduce page accesses required to look up signatures. In addition, P-FSSF is flexible enough to have its concrete organization tuned to real application environments. We develop formulas to estimate the retrieval cost of P-FSSF in the context of the general set-valued object retrieval. Also, formulas to tell the update and storage costs are derived. Then, the processing cost of P-FSSF is shown to be lower than the other existing signature file organizations in general. We also show that *Partitioned Bit-Sliced Signature File (P-BSSF)*, which is a special case of P-FSSF, is appropriate organization in most probable cases through the study of the optimal parameter values for P-FSSF.

1 Introduction

Modern database and information retrieval systems have to support storage and retrieval of a variety of data objects. In one dimension, they have to manipulate text, image, video and audio data as well as fixed record structures. In another dimension, they have to support complex data structures built with tuples, sets, lists, and so on. To attain proper performance, development of efficient access methods applicable to object manipulation in those systems is very important. The signature file is one of promising access methods to facilitate object manipulation in modern database and information retrieval systems [3, 4, 5, 6, 17]. The signature file is a collection of signatures, each of which is a bit vector formed for a data object representing its properties. The signature file access method was originally designed for text retrieval, and a lot of study has been done in the context [3, 4, 5, 6]. 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 [15], to the Prolog clause retrieval [14], to the image retrieval [13], to the set-valued object retrieval [7, 9], and to the object navigation in the object-oriented databases [11, 20].

We have proposed the application of superimposed coded signature files to set-valued object retrieval (shortly, *set retrieval*) and showed its potential applicability [7, 9]. The set retrieval here means finding objects based on the set containment conditions given a large number of set-valued objects stored in the database. The concept of set retrieval is general and 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

* Current Affiliation: NTT Data Communications Systems Corp.

database systems and object-oriented database systems, "is-subset" queries have to be supported as well as "has-subset" queries [1].

It is known that, in the retrieval with the signature file, 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 [4, 17]. Most of the organization schemes are classified into two groups. In the first group, the signature of a target object is decomposed into fragments and they are stored in different pages. We call this scheme *vertical decomposition*. The *Frame-Sliced Signature File (FSSF)* [12] and *Bit-Sliced Signature File (BSSF)* [15], which is a special case of FSSF, represent the first group. In the second group, the collection of signatures is divided into partitions, and different pages are allocated to the partitions. We call this scheme *horizontal decomposition*. The *Partitioned Signature File* [10] and *Quick Filer* [22] represent the second group.

In this paper, we present a new signature file organization method, named *Partitioned Frame-Sliced Signature File (P-FSSF)*, and study its performance. P-FSSF incorporates the vertical decomposition scheme of the Frame-Slice approach and the horizontal decomposition scheme of the Partitioned Signature File approach. Thus, the retrieval cost of P-FSSF is, in general, lower than the other existing signature file organizations. In addition, it is flexible enough to have its concrete organization tuned to real application environments by deciding appropriate parameter values. We develop formulas to estimate the cost of retrieval with P-FSSF in the context of the set retrieval. Also, formulas to tell the update and storage costs are derived. Then, the processing cost of P-FSSF is intensively analyzed with the cost formulas.

Except for our study [18], there have been two proposals of signature file organizations incorporating both the vertical and the horizontal decomposition schemes, to the best of our knowledge. One is the *Two-dimensional Dynamic Signature File (TDSF)* [8], and the other is the *Key-Based Bit-Sliced Signature File* [21]. The former combines the Frame-Slice-based vertical decomposition and the horizontal decomposition using the extendible hashing. The latter combines the Bit-Slice-based vertical decomposition and the Partitioned Signature File approach. The original contribution of our study in the light of the related works can be summarized as follows.

(1) Both in TDSF and the Key-Based Bit-Sliced Signature File, the original signature is directly used for horizontal decomposition. For this reason, to avoid the skew in the horizontal decomposition, each signature has to satisfy the condition that half the bits are set to "1." Since P-FSSF is organized to use another set of signatures for the horizontal decomposition, P-FSSF is rid of this restriction and has more flexibility. As exemplified in our previous study [7], the total retrieval cost is sometimes much lower for some kinds of queries if the number of bits set to 1 in a signature is rather small. This option cannot be applicable to TDSF and the Key-Based Bit-Sliced Signature File.

(2) We derive cost formulas for P-FSSF and study its performance in processing "has-subset" and "is-subset" queries in the context of set retrieval. The studies in [8, 21] consider only "has-subset" queries under the restriction mentioned in (1).

(3) We show that *Partitioned Bit-Sliced Signature File (P-BSSF)*, which is a special case of P-FSSF, is appropriate organization in most probable cases through the study of the optimal parameter values for several combinations of retrieval and update requests. The optimal selection of the parameter values has not been reported in [8, 21].

(4) We have proposed an efficient query processing strategy named *smart retrieval* for BSSF in [7]. The smart retrieval strategy contributes to the reduction of the total retrieval cost using partial look-up of signatures. The

smart retrieval is also investigated in this study and is shown to be beneficial even in the signature file organization incorporating the vertical and horizontal decomposition schemes.

The remaining part of this paper is organized as follows. In Section 2, we give an overview of set retrieval with the signature file in this context, and explain the vertical and horizontal decompositions in the signature file organization. Section 3 presents P-FSSF and its query and update processing schemes. In Section 4, we derive formulas to estimate the cost involved in query and update processing with P-FSSF. In Section 5, the processing cost of P-FSSF is analyzed based on the formulas derived in Section 4. In Section 6, we show that P-FSSF can attain better performance than TDSF and the Key-Based Bit-Sliced Signature File, and that P-BSSF, which is a special case of P-FSSF, is an appropriate organization in most probable cases through the study of the optimal parameter values. In Section 7, we discuss further reduction of retrieval cost with the smart retrieval strategy. Section 8 concludes this paper.

2 Preliminaries

2.1 Set Retrieval with Signature Files

We explain set retrieval in our context based on the objects having set attribute values shown in Fig. 1. Each row in Fig. 1 corresponds to a sales department object. Each object is tagged with the object identifier (OID) and has attributes DNAME and SITEMS. DNAME represents the department name. SITEMS is a set attribute and represents the set of sales items of each department. Object retrieval based on the SITEMS attribute value, for instance SITEMS $\supseteq$ {"pen", "ink"}, is an example of set retrieval considered in this paper. In this case, each SITEMS attribute value is called a *target set*, and {"pen", "ink"} 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.

SALES_DEPARTMENT		
OID	DNAME	SITMES
		INAME
oid1	Stationary	pen
		pencil
		ink
oid2	Toy	doll
		cards
		pencil
		drum

Fig. 1. Objects Having Set Attribute Values

element		element signature
pen	$\rightarrow$	00010001
pencil	$\rightarrow$	01001000
ink	$\rightarrow$	00010010
		$\Downarrow$
target signature		01011011

Fig. 2. Generation of a Target Signature

As mentioned in Section 1, a signature is a bit vector formed for each object. For set retrieval, a *target signature* is generated for each target set and stored in the signature file. 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." Then, a target signature is obtained by bitwise OR-ing (*superimposed coding*) element signatures of all the elements in the target set and stored in the signature file with the corresponding OID. Fig. 2 shows the generation of the target

signature for set {"pen", "pencil", "ink"}. The number of "1"'s in the signature is called the w*eight*.

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 a potential match. If the target signature satisfies the following conditions, 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 fail the test are called *false drops*, while the qualified objects are called *actual drops*.

2.2 Vertical and Horizontal Decomposition Schemes

It is known that the page access required to look up signatures is a dominant cost factor in the retrieval with the signature file. The vertical decomposition and the horizontal decomposition are used in some signature file organizations to reduce the page access cost. We briefly explain FSSF and the FP-Partitioned Signature File to present basic ideas of the vertical and horizontal decomposition schemes.

Frame-Sliced Signature File (FSSF) FSSF was proposed by Lin and Faloutsos [12]. Let the signature length be F. In FSSF, each bit vector of length F is vertically decomposed into k *frames* of s bits each ($k \cdot s = F$). The set signature is generated as follows: For each element in the set, n ($n \leq k$) distinct frames are selected and m bit positions are set to "1" in each selected frame by hashing to generate an element signature. Then, the set signature is obtained by bit-wise OR-ing the element signatures. Suppose we are processing a $T \supseteq Q$ query and only n_q frames includes "1" in the query signature. Then, only the n_q frames out of k need to be looked up to find drops. This is why the vertical decomposition in FSSF contributes to the reduction of the signature look-up cost. In processing $T \subseteq Q$ queries, similar reduction can be attained by focussing on bit positions set to "0" in the query signature. When $s = 1$, namely each frame has one bit length, FSSF is called the Bit-Sliced Signature File (BSSF) [15]. The retrieval cost of BSSF is lower than FSSF in general, while update is rather expensive in BSSF.

FP-Partitioned Signature File The Partitioned Signature File was proposed by Lee and Leng [10], and the partitioning based on the *Fixed Prefix (FP) method* is explained here. Basic idea of the *FP-Partitioned Signature File* is to decompose the set of signatures into *partitions* using the fixed number of prefix bits as a key. For example, when two bit prefix is used as a key, we get four partitions corresponding to "00" through "11." Suppose we are processing a $T \supseteq Q$ query and the two bit prefix of the query signature is "01." Then, only two partitions out of four, corresponding to "01" and "11," need to be looked up to find drops. This is why the horizontal decomposition in the FP-Partitioned Signature File contributes to the reduction of the signature look-up cost. In processing $T \subseteq Q$ queries, similar reduction can be attained by focussing on bit positions set to "0" in the query signature. To avoid the skew in assigning partitions to signatures, each signature has to satisfy the condition that half the bits are set to "1."

As for $T \supseteq Q$ queries, the vertical decomposition in FSSF and BSSF attains much cost reduction when the query signature weight is small, while the horizontal decomposition in the FP-Partitioned Signature File is effective when the query signature weight is large. This is also advocated in [21]. As for $T \subseteq Q$ queries, the vertical decomposition gives much contribution when the query signature weight is large, while the horizontal decomposition is effective when the query signature weight is small. Therefore, we can get very efficient signature file organization by incorporating both the vertical and horizontal decomposition schemes.

3 Partitioned Frame-Sliced Signature File (P-FSSF)

3.1 Organization of P-FSSF

Partitioned Frame-Sliced Signature File (P-FSSF) is a new signature file organization method that combines the Frame-Slice-based vertical decomposition and the FP-based horizontal decomposition. Fig. 3 illustrates the organization of P-FSSF. The storage space of signatures is horizontally decomposed into *partitions* and each partition is vertically decomposed into *frame-slices*. Each frame-slice corresponds to one disk page.

In P-FSSF, two target signatures, namely *target FP signature* and *target FSSF signature*, are created using different hash functions. Only the target FP signature has to satisfy the condition that half the bits are set to "1." Only the target FSSF signatures are actually stored in P-FSSF. Procedure to register a new target object into P-FSSF is as follows:

1. For each set element, an *element FP signature* of f-bit length is created. The element FP signature weight is set to m_{opt} $(= \frac{f}{D_t} \ln 2)$, where D_t is the cardinality of the target set. Then the target FP signature $S_{T\{FP\}}$ is created by the superimposed coding. It is known that the expected target signature weight is $\frac{f}{2}$, when the element signature weight is m_{opt} $(= \frac{f}{D_t} \ln 2)$ [5, 16].
2. The h-bit prefix $S'_{T\{FP\}}$ of the target FP signature $S_{T\{FP\}}$ is used to decide the partition to store the target FSSF signature.
3. The target FSSF signature $S_{T\{FSSF\}}$ is created following the FSSF method. $S_{T\{FSSF\}}$ has F-bit length and is divided into k frames, each of which has s-bit length $(k \cdot s = F)$. To create an *element FSSF signature*, n $(n \leq k)$ frames are selected, and m bits are set to "1," for each selected frame. Then $S_{T\{FSSF\}}$ is created by using the superimposed-coding and stored in the partition determined in Step 2. We call the frames of $S_{T\{FSSF\}}$ in which some bits are set to "1" as *gray frames* and, in particular, the frames in which all the bits are set to "1" as *black frames*.
4. The OID is stored in the corresponding position in the OID file.

In Fig. 3, P-FSSF has the parameters $h = 2$, $k = 4$, $s = 2$, and $F = k \cdot s = 8$.

3.2 Retrieval Algorithm

When a query set is given, the query is processed as follows.

1. In the same way as the target signature, the query FP signature $S_{Q\{FP\}}$ (and its prefix $S'_{Q\{FP\}}$) and the query FSSF signature $S_{Q\{FSSF\}}$ are created. Gray frames and black frames of $S_{Q\{FSSF\}}$ are defined in a similar way.

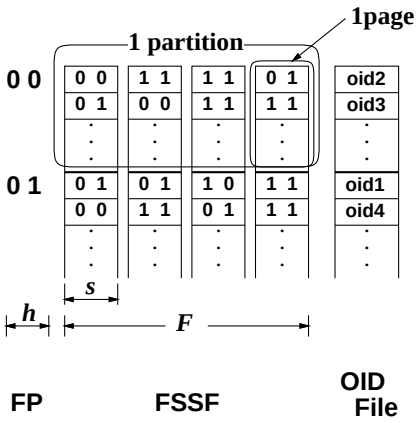

Fig. 3. Partitioned Frame-Sliced Signature File (P-FSSF)

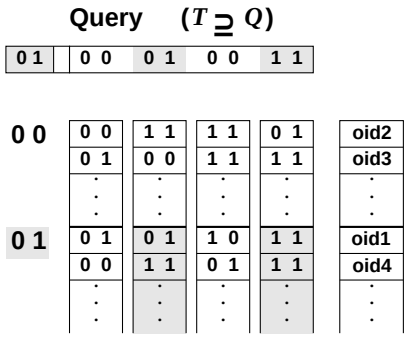

Fig. 4. Query Processing with P-FSSF

2. $S'_{Q\{FP\}}$ is used to decide the partitions to be looked up. A partition needs to be looked up when the following condition is satisfied.

$$T \supseteq Q : S'_{T\{FP\}} \wedge S'_{Q\{FP\}} = S'_{Q\{FP\}}$$
$$T \subseteq Q : S'_{T\{FP\}} \wedge S'_{Q\{FP\}} = S'_{T\{FP\}}$$

3. For each partition selected in Step 2, the target FSSF signatures are compared with the query FSSF signature $S_{Q\{FSSF\}}$. If $S_{T\{FSSF\}}$ satisfies the following condition, it becomes a drop.

$$T \supseteq Q : S_{T\{FSSF\}} \wedge S_{Q\{FSSF\}} = S_{Q\{FSSF\}}$$
$$T \subseteq Q : S_{T\{FSSF\}} \wedge S_{Q\{FSSF\}} = S_{T\{FSSF\}}$$

   Note that only the frame-slices corresponding to gray frames ($T \supseteq Q$) or non-black frames ($T \subseteq Q$) of $S_{Q\{FSSF\}}$ in each partition need to be looked up to check these conditions as explained for FSSF in Subsection 2.2.
4. False drop resolution is performed, and actual drops are returned.

Figure 4 illustrates the query processing with P-FSSF for $T \supseteq Q$. Since the prefix of the query FP signature is "01," the second and fourth partitions need to be looked up. In the second and fourth partitions, only the second and last frame-slices are to be looked up.

3.3 Update Algorithm

Basic procedure to register a new object into P-FSSF has been presented in Subsection 3.1. In this subsection, update algorithm for deletion of an object is described. We assume that the OID to be deleted and its target FP and FSSF signatures ($S_{T\{FSSF\}}$ and $S_{T\{FP\}}$) are given at deletion time.

1. $S'_{T\{FP\}}$ is used to decide the partition where the target FSSF signature $S_{T\{FSSF\}}$ is stored.
2. The part of the OID file corresponding to the partition selected in Step 1 is scanned. If the OID to be deleted is found, the deletion bit is set at the position of the OID file.

3. The target FSSF signature in the partition is cleared to "0"'s. Actually, for each bit position set to "1" in $S_{T\{FSSF\}}$, the corresponding position in the frame-slice is reset to "0."

4 Cost Model

In this section, formulas to estimate the retrieval cost RC, the update cost UC, and the storage cost SC of P-FSSF are derived. The retrieval cost and the update cost are estimated based on disk page I/Os, and the storage cost is estimated by the number of disk pages to store P-FSSF. Symbols and their definitions are summarized in Table 1.

4.1 False Drop Probability

False drop probability [5] is an important measure for estimating the performance of signature files and given by Eq. (1).

$$Fd = \frac{\text{false drops}}{N - \text{actual drops}} \tag{1}$$

In the following, we estimate the false drop probability of P-FSSF. Given a frame of the target FSSF signature, the probability that it is a gray frame is estimated as follows.

$$1 - \left(1 - \frac{n}{k}\right)^{D_{\mathrm{t}}}$$

Similar remark applies to the query FSSF signature. Therefore, for each target FSSF signature and query FSSF signature, the expected numbers of gray frames are derived as Eqs. (2) and (3), respectively.

$$n_{\mathrm{t}} = k\left(1 - \left(1 - \frac{n}{k}\right)^{D_{\mathrm{t}}}\right) \tag{2}$$

$$n_{\mathrm{q}} = k\left(1 - \left(1 - \frac{n}{k}\right)^{D_{\mathrm{q}}}\right) \tag{3}$$

For each gray frame of the target FSSF signature, Eq. (4) gives the expected number of elements in a target set which set some bits in the frame to "1."

$$D'_{\mathrm{t}} = \frac{nD_{\mathrm{t}}}{n_{\mathrm{t}}} \tag{4}$$

Similar remark applies to each gray frame of the query FSSF signature, and we get Eq. (5).

$$D'_{\mathrm{q}} = \frac{nD_{\mathrm{q}}}{n_{\mathrm{q}}} \tag{5}$$

For each target FSSF signature and query FSSF signature, the expected weight of a gray frame is given by Eqs. (6) and (7).

$$m_{\mathrm{t}} = s\left(1 - \left(1 - \frac{m}{s}\right)^{D'_{\mathrm{t}}}\right) \tag{6}$$

$$m_{\mathrm{q}} = s\left(1 - \left(1 - \frac{m}{s}\right)^{D'_{\mathrm{q}}}\right) \tag{7}$$

Therefore, the false drop probabilities for $T \supseteq Q$ and $T \subseteq Q$ can be estimated by Eqs. (8) and (9), respectively.

$$Fd_{\{T \supseteq Q\}} = \left\{ \left(1 - \left(1 - \frac{n}{k}\right)^{D_t}\right) \cdot \left(1 - \left(1 - \frac{m}{s}\right)^{D'_t}\right)^{m_q} \right\}^{n_q} \tag{8}$$

$$Fd_{\{T \subseteq Q\}} = \left\{ \left(1 - \left(1 - \frac{n}{k}\right)^{D_q}\right) \cdot \left(1 - \left(1 - \frac{m}{s}\right)^{D'_q}\right)^{m_t} \right\}^{n_t} \tag{9}$$

4.2 Retrieval Cost Model

The retrieval cost RC is estimated by Eq. (10).

$$RC = LC_{sig} + LC_{OID} + P_o(A + PAR \cdot Fd(N - A)) \tag{10}$$

Here, LC_{sig} is the cost to look up the target FSSF signatures, LC_{OID} is the access cost for the OID file, and $P_o(A + PAR \cdot Fd(N - A))$ is the cost for false drop resolution.

LC_{sig} is given by Eqs. (11) and (12)

$$LC_{sig\{T \supseteq Q\}} = n_q \cdot PAR_{\{T \supseteq Q\}} \cdot \left\lceil \frac{N}{\lfloor \frac{Pb \cdot \alpha}{s} \rfloor} \right\rceil \tag{11}$$

$$LC_{sig\{T \subseteq Q\}} = (k - n_q \cdot p_q) \cdot PAR_{\{T \subseteq Q\}} \cdot \left\lceil \frac{N}{\lfloor \frac{Pb \cdot \alpha}{s} \rfloor} \right\rceil, \tag{12}$$

where p_q denotes the probability that a gray frame of the query FSSF signature happens to be black and given by $(1 - (1 - \frac{m}{s})^{D'_q})^s$.

PAR (*Partition Activation Ratio*) is the ratio of partitions to be looked up. PAR is expressed as follows.

$$PAR_{\{T \supseteq Q\}} \approx \left(1 - \frac{w(S_{Q\{FP\}})}{2f}\right)^h \tag{13}$$

$$PAR_{\{T \subseteq Q\}} \approx \left(\frac{f + w(S_{Q\{FP\}})}{2f}\right)^h, \tag{14}$$

where $w(S_{Q\{FP\}})$ is the weight of the query FP signature and given by Eq. (15).

$$w(S_{Q\{FP\}}) = f\left(1 - (1 - m_{\mathrm{opt}}/f)^{D_q}\right) \approx f\left(1 - (1/2)^{\frac{D_q}{D_t}}\right) \tag{15}$$

Derivation of $PAR_{\{T \supseteq Q\}}$ is given in [2] in detail and $PAR_{\{T \subseteq Q\}}$ is obtained in a similar way. LC_{OID} is derived by using the Yao's function [19].

$$LC_{OID} = SC_{OID} \cdot \left[1 - \prod_{i=1}^{A+Fd(N-A)} \frac{N \cdot (1 - 1/SC_{OID}) - i + 1}{N - i + 1}\right], \tag{16}$$

where $SC_{OID} = \left\lceil \frac{N}{\lfloor \frac{P \cdot \alpha}{oid} \rfloor} \right\rceil$ is the storage cost of the OID file. The number of actual drops A can be estimated in the same manner as in our previous work [7].

4.3 Update Cost Model

When updating P-FSSF, the partition to be updated is determined by the prefix of the target FP signature. Next, the OID file corresponding to the partition is scanned and modified. Then n_t frame-slices in the partition is set to "1" (for insertion) or "0" (for deletion). The update cost is expressed by Eq. (17).

$$UC = 2n_t + \frac{b \cdot oid}{2 \cdot s} + 1 \tag{17}$$

4.4 Storage Cost Model

The storage cost of P-FSSF is the sum of the storage cost for the FSSF signatures and that for the OID file, and is given by Eq. (18).

$$SC = \left\lceil \frac{N}{\lfloor \frac{Pb \cdot \alpha}{s} \rfloor} \right\rceil \cdot k + \left\lceil \frac{N}{\lfloor \frac{P \cdot \alpha}{oid} \rfloor} \right\rceil \tag{18}$$

Table 1. Symbols and Their Definitions

Symbol	Definition
D_t	cardinality of a target set
D_q	cardinality of a query set
N	number of objects
P	a disk page size in bytes
b	number of bits per byte
oid	an OID size in bytes
P_o	number of page accesses to fetch an object
F, f	signature size in bits
A	number of actual drops
Fd	false drop probability
α	partition load ratio
h	prefix size in bits
n	number of frames selected in element signature generation
s	frame width in bits
k	number of frames ($k \cdot s = F$)
m	number of bits set in each selected frame

Table 2. Parameter Values

Symbol	Value
D_t	cardinality of a target set (= 100)
N	number of objects (= 800,000)
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 (= 1024)
α	partition load factor (= 0.75)

5 Cost Analysis

In this section, the processing and storage cost of P-FSSF is analyzed based on the formulas derived in Section 4. To clarify the characteristics of P-FSSF, the costs of FSSF, BSSF, FP-Partitioned Signature File are also shown. Cost formulas for those signature file organizations can be derived in a similar manner to those for P-FSSF, and are omitted here for space limitation. The cost of P-BSSF, a special case of P-FSSF, is also analyzed. The cost formulas for P-BSSF are those for P-FSSF with $s = 1$. Cost analysis for the parameter values shown in Table 2 is presented below.

5.1 Retrieval Cost

Retrieval Cost for $T \supseteq Q$ The retrieval cost for $T \supseteq Q$ is shown in Fig. 5. As indicated in Subsection 2.2, the cost of the FP-Partitioned Signature File is prohibitively high for small D_q values, while that of FSSF drastically increases as D_q becomes large. In contrast to this, the cost of P-FSSF is fairly stable for a wide range of D_q values. While BSSF can efficiently process $T \supseteq Q$ queries, P-BSSF, a special case of P-FSSF, with the same element signature weight m is generally more efficient than BSSF. Note that the retrieval cost of BSSF and P-BSSF is lower for smaller m values.

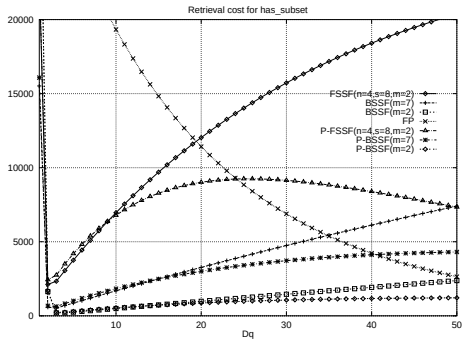

Fig. 5. Retrieval Cost for $T \supseteq Q$

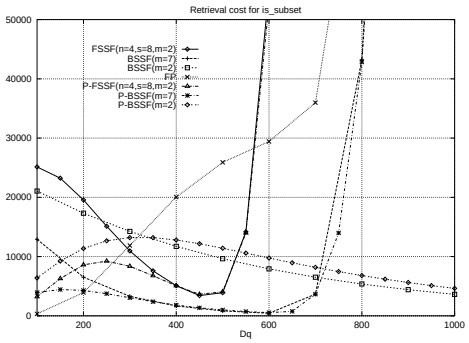

Fig. 6. Retrieval Cost for $T \subseteq Q$

Retrieval Cost for $T \subseteq Q$ The retrieval cost for $T \subseteq Q$ is shown in Fig. 6. In contrast of the case of $T \subseteq Q$, the cost of the FP-Partitioned Signature File monotonically increases as D_q becomes large and is prohibitively high for large D_q values.

The cost of FSSF is rather high for small D_q values, and decreases as D_q becomes large. However, it starts increasing for large D_q values due to the increase of false drops. Although the behavior of P-FSSF is almost same as that of FSSF with the same m value for large D_q, it gives a lot of cost reduction for small D_q. While BSSF can also efficiently process $T \subseteq Q$ queries, the corresponding P-BSSF is generally more efficient. Note that P-BSSF (BSSF) with $m = 2$ is not as efficient as that with $m = 7$, the former is applicable to a wide range of D_q values.

5.2 Update Cost

Update cost for each signture file organization is shown in Table 3. UC_I and UC_D stand for insertion cost and deletion cost, respectively. For FSSF and BSSF, we assume that a new signature is inserted at the end of the signature file. The update cost of the FP-Partitioned Signature File is lowest both for insertion and deletion. The reason is that the partition to be updated can be efficiently determined by the partitioning key in the FP-Partitioned Signature File, and that the target signature is stored as a whole in a page. Although the insertion cost of P-FSSF is slightly higher than that of the corresponding FSSF, the deletion cost is much less since the partition to be updated can be determined by the prefix of the target FP signature. Note that the update cost of P-BSSF (BSSF) can be quite high since one signature is vertically decomposed and stored in many bit-slices.

5.3 Storage Cost

Storage cost for each signature file organization is shown in Table 4. The storage costs of FSSF and BSSF are lower than those of the FP-Partitioned Signature File, P-FSSF, and P-BSSF. The reason is that the partition load ratio α is set to 75%, which means that 25% of the storage space is reserved for incoming insertions of new signatures. In contrast to this, no extra space is reserved in FSSF and BSSF since insertion always goes to the end of the file.

Table 3. Update Cost

Organization	UC_I	UC_D
FSSF ($n = 4$, $s = 8$, $m = 2$)	248	1029
BSSF ($m = 7$)	1018	1799
BSSF ($m = 2$)	366	1147
FP	4	2
P-FSSF ($n = 4$, $s = 8$, $m = 2$)	279	279
P-BSSF ($m = 7$)	1049	1049
P-BSSF ($m = 2$)	397	397

Table 4. Storage Cost

Organization	SC
FSSF ($n = 4$, $s = 8$, $m = 2$)	26651
BSSF ($m = 7$)	27163
BSSF ($m = 2$)	27163
FP	34897
P-FSSF ($n = 4$, $s = 8$, $m = 2$)	35492
P-BSSF ($m = 7$)	35876
P-BSSF ($m = 2$)	35876

6 Optimal Configuration of P-FSSF

In Section 5, we have analyzed the behavior of P-FSSF (including P-BSSF as its special case) in the light of FSSF, BSSF, and the FP-Partitioned Signature File. Through the analysis, we have presented the advantage of incorporating both the vertical and horizontal decomposition schemes.

As mentioned in Section 2, TDSF and the Key-Based Bit-Sliced Signature File also incorporate the vertical and horizontal decomposition schemes. However, each signature in those organization has to satisfy the condition that half the bits are set to "1." In the context of our discussion, this restriction is equivalent to the condition $n \cdot m = \frac{F}{D_t} \ln 2$ as explained in Subsection 3.1 [5, 16]. P-FSSF is rid of this restriction. To investigate the advantage of this flexibility, we derive the optimal configuration of P-FSSF with and without the restriction $n \cdot m = \frac{F}{D_t} \ln 2$.

We calculate the values for s, n, and m which give the cost minimal P-FSSF by the hill-climbing method fixing the other parameters D_t, N, F, α, etc. The cost C is defined to consider the update cost as well as the retrieval cost:

$$C = (1 - \delta)RC + \delta \cdot UC, \tag{19}$$

where RC and UC denote the retrieval cost (Eq. (10)) and the update cost (Eq. (17)), respectively, and δ gives the occurrence frequency of updates. $\delta = 0$ means the retrieval-only case, and $\delta = 0.5$ means the case where retrieval and update occur in equal frequency.

Fig. 7 and Fig. 8 show the costs of the optimal P-FSSF for each D_q value with and without the restriction $n \cdot m = \frac{F}{D_t} \ln 2$. In Fig. 7 and Fig. 8, $D_t = 100$ and $F = 1024$. The costs with the restriction are tagged with RESTRICT. As shown in Fig. 7, for $T \supseteq Q$, the optimal P-FSSF without the restriction gives far cheaper cost than that with the restriction for the most D_q values. It also holds for $T \subseteq Q$ except for large D_q values. Therefore, we can say that the flexibility of P-FSSF can contribute to the reduction of the processing cost when appropriate parameter values are selected.

As shown in Section 5, although the retrieval cost of P-BSSF, namely P-FSSF with $s = 1$, is fairly low, its update cost is rather high. We have counted

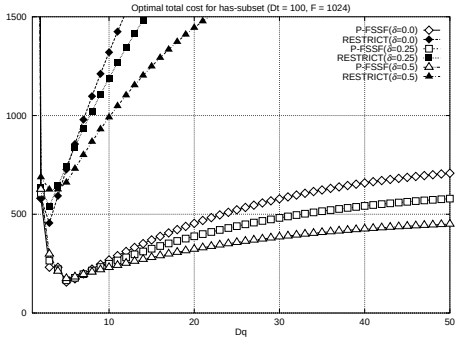

Fig. 7. Optimal Cost of P-BSSF for $T \supseteq Q$

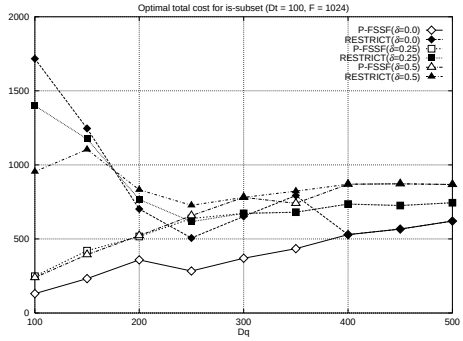

Fig. 8. Optimal Cost of P-BSSF for $T \subseteq Q$

the cases where the total cost optimal P-FSSF happens to be P-BSSF, namely $s = 1$. Table 5 shows the percentage of the cases that P-BSSF is optimal. For $T \supseteq Q$, the optimal P-FSSF happens to be P-BSSF in all cases considered. For $T \subseteq Q$, P-BSSF is optimal in more than 80% of the cases. Therefore, we can say that P-BSSF is appropriate organization in most probable cases.

Table 5. Percentage of the Cases such that P-BSSF is Optimal

		$\delta = 0.0$	$\delta = 0.25$	$\delta = 0.5$
$D_t = 10$	$T \supseteq Q$	100%	100%	100%
$F = 256$	$T \subseteq Q$	90%	90%	80%
$D_t = 100$	$T \supseteq Q$	100%	100%	100%
$F = 1024$	$T \subseteq Q$	85%	85%	85%

7 Smart Retrieval Strategy

The *smart retrieval* was proposed in [7] as an efficient query evaluation strategy for BSSF. In the query processing of signature files, the final decision that a candidate object actually satisfies the query condition or not is performed in the false drop resolution stage. Therefore, it is often beneficial for the retrieval cost reduction to omit looking up some bit-slices even if false drops increases to some extent. The smart retrieval strategy uses this property. Although the smart retrieval strategy is generally applicable to the signature file organization incorporating the vertical decomposition, it is most effective in the bit-slice-based vertical decomposition. Since P-BSSF incorporates bit-slicing, the smart retrieval strategy is also beneficial for P-BSSF.

We consider the retrieval cost of P-BSSF with $m = 2$ for $T \supseteq Q$ shown in Fig. 5. The retrieval cost takes the minimal value for $D_q = 4$. When $D_q \leq 4$, the retrieval cost mainly consists of the false drop resolution cost. On the other hand, when $D_q > 4$, the most of the retrieval cost is due to the look-up of the bit-slices, while the number of false drops is very small. Therefore, the look-up of some of the bit-slices can be omitted without increasing the total retrieval cost for $D_q > 4$. In this case, the smart retrieval strategy for P-BSSF is implemented as follows.

1. When $D_q \leq 4$, the query is processed ordinarily.
2. When $D_q > 4$, the query FP signature is created ordinarily and the partition to be looked up is determined. Then, the query FSSF (actually BSSF) signature is created from any four elements in the query set and the bit-slices are looked up as if $D_q = 4$.

The retrieval cost of P-BSSF for $T \supseteq Q$ following the smart retrieval strategy is shown in Fig. 9. The smart retrieval strategy is also applicable to $T \subseteq Q$, and its effect is shown in Fig. 10. As shown in Fig. 9 and Fig. 10, the smart retrieval strategy indeed contributes to the retrieval cost reduction of P-BSSF.

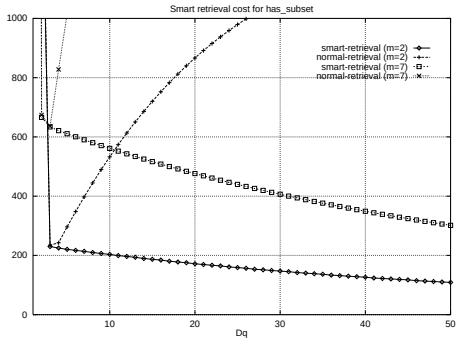

Fig. 9. Retrieval Cost for $T \supseteq Q$ with Smart Retrieval Strategy

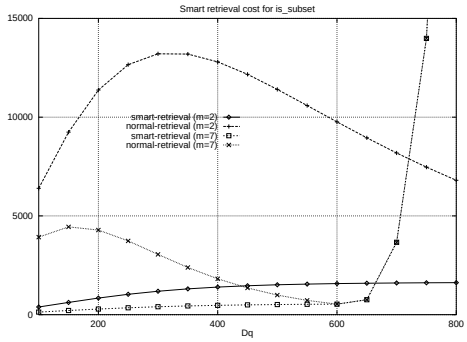

Fig. 10. Retrieval Cost for $T \subseteq Q$ with Smart Retrieval Strategy

8 Conclusion

In this paper, we have presented a new signature file organization method, named *Partitioned Frame-Sliced Signature File (P-FSSF)*, and studied its performance in the context of set retrieval. P-FSSF incorporates the vertical decomposition scheme of the Frame-Slice approach and the horizontal decomposition scheme of the FP-Partitioned Signature File approach. We have derived formulas to estimate the retrieval cost, update cost, and storage cost of P-FSSF. Through the analysis of the behavior of P-FSSF, we have demonstrated the advantage of incorporating both the vertical and horizontal decomposition schemes. Then we have shown that the flexibility of P-FSSF (namely, without the restriction that $n \cdot s = \frac{F}{D_t} \ln 2$) can contribute to the cost reduction in query and update processing. Furthermore, it has been shown that *Partitioned Bit-Sliced Signature File (P-BSSF)*, a special case of P-FSSF with $s = 1$, is appropriate organization in most probable cases through the intensive analysis of the optimal configuration of P-FSSF. Finally, the smart retrieval strategy has been proved to contribute to the reduction of the retrieval cost of P-BSSF.

The future research issues include cost evaluation including CPU processing time, evaluation in more practical context having skew in D_t values, and refinement of the smart retrieval strategy.

Acknowledgement

The authors appreciate collaboration with many members of Database Laboratory, University of Tsukuba, and Laboratory for Database and Media Systems (Uemura Laboratory), NAIST. This work was partially supported by the Grand-in-Aid for Science Research on Priority Areas from the Ministry of Education, Science, Sports and Culture of Japan.

References

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

2. P. Ciaccia and P. Zezula. "Estimating Accesses in Partitioned Signature File Organizations,". *ACM Trans. Inf. Syst.*, 11(2):133–142, 1993.
3. C. Faloutsos. "Access Methods for Text,". *ACM Comput. Surv.*, 17(1):49–74, 1985.
4. C. Faloutsos. "Signature-Based Text Retrieval Methods: A Survey,". *IEEE Database Eng.*, 13(1):25–32, Mar. 1990.
5. 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.
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, July 1987.
7. Y. Ishikawa, H. Kitagawa, and N. Ohbo. "Evaluation of Signature Files as Set Access Facilities in OODBs,". In *Proc. ACM SIGMOD*, pages 247–256, Washington, D.C., 1993.
8. J. K. Kim and J. W. Chang. "A Two-dimensional Dynamic Signature File Method,". In *Intl. Symp. on Advanced Database Technologies and Their Integration (ADTI'94)*, pages 63–70, Nara, Japan, 1994.
9. H. Kitagawa, Y. Fukushima, Y. Ishikawa, and N. Ohbo. "Estimation of False Drops in Set-valued Object Retrieval with Signature Files,". In *Proc. 4th Intl. Conf. on Foundations of Data Organization and Algorithms (FODO '93)*, pages 146–163, 1993.
10. D. L. Lee and C. Leng. "Partitioned Signature Files: Design Issues and Performance Evaluation,". *ACM Trans. Off. Inf. Syst.*, 7(2):158–180, 1989.
11. W. Lee and D. L. Lee. "Signature File Methods for Indexing Object-Oriented Database Systems,". In *Intl. Computer Science Conf. (ICSC)*, pages 616–622, Hong Kong, 1992.
12. Z. Lin and C. Faloutsos. "Frame-Sliced Signature Files,". *IEEE Trans. Knowl. and Data Eng.*, 4(3):281–289, 1992.
13. F. Rabitti and P. Savino. "Image Query Processing Based on Multi-level Signatures,". In *Proc. of 14th Annual Intl. ACM/SIGIR Conf. on Research and Development in Information Retrieval*, pages 305–314, Chicago, Illinois, Oct. 1991.
14. K. Ramamohanarao and J. A. Shepherd. "A Superimposed Codeword Indexing Scheme for Very Large Prolog Databases,". In *Proc. of the Third Intl. Conf. on Logic Programming*, pages 569–576, London, 1986.
15. C. S. Roberts. "Partial-Match Retrieval via the Method of Superimposed Codes,". *Proc. IEEE*, 67(12):1624–1642, Dec. 1979.
16. S. Stiassny. "Mathematical Analysys of Various Superimposed Coding Methods,". *American Documentation*, 11(2):155–169, 1960.
17. 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.
18. N. Watanabe and H. Kitagawa. "Design and Evaluation of Partitioned Bit-Sliced Signature File,". *IPS Japan SIGDBS*, 95-DBS-104:1–8, July 1995. (in Japanese).
19. S. B. Yao. "Approximating Block Accesses in Database Organizations,". *CACM*, 20(4):260–261, 1977.
20. H.-S. Yong, S. Lee, and H.-J. Kim. "Applying Signatures for Forward Traversal Query Processing in Object-Oriented Databases,". In *Proc. of 10th Intl. Conf. on Data Engineering*, pages 518–525, Houston, Texas, Feb. 1994. IEEE.
21. P. Zezula, P. Ciaccia, and P. Tiberio. "Key-Based Partitioned Bit-Sliced Signature File,". *ACM SIGIR Forum*, 29(2):20–34, Fall 1995.
22. P. Zezula, F. Rabitti, and P. Tiberio. "Dynamic Partitioning of Signature Files,". *ACM Trans. Inf. Syst.*, 9(4):336–369, 1991.

This article was processed using the LaTeX macro package with LLNCS style