255
C.1

THRESHOLD SEQUENCING ALGORITHMS

BY

MOHAMAD A. NAAMANI

A Thesis Submitted in Partial Fulfillment of the Requirement for the Degree of Master of Science in Computer Science at the Lebanese American University

Mohamad Naamani

[redacted]

Lebanese American University

Graduate Studies

We hereby approve the thesis of Mr. Mohamad Naamani, candidate for the Master degree of Computer Science.*

Dr. Issam Moghrabi [redacted]

Dr. Adnan Hamzeh [redacted]

Dr. Abdul Nasser Kassar [redacted]

* We also certify that written approval has been obtained for any proprietary material contained therein.

TABLE OF CONTENTS:

LIST OF TABLES iv

ACKNOWLEDGMENTS v

ABSTRACT vi

1. INTRODUCTION 1

2. INFORMATION RETRIEVAL SYSTEM 4

2.1 THE DOCUMENT SELECTION SUBSYSTEM 4

2.2 THE INDEXING SUBSYSTEM 5

2.3 THE VOCABULARY SUBSYSTEM 5

2.4 THE SEARCING SUBSYSTEM 6

2.5 THE USER INTERFACE SUBSYSTEM 7

2.6 THE MATCHING SUBSYSTEM 7

3. DOCUMENT CLUSTERING 8

3.1 HIERARCHICAL CLUSTER GENERATION 9

3.2 HEURISTIC CLUSTER GENERATION 11

4. DOCUMENT SEQUENCING 13

4.1 CLUSTERING APPROACH 13

4.2 HAMMING DISTANCE 13

4.3 SEQUENCING ALGORITHM 14

5. COMPARING THE CLASSIFICATION ALGORITHMS 16

5.1 DOCUMENT INSERTION 16

5.2 DOCUMENT DELETION 17

6. EXPERIMENTAL WORK 19

6.1 NEW SEQUENCING METHODS 19

6.2 QUERY PROCESSING 20

6.3 KEYTERM WEIGHTING 20

6.4 HAMMING DISTANCE OF THE UNSEQUENCED FILE 22

6.5 CHOOSING THRESHOLDS FOR SEQUENCING 2 22

6.6 HAMMING DISTANCE OF THE SEQUENCED FILE 23

6.7 RELATIONSHIP BETWEEN HD OF UNSEQ. AND SEQ. FILES 23

7. PROGRAM DESIGN 24

7.1 PROGRAMS WRITTEN 24

7.2 PARAMETERS ADJUSTABLE DURING SIMULATION 24

7.3 RECORDS GENERATION 25

7.4 FILE SEQUENCING 25

7.5 QUERY GENERATION 25

7.6 QUERY PROCESSING 26

7.7 IMPLEMENTATION 26

8. EMPIRICAL RESULTS 28

8.1 HAMMING DISTANCE OF THE UNSEQUENCED FILE 28

8.2 CHOOSING THRESHOLD 5 FOR SEQUENCE 2 30

8.3 EFFICIENCY OF THE SEQUENCES 32

8.4 PERFORMANCE OF THE SEQUENCES DURING QUERY PROCESSING 32

8.4.1 PERCENTAGE REDUCTION IN BLOCK ACCESS 32

8.4.2 TREND OF PERCENTAGE REDUCTION 33

8.5 EVALUATING THE KEY TERM WEIGHTING SCHEME 36

9. CONCLUSIONS AND FURTHER RESEARCH 40

9.1 CONCLUSIONS 40

9.2 FURTHER RESEARCH 40

REFERENCES 42

LIST OF TABLES:

TABLE 8.1 POSSIBLE HD BETWEEN ADJACENT RECORDS	**31**
TABLE 8.2 MAXIMUM THRESHOLDS FOR DIFFERENT WEIGHTS	**31**
TABLE 8.3 PERCENTAGE OF REDUCTION IN BLOCK ACCESSES	**35**
TABLE 8.4 AVG. NUMBER OF BLOCK ACCESSES WITH WEIGHT OF 3	**37**
TABLE 8.5 AVG. NUMBER OF BLOCK ACCESSES WITHWEIGHT OF 5	**37**

ACKNOWLEDGMENTS:

I would like to thank Dr. Issam Moghrabi for his guidance and support during the development of this thesis, and through out the years of my M.S. and B.S. degrees.

I would also like to thank Dr. Ahmad Kabbani for his help during the years I spent at LAU.

ABSTRACT:

Response time in an information system can be improved by reducing the number of blocks accessed when retrieving a document set. One approach is to restructure the document base in such a way that similar documents are placed close together in the file space. This ensures a greater probability that documents will be co-located within the same block.

This thesis is based on a file sequencing algorithm proposed by Lowden [Lowden 1985) and attempts to modify and improve the algorithm proposed. A series of three sequencing algorithms are considered. The effect of associating weights to key terms during document sequencing is the major area of interest. Simulation with computer programs and statistical theories are used to analyze the performance of these algorithms.

CHAPTER 1: INTRODUCTION

Nowadays more and more information is generated and put into circulation. Places like libraries and offices require computerized systems to handle all the information for them and make query processing as efficient as possible. Therefore for the last three decades Information Retrieval has been an active area of research interest. New research work is still emerging quickly.

Generally speaking information retrieval is a discipline involved with the organization, structuring, analysis, storage, searching and dissemination of information. Information retrieval Systems are designed to make available a given stored collection of information items with the objective of providing, in response to a user query, references that would contain the information desired by the user. In other words, the system is intended to identify which documents the user should read in order to satisfy his/her information requirements.

A document may or may not be relevant to a user query depending on many factors concerning the document (e.g. its scope, how it is written) as well as numerous user characteristics (e.g. why the search is initiated, user's previous knowledge). In any case, whatever the information retrieval system does, if a document is judged by the user to be of interest, it is relevant; otherwise, it is non-relevant. Since many factors may influence the judgment concerning relevance in a complex way, satisfying user requirements is therefore not a simple task.

It is common, in information retrieval, to suppose that each document is indexed by a set of 'content' identifiers that are variously known as key terms, index terms or subject indicators. This would require the application of some automatic or manual indexing technique to the full text or some surrogate of the documents in order to identify the key terms to be used in their representation. In addition to the selection of index terms to represent documents, it is also common to associate weights that reflect the importance of each key term [Wong et al 1986).

In the context of information retrieval, classification is required to group the documents in such a way that retrieval will be faster. It would appear, therefore, that documents are grouped because they are in some sense related to each other; but more basically, they are grouped because they are likely to be *wanted* together [van Rijsbergen 1979).

With all these properties necessarily to be catered by information retrieval systems, it is easy to see that designing a competitive system is very challenging.

A detailed account of the structure and functions performed by a conventional information retrieval system will be given in Chapter 2.

In 1985, Lowden [Lowden 1985] proposed a file sequencing algorithm to reduce the access time required for information retrieval. His work was extended by Thompson [Thompson 1989] who introduced a similar algorithm that sequenced a file in a significantly reduced amount of time. His work included a large number of simulations to determine the effectiveness and duration of the algorithm. The conclusions obtained, while clearly indicating a reduced sequencing time, contained a number of unexpected and unexplained results.

The purpose of this report is fourfold:

1. To investigate various document classification algorithms and compare their performance during document retrieval.
2. To compare Lowden's, Thompson's and one other similar sequencing algorithm to determine the effectiveness and duration of each.
3. To investigate the effect of 'weighted-key terms' on the file sequencing efficiency of the above algorithms.
4. To analyze statistically the file sequencing algorithms with a view to explaining the results obtained.

CHAPTER 2: INFORMATION RETRIEVAL SYSTEN

Information retrieval deals with the representation, storage, and access to documents or representatives of documents. The input information is likely to include the natural language text of the documents or document excerpts and abstracts. The output of an information retrieval system in response to a search request consists of sets of references. These references are intended to provide the system users with information about items of potential interest [Salton 1983).

An information retrieval system may be considered to comprise of six major subsystems:

- the document selection subsystem
- the indexing subsystem
- the vocabulary subsystem
- the searching subsystem
- the user interface subsystem
- the matching subsystem [Lancaster 1979)

2.1 THE DOCUMENT SELECTION SUBSYSTEM

Input documents are selected according to some specified criteria. It is usually performed manually since selectors such as librarians may have direct judgment on the content of the documents. They are then organized and controlled such that they can be identified and located in response to various user demands. Organization and control activities include classification, cataloging, indexing and abstracting; these functions are performed by the following subsystems.

2.2 THE INDEXING SUBSYSTEM

The process of constructing document surrogates by assigning identifiers to text items is known as indexing. In the past, indexing operations were normally performed intellectually by subject experts, or by trained persons with experience in assigning content descriptions. More recently, the original text of information items *has* frequently been used as a basis for indexing, and text analysis is now often controlled by automatic, machine-performed procedures [Salton 19891.

The indexing process involves conceptual analysis of a document and translation of the concept to a particular vocabulary. Such a vocabulary might be a list of subject heading, a classification scheme, a thesaurus, or simply a list of key terms or phrases [Lancaster 1979].

2.3 THE VOCABULARY SUBSYSTEM

Storage space is always an expensive resource in information systems. Storing all key terms or phrases required for representing the documents is therefore impractical.

It is useful first to remove word suffixes, thereby reducing the original words to word stem form. A variety of different forms of the same word, for example analysis, analyzing, analyzer, analyzed, and analyzing can be reduced to a common word stem "analy" [Salton 1983].

The vocabulary subsystem provides a wide range of support to the translation processes in the indexing and searching subsystems.

2.4 THE SEARCHING SUBSYSTEM

All search strategies are based on comparison between the query and the stored documents. The search strategies to be employed in an information retrieval system depend mainly on the file structure of the system. Different file structures would have different strategies and appropriateness for searching. Sometimes comparisons may only be achieved by comparing the query with cluster profiles (cluster representatives) indirectly [van Rijsbergen 1979).

The following Retrieval Models are mostly employed in searching process. They define the methods for comparing the similarity between a document and a query.

(1) The Vector Space Model

- which represents both queries and documents by term sets and computes global similarities between them.

(2) The Probabilistic Retrieval Model

- which is based on the idea that since relevance of a document with respect to a query is a matter of degree, then when the document and query vectors are sufficiently similar, the corresponding probability of relevance is large enough to make it reasonable to retrieve the document in answer to the query.

(3) The Boolean Family of Retrieval Models

- which compare Boolean query statements (i.e. the ones with AND, OR and NOT operators) with the term sets used to identify document content. (Salton 1989)

2.5 THE USER INTERFACE SUBSYSTEM

The user interface subsystem is the direct part of an information retrieval system to interact with its users. A user-friendly interface is therefore highly important for a successful system.

The use of query languages that have structural query syntax discourages many users as they are inflexible to use, and require the users to learn the correct syntax. Therefore current research interest is on the development of natural language interfaces.

2.6 THE MATCHING SUBSYSTEM

Matching query with existing documents plays an important role in query processing. Different file organizations have been used to aid the matching process and the most common ones are the Inverted List and the Multi-list file organizations [Ashford 1989].

CHAPTER 3: DOCUMENT CLUSTERING

One disadvantage of conventional information retrieval methodology, using inverted index files, is that the information pertaining to a document is scattered among many different inverted-term lists. Furthermore, information relating to different documents with similar term assignments may not be in close proximity within the file system. Hence, during retrieval, documents satisfying queries would not appear close together. This problem may be overcome by the use of *clustering* operations that group into common areas items judged to be similar [Salton 1989).

In general, a clustered file provides efficient file access by limiting the searches to those document clusters which appear to be the most similar to the corresponding queries. At the same time, clustered file searches may also be effective in retrieving the wanted items whenever the so-called *Cluster Hypothesis* applies which states that:

- closely associated documents tend to be relevant to the same request [van Rijsbergen 1979]

According to the cluster hypothesis, it is obvious that storing similar documents together would reduce the time required for retrieval.

Various classification methods have been proposed by various researchers. However, for a good classification method, a few criteria should be emphasized:

- the method produces a clustering which is unlikely to be altered drastically when further objects are incorporated

- the method is stable in the sense that small errors in the description of the objects lead to small changes in the clustering
- the method is independent of the initial ordering of the objects [van Rijsbergen 1979]

Two approaches of cluster-generation will be introduced in the following sections. A file sequencing algorithm which serves the same purpose will be introduced in next chapter Their pros and cons will be discussed in Chapter 5.

3.1 HIERARCHICAL CLUSTER GENERATION

Two main strategies can be used to carry out the cluster generation. Firstly a complete list of all pair wise document similarities can be constructed, in which case it is necessary to employ a grouping mechanism capable of assembling into common clusters documents with sufficiently large pair wise similarities. Alternatively, *heuristic methods* can be used that do not require

Pair wise document similarities to be computed [Salton 1989]. The heuristic methods will be discussed in next section.

When clustering does depend on pair wise document similarities, a document similarity matrix is conveniently used as a starting point, followed by a comparison of all distinct pairs of matrix rows to be used for document clustering. Suppose there are N documents in total, the pair wise comparison of matrix rows produces N(N-1)/2 different pair wise similarity coefficients for the documents. No matter what specific clustering method is used, the clustering process can be carried out either *diversively* or *agglomeratively* [Jain 1958]. In the former case, the complete collection is assumed to represent one complete cluster that is subsequently broken down into smaller pieces. In the latter, individual document similarities are used as a starting point, and a gluing operation collects similar documents into larger groups [Murray 1972). In both cases, a document hierarchy called a dendrogram would be produced [Jain 1988). Here the agglomerative process, which is more commonly employed in information retrieval systems, will be considered.

A simple program outline for a hierarchical, agglomerative clustering process is given below:

1. Compute all pair wise document-document similarity coefficients.
2. Place each of the N documents into a class of its own.

3. Form a new cluster by combining the *most similar pair* of current clusters i and j; update the similarity matrix by deleting the rows and columns corresponding to I and j and then calculate the entries in the row corresponding to the new cluster i + j.
4. Repeat step 3 if the number of clusters remaining is greater than 1.

Each item is thus placed in its own class and then the two most similar items are combined into a single class; the combining process is continued until only a single class of objects remains. When two single documents are combined into a class, the clustering criterion is simply the size of the similarity coefficient between them. When larger clusters must be combined, a criterion of closeness between clusters must be specified. Several possibilities present themselves:

•In *Single-Link* clustering, the similarity between a pair of clusters is taken to be the similarity between the *most similar* pair of documents, one of which appears in each cluster. Thus each cluster member will be more similar to at least one member in that same cluster than to any member of another cluster.

•In *Complete-Link* clustering, the similarity between the *least similar* pair of documents from the two clusters is used as the cluster similarity. Thus each cluster member is more similar to the most dissimilar member of that cluster than to the most dissimilar member of any other cluster.

• *Group-average* clustering is a compromise between the extremes of the single-link and complete-link Systems, in that each cluster member has a greater average similarity to the remaining members of that cluster than it does to all members of any other cluster. [Murray 1972)

It can be seen that a single-link system produces a small number of large, poorly linked clusters, whereas the complete-link process produces a much larger number of small, tightly linked groupings. Because each document in a complete-link cluster is guaranteed to resemble all other documents in that cluster (at the stated similarity level), the complete-link clustering system may be better adapted to retrieval than the single-link clusters, where similarities between documents may be very low. Unfortunately, a complete-link cluster generation is generally more expensive to perform than a single-link process.

3.2 HEURISTIC CLUSTER GENERATION

The hierarchical clustering strategies just described are based on prior knowledge of all pair wise similarities between documents. Therefore the corresponding cluster-generation methods are relatively expensive to perform. In return, these methods produce a unique set of well-formed clusters for each set of data, regardless of the order in which the similarity pairs are introduced into the clustering process. Furthermore, the resulting cluster hierarchy is stable in that small changes in input data do not lead to large rearrangements in the cluster structure.

These desirable clustering properties are lost when heuristic clustering methods are used;

heuristic methods produce rough cluster arrangements rapidly at relatively little expense. The simplest heuristic process, a *one-pass* procedure, takes the elements to be clustered one at a time in arbitrary order, requiring no advance knowledge of document similarities. The first document is initially taken and placed into a cluster of its own. Each subsequent document is then compared against all existing clusters, and is placed in a previously existing cluster whenever it is similar enough to that cluster. If a new document is not sufficiently similar to any existing cluster, the new document forms a cluster of its own. This process is continued until the last element is processed.

A somewhat different heuristic clustering method is based on identifying individual documents that lie in dense regions of the document space, that is, documents surrounded by many other documents in close proximity. When such a document is located, it is used as a cluster seed, and all sufficiently similar documents in the same general vicinity then form a common class of documents. More specifically, a *density test* is performed sequentially for all documents not yet clustered. Whenever a document passes the density test by exhibiting a sufficient number of close neighbors, a clustering operation forms a new cluster around the document. The density-test process is then resumed with the next document not yet clustered. At the end of the process, some documents remain unclustered either because they failed the density test, or because they were too far removed from cluster seeds to fit into any existing cluster. Such 'loose' items may either be left unclustered, or assembled into cluster structures of their own [Salton 1989].

Although the desirable properties of the hierarchical clustering strategies are not retained by the use of the methods described here, the time saved during clustering makes it advantages to perform clustering heuristically.

CHAPTER 4: DOCUMENT SEQUENCING

In 1985, Lowden proposed an algorithm for sequencing a multi-attribute file, which can be used in a situation where the query pattern is unknown and the term content equiprobable [Lowden 1985].

Documents in a file are represented by records with multiple attributes. The algorithm physically locates similar multi-attribute records close to each other and this increases the probability that two or more records satisfying a general query will reside in the same block of secondary storage, thereby reducing the total number of blocks accessed.

4.1 CLUSTERING APPROACH

For a multi-attribute file of N records <R_1, R_2, ..., R_N > each comprising of m attributes <a_1, a_2, ..., $a_{m>}$ >, it is sorted provided that the difference between consecutive records is a minimum. i.e. for the set of N records, they are sorted if

$$\sum_{i=1}^{N-1} d(\mathbf{R}_i, \mathbf{R}_{i+1})$$

is a minimum for all possible orderings of R_i 's, where d(R_i, R_{i+1}) is defined to be the difference between records R_i and R_{i+1}.

4.2 HAMMING DISTANCE

The difference between records can be measured by the following method. The difference calculated would be called the *Hamming Distance.*

Given records

$R_p=<a_{p1}, a_{p2}, \dots, a_{pm}>$ and

$R_q=<a_{q1}, a_{q2}, \dots, a_{qm}>$

each containing m distinct attributes, the Hamming Distance between them may be defined as

$$d(R_p, R_q) = \sum_{i=1}^{m} f(a_{pi}, a_{qi})$$

$$\text{where } f(a_{pi}, a_{qi}) = \begin{cases} 1 \text{ if } a_{pi} \neq a_{qi} & \forall i \quad \text{where } 1 \le i \le m \\ 0 \text{ otherwise} \end{cases}$$

4.3 SEOUENCING ALGORITHM

For a file with N records, a square matrix S of size N is constructed with S_{ij} as the Hamming Distance between R_i and R_j. The minimal spanning path is the sequence of records for which the sum of S_{ij}'s is a minimum.

The problem of finding such a minimal spanning path is exactly equivalent to the Traveling Salesman problem which is NP hard. Because an exact computation of such a path would be too expensive, an algorithm has been proposed that approximates the minimal spanning path. A near-optimal path would thus be constructed. The algorithm is listed below.

Let the set of N records be represented as nodes of a graph (N,d) in which the values for the edges between nodes are given by the difference matrix S. An ordered set of nodes which is a subset of N gives a path T which is called a sub path of the graph (N,d).

Thus given a graph (N,d), a sub path T and a node m in N which is not in the set of nodes forming the path T, then the path (T,m) can be constructed as follows.

If T passes through only a single node i then construct path (T,m) as the two node path consisting of edge (i,m) or (m,i).

If T passes through more than one node then:

(1) find an edge (x,y) between the nodes of T which minimizes

$$d(x,m) + d(m,y) - d(x,y)$$

(2) replace edge (x,y) with edges (x,m) and (m,y).

Sequencing a file of N records with this algorithm would then require $O(N^2)$ comparisons.

CHAPTER 5: COMPARING THE CLASSIFICATION ALGORTHNS

Some hierarchical and heuristic cluster-generation algorithms and a file sequencing algorithm have been presented in the previous two chapters. They both aim at reducing the time needed during retrieval. Although they share this common aim, they still have several differences due to the cluster/sequence-generation methods employed. Their characteristics will be compared in this chapter.

In hierarchical cluster generation, pair wise document similarity coefficients are calculated. Clusters are generated according to the magnitude of the coefficients. Documents which have highest similarity coefficients will be combined into a cluster immediately. Therefore information concerning all documents must be obtained in the first place.

For the file sequencing algorithm, document insertion is done sequentially. The order of the documents in the unsequenced file would therefore directly affect the quality of the resulting sequenced file. It is very different from the files generated by the hierarchical clustering methods which consider all the existing documents at the same time.

5.1 DOCUMENT INSERTION

Document insertion by both types of classification methods is the same: insertion is done by looking up the file in the same way as query processing. The document which is waiting for insertion is treated as a query. It is then inserted into the location where its neighbors are most similar to it.

Therefore there is a trade-off between efficiency and effectiveness.

Some heuristic methods of cluster-generation have been discussed in Chapter 3. They are, in general, more efficient than the hierarchical cluster-generation methods. However, the clusters generated are less effective than those generated hierarchically.

In the next chapter a few heuristic modifications of the file sequencing algorithm will be introduced. Their efficiency and effectiveness will be investigated both experimentally and statistically.

CHAPTER 6: EXPERIMENTAL WORK

Lowden's sequencing algorithm re-sequences an unsequenced file to a near-optimal form. However it requires O(n^2) comparisons of records for the initial sequence construction. Hence it is too expensive for a file with a large number of records or that is updated frequently.

In order to enhance the, efficiency of the algorithm, it is possible to modify the insertion strategy so as to reduce the number of comparisons by a certain amount while maintaining the effectiveness of the algorithm.

Apart from enhancing the efficiency of the algorithm, it is also possible to introduce a *key term weighting scheme* into the algorithm during the calculation of Hamming Distance. Records containing key terms with higher weights would have higher likelihood to be clustered together. It would help to reduce the number of block accesses during query processing.

The performance of different sequencing algorithms with key term weighting will be analyzed both experimentally and statistically with the help of computer programs.

6.1 NEW SEQUENCING METHODS

Lowden's algorithm (i.e. *Sequence 1)* inserts a new record into a position in the partially sequenced file such that the total Hamming Distance is minimized. However due to its high computational Cost, two new sequences will be introduced to enhance its efficiency.

Sequence 2 is a modification of Sequence 1. A new record is inserted into a position such that the total Hamming Distance of the file is increased by less than a threshold t. If such a position is found during the scanning phase of the partially sequenced file, the new record is inserted without further scanning and thus reducing the number of comparisons needed. It is noticeable that when t is very small, we will be faced with an algorithm very similar to Sequence 1. On the other hand, when t is very large, the sequenced file would be poorly sequenced. Therefore the quality of the sequenced file is directly related to the threshold of insertion.

Sequence 3 is a similar modification of Sequence 1 but insertion is done according to the criteria that the average Hamming Distance of the file is not increased. The organization of the unsequenced file may therefore affect the quality of the sequenced file However if the file size is large enough, this problem will not be significant.

The performance of these sequences will be analyzed through the help of computer programs.

6.2 QUERY PROCESSING

Query processing is done by retrieving the records which contain at least the same set of key terms specified in the queries. The number of blocks accessed is therefore dependent on the block size and the distribution of the records. It is the number of blocks accessed which affects the time taken for query processing since they are done by physical disk access. The sequencing schemes described in the previous section are therefore aimed at reducing the time needed for query processing by clustering similar records together.

The performance of those algorithms will be compared by the reduction in block accesses after sequencing.

6.3 KEY TERM WEIGHTING

According to the original work of Lowden [Lowden 1985] all key terms are of equal weight. However in real life some key terms are more important than the others and therefore would have higher chance to appear in queries. In order to simulate this situation, a weight is now given to each key term. During sequencing, records containing key terms with higher weights would have higher likelihood to be clustered together. It can help to reduce the number of block accesses during query processing. The approach can be achieved by modifying the method of calculation of the Hamming Distance. The idea can be illustrated by the following example.

Suppose there are seven key terms (a, b, c, d, e, f, g) with weights (1, 1, 2, 1, 1, 4, 3) respectively and two records R1 and R2 of the following form:

R1 = < a, b, C, d, e >

R2 = < b, C, e, f, g >

By the original method of calculation of Hamming Distance, i.e. each key term is of equal weight, HD(R1,R2) would be equal to 2 since only b, C and e are in both records.

According to the new key term weighting scheme, a, b, d and e are normal key terms while C, f and g are more important key terms and therefore have higher weights. The magnitude of the weights reflects the importance of them in compare with the normal key terms. Under this key term weighting scheme, the new HD(R1,R2) would be evaluated by the following formula:

$$HD(R1,R2) = \text{original HD} - \Sigma[\text{ weight(key term in common)} - 1]$$

$$= 2 - (1 - 1) - (2 - 1) - (1 - 1) = 1$$

The performance of all three sequences with and without this key term weighting scheme will be investigated. Their performance under the adjustment of the weights of key terms, number of attributes in queries and the size of a block will also be investigated.

In addition to simulating the performance using computer programs, it can also be estimated by statistical methods. The experimental results will be compared with the theoretical results so as to validate their correctness.

6.4 HAMMING DISTANCE OF THE UNSEQUENCED FILE

The total Hamming Distance of an unsequenced file is dependent on its characteristics such as the number of attributes in a record and the number of possible key terms. It is therefore possible to estimate its total Hamming Distance with the use of some statistical models.

The experimental result will be compared with the theoretical result in order to justify the statistical model.

6.5 CHOOSING THRESHOLDS FOR SEQUENCING 2

According to the key term weighting scheme, it is clear that different thresholds are needed for key terms with different weights. Some thresholds may be more effective than the others to a particular set of weights. Therefore choosing a suitable threshold for a particular situation is crucial.

Thresholds may be chosen according to experience or by statistical estimation. Both approach will be carried out to investigate the effect of different thresholds. It is possible that an optimal threshold could be found which suits all situations.

This part will be carried out using a statistical technique called *Multiple Regression* It is a general statistical technique through which one can analyze the relationship between a dependent or criterion variable and a set of independent or predictor variables. The value of the dependent variable can also be predicted by the values of the set of independent variables.

The results from Regression Analysis will be justified by a series of statistical *Hypothesis Testing.*

6.6 HAMMING DISTANCE OF THE SEQUENCED FILE

The Hamming Distance of the sequenced files will depend on the number of attributes in a record, the number of possible key terms and their respective weights. Multiple Regression will be used to find out their relationship. The result can help to predict the Hamming Distance of the sequenced files before sequencing is carried out.

6.7 RELATIONSHIP BETWEEN HD OF UNSEQUENCED AND SEQUENCED FILES

Apart from the factors mentioned above, the structure of the unsequenced file is an important factor which affects the resulting Hamming Distance of the sequenced files. If the relationship between the Hamming Distance of the sequenced files and the unsequenced one can be found, it may be possible to predict the Hamming Distance of the sequenced files in advance. Multiple Regression will be used to investigate this relationship.

CHAPTER 7: PROGRAM DESIGN

7.1 PROGRAMS WRITTEN

In order to investigate the performance of the sequencing algorithms, seven programs were written in the C language to simulate sequence generation and query processing.

Functions of the programs written are listed below:

- *1) file.c* - generates unsequenced files
- *2) cal_hd.c* - calculates the Hamming Distance of a file
- *3) query.c* - generates queries
- *4) first.c* - sequences unsequenced files by Sequence 1
- *5) secon d.c* - sequences unsequenced files by Sequence 2
- *6) third.c* - sequences unsequenced files by Sequence 3
- *7) match.c* - matching queries to records

7.2 PARAMETERS ADJUSTABLE DURING SIMULATION

Several parameters were incorporated to be adjustable for experimental purpose. They were:

- No. of records in a file
- No. of attributes in a record
- Block size
- No. of key terms available

- Weights of the key terms
- No. of queries to be generated
- No. of attributes in each query
- The threshold of Sequence 2

7.3RECORDS GENERATION

Key terms in records are represented by integers. They are generated randomly by a random number generator which uses the system clock as the seed. It ensures that files generated are unique.

Each key term can only appear once in each record. If the random number generated is the same as any of the key terms already in the record, it will be regenerated.

The program FILE.C is responsible for generating unsequenced files which will be stored for further processing.

7.4 FILE SEOUENCING

Weights of the key terms are held in a file called WEIGHT. Sequencing is done according to the weights specified in the file using the key term weighting scheme described in the previous chapter. After sequencing, the sequenced file, its total Hamming Distance and the number of comparisons used are stored.

7.5 QUERY GENERATION

Queries are generated by the program QUERY.C. They are also represented by integers which are generated by a random number generator. However the generation scheme is different from the one being used in file generation since queries are generated according to the weights of the key terms specified in the file WEIGHT. The probability of a key term being generated is directly proportional to its weight.

Although the random number generator of C language does not have this special feature, it is possible to attain the desired effect by doing a transformation on the random numbers generated. Instead of generating the random numbers which represent the key terms directly, a new set of random numbers are used. Within this set, the number of members representing a key term is directly proportional to its weight. For example, a set of key terms {a, b, C, d} with weights {1, 2, 3, 1}, the set of random numbers generated would range from 0 to 6 which represents {a, b, b, C, C, c, d}. In this way, the chance of a key term being generated would be directly proportional to its weight.

7.6 QUERY PROCESSING

Queries generated are matched against the records. A record will only be retrieved if it contains the key terms specified in the queries. The matching process is done by the program MATCH.C. The average number of block accesses will be printed after execution.

7.7 IMPLEMENTATION

As experiments of various forms are to be carried out, the programs must therefore be designed to be as flexible as possible to suit general purposes. Also sequencing a file with a thousand records is too time-consuming to be done interactively. Therefore batch processing is an optimal choice.

In order to facilitate batch processing, the function of each program is designed to be as specific as possible. Inputs to each program are passed by command line arguments. i.e. all the parameters needed for program execution are typed right after the program name on the command line of the operation system. The programs together with their parameters can be pre-typed in a file for batch execution. Various types of experiments can therefore be carried out by altering the order of execution of the programs and their corresponding parameters.

CHAPTER 8: EMPIRICAL RESULTS

8.1 HAMMING DISTANCE OF THE UNSEQUENCED FILE

The unsequenced files are generated such that each key term is represented by a random number generated by the random number generator of the C language. It is possible to estimate the Hamming Distance of an unsequenced file with the help of statistical probability distribution theories since the probability of a specific record being generated is known.

Since each key term can only appear once in each record, sampling is therefore done by *Simple Random Sampling without Replacement (SRSWOR).* Let r be the number of random numbers available to be generated and n be the number of attributes in a record, the number of possible unique records to be generated will be equal to rCn. That means the probability of each record being generated is equal to 1 / rCn.

After the first possible record, generated with x standard statistical *Distribution.,*has been generated from the set of rCn, the probability of the second one being "key terms" equal to the first one follows a probability distribution - *the Hypergeometric*

Suppose there are r preferred objects out of a population of N objects. A sample of n objects are taken from these N. The probability of having x preferred objects in this sample is equal to:

$$ {}_N C_x \left[r / n \ldots r-(x-1) / N-(x-1) \right] \left[(N-r)/(N-x) \ldots \{(N-r)-(n-x-1)\}/\{N-(n-1)\} \right] $$

$$ = \{ {}_r C_x \; {}_{N-R} C_{N-X} \} / {}_N C_n $$

This is the so-called Hypergeometric Distribution.

The first record of the unsequenced file, having n distinct attributes, is drawn from a population of r key terms. For the second record, the key terms are drawn from the same population but with n key terms which are already present in the first one. Therefore the probability of having x key terms equal to the first one is:

$$P(X) = {}_{n}C_{x} \quad {}_{r-n}C_{n-x} \quad / \quad {}_{r}C_{n}$$

As records are generated independently of each other, when generating the kth record, the probability of having x matches with the (k-1)th record is still equal to p(x).

According to this principle, the total Hamming Distance of the file can be estimated by calculating the expected value of the Hamming Distance. When the number of matches is x, the Hamming Distance between the two records is (n-x). Therefore the expected total Hamming Distance is:

$$E(\text{total HD}) = (N - 1) * \sum_{X=0} (n - x)\, p\,(x)$$

where N = total number of records in the file

Experimental results generated from computer programs are compared with the values calculated from this expression. A file size of 1000 was *used.* The result is plotted and a diagonal is drawn in the figure. The experimental results would be closer to the theoretical values if the points were distributed near to the diagonal. From this figure it is clear that the result is quite satisfactory.

8.2 CHOOSING THRESHOLD 5 FOR SEQUENCE 2

When sequencing the records according to the key term weighting scheme with Sequence 2, different thresholds would be needed for different weights of the key terms since insertion of new records can only be done if the increment in total Hamming Distance is less than the threshold. If the threshold is set to too high, the records will not be sequenced at all. If it is too low, the number of comparisons will be too high. Therefore choosing suitable thresholds is crucial for getting satisfactory sequenced files.

In order to set a reasonable value for the threshold, the first thing to consider is the possible Hamming Distance between two records. The threshold should be at least smaller than the mean of the possible Hamming Distance. That means insertion can only be done if the records are at least 50% similar to each other. Otherwise the insertion would be meaningless.

To simplify the situation for analysis purpose, during the experiments the number of attributes in a record is fixed to 10 and only one key term will have a higher weight. All other key terms have weights equal to one.

Let w be the weight of one of the key terms. If there are x key terms in common within two adjacent records and they contain the higher-weighted key term, the Hamming Distance between these two records will be [(10-x) - (w-1)]. On the other hand, if they do not contain the higher-weighted key term, the Hamming Distance between them will be (10 - x). Therefore the following relationship can be obtained:

no. of key terms matched between two records (x)	possible HD	presence of higher-weighted key term	Prob(HD/X)
10	1-W	y	0.5
	0	n	0.5
9	2-w	y	0.5
	1	n	0.5
1	10-w	y	0.5
	9	n	0.5
0	10	-	1

Table 8.1 Possible HD between adjacent records and the probability of attaining them.

According to the description in section 8.1, the number of matches between two records follows the Hypergeometric Distribution. Hence the expected value of the Hamming Distance between two records would be equal to:

- $E(HD) = \Sigma$ possible $HDi * P(HDi \cap X)$

  $= \Sigma$ possible $HDi * P(X) * P(HDi/X)$

- It is obvious that any threshold higher than this value would be meaningless.

From calculation, the following values are obtained:

w	maximum threshold
1	5.0
2	4.5
3	4.0
4	3.5
5	3.0

Table 8.2 Maximum thresholds for different values of weight.

An experiment has been done to investigate the performance of different thresholds. A ratio of the total Hamming Distance to the number of comparisons of records is plotted. Sequence 2.n means the threshold is being set to n. It is very clear that a significant decrease in the Hamming Distance is found when the threshold is smaller than 4. In addition Sequence 2.2 can approximate Sequence 1 very well under all three different values of the weights. This matches the theoretical result given above: if the weight is not greater than 5, a satisfactory result will be obtained if the threshold is not greater than 3.0.

As a threshold of 2 is very suitable for all the values of weights used in the experiments, the threshold of Sequence 2 will be fixed to have a value of 2 for the remaining experiments.

8.3 EFFICIENCY OF THE SEOUENCES

Sequence 1 gives the best sequence among all three sequencing algorithms and therefore would require higher number of comparisons during sequencing. However, it is too expensive with respect to computational cost. Therefore approximation algorithms such as Sequence 2 and 3 are introduced to reduce the computational cost.

The results obtained show that when the weight is small, Sequence 2 and 3 can approximate Sequence 1 very well while reducing the number of comparisons needed by at least half. Although when the weight equals to five the approximation is not that satisfactory, they can still maintain a very low total Hamming Distance in comparison with the unsequenced file.

8.4 PERFORMANCE OF THE SEQUENCES DURING OUERY PROCESSING

8.4.1 PERCENTAGE REDUCTION IN BLOCK ACCESSES

When matching queries against the records, a retrieval will be done only if the record contains all the key terms specified in the queries. The number of blocks accessed is therefore dependent on the block size. The objective of these three sequences is to reduce the total number of block accesses in compared with an unsequenced file during query processing.

A series of experiments have been carried out to determine the reduction in block accesses after sequencing. A total of 1000 records and 100 queries were generated. Block size of 5, 10, 15 and 20 have been used. The results of having weights of 1, 3 and 5 are plotted in 3 different figures. Each point on the graphs is the average of the block sizes being used.

The first figure shows that when weight is equal to 1, the performance of all three sequences are very similar to each other.

In the second figure, the weight is raised to 3. The performance of Sequence 1 has 4 percent of improvement while that of Sequence 2 and 3 are decreased by a certain amount. The last figure shows that a similar result is obtained when the weight is set to 5.

These results highlight the clustering ability of the sequences. Under the key term weighting scheme, records containing the higher weighted key term would have a higher probability of being clustered together. Sequence 1 is the only sequence among all three which can maintain the clustering ability at a near-optimal level at all times. However its approximations (Sequence 2 and 3), can only resemble the clustering ability of Sequence 1 to a certain extent. As the higher weighted key term would appear more frequently in the queries, the less efficient sequencing processes (Sequence 2 and 3) would result in a degradation in performance.

8.4.2 TREND OF THE PERCENTAGE REDUCTION

Figures from experimental results showed that the percentage reduction in block accesses would be highest when the number of attributes in queries is around 3. When the number of attributes in queries is too small or too large, the percentage reduction is decreased.

This trend of percentage reduction is due to the distribution of the records matched. When there is only 1 attribute in each query, a large number of records in the file would be matched with the queries. Therefore whether the records are sequenced or not does not have much effect on the number of block accesses.

When the number of attributes in each query is large, the number of records in the file that match with the queries would be very limited. Therefore, even when using the sequencing algorithms that are quite efficient, the number of block accesses would not be greatly reduced.

The maximum value of the percentage reduction in accesses can be estimated by using statistical methods.The probability of a query matching a particular record is given by:

$$P(\text{matching}) = n/r\ldots\ldots(n-x+1)/(r-x+1)$$

Where n = number of attributes in a record

x = number of attributes in the query

r = number of possible keytems

The probability of obtaining y records in a file that matches the query 'can be found with the help of this equation. It is also possible to estimate the percentage reduction in block accesses.

The expected number of block *accesses* in the unsequenced file when there are y matches is approximately equal to y if y is not greater than the number of blocks in the file. This is because the matched records may be distributed among all the blocks in the file. If y is greater than the number of blocks in that file, the expected number of block accesses would be equal to the number of blocks in the file.

In the sequenced file, the estimation of number of block accesses could be done by making the assumption that sequencing will place all matched records as close as possible. Therefore the expected number of block accesses would be equal to [(y-1) div block_size) + 1.

The difference between these two expected values is the reduction in block accesses after sequencing. With this value, it is possible to approximate the expected percentage reduction in block accesses with different number of attributes in queries. It is given by the following equation:

$$\sum_{y=0}^{N} E(\text{\% reduction in block accesses} \mid y)\, P(\text{ y matches})$$

$$= \sum_{y=0}^{N} E(\text{\% reduction in block accesses} \mid y)\ {}_{N}C_{y}\, p^{y} (1-p)^{N-y}$$

where N = no. of records in the file

p = P(match)

The results of the above estimation is summarized in the following table:

No. of attribute in query	% reduction
1	49.55
2	75.87
3	88.83
4	88.96
5	87.22
6	81.21

Table 8.3 Optimal percentage reduction in block accesses with different number of attributes in queries.

The percentage reduction in block accesses in this table is much greater than the experimental values, as the estimation is based on the assumption of optimal sequencing which can never be attained even with Sequence 1. Although the estimated values are not directly comparable with the experimental ones, they still show a trend which is very similar to the experimental one: the maximum reduction is attained when the number of attributes in queries is around 3.

8.5EVALUATING THE KEY TERM WEIGHTING SCHEME

The objective of the key term weighting scheme is to increase the likelihood to cluster the records containing important key terms together so as to reduce the number of block accesses during query processing.

The performance of the sequencing algorithms under the scheme is compared with their performance when rating all key terms with equal weight. The results are shown in Table 8.4 and 8.5. The number of attributes in queries is 3, and the weight of one of the key terms is set to 3 or 5.

weight = 3

blk size	5	10	15	20
sequencing without weight				
avg. no. of block accesses				
unseq	84.06	66.07	53.84	44.33
1	57.23	44.27	37.87	32.86
2	59.31	46.20	39.11	34.02
3	58.76	46.01	38.87	33.95
sequencing with weight				
avg. no. of block accesses				
unseq	84.06	66.07	53.84	44.33
1	55.54	42.15	35.17	30.18
2	66.42	51.05	41.97	34.81
3	64.74	50.00	40.72	33.91

Table 8.4 Average no. of block accesses when the weight of the higher-weighted key term is 3.

weight = 5

blk size	5	10	15	20
sequencing without weight				
avg. no. of block accesses				
unseq	82.86	65.33	53.57	44.08
1	56.60	44.53	37.60	33.08
2	59.14	46.07	38.70	33.92
3	58.90	46.36	39.22	34.12
sequencing with weight				
avg. no. of block accesses				
unseq	82.86	65.33	53.57	44.08
1	53.49	40.08	32.78	28.29
2	69.52	51.24	40.62	33.28
3	69.38	51.46	40.54	33.09

Table 8.5 Average no. of block accesses when the weight of the higher-weighted key term is 5.

As can be seen from the tables, Sequence l's performance is improved with the use of the key term weighting scheme; it conforms to the objective of the scheme.

The results of Sequence 2 and 3 were contradictory: the number of block accesses increases when weighting is involved during sequencing. Since the Hamming Distance between two records would be reduced if they contain the higher-weighted key term, it is obvious that the increment in total Hamming Distance would be much smaller with the use of the key term weighting scheme. Although both Sequence 2 and 3 can adequately cluster the records containing the higher-weighted key term together, due to the relatively small Hamming Distance between them, they cannot sequence the records within this set comprehensively. Under the insertion strategy of these two algorithms, insertion within this set of records would not be as satisfactory as within the other records since the increment in total Hamming Distance during insertion may always fall below the insertion threshold. Intra-set sequencing would therefore be affected. As a result, both Sequence 2 and 3 become less efficient when sequencing under the key term weighting scheme.

Another significant result is observed from the above tables:

the difference between the number of block accesses with and without the use of the key term weighting scheme is decreasing gradually while the block size is increasing. This phenomenon can be explained by the above interpretation: neither of the two sequences are able to adequately sequence the records containing the higher-weighted key term.

This was found to be related to the Hypergeometric Distribution. Multiple Regression was used to estimate the total Hamming Distance of sequenced files. The analysis was carried out firstly with, and then without, the use of the total Hamming Distance of the unsequenced file. Other factors under consideration were the number of attributes in the file, the number of possible key terms and the weight of the higher-weighted key term. The results showed that the prediction accuracy of the regression lines was slightly improved with the use of the total Hamming Distance of the unsequenced file. However, this difference will gradually become insignificant as the file size and the number of experiments carried out increases since the total Hamming Distance of the unsequenced file will asymptotically approach constant.

The percentage reduction in block accesses during query processing was found to be optimal when the number of attributes in queries is approximately 3. This trend has been analyzed by statistical techniques in Section 8.4.2, and the results showed that the optimal performance is obtained when the number of attributes in queries is around 3 to 4.

CHAPTER 9: CONCLUSIONS AND FURTHER RESEARCH

9.1 CONCLUSIONS

The characteristics of various file clustering and sequencing algorithms were discussed in previous chapters. A comparison of them were given in Chapter 5. Heuristic modifications of them were also introduced.

Simulations of Lowden's file sequencing algorithm (Sequence 1) and its two heuristic modifications, Sequence 2 and 3, have been performed. A key term weighting scheme has also been introduced, and its implication on query processing was compared with files without the association of key term weighting. Empirical results indicated that Sequence 2 will have the best performance when the threshold is set equal to 2. The results also showed that the heuristic modifications are less effective with the association of key term weighting. However, the efficiency of file sequencing is significantly improved over the original algorithm (Sequence 1). Therefore the trade-off of efficiency and effectiveness is once more shown.

In addition to experimental analysis, statistical techniques were also employed to estimate the Hamming Distance of various kinds of files. The total Hamming Distance of unsequenced files

9.2 FURTHER RESEARCH

A key term weighting scheme has been employed during sequencing. However, due to the limitation of time, only one of the available key terms was associated a higher weight during analysis. Further analysis would be possible for investigating the effect of associating various magnitude of weights to the key terms. It would surely be a more interesting and challenging task.

The information retrieval system introduced in an earlier chapter is a conventional one. Current research interest is now on a new kind of system called the *Hypertext* system. Interested researchers can refer to various papers released recently in the Computer Journal Vol.35, No.3, 1992.

References

1. **Cardenas, A.F.,** *Analysis and performance of inverted database structure,* Communications of the ACM, 18, 5, 253-263, 1975.

2. **Comer, D.,** *The ubiquitous B-tree,* ACM computing surveys, 11, 2, 121-138, 1979.

3. **Guttman, A.,** *R-trees: a dynamic index structure for spatial searching,* Proceedings of the 1984 ACM SIGMOD International conference on the mgt. of data, 47-57, 1984.

4. **Larson, P.A.,** *Linear Hashing with partial expansions,* Proceedings of the 6th international conference on VLDB, 224-232, 1980.

5. **Nievergelt, J. and Hinterberger, H.,** *The grid file: An adaptable symmetric multikey file structure,* ACM Transactions on database systems, 9, 1, 38-71, 1984.

6. **Jain, A.K. and Dubes, R.C.,** *Algorithms for Clustering Data,* Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

7. **King, B.,** *Step-wise clustering procedures,* Journal of the American Statistical Association, **69**, 1967.

8. **Lowden, B. G. T.,** *An Approach To Multikey Sequencing In An Equiprobable Key term Retrieval Simulation,* Proceedings of 8th ACM SIGIR Conf. on Research and Development in Information Retrieval, 1985.

9. **Mauldin, M.L.,** *Conceptual Information Retrieval: A case study in adaptive partial parsing,* Kluwer Academic publishers, 1991.

10. **Moghrabi, I.A.R.,** *Expert Systems and Their Use in Libraries,* Sci-Quest, **5**, Issue 2,1995.

11. **Salton, G.,** *Dynamic Information and library processing,* Prentice-Hall, Englewood Cliffs, new Jersey, 1975.

12. **Salton, G. and McGill, M.J.,** *Introduction to Modern Information Retrieval,* 1983, McGraw Hill.

13. **Wong, S.K.M., Zierko, W. and Ranghvan, V.V.,** *On Modeling of information retrieval concepts in vector spaces,* ACM Transactions on Database Systems, **12,** No.2, 1987.

14. **Yu, C.T. Lam, K. Suen, and siu, M.K.,** *Adaptive Record clustering,* ACM Transactions on database systems, 10, 2, 180-204, 1985.

15. **Yu, C.T. Luk, W.S. and Siu, M.K.,** *On the estimation of the number of desired records with respect to a given query,* ACM Transactions on database systems, 3, 1, 41-56, 1978.

16. **Ymmanuel, M.B.** -*Clustering Techniques for Record Databases,* M.Sc Dissertation (University of Essex 1993)