HIROSHIMA MATH. J.
8 (1978), 515–544

Combinatorial File Organization Schemes and their Experimental Evaluation

Hideto IKEDA

(Received May 20, 1978)

1. Introduction

In an information storage and retrieval system, the secondary indexes are organized in order to retrieve pertinent records quickly from the master file. For such purpose, a set of accession numbers of pertinent records, called a bucket, is sometimes organized to each of the indexes. An inverted file organization scheme of order one (abbreviated by IFS_1) is a method of organizing these sets. For an index specified by a query of order one, the accession numbers of pertinent records in the master file can be found quickly, because the corresponding set of accession numbers is much smaller in size than the master file and is organized in consecutive locations.

In order to answer the retrieval request for a query of order two, which specifies two indexes simultaneously, in IFS_1, it is necessary to execute an AND-operation on two sets of accession numbers. Such an operation may sometimes require a considerable amount of the computer time. If a set of accession numbers of all records pertinent to each query of order two is prepared in advance, the answer can be obtained more quickly. An inverted file organization scheme of order two (abbreviated by IFS_2) is a scheme obtained by a collection of such sets. Such scheme might be effective in the retrieval of a query of order two. The accession number of a record must be stored quite redundantly in a number of storage locations. The scheme IFS_2, therefore, has serious disadvantages in space and time.

Several attempts have been done to reduce the disadvantages of IFS_2. Among others, the balanced file organization scheme of order two (abbreviated by BFS_2) by Abraham, Ghosh, and Ray-Chaudhuri [1], the new balanced file organization scheme of order two (abbreviated by $NBFS_2$) by Chow [6], and our Hiroshima University balanced file organization scheme of order two (abbreviated by $HUBFS_2$) by Yamamoto, Ikeda (the present author), Shige-eda, Ushio and Hamada [17], have been proposed.

A generalized Hiroshima University balanced file organization scheme of order two (abbreviated by $GHUBFS_2$) proposed in our above mentioned paper [17] is so designed that the queries of order one as well as order two can be retrieved quickly. Not only it has, though theoretically, the least redundancy among generalized balanced file organization schemes of order two but also it can have the same redundancy with IFS_1 by an appropriate selection of parameters.

It is assumed in these schemes that each record is characterized by binary-valued attributes. File organization schemes for the records with multiple-valued attributes can be seen in several literatures (e.g., [2], [3], [4], [7], [18], [19] and [20]). Furthermore, different kinds of file organization schemes having a consecutive retrieval property have been considered in some literatures (cf. [8], [9], [10], [11] and [21]).

In this paper, *we shall confine ourselves to the file organization schemes for records with binary-valued attributes and consider* $GHUBFS_2$ *in detail by giving its addressing function explicitly. The implementations of* $GHUBFS_2$ *and* IFS_1 *to an on-line document retrieval system which has been announced in* Ikeda [12] *are also presented.* This system is developed in order to evaluate experimentally $GHUBFS_2$ and IFS_1 by comparing the execution times for retrieval of queries of order one as well as of order two and for storing given bibliographic data in the system.

In Section 2, a combinatorial file organization scheme of order k is defined formally in relation to the addressing function which corresponds each query of order k to the address of its bucket. IFS_1 and IFS_2 are also defined there. In Section 3, a balanced file organization scheme of order k (abbreviated by BFS_k) is defined, and addressing functions of BFS_2, $NBFS_2$ and $HUBFS_2$ are discussed.

In Section 4, we give an explicit expression of an addressing function of $HUBFS_2$ by showing more explicitly a claw-decomposition of a complete graph (cf. [16]). In Section 5, a generalized balanced file organization scheme of order k is defined formally, and $GHUBFS_2$ is also considered. By using the addressing function of $HUBFS_2$ given in Theorem 2, we give an addressing function of $GHUBFS_2$ explicitly in Theorem 3. The redundancy of a file organization scheme and the theoretical evaluation of the redundancy are discussed in Section 6.

Finally in Section 7, a result of the experimental evaluation of $GHUBFS_2$ and IFS_1 by the growth of data will be given. The result suggests that $GHUBFS_2$ can answer retrieval requests for queries of order two much faster than IFS_1 and can reduce the number of buckets organized for queries of order two in IFS_2. Although the space requirement of $GHUBFS_2$ is slightly larger than that of IFS_1, the time requirement of $GHUBFS_2$ for storing is nearly as large as that of IFS_1, since the number of access to the buckets for storing a record can be reduced to the same by an appropriate selection of parameters.

2. Definition of a combinatorial file organization scheme

In this section, the basic consideration on some concepts and notations related to combinatorial file organization schemes will be treated. A familiar scheme called an *inverted file organization scheme* (IFS, inventor unknown) will also be summarized.

Let $\boldsymbol{\Omega}$ be a collection of records, called a master file. Suppose that every record ω of the file $\boldsymbol{\Omega}$ has an *identification*, or an *accession number*, which is denoted by a_ω. Suppose, furthermore, that every record $\omega \in \boldsymbol{\Omega}$ is characterized by l attributes $A_1, A_2, \ldots, A_l$ or equivalently by an l-dimensional 0–1 vector

$$X(\omega) = (\delta_1 \delta_2 \cdots \delta_l) \in \{0, 1\}^l$$

where δ_i takes 1 when ω has the ith attribute A_i and takes 0 otherwise.

A *canonical query of order k* with respect to $\boldsymbol{\Omega}$ is a retrieval request which is specified by a set $Q_{i_1 i_2 \cdots i_k}$ of k attributes $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$, where $1 \leq i_1 < i_2 < \cdots < i_k \leq l$. The *set* $\rho(Q_{i_1 i_2 \cdots i_k})$ *of records relevant to a canonical query* $Q_{i_1 i_2 \cdots i_k}$ *of order k* is defined by

$$\rho(Q_{i_1 i_2 \cdots i_k}) = \{\omega \in \boldsymbol{\Omega} | X(\omega) = (\delta_1 \delta_2 \cdots \delta_l),\ \delta_{i_1} = \delta_{i_2} = \cdots = \delta_{i_k} = 1\}.$$

It is easy to show that

$$\rho(Q_{i_1 i_2 \cdots i_k}) = \bigcap_{\nu=1}^{k} \rho(Q_{i_\nu})$$

holds by definition. In the following, a canonical query of order k will be referred to a *query of order k* or a *kth order query*.

In general, a *retrieval request* is to search every record whose characteristic vector belongs to a given subset $\boldsymbol{\Delta}$ of the space of all l-dimensional nonzero 0–1 vectors.

Because the set $\{\omega \in \boldsymbol{\Omega} | X(\omega) \in \boldsymbol{\Delta}\}$ is constructed from the sets $\rho(Q_i)$, $i = 1, 2, \ldots, l$, by the set-operations $\cap$, $\cup$ and $-$, we have the following

LEMMA 1. *If we can answer any query Q_i of order one, i.e., if we can search every record relevant to any Q_i, then we can also answer any retrieval request.*

In order to answer a retrieval request, the set of accession numbers of all records relevant to the request is often organized in the computer storage. Such a set of accession numbers is often called a *secondary index*. A combinatorial file organization scheme is a method of organizing a collection of secondary indexes with respect to a set of queries. Thus an essential part of a file organization scheme is to define an addressing function of the set of queries, in so far as the intrastructure of secondary indexes is disregarded.

DEFINITION 1. *A combinatorial file organization scheme of order k is defined to be a function β from the set of all canonical queries of order k onto the set of integers* $\{1, 2,\dots, b\}$.

The set $\{1, 2,\dots, b\}$ is sometimes referred to the *address set* and the function β is referred to the *addressing function* of the scheme. The secondary index

$$B_t = \{a_\omega | \omega \in \rho(Q) \quad \text{for some} \quad Q \in \beta^{-1}(t)\}$$

is called the tth *bucket* for any $t=1, 2,\dots, b$. The number t is called the *address* or the *bucket identification number* of B_t.

An *inverted file organization scheme of order one* (abbreviated by IFS_1) is a primitive type of combinatorial file organization scheme of order one.

DEFINITION 2. *A file organization scheme of order one with l attributes is an* IFS_1, *if its addressing function β is a one-to-one function from the set of all queries of order one onto the address set* $\{1, 2,\dots, l\}$, *e.g., by* $\beta(Q_i)=i$.

Consider, for example, a file $\boldsymbol{\Omega}$ which is composed of ten records $\omega_1, \omega_2,\dots, \omega_{10}$ characterized by three attributes A_1, A_2 and A_3. Figure 2.1 is an incidence matrix of records vs attributes, that is, the mth column vector of the matrix indicates the characteristic vector of a record ω_m.

$$\begin{array}{c|cccccccccc}
 & \omega_1 & \omega_2 & \omega_3 & \omega_4 & \omega_5 & \omega_6 & \omega_7 & \omega_8 & \omega_9 & \omega_{10} \\
\hline
A_1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
A_2 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
A_3 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1
\end{array}$$

Figure 2.1

The buckets are

$$B_1 = \{a_1, a_3, a_4, a_6, a_7, a_8, a_9\},$$

$$B_2 = \{a_1, a_2, a_6, a_7\},$$

$$B_3 = \{a_2, a_5, a_6, a_8, a_{10}\},$$

where $a_m = a_{\omega_m}$ is the accession number of a record ω_m.

If an IFS_1 is organized, the access to the ith bucket B_i is sufficient to answer a query Q_i of order one. If it is necessary to display the contents of relevant records, they are obtained by accession numbers in the bucket.

For a query Q_{ij} of order two, however, an AND-operation has to be performed on two sets B_i and B_j. This operation may sometimes require a considerable amount of the computer time.

If every set of accession numbers of all records relevant to each query of order two is prepared in advance, then the answer can be obtained more quickly. Such a file organization scheme of order two is called an *inverted file organization scheme of order two* (abbreviated by IFS_2).

DEFINITION 3. *A file organization scheme of order two with l attributes is an* IFS_2, *if its addressing function β is a one-to-one function from the set* $\{Q_{ij} | 1 \leq i < j \leq l\}$ *of all queries of order two onto the address set* $\{1, 2, \ldots, l(l-1)/2\}$, e.g.,

$$\beta(Q_{ij}) = (2l - i)(i - 1)/2 - i + j.$$

Although IFS_2 might be effective in the retrieval of a query of order two, it has serious disadvantages in the retrieval of a query of order one. Namely for a query Q_i of order one with respect to l attributes, it is necessary to perform an OR-operation

$$\bigcup_{j=1}^{i-1} B_{\beta(Q_{ji})} \cup \bigcup_{j=i+1}^{l} B_{\beta(Q_{ij})} \tag{2.1}$$

Furthermore, when there is a record ω in $\boldsymbol{\Omega}$ having a characteristic vector $(0\cdots0\overset{i}{1}0\cdots0)$, then ω is included in $\rho(Q_i)$, but its accession number a_ω is not included in the set (2.1). This indicates that it is not sufficient to construct the set (2.1) for the retrieval of Q_i.

Not only IFS_2 but also every file organization scheme of order two has this incompleteness for the retrieval of a query of order one.

3. Balanced file organization schemes

DEFINITION 4. *A file organization scheme of order k with l attributes and b addresses is said to be balanced if the addressing function β satisfies*

$$|\beta^{-1}(t)| = c, \qquad \textit{for every} \quad t = 1, 2, \ldots, b,$$

where c is a positive integer and $|S|$ denotes the number of elements in a finite set S.

It is easy to show that the constant c has to satisfy

$$\binom{l}{k} = bc, \tag{3.1}$$

and c is called the *number of queries corresponding to a bucket.* A balanced file organization scheme of order k is abbreviated by BFS_k.

For example, an IFS_1 and an IFS_2 are balanced file organization schemes with $c=1$.

Now we give an example of BFS_1 with $c\neq 1$. Assume that l is an integral multiple of b, and consider a function β given by

$$\beta(\mathrm{Q}_i) = [(i + c - 1)/c] \in \{1, 2,\dots, b\} \qquad \text{for} \quad i = 1, 2,\dots, l,$$

where $c=l/b$ and $[p]$ denotes the largest integer not exceeding p. Then this function β defines a BFS_1, and

$$\beta^{-1}(t) = \{\mathrm{Q}_{(t-1)c+1}, \mathrm{Q}_{(t-1)c+2},\dots, \mathrm{Q}_{tc}\} \qquad \text{for} \quad t = 1, 2,\dots, b.$$

In the rest of this section, we shall define various file organization schemes of order two by giving their addressing functions.

Abraham, Ghosh and Ray-Chaudhuri [1] have constructed a BFS_2 by using a finite projective geometry. Consider a finite projective geometry $\mathrm{PG}(N, s)$ of N-dimension based on the Galois field $\mathrm{GF}(s)$, where $s=p^n$, p is a prime integer and n is a positive integer. Associate each point P_i ($i=1, 2,\dots, l$; $l=s^N+s^{N-1}+\cdots+s+1$) of $\mathrm{PG}(N, s)$ with an attribute A_i. Assign each line L_t in $\mathrm{PG}(N, s)$ to a number t in $\{1, 2,\dots, b\}$ where $b=(s^{N+1}-1)(s^N-1)/(s^2-1)(s-1)$. Then for a given query Q_{ij} of order two, there corresponds a unique line L_t passing through two points P_i and P_j, and we can define a function β by

$$\beta(\mathrm{Q}_{ij}) = t.$$

Since each line of $\mathrm{PG}(N, s)$ contains $s+1$ points, we have

$$|\beta^{-1}(t)| = \binom{s+1}{2}$$

for all t in $\{1, 2,\dots, b\}$, and hence β defines a BFS_2.

They [1] have also constructed a BFS_2 by using a finite Euclidean geometry. Consider a finite Euclidean geometry $\mathrm{EG}(N, s)$ of N-dimension based on the Galois field $\mathrm{GF}(s)$, where $s=p^n$, p is a prime integer and n is a positive integer. Associate each point with an attribute, and a query of order two can correspond uniquely to a line. This correspondence defines an addressing function of a BFS_2 based on $\mathrm{EG}(N, s)$.

More generally, such schemes can be also constructed by giving a balanced incomplete block design (BIBD) (cf. Ray-chaudhuri [13] or Bose and Koch [4]).

Chow [6] proposed some BFS_2 based on an ordering of the queries of order two (abbreviated by NBFS_2 after Chow). He gave various addressing functions. Some of them are given as follows:

$$\beta(\mathrm{Q}_{ij}) = [\{(i - 1)l + j - i(i - 1)/2 + c - 1\}/c],$$

$$\beta'(\mathrm{Q}_{ij}) = [\{(j-1)(j-2)/2 + i + c - 1\}/c].$$

Furthermore, Yamamoto, Ikeda, Shige-eda, Ushio and Hamada [17] have designed a different type of BFS_2, called an HUBFS_2. If we identify each attribute A_i with a vertex v_i, then each query Q_{ij} of order two is identified with an edge E_{ij} connecting between vertices v_i and v_j. A set B of queries of order two can, therefore, be considered as a graph, which is called a *graphical structure* of B.

DEFINITION 5. *A file organization scheme of order two with l attributes and b addresses is called an* HUBFS_2, *if its addressing function β satisfies that the graphical structure of every set $\beta^{-1}(t)$* ($\mathrm{t}=1, 2,\ldots, b$) *is a claw with c edges connecting between one root-point and c vertices* (*or a complete bipartite graph* $\mathrm{K}_{1,c}$), *where* $bc=\binom{l}{2}$.

Then, the following theorem has been proved.

THEOREM 1 ([17, Th. 4.1]). *An* HUBFS_2 *with l attributes and b addresses can be constructed if and only if the following conditions* (i) *and* (ii) *are satisfied*:

(i) $\binom{l}{2}$ *is an integral multiple b, and*

(ii) $l \geq 2c$, *where* $c=\binom{l}{2}\Big/b$.

An addressing function β of HUBFS_2 will be given explicitly in the following section. Table 3.1 illustrates an example of HUBFS_2 with parameters $l=9$, $b=12$, $c=3$.

Table 3.1

HUBFS_2 with $l=9$, $b=12$, $c=3$

Address of bucket

j \ i	1	2	3	4	5	6	7	8
9	12	12	6	12	11	9	11	11
8	10	5	6	10	8	9	10	
7	3	3	3	7	8	9		
6	2	2	2	7	8			
5	1	1	1	7				
4	4	5	6					
3	4	5						
2	4							
1								

Buckets and corresponding queries

Bucket	Queries		
B_1	Q_{15}	Q_{25}	Q_{35}
B_2	Q_{16}	Q_{26}	Q_{36}
B_3	Q_{17}	Q_{27}	Q_{37}
B_4	Q_{12}	Q_{13}	Q_{14}
B_5	Q_{23}	Q_{24}	Q_{28}
B_6	Q_{34}	Q_{38}	Q_{39}
B_7	Q_{45}	Q_{46}	Q_{47}
B_8	Q_{56}	Q_{57}	Q_{58}
B_9	Q_{67}	Q_{68}	Q_{69}
B_{10}	Q_{18}	Q_{48}	Q_{78}
B_{11}	Q_{59}	Q_{79}	Q_{89}
B_{12}	Q_{19}	Q_{29}	Q_{49}

4. An explicit definition of an addressing function of HUBFS_2

Let T be the triangular set of lattice points $\{(i, j) | 1 \leq i < j \leq l\}$ in the Euclidean plane. The set of c edges of a claw $K_{1,c}$ can be identified with a subset of c lattice points standing together on the same ith row and/or ith column. Such a subset may be called a *claw-type subset of* T.

Then we see immediately the following by Definition 5.

LEMMA 2. *Let β be an addressing function of a file organization scheme. If every set $\{(i, j) | \beta(\mathrm{Q}_{ij}) = t\}$ $(t = 1, 2, \ldots, b)$ is a claw-type subset of* T *with c points, then β defines an* HUBFS_2.

We have proved the sufficiency of Theorem 1 by using this lemma and by giving only an algorithm of the decomposition of T into mutually disjoint b claw-type subsets with c points.

In the rest of this section, we shall define explicitly an addressing function β of an HUBFS_2 by giving such a decomposition. For this purpose, we prepare the following lemma.

LEMMA 3 (cf. [16, Th. 1.1]). *Suppose that given nonnegative integers $r_1, r_2 \ldots, r_m$ and $s_1, s_2, \ldots, s_n$ satisfy the conditions*

$$r_i = r \quad \text{if} \quad 1 \leq i \leq a, \qquad r_i = r + 1 \quad \text{if} \quad a + 1 \leq i \leq m,$$

for some a in $\{0, 1, \ldots, m\}$ and r; $0 \leq s_j \leq m$ for $j = 1, 2, \ldots, n$; and

$$\sum_{j=1}^{n} s_j = \sum_{i=1}^{m} r_i = rm + m - a.$$

If (t_{ij}) is the $m \times n$ 0–1 matrix defined by

$$t_{ij} = \begin{cases} 1 & \text{if} \quad i \equiv a + \sum_{k=1}^{j-1} s_k + s \ (\mathrm{mod}\, m) \qquad \text{for some } s \text{ in } \{1, 2, \ldots, s_j\}, \\ 0 & \text{otherwise,} \end{cases}$$

then its row and column sum vectors are $(r_1, r_2, \ldots, r_m)$ and $(s_1, s_2, \ldots, s_n)$, respectively, i.e., $\sum_{j=1}^{n} t_{ij} = r_i$ and $\sum_{i=1}^{m} t_{ij} = s_j$.

PROOF. For $j = 1, 2, \ldots, n$, consider the set $\mathrm{U}_j = \{a + \sum_{k=1}^{j-1} s_k + s | s = 1, 2, \ldots, s_j\}$. Since $s_j \leq m$ by assumption, we see immediately that $u \equiv u' \ (\mathrm{mod}\, m)$ for $u, u' \in \mathrm{U}_j$ implies $u = u'$. Thus we see $\sum_{i=1}^{m} t_{ij} = s_j$ by definition.

By assumption $\sum_{j=1}^{n} s_j = (r+1)m - a$, we have $\bigcup_{j=1}^{n} \mathrm{U}_j = \{a+1, a+2, \ldots, (r+$

$1)m\}$. For $i=1, 2,\ldots, m$, this set contains integers $pm+i$ for

$$p = 1, 2,\ldots, r \quad \text{if} \quad i \le a, \quad \text{and} \quad p = 0, 1, 2,\ldots, r \quad \text{if} \quad i \ge a + 1.$$

Thus we see immediately $\sum_{j=1}^{n} t_{ij}=r_i$ by definition. This completes the proof.

For integers p and m, let $[p]_m$ be the integer q satisfying $1\le q\le m$ and $p\equiv q \pmod m$, i.e., $[p]_m=(p-1)(\mathrm{mod}\, m)+1$. Also, for integers a_1, a_2, b and m, we shall use the notation $[a_1]_m\to b\to[a_2]_m$ in the following sense:

(i) If $[a_1]_m\le[a_2]_m$, then $[a_1]_m<b<[a_2]_m$.

(ii) If $[a_1]_m>[a_2]_m$, then $[a_1]_m<b\le m$ or $1\le b<[a_2]_m$.

Then, the condition $t_{ij}=1$ in Lemma 3 is restated by the condition

$$[a + \sum_{k=1}^{j-1} s_k]_m \longrightarrow i \longrightarrow [a + 1 + \sum_{k=1}^{j} s_k]_m.$$

Thus, we see easily that Lemma 3 for $s_j=n-j$ $(j=1, 2,\ldots, n)$ is restated by the set $\mathrm{I}=\{(i, j)|t_{ij}=1\}$ as follows:

LEMMA 4. *Suppose that given integers a, n and m satisfy*

$$m \ge n \ge 1, \quad m \ge a \ge 0, \quad \textit{and}$$

$$a + n(n-1)/2 = (b_1 + 1)m \qquad \textit{for some integer } b_1 \ge 0.$$

Let $\mathrm{I}(a, m, n)$ be the subset of the product set $\mathrm{Q}(m, n)=\{1, 2,\ldots, m\}\times\{1, 2,\ldots, n\}$ consisting of all $(i, j)\in\mathrm{Q}(m, n)$ such that

$$[a + \sum_{k=1}^{j-1}(n-k)]_m \longrightarrow i \longrightarrow [a + 1 + \sum_{k=1}^{j}(n-k)]_m;$$

and decompose $\mathrm{Q}(m, n)$ into three mutually disjoint subsets

$$\mathrm{K}(a, m, n) = \mathrm{I}(a, m, n) \cap \mathrm{Q}(a, n), \quad \mathrm{L}(a, m, n) = \mathrm{I}(a, m, n) - \mathrm{K}(a, m, n),$$

and $\mathrm{M}(a, m, n)=\mathrm{Q}(m, n)-\mathrm{I}(a, m, n)$. Then the following hold:

$$|\{j|(i, j)\in\mathrm{K}(a, m, n)\}| = b_1 \qquad \textit{for} \quad i = 1, 2,\ldots, a,$$

$$|\{j|(i, j)\in\mathrm{L}(a, m, n)\}| = b_1 + 1 \qquad \textit{for} \quad i = a + 1, a + 2,\ldots, m;$$

$$|\{i|(i, j)\in\mathrm{M}(a, m, n)\}| = m - n + j \qquad \textit{for} \quad j = 1, 2,\ldots, n.$$

Also, for the case that $m+1\le n\le 2m+1$ and

$s_j = m+1-j\ (j=1, 2,\dots, m+1),\quad s_j = n-j\ (j=m+2, m+3,\dots, n),$

Lemma 3 is the following form:

LEMMA 5. *Suppose that given integers a, n and m satisfy*

$$m \geq n' = n-m-1 \geq 1,\quad m \geq a \geq 0,\quad \text{and}$$

$$a' + n'(n'-1)/2 = (b_2+1)m \qquad \text{for some integer } b_2 \geq 0,$$

where $a'=a+m(m+1)/2$. Let $\mathrm{I}'(a, m, n)$ be the subset of $\mathrm{Q}(m, n)$ consisting of all (i, j) such that

$$[a+\sum_{k=1}^{j-1}(m+1-k)]_m \longrightarrow i \longrightarrow [a+1+\sum_{k=1}^{j}(m+1-k)]_m$$
$$\text{if}\quad 1 \leq j \leq m+1,$$

$$[a'+\sum_{k=m+2}^{j-1}(n-k)]_m \longrightarrow i \longrightarrow [a'+1+\sum_{k=m+2}^{j}(n-k)]_m$$
$$\text{if}\quad m+2 \leq j \leq n;$$

and decompose $\mathrm{Q}(m, n)$ into three mutually disjoint subsets

$$\mathrm{K}'(a, m, n) = \mathrm{I}'(a, m, n) \cap \mathrm{Q}(a, n),$$

$$\mathrm{L}'(a, m, n) = \mathrm{I}'(a, m, n) - \mathrm{K}'(a, m, n),$$

and $\mathrm{M}'(a, m, n)=\mathrm{Q}(m, n)-\mathrm{I}'(a, m, n)$. Then the following hold:

$$|\{j|(i, j) \in \mathrm{K}'(a, m, n)\}| = b_2 \qquad \text{for}\quad i=1, 2,\dots, a,$$

$$|\{j|(i, j) \in \mathrm{L}'(a, m, n)\}| = b_2+1 \qquad \text{for}\quad i=a+1, a+2,\dots, m;$$

$$|\{i|(i, j) \in \mathrm{M}'(a, m, n)\}| = \begin{cases} j-1 & \text{for}\quad j=1, 2,\dots, m+1, \\ m-n+j & \text{for}\quad j=\mathrm{m}+2, m+3,\dots, n. \end{cases}$$

Now, let l, b and c be integers satisfying (i) and (ii) in Theorem 1, and consider the three cases $2c \leq l < 3c$, $3c \leq l < 4c$ and $4c \leq l$. Then we can define a decomposition of the triangular set T into b claw-type subsets with c points and an addressing function $\beta^{(n)}$ from the set $\{\mathrm{Q}_{ij}|1 \leq i < j \leq l\}$ of all queries of order two onto the address set $\{1, 2,\dots, b\}$ as follows, where $bc=\binom{l}{2}$.

Case 1: $2c \leq l < 3c$. Put

$$l = 2c+r,\quad b = \binom{l}{2}\Big/c = 2c-1+2r+b_1.$$

Then $0 \leq r < c$, $b_1 = r(r-1)/2c$, and b_1 is zero for $r=0$ or 1, and b_1 is an integer satisfying $0 \leq b_1 < (r-1)/2$. The triangular set T can be decomposed into thirteen disjoint subsets shown in Table 4.1, where K, L and M are obtained by the transposition of two coordinates and a parallel transformation from the sets in Lemma 4 for $a = c-1-b_1$, $n=r$, $m=c-1$ which satisfy

$$c - 1 - b_1 + r(r-1)/2 = (b_1 + 1)(c - 1)\,;$$

and then the function $\beta^{(1)}$ is defined there. By Figure 4.1 and by using the above equality and the ones in Lemma 4, we see easily that $\{(i, j) | \beta^{(1)}(Q_{ij}) = t\}$ is a claw-type subset of T with c points for any $t = 1, 2, \ldots, b$. Thus $\beta^{(1)}$ is an addressing function of an $HUBFS_2$ by Lemma 2.

Table 4.1

Definition of an addressing function of an $HUBFS_2$ $(2c \leq l < 3c)$

Definition of the area	$\beta^{(1)}(Q_{ij})$
A: $1 \leq i < j \leq c+1$	i
B: $c+1 \leq i < j \leq c+r+1$	i
C: $c+r+1 \leq i < j \leq l$	$j-1$
D: $1 \leq i \leq c$, $c+2 \leq j \leq c+r+1$	$j+c+r-2$
E: $1 \leq i$, $c+r+2 \leq j$, $i+j \leq 2c+r-b_1+1$	$j-1$
F: $i \leq c-b_1-1$, $j \leq 2c+r-b_1$, $i+j \geq 2c+r-b_1+2$	i
G: $1 \leq i \leq c-b_1-1$, $2c+r-b_1+1 \leq j \leq l$	$j+r+b_1-1$
H: $c-b_1 \leq i \leq c$, $c+r+2 \leq j \leq 2c+r-b_1$	i
I: $c-b_1 \leq i$, $2c+r-b_1+1 \leq j$, $i+j \leq l+c-b_1+1$	$j-1$
J: $i \leq c$, $j \leq l$, $i+j \geq l+c-b_1+2$	i
K: $(j-c-r-1, i-c) \in K(c-1-b_1, c-1, r)$	$j-1$
L: $(j-c-r-1, i-c) \in L(c-1-b_1, c-1, r)$	$j+r+b_1-1$
M: $(j-c-r-1, i-c) \in M(c-1-b_1, c-1, r)$	i

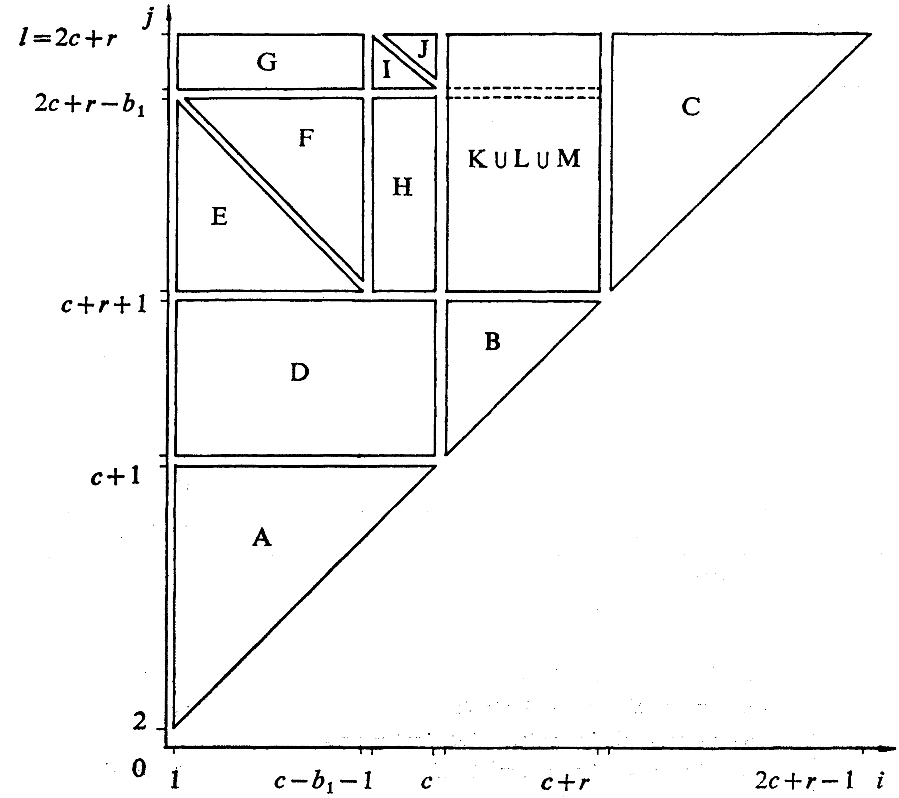

Figure 4.1 $(2c \leq l < 3c)$

Case 2: $3c \leq l < 4c$. Put

$$l = 3c + r, \quad b = \binom{l}{2}\Big/c = 4c + 3r - 1 + b_2.$$

Then $0 \leq r < c$, $b_2 = \{c(c-1) + r(r-1)\}/2c$, and b_2 is a positive integer satisfying $(c-1)/2 \leq b_2 < c-1$. In this case, T will be divided into the fifteen disjoint subsets shown in Table 4.2, where K′, L′ and M′ are obtained by the transposition of two coordinates and a parallel transformation from the sets in Lemma 5 for $a = c-1-b_2$, $n = c+r$, $m = c-1$ which satisfy

$$c - 1 - b_2 + c(c-1)/2 + r(r-1)/2 = (b_2 + 1)(c-1);$$

and then the function $\beta^{(2)}$ is defined there. By Figure 4.2 and by using the above equality and the ones in Lemma 5, we see easily that $\{(i, j) | \beta^{(2)}(Q_{ij}) = t\}$ is a claw-type subset of T with c points for any t. Thus $\beta^{(2)}$ is an addressing function of an HUBFS_2 by Lemma 2.

Table 4.2

Definition of an addressing function of an HUBFS$_2$ $(3c \le l < 4c)$

Definition of the area	$\beta^{(2)}(Q_{ij})$
A_1 : $1 \le i < j \le c+1$	i
A_2 : $c+1 \le i < j \le 2c+1$	i
B : $2c+1 \le i < j \le 2c+r+1$	i
C : $2c+r+1 \le i < j \le l$	$j-1$
D_1 : $1 \le i \le c$, $c+2 \le j \le 2c+r+1$	$j+2c+r-2$
D_2 : $c+1 \le i \le 2c$, $2c+2 \le j \le 2c+r+1$	$j+2c+2r-2$
E : $1 \le i$, $2c+r+2 \le j$, $i+j \le l-b_2+1$	$j-1$
F : $i \le c-b_2-1$, $j \le l-b_2$, $i+j \ge l-b_2+2$	i
G : $1 \le i \le c-b_2-1$, $l-b_2+1 \le j \le l$	$j+c+2r+b_2-1$
H : $c-b_2 \le i \le c$, $2c+r+2 \le j \le l-b_2$	i
I : $c-b_2 \le i$, $l-b_2+1 \le j$, $i+j \le l+c-b_2+1$	$j-1$
J : $i \le c$, $j \le l$, $i+j \ge l+c-b_2+2$	i
K′ : $(j-2c-r-1, i-c) \in K'(c-1-b_2, c+r, c-1)$	$j-1$
L′ : $(j-2c-r-1, i-c) \in L'(c-1-b_2, c+r, c-1)$	$j+c+2r+b_2-1$
M′ : $(j-2c-r-1, i-c) \in M'(c-1-b_2, c+r, c-1)$	i

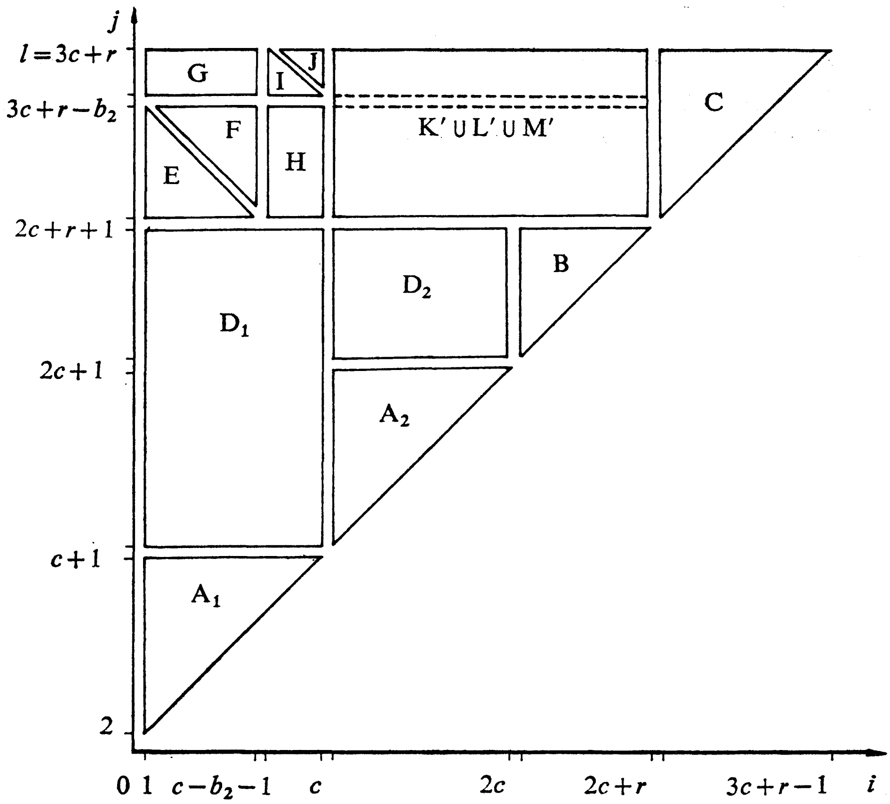

Figure 4.2 $(3c \le l < 4c)$

Case 3: $4c \leq l$. There exist positive integers n and l_0 satisfying

$$l = 2nc + l_0 \quad \text{and} \quad 2c \leq l_0 < 4c.$$

In this case, T can be divided into $2n+1$ subsets shown in Table 4.3, and we can apply the considerations for Case 1 or Case 2 on T_0 since $2c \leq l_0 < 4c$, and that for Case 1 with $l = 2c+1$ on V_p; and then the function $\beta^{(3)}$ is defined there. By Figure 4.3 and by the results for Case 1 or Case 2, we see easily that $\beta^{(3)}$ satisfies the condition in Lemma 2 and hence $\beta^{(3)}$ is an addressing function of an $HUBFS_2$.

Table 4.3

Definition of an addressing function of an $HUBFS_2$ $(4c \leq l)$

Definition of the area	$\beta^{(3)}(Q_{ij})$
T_0: $1 \leq i < j \leq l_0$	$\beta^{(1)}(Q_{ij})$ if $l_0 < 3c$ $\beta^{(2)}(Q_{ij})$ if $l_0 \geq 3c$
V_p: $l_p \leq i < j \leq l_{p+1}$	$\beta^{(1)}(Q_{i'j'}) + (2c+1)p + l_0(l_0-1)/2c$
U_p: $1 \leq i \leq l_p - 1$, $l_p \leq j \leq l_{p+1}$ for $p = 0, 1, \ldots, n-1$	$i + \left\lceil \frac{j - l_p - 1}{c} \right\rceil (l_p - 1) + p(p-1)(2c-1) + 2p$ $+ (2c+1)n + l_0(l_0-1)/2c$

where $l_p = l_0 + 2pc$, $i' = i - l_p + 1$ and $j' = j - l_p + 1$.

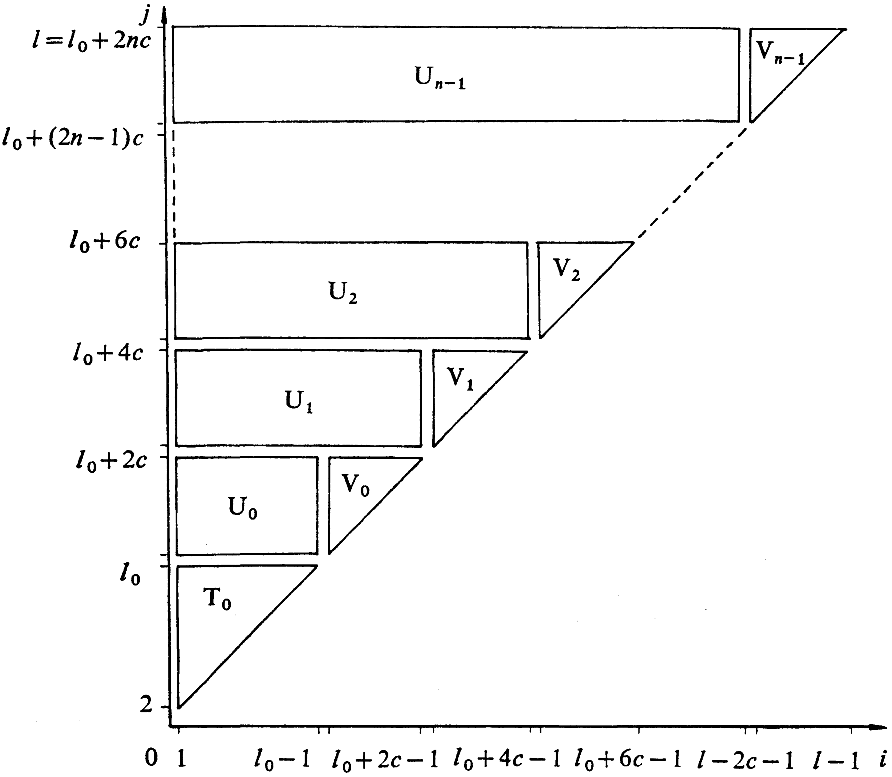

Figure 4.3 ($l \geq 4c$)

By the above arguments, we have proved the following

THFOREM 2. *Let l, b and c be the integers satisfying the conditions* (i) *and* (ii) *in Theorem* 1, *and consider the following three cases*:

Case 1: $2c \leq l < 3c$, *Case* 2: $3c \leq l < 4c$, *Case* 3: $4c \leq l$.

Then, the function $\beta = \beta^{(n)}$ given in Table 4.*n for Case n* ($n = 1, 2, 3$) *is an addressing function of an* HUBFS_2 *with l attributes and b addresses.*

5. Generalized balanced file organization schemes

As it is stated in Section 2, a file organization scheme of order two has serious disadvantages in the retrieval of a query of order one. In order to overcome such disadvantages, we shall define a *generalized balanced file organization scheme of order k* (abbreviated by GBFS_k) as follows.

DEFINITION 6. *A* GBFS_k *is defined to be a k-tuple* $(\beta_1, \beta_2, \ldots, \beta_k)$ *of func-*

tions satisfying the following conditions:

(i) β_v *is an addressing function from the set of all queries of order* v *onto the set* $I_v(\subset\{1, 2,\dots, b\})$,

(ii) $|I_v|$ *is an divisor of* $\binom{l}{v}$, *and* $|\beta_v^{-1}(t)|=\binom{l}{v}/|I_v|$ *for every* $t\in I_v$, *and for* $v=1, 2,\dots, k$,

and

(iii) $\bigcup_{v=1}^{k} I_v = \{1, 2,\dots, b\}$.

The integer b is called the *total number of addresses* and the integer $|I_v|$ is called the *number of addresses corresponding to queries of order* v. In $GBFS_k$, the tth bucket B_t is defined to be the set $\bigcup_{v\in J}\{a_\omega|\rho(Q)\ni\omega$ for some $Q\in\beta_v^{-1}(t)\}$ where $J=\{v|t\in I_v\}$.

For instance, the pair (β_1, β_2), defined by

$$\beta_1(Q_i) = i \qquad \text{for } 1\leq i\leq l, \text{ and}$$

$$\beta_2(Q_{ij}) = \frac{(2l-i-1)i}{2} - l + j \qquad \text{for } 1\leq i<j\leq l,$$

is a pair of addressing functions of a $GBFS_2$.

Let β_2 be an addressing function of a BFS_2 with l attributes and b_2 addresses. Define a function β_1 from the set of all queries of order one into the set $\{1, 2,\dots, b\}$ as $\beta_1(Q_i)=b_2+i$ for $i=1, 2,\dots, l$ where $b=b_2+l$. Then the pair (β_1, β_2) defines a $GBFS_2$.

Another example of $GBFS_2$ can be seen in Yamamoto, Teramoto and Futagami [20].

Now, we consider a *generalized Hiroshima University balanced file organization scheme of order two* (abbreviated by $GHUBFS_2$).

DEFINITION 7 ([17]). *A* $GBFS_2$ *with* l *attributes and* b *addresses is a* $GHUBFS_2$, *if its pair* (β_1, β_2) *of addressing functions satisfies the following conditions* (i), (ii) *and* (iii):

(i) *The image* I_2 *of* β_2 *is* $\{1, 2,\dots, b\}$ *and* β_2 *is an addressing function of an* $HUBFS_2$,

(ii) β_1 *is a one-to-one function onto* $I_1=\{1, 2,\dots, l\}$, *and*

(iii) $\beta_1^{-1}(t)$ *is the root-point of a claw corresponding to* $\beta_2^{-1}(t)$ *for all* $t\in\{1, 2,\dots, l\}$.

We have shown in [17, Th. 6.2] that a $GHUBFS_2$ can be constructed if and only if l and b satisfy the conditions (i) and (ii) in Theorem 1 and $l>2c$ in addition, i.e.,

$$bc = \binom{l}{2} \quad \text{and} \quad l > 2c \qquad \text{for some integer } c. \tag{5.1}$$

By Definitions 5 and 7 and Theorem 2, we see immediately that an explicit expression of a pair of addressing functions of a GHBFS_2 is given in the following

THEOREM 3. *Let l, b and c be given positive integers satisfying* (5.1). *Then, in addition to the addressing function $\beta_2 = \beta$ of an* HUBFS_2 *given in* Theorem 2, *we can define a function β_1 from the set $\{Q_i | 1 \le i \le l\}$ of all queries of order one onto $\{1, 2, \ldots, l\}$ so that (β_1, β_2) is a pair of addressing functions of a* GHUBFS_2, *as follows*:

Case 1. $2c < l < 3c$.

$$\beta_1(Q_i) = \beta_1^{(1)}(Q_i) = \begin{cases} i & \text{if } 1 \le i \le c + r, \\ 2c + 2r - 1 & \text{if } i = c + r + 1, \\ i - 1 & \text{if } c + r + 2 \le i \le l, \end{cases}$$

where $l = 2c + r$.

Case 2. $3c \le l < 4c$.

$$\beta_1(Q_i) = \beta_1^{(2)}(Q_i) = \begin{cases} i & \text{if } 1 \le i \le 2c + r, \\ 4c + 2r - 1 & \text{if } i = 2c + r + 1, \\ i - 1 & \text{if } 2c + r + 2 \le i \le l, \end{cases}$$

where $l = 3c + r$.

Case 3. $4c \le l$.

$$\beta_1(Q_i) = \begin{cases} \beta_1^{(1)}(Q_i) & \text{if } l_0 \le 3c,\ 1 \le i \le l_0 - 1, \\ \beta_1^{(2)}(Q_i) & \text{if } l_0 \le 3c,\ 1 \le i \le l_0 - 1, \\ \beta_1^{(1)}(Q_{i'}) + (2c+1)p + \binom{l_0}{2}\Big/c & \text{if } l_p \le i \le l_{p+1} - 1, \end{cases}$$

where $l = 2nc + l_0$, $2c \le l_0 < 4c$, $l_p = l_0 + 2pc$, $i' = i - l_p + 1$ *for* $p = 1, 2, \ldots, n-1$.

6. Theoretical evaluation of balanced file organization schemes

Consider a GBFS_k with a k-tuple $(\beta_1, \beta_2, \ldots, \beta_k)$ of addressing functions. In order to store a record ω with h attributes $A_{i_1}, A_{i_2}, \ldots, A_{i_h}$ $(1 \le i_1 < i_2 < \cdots < i_h \le l)$ in the scheme, it is necessary to perform the following procedures.

(i) *Construct sets of v-tuples,*

$$S_v = \{(\alpha_1, \alpha_2,\dots, \alpha_v) | \alpha_1 < \alpha_2 < \cdots < \alpha_v \quad \textit{and}$$
$$\{\alpha_1, \alpha_2,\dots, \alpha_v\} \subset \{i_1,\dots, i_n\}\}, \qquad \textit{for all} \quad v = 1, 2,\dots, k.$$

(ii) *Construct sets of addresses*

$$D_v = \{B_v(Q_{\alpha_1\alpha_2\cdots\alpha_v}) | (\alpha_1, \alpha_2,\dots, \alpha_v) \in S_v\}, \qquad \textit{for all} \quad v = 1, 2,\dots, k.$$

(iii) *Set* $B_t = B_t \cup \{a_\omega\}$ *for all* $t \in \bigcup_{v=1}^{k} D_v$.

An accession number of a record, therefore, will be stored $|\bigcup_{v=1}^{k} D_v|$ times.

DEFINITION 8. *The redundancy* $R(\beta_1, \beta_2,\dots, \beta_k)$ *of a* $GBFS_k$ *is defined as*

$$R(\beta_1, \beta_2,\dots, \beta_k) = \sum_{t=1}^{b} |B_t|/|\boldsymbol{\Omega}|$$

where $|B_t|/|\boldsymbol{\Omega}|$ *is the relative frequency of an accession number of a record being stored in* B_t *and b is the total number of addresses.*

In order to discuss theoretically the redundancy of file organization schemes, a probability distribution of records plays an important role. A class of probability distributions of records has been presented in our previous paper [17]. This class includes the uniform distribution which has been used so far in the theoretical evaluation of the redundancies of file organization schemes.

Let $P(\cdot)$ be a probability distribution over $\{0, 1\}^l$ induced by the probability distribution of records over $\boldsymbol{\Omega}$ through X.

A probability distribution $P(\cdot)$ is said to be *permutation invariant* if it satisfies

$$P(\sigma\boldsymbol{\delta}) = P(\boldsymbol{\delta}),$$

for any $\boldsymbol{\delta} \in \{0, 1\}^l$ and any permutation σ of $\{1, 2,\dots, l\}$.

Then it has been shown in [17, Lemma 2.1] that $P(\boldsymbol{\delta})$ depends only on the weight $w(\boldsymbol{\delta}) = \sum_{i=1}^{l} \delta_i$ of $\boldsymbol{\delta} = (\delta_1\delta_2\cdots\delta_l)$, that is, the formula

$$P(\boldsymbol{\delta}) = p_{w(\delta)}$$

holds where p_w is a function of w.

The theoretical redundancies of the above-mentioned file organization schemes under the record distribution are as follows (cf. [17, (3.2), (3.5), (4.2), (6.2)]):

$$R(IFS_1) = l\sum_{w=0}^{l} \binom{l-1}{w-1} p_w,$$

$$\mathrm{R(IFS_2)} = \frac{l(l-1)}{2}\sum_{w=0}^{l}\binom{l-2}{w-2}p_w,$$

$\mathrm{R(BFS_2}$ *based on* $\mathrm{PG}(N, s))$

$$= \frac{(s^{N+1}-1)(s^N-1)}{(s^2-1)(s-1)}\sum_{w=0}^{l}\sum_{j=1}^{s-1}\left\{\binom{l-j}{w-1} - (s-1)\binom{l-s}{x-1}\right\}p_w,$$

$\mathrm{R(BFS_2}$ *based on* $\mathrm{EG}(N, s))$

$$= \frac{s^{N-1}(s^N-1)}{s-1}\sum_{w=0}^{l}\sum_{j=1}^{s-1}\left\{\binom{l-j}{w-1} - (s-1)\binom{l-s}{w-1}\right\}p_w,$$

$$\mathrm{R(HUBFS_2)} = \frac{l(l-1)}{2c}\sum_{w=0}^{l}\left\{\binom{l-1}{w-1} - \binom{l-c-1}{w-1}\right\}p_w,$$

$$\mathrm{R(GHUBFS_2)} = \sum_{w=0}^{l}\left\{\frac{l(l-1)}{2c}\binom{l-1}{w-1} - \left(\frac{l(l-1)}{2c} - l\right)\binom{l-c-1}{w-1}\right\}p_w,$$

where l is the number of attributes, c is the number of queries corresponding to a bucket and p_w is a probability function on $w \in \{1, 2, \ldots, l\}$. In $\mathrm{NBFS_2}$, it is difficult to compute the redundancy of the scheme, since the graphical structure of the buckets is not homogeneous all over the scheme.

Under such probability distribution of records, we have shown the following

THEOREM 4 (cf. [17, Th. 3.1, 6. 1]). $\mathrm{HUBFS_2}$ *has the least redundancy among the balanced file organization schemes of order two. Moreover,* $\mathrm{GHUBFS_2}$ *has the least redundancy among all the generalized balanced file organization schemes of order two provided that the addressing function* β_1 *of order one is one-to-one.*

7. Experimental evaluation of $\mathrm{GHUBFS_2}$

In the rest of this paper we shall describe results of the experimental evaluation of $\mathrm{GHUBFS_2}$ by using actual bibliographic data. The contents of this section are the detailed descriptions of the author's results announced in [12]. Though not only $\mathrm{GHUBFS_2}$ and $\mathrm{IFS_1}$, but also a file organization scheme having the consecutive retrieval property with redundancy defined by Ghosh [11] is treated in [12], the latter has been excluded here, because $\mathrm{GHUBFS_2}$ has to be evaluated in comparison with a standard scheme, namely, $\mathrm{IFS_1}$.

An on-line retrieval and batch storing system for the bibliographic data has been implemented in order to compare the scheme $\mathrm{GHUBFS_2}$ with $\mathrm{IFS_1}$. Especially, the execution time for retrieval of queries of order two as well as of order one and for storing the bibliographic data into the system have been compared.

The bibliographic data of the graph theory by Turner [14] are used. Each record of the data has the same format which is illustrated in Figure 7.1.

```
$000153$02005CHO75$04037PERMUTATIONS WITH RESTRICTE
D POSITON$06008MATRICES$06011ENUMEIRATION$11014MATH
. COMP. 16$17007222–226$190041962$37009F. HARARY$99

RECORD LAYOUT
01 ARTICLE
   02 BEGIN        3 bytes, alphanumeirc, value '$00'
   02 LENART       4 bytes, zoned decimal, format 9999
   02 SEGMENT      occurs a variable number of times
      03 TAG       3 bytes, alphnumeric
      03 LENSEG    3 bytes, zoned decimal, format 999
      03 RECSEG    alphanmeric, variable length given by LENSEG
   02 END          3 bytes, alphanumeric, value '$99'

TAG-LIST (Extracts)
  $00 THE BEGINNING OF ARTICLE
  $02 ARTICLE NUMBER
  $04 TITLE
  $06 ADDED KEYWORD
  $11 REFERENCE RECORD FOR ARTICLE
  $17 PAGES
  $19 YEAR
  $37 AUTHOR
  $99 THE END OF ARTICLE
```

Figure 7.1 Bibliographic data structure

In the system, eighty-one keywords are treated as attributes which characterize the records. These 81 keywords consist of 80 keywords given by Turner [14] and an additional keyword, since $l=81$ is convenient to construct a particular GHUBFS$_2$ of interest. The term 'LINEAR, is selected for an additional keyword by its frequency in use. These 81 keywords will serve well in the document retrieval on such a specific field as the theory of graphs.

A query of order one corresponds to a retrieval request such as "*Find every document including the word, say,* 'ALGORITHMS', *in the segments having the tags* $ 04 *and* $ 06.". If a retrieval request is given in the form "*Find every document related to the decomposition problem of a graph into complete subgraphs.*', then we can interpret, but not precisely, the request as a query Q_{ij} of order two, where A_i and A_j are associated with the word 'DECOMPOSITION' and the phrase 'COMPLETE SUBGRAPHS' respectively.

For a given word or phrase K with respect to a query, it is necessary to transform K to its identification number i, called the (*keyword*) *number* of K. In this system, we have organized a hashing table in the main memory for the key-to-number transformation. The determination procedure of the number of a keyword K by searching the hashing table is called *Algorithm* I.

In a universal document retrieval system, however, much more keywords must be handled and the sequential search by the alphabetical order of keywords as well as the direct search has to be supported. Thus the keywords would have to be organized by the balanced tree structure on a random access storage. In order to evaluate the organization schemes of secondary indexes precisely, the use of in-core hashing table might be desirable in an experimental system, since it reduces the effects of the key-to-number transformation.

To answer a query Q_i of order one (*first order search*) is to create a set F which consists of accession numbers of all records relevant to Q_i, i.e.,

$$F = \{a_\omega | X(\omega) = (\delta_1 \delta_2 \cdots \delta_{81}),\ \delta_i = 1\}.$$

To answer a query Q_{ij} of order two (*second order search*) is to create

$$F = \{a_\omega | X(\omega) = (\delta_1 \delta_2 \cdots \delta_{81}),\ \delta_i = 1 \text{ and } \delta_j = 1\}.$$

In order to store the actual record shown in Figure 7.1, the *text file* is organized. The file is implemented as a DAM (Direct Access Method) file on a random access storage. Each fixed-length (440 bytes) block of the file is used, and a record ω of actual bibliographic data is stored in it. A relative block number m is used as an accession number (4 bytes) of a record being stored in it, that is $a_\omega = m$. If the length of a record is greater than the size of a block, the overflowing part of the record will be stored in an empty block and will be connected by a pointer as shown in Figure 7.2.

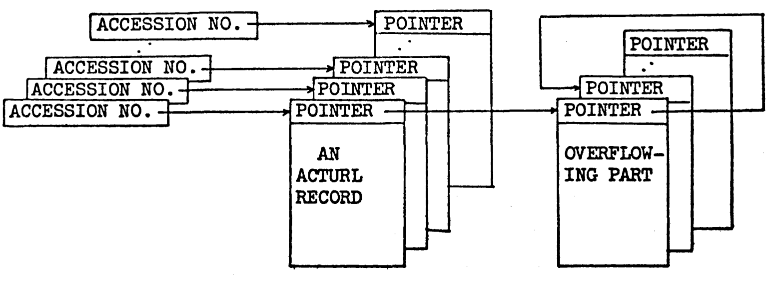

Figure 7.2 Structure of the text file

The handling overflow when storing records on the text file is managed automatically by the system.

Implementation of IFS_1. The scheme IFS_1 presented in Section 2 has been implemented as follows. The 81 buckets of the scheme are organized as a DAM file on a random access storage. The tth bucket B_t corresponds to the fixed-length (440 bytes) block with its relative block number t. More than 81 blocks are prepared and every block with its relative block number exceeding 81 is used for the overflow area. A block can contain 108 accession numbers. If the size of a block is not large enough for storing the number of accession numbers of the corresponding bucket, handling overflow is also managed automatically by the system.

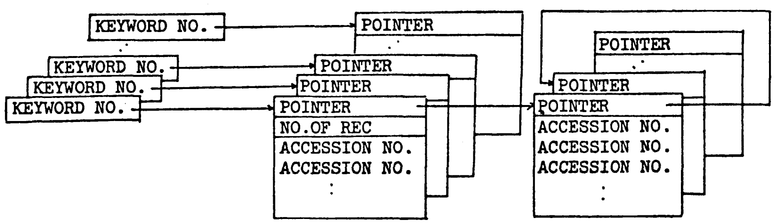

Figure 7.3 Structure of the secondary indexes of IFS_1

In order to store the records, the system performs the following procedures.

IS–1. *Set* $m=0$.

IS–2. *If* $\boldsymbol{\Omega}$ *is not empty, get a record* ω *in* $\boldsymbol{\Omega}$ *or else halt.*

IS–3. *Set* $m=m+1$ *and store* ω *into mth block of the text file.*

IS–4. *Construct a set*
$S_1^{(\omega)}=\{i \mid \textit{the ith keyword } A_i \textit{ relevant to } \omega \textit{ for } i=1,2,\ldots,81\}$.
If $S_1^{(\omega)}=\phi$, *then go to* IS–2.

IS–5. *Set* $B_i=B_i \cup \{m\}$ *for all* $i \in S_1^{(\omega)}$ *and set* $\boldsymbol{\Omega}=\boldsymbol{\Omega}-\{\omega\}$.
Go to IS–2.

In order to perform the procedure IS–4, the system executes the extraction of words from the segments having tags \$ 04 and \$ 06, and the matching of each extracted word to the registered keywords. For IS–5, the system uses the ith block to store the accession number m unless the ith block is full of accession numbers. The system further uses an overflow area if and only if the block is full of accession numbers.

The algorithm for first-order searches is as follows.

IR1–1. *Get a keyword* K *from the terminal.*

IR1–2. *Determine the keyword number* i *of* K *by Algorithm* I.

IR1–3. *Create a set* F *by the transmission of all accession numbers in* B_i.

The algorithm for second-order searches is as follows.

IR2–1. *Get two keywords* K *and* K′ *from the terminal.*

IR2–2. *Determine the keyword numbers* i *and* i' *of* K and K′, *respectively, by Algorithm* I.

IR2–3. *Create a set* F *by the operation*

$$\mathrm{F} = \mathrm{B}_i \cap \mathrm{B}_{i'}.$$

Implementation of GHUBFS$_2$. A GHUBFS$_2$ presented in Section 5 with parameters $l=81$, $c=40$, $b=l(l-1)/(2c)=81$ has been implemented as follows. In such a special case, addressing functions β_1 and β_2 which are simpler than those given in Section 4 can be obtained. The functions used here are as follows:

$$\beta_1(\mathrm{Q}_i) = i$$

$$\beta_2(\mathrm{Q}_{ij}) = \begin{cases} i & \text{if} \quad i < 40 \text{ and } j \leq 40 \quad \text{or} \quad i \leq 40 \text{ and } j > 81 - i, \\ j & \text{otherwise.} \end{cases}$$

Note that the function β_2 partitions the set of all queries of order two into 81 subsets and that the graphical structure of each subset is a claw with 40 edges. The address of each subset is illustrated in Figure 7.4.

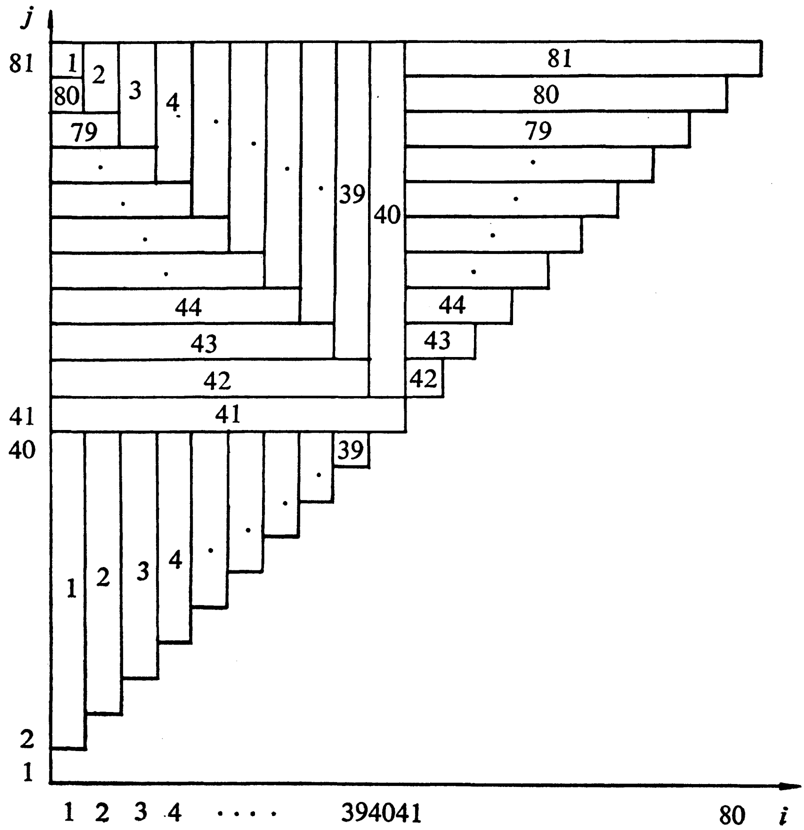

Figure 7.4 Partition of all second order queries into claw-type subsets

In our system, a bucket consists of two types of subbuckets illustrated in Figure 7.5. One is a *claw-type subbucket* $B_t^{(C)}$ which includes not only accession numbers but also keyword numbers explained in the storing algorithm. Another is an *inverted-type subbucket* $B_t^{(I)}$ in which only accession numbers are stored. The size of each subbucket is 220 bytes. The size of a bucket $B_t^{(C)} \cup B_t^{(I)}$ is therefore equal to that of IFS_1.

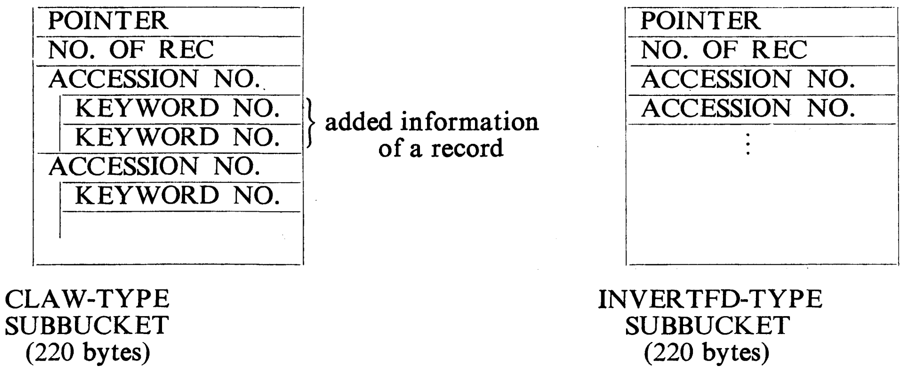

Figure 7.5 Structure of buckets of $GHUBFS_2$

In order to store the records, the system performs the following procedures.

HS–1. *Set* $m=0$.

HS–2. *If* $\boldsymbol{\Omega}$ *is not empty, get a record* ω *in* $\boldsymbol{\Omega}$, *or else halt.*

HS–3. *Set* $m=m+1$ *and store* ω *into mth block of the text file.*

HS–4. *Construct two sets*

$$S_1^{(\omega)} = \{i|X(\omega) = (\delta_1\delta_2\cdots\delta_{81}),\ \delta_i = 1\},$$

$$S_1^{(\omega)} = \{(i, j)|i < j \text{ and } i, j \in S_1^{(\omega)}\}.$$

HS–5. *Construct two sets*

$$D_1^{(\omega)} = \{\beta_1(Q_i)|i \in S_1^{(\omega)}\},$$

$$D_2^{(\omega)} = \{\beta_2(Q_{ij})|(i, j) \in S_2^{(\omega)}\}.$$

HS–6. *Construct*

$$J_t^{(\omega)} = \{i, j|(i, j) \in S_2^{(\omega)},\ \beta_2(Q_{ij}) = t\} - \{t\}$$

and then set

$$B_t^{(C)} = B_t^{(C)} \cup \{(m, L(J_t^{(\omega)}))\} \qquad \textit{for all} \quad t \in D_2^{(\omega)},$$

where $(m, L(J_t^{(\omega)}))$ *denotes a list consisting* m *and a list* $L(J_t^{(\omega)})$ *of all* *elements in* $J_t^{(\omega)}$. *Further set*

$$B_t^{(I)} = B_t^{(I)} \cup \{m\} \qquad \textit{for all} \quad t \in D_1^{(\omega)} - D_2^{(\omega)},$$

The algorithm for first-order searches is as follows.

HR1–1. *Get a keyword* K *from the terminal.*

HR1–2. *Determine the keyword number* i *of* K *by Algorithm* I.

HR1–3. *Create a set* F *by the operation*

$$F = \{m|(m, L(J)) \in B_{\beta(Q_i)}^{(C)}\} \cup B_{\beta(Q_i)}^{(I)}.$$

The algorithm for second-order searches is as follows.

HR2–1. *Get two keywords* K *and* K′ *from the terminal.*

HR2–2. *Determine the keyword number* i *and* i' *of* K *and* K′, *respectively, by Algorithm* I.

HR2–3. *Create a set* F *by the operation*

$$F = \{m|(m, L(J)) \in B_{\beta(Q_{ii'})}^{(C)}, (\{i, i'\} - \{\beta_2(Q_{ii'})\}) \cap J \neq \phi\}.$$

Results of the experimentation and discussions. Table 7.1 shows the frequency distribution of the number of keywords in a record of our bibliographic data $\boldsymbol{\Omega}$.

Table 7.1 Frequency distribution of records

No. of keywords	0	1	2	3	4	5	6	7
Frequency	165	711	560	213	70	13	6	0

The redundancy, or the average number of times the accession number a_ω *of a record* ω *having been stored in the buckets, of* $GHUBFS_2$ *as well as* IFS_1 *is* 1.64.

The system operates within the framework of OS7 operating system for HITAC 8700 computer at Hiroshima University Computing Center. OS7 supports an on-line processor which permits the conversational interaction with a user's program. Our experimental on-line system works within this framework.

We have performed the evaluation of IFS_1 and $GHUBFS_2$ by using not only the original 1,738 records but also two, three, four, five and twenty-five times as many records. The number of records, the size of the text file, the size of the secondary indexes and the time needed to store and shown in Table 7.2. *Some differences among two schemes in CPU time and elapse time used for storing in the buckets can be seen there. These results may be explained by the difference of addressing algorithms and the structure of buckets. There are,* *however, relatively small differences in CPU time and elapse time during the storing phase.*

Table 7.2

Performance statistics for storing*

No. of records		1738	3476	5214	6952	8690	43450
Size of text file		319	638	957	1276	1595	7975
Size of secondary indexes	IFS_1	11	23	34	45	57	272
	$GHUBFS_2$	25	49	74	99	123	624
CTIME1	IFS_1	1.5	2.9	4.4	5.9	7.5	38.5
	$GHUBFS_2$	3.7	7.9	10.5	14.0	17.4	87.2
ETIME1	IFS_1	5.6	7.3	8.8	10.5	12.2	47.0
	$GHUBFS_2$	11.8	16.3	20.8	25.2	29.9	122.2
CTIME2	IFS_1	145.0	285.4	426.0	566.6	707.2	3519.4
	$GHUBFS_2$	147.1	289.6	432.2	574.8	717.4	3570.1
ETIME2	IFS_1	234.6	391.4	548.6	705.7	863.2	4007.4
	$GHUBFS_2$	242.3	403.5	565.3	727.2	888.8	4123.8

*) All times are expressed in second and file sizes in KB.
CTIME1 = CPU time for storing in the buckets.
ETIME1 = Time elapsed for storing in the buckets.
CTIME2 = CPU time during the storing phase.
ETIME2 = Time elapsed during the storing phase.

Figure 7.6 illustrates the performance characteristics of the first-order searches by the growth of data. IFS_1 *is, of course, faster than* $GHUBFS_2$ *and the difference may be explained mainly by the effect of the set operation* HR1–3. If each bucket of $GHUBFS_2$ were not partitioned into two types of subbuckets and were organized on a block as a merged bucket $B_t^{(C)} \cup B_t^{(I)}$, then it might be expected that the first-order search in $GHUBFS_2$ would become more faster.

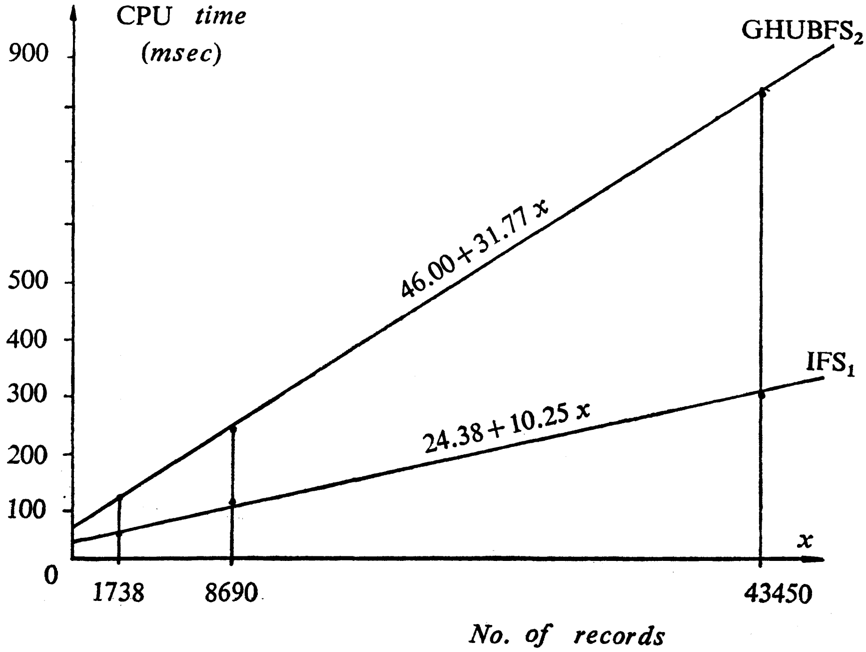

Figure 7.6 Average CPU time for search of order 1

The performance characteristics of the second-order searches illustrated in Figure 7.7 show that GHUBFS$_2$ *is much faster than* IFS$_1$. *This indicates that the selection procedure* HR2–3 *needed for search in the claw-type subbucket of* GHUBFS$_2$ *is not so serious when compared with the set-operation* IR2–3 *needed in* IFS$_1$.

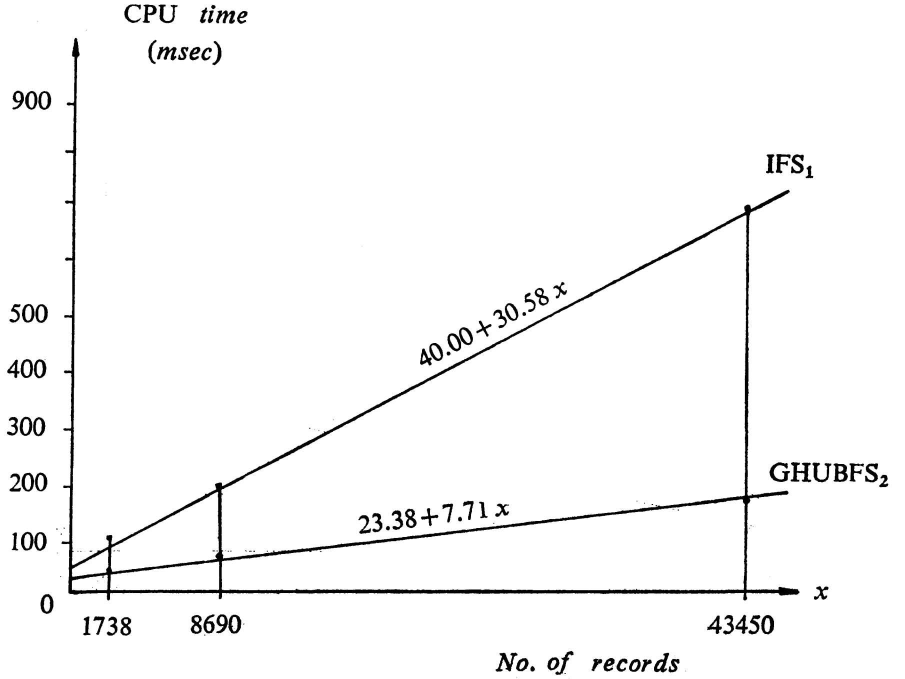

Figure 7.7 Average CPU time for search of order 2

Our experimental study shows that the performance of an information storage and retrieval system depends greatly on the selection of a file organization scheme. The results, however, depend deeply on the structure of data, the number of data, the number of attributes, parameters of the scheme, the size of a bucket of the secondary index, the access method to the file, etc..

The performance of GHUBFS$_2$ may depend on its parameter c, the number of second-order queries corresponding to a bucket. In our implemented system, $c=(l-1)/2$ is selected in order to reduce the redundancy to the least.

Acknowlegments: I would like to express my hearty thanks to Professor S. Yamamoto for his valuable guidance all over this work. I also wish to thank to Miss K. Futagami for her contribution on the detailed programming of the experimental system.

I am grateful to Professor M. Sugawara for his encouragement during the preparation of this paper.

This study is supported in part by the Grant of Ministry of Education, Science and Culture of Japan.

References

[1] C. T. Abraham, S. P. Ghosh and D. K. Ray-Chaudhuri, File organization schemes based on finite geometries, *Information and Control* **12** (1968), 143–163.

[2] G. Berman, The application of difference sets to the design of a balanced multiple-valued filing scheme, *Information and Control* **32** (1976), 128–138.

[3] R. C. Bose, C. T. Abraham and S. P. Ghosh, File organization of records with multiple-valued attributes for multi-attributes queries, *Combinatorial Mathematics and Its* Applications, UNC Press, 1969, 277–297.

[4] R. C. Bose and G. G. Koch, The design of combinatorial information retrieval systems for files with multiple-valued attributes, *SIAM J. Appl. Math.* **17** (1969), 1203–1214.

[5] A. F. Cardenas, Evaluation and selection of file organization — A model and system, *Comm. ACM* **16** (1973), 540–548.

[6] D. K. Chow, New balanced-file organization schemes, *Information and Control* **15** (1969), 377–396.

[7] S. P. Ghosh, Organization of records with unequal multiple-valued attributes and combinatorial queries of order 2, *Information Sci.* **1** (1969), 363–380.

[8] S. P. Ghosh, File organization: The consecutive retrieval property, *Comm. ACM* **15** (1972), 802–808.

[9] S. P. Ghosh, On the theory of consecutive storage of relevant records, *Information Sci.* **6** (1973), 1–9.

[10] S. P. Ghosh, File organization: Consecutive storage of relevant records on drum-type storage, *Information and Control* **25** (1974), 145–165.

[11] S. P. Ghosh, The consecutive storage of relevant records with redundancy, *Comm. ACM* **18** (1975), 464–471.

[12] H. Ikeda, Evaluation of combinatorial file organization schemes, *Proc. of* 3*rd VLDB Conference* (1977), 231–235.

[13] D. K. Ray-Chaudhuri, Combinatorial information retrieval systems for files, *SIAM J. Appl. Math.* **16** (1968), 973–992.

[14] J. Turner, Key-word indexed bibliography of graph theory, *Proof Techniques in Graph Theory* (ed. F. Harary), Academic Press, 1969, 189–330.

[15] A. Waksman and M. W. Green, On the consecutive retrieval property in file organization, *IEEE Trans. Computers* **C-23** (1974), 173–174.

[16] S. Yamamoto, H. Ikeda, S. Shige-eda, K. Ushio and N. Hamada, On claw-decomposition of complete graphs and complete bigraphs, *Hiroshima Math. J.* **5** (1975), 33–42.

[17] S. Yamamoto, H. Ikeda, S. Shige-eda, K. Ushio and N. Hamada, Design of a new balanced file organization scheme with the least redundancy, *Information and Control* **28** (1975), 156–175.

[18] S. Yamamoto, S. Tazawa, K. Ushio and H. Ikeda, Design of a balanced multiple-valued file organization scheme with the least redundancy, presented at the *Third International Conference on Very Large Data Bases,* Oct. 1977, and to be published in *ACM Transactions on Database Systems.*

[19] S. Yamamoto, S. Tazawa, K. Ushio and H. Ikeda, Design of a generalized balanced multiple-valued file organization scheme of order two, to appear in *Proc. ACM-SIGMOD* 1978.

[20] S. Yamamoto, T. Teramoto and K. Futagami, Design of a balanced multiple-valued filing scheme of order two based on cyclically generated spread in finite projective geometry,

Information and Control **21** (1972), 72–91.

[21] S. Yamamoto, K. Ushio, H. Ikeda, S. Tazawa, F. Tamari and N. Hamada, Partition of a query set into minimal number of subests having consecutive retrieval property, *J. Statist. Planning Inf.* **1** (1977), 41–51.

Computing Center,
Hiroshima University