Clustering Records in Information Retrieval Systems

Maisaa A.K. Al-Saffar

March 1995

RT
138

Clustering Records in Information Retrieval Systems

Maisaa A.K. Al-Saffar
B.Sc., Beirut University College

THESIS

Submitted in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science
at the Lebanese American University
March 1995

Signatures Redacted

Dr. Issam Moghrabi (Supervisor) Signatures Redacted
Assistant Professor of Computer Science
Lebanese American University

Dr. George Nasr
Assistant Professor of Electrical and Computer Engineering
Lebanese American University

Signatures Redacted

Dr. Haidar Harmanani
Assistant Professor of Computer Engineering and Computer Science
Lebanese American University

Abstract

This project is a study of the space density of a file and how it affects retrieval time. The number of blocks that are retrieved when a query is made to the document is used as a measure of response time. The space density is an indication of how close related records are placed. The hamming distance of a file is used as a measure of space density. A sequencing algorithm based on threshold values that reorders records in a file so as to increase average record similarity is used to obtain varying space densities. Simulation experiments conducted proved that a great reduction in response time is yielded after the restructuring of a file with a reasonable amount of work required in sequencing. Other terms such as block size and terms in a query which affect response time are studied. Results from the experiments are shown graphically. Statistical methods are used to confirm the experimental results.

Acknowledgements

In working on this project, I've been fortunate to have the supervision and guidance of Dr. I. A. R. Moghrabi. I gratefully acknowledge his numerous suggestions and constructive criticisms.

I further acknowledge my debt to my family whose ongoing financial and moral support saw me through the years at this university.

Finally, I lack the words to thank my friend Amer Bawab whose precious time and encouraging attitude helped me complete this project.

Table of Contents

Chapter 1

Introduction

Information retrieval (IR) is a discipline which involves the organisation, structuring, searching, and dissemination of information. The information retrieval system is designed to make available a collection of stored information in response to a query with the objective of providing references that contain the information required.

Organisation is required in IR systems to group the documents in such a way that retrieval is accelerated. Documents or records in a database file are grouped because they are in some sense related to each other and are therefore likely to be retrieved together.

Normally, queries to a database file are inefficiently handled by scanning every record in the file, or by the use of indexes for faster access to the records. Inverted and multilist file systems together with their associated directories index documents or records by a set of identifiers that are known as keyterms or index terms. Before fetching the data file the query processor of the inverted file system consults the directory decoder that returns from the directory a set of addresses of records that satisfy the query. The generation of these addresses is then followed by the retrieval of the blocks that contain these records. Total response time is therefore dependent on the time taken to translate the query into addresses together with the time needed to retrieve the required blocks. Hence the less blocks retrieved for a query, the smaller the total response time.

To date, most techniques for improving response time, in an equiprobable keyterm situation, have been concerned with rephrasing the query by employing strategies such as identifying common subexpressions and combining certain

operations to minimise directory searching for records. However, little attention has been given to the issue of how main file records might be sequenced to minimise the number of blocks retrieved in a general query.

The purpose of this project is a sequencing algorithm that enhances the efficiency of current sequencing algorithms. The existing algorithm [Lowden, 85] is only concerned with sequencing the records so as to minimise, as far as possible, the sum of the hamming distances between them. This ensures that records which have similar properties are in close proximity to each other in the file space. However this file structuring algorithm required comparisons of the order of n^2.

In our proposed sequencing algorithm, a different insertion strategy is adopted so as to reduce the number of comparisons made for the sequence construction while maintaining the effectiveness of the algorithm.

The enhanced version of the current algorithm is based on the specification of a threshold value. Insertion of incoming records occurs immediately once the threshold condition is satisfied. For the rest of this report, we refer to the current algorithm as sequencing algorithm1 and to its enhanced version as sequencing algorithm 2.

The main purpose of this project is to sequence a file so that its total hamming distance is minimised in some sense. It is demonstrated in this report that when a file's hamming distance is reduced, the number of blocks retrieved when a query set is matched against it is also reduced. Reduction in blocks retrieved implies reduction in response time.

An analysis of the effects of varying block size and terms in a query is carried out to find ways of further improving response time.

used to estimate the hamming distance of the random or
also the hamming distance after sequencing using
The percentage reduction in blocks retrieved after
ated. As part of this project. statistical formulae are
e these estimates.

Chapter 2

Clustering of Database Records

2.1 File System Description

The files used in this project consist of records of fixed length. An example of a file system is an employee file where each record represents an employee details and has a fixed number of keyterms.

A key conversion system is assumed to convert keyterms into numerical values within some range. That is, there is a direct translation or mapping of keyterms into numerical values. Table 0 below shows the coding of attribute values for the employee file system.

Attribute	Attribute Value	Code
Starting Date	1980 - 1985	1
Starting Date	1986 - 1991	2
Starting Date	1992 - 1996	3
Position	Operator	1
Position	Programmer	2
Position	Systems Analyst	3
Salary	< 500	1
Salary	500 - 1000	2
Salary	> 1000	3
Marital Status	Single	1
Marital Status	Married	2
Marital Status	Divorced	3

Table 0

If our file is very large, it would not fit within a single block of storage media but would occupy several blocks. When a record is to be read, the block in which this record is found is retrieved. If all records satisfying a given query are confined to 1 block then only a single block will be retrieved. On the other hand, if all the records to be retrieved, say n are in different blocks then n blocks will be retrieved.

Usually queries are requests to retrieve similar records, i.e. records that have certain keyterms in common. If these similar records happen to be placed together then the retrieval of these records will require the retrieval of few blocks. The hamming distance explained in the next section gives an indication of how the proximity or disparity between records in a file can be measured.

2.2 Hamming Distance

The hamming distance between 2 records is the number of noncommon keyterms between them. For example, given the 2 records shown below,

1	2	3	1
1	2	3	2

the 2 records have out of 4 keyterms, 3 keyterms in common (1, 2, 3) and 1 keyterm not in common. The hamming distance between these 2 records is therefore 1.

The total hamming distance of a file is the sum of all hamming distances between adjacent records. For example, consider a small file such as the one shown as Sample 1 below.

Starting Date	Position	Salary	Status
1	2	3	1
2	1	2	2
1	2	1	3
3	3	2	2

Sample 1

The hamming distance between the

- 1st and 2nd records is 4
- 2nd and 3rd records is 4
- 3rd and 4th records is 4

The total hamming distance is therefore = 4 + 4 + 4 = 12. By dividing this number by 3 (the number of records - 1) the average hamming distance can be obtained as 4. This implies that for Sample 1, on the average adjacent records have 4 keyterms not in common.

From the previous section, it is advisable to have files with as small average hamming distance as possible so that fewer blocks are retrieved when queries are made. In order to achieve a small average hamming distance, rearrangement of the records of the file is required to place similar records together. This rearrangement is usually termed as sequencing or clustering. The next 2 sections describe two forms of sequencing.

2.3 Sequencing Algorithm 1

This algorithm sequences an unsequenced file to near-optimal form. The sequencing process involves retrieving the records in the order they are originally

arranged and finding for them new positions among the already sequenced records. The position that is sought for each incoming record is one leading to the minimum increase in hamming distance, thereby, the resultant file would have the least possible hamming distance. All possible positions at which an incoming record can be placed are tested. For each position, the increase in hamming distance is computed and track is kept of the position giving rise to the smallest increase in hamming distance where the incoming record is ultimately placed. The sequencing stops when the sequenced list of records consists of all the file records.

The first record retrieved from the unsequenced list of records has only 1 possible insertion position. The 2nd has 2 since it can be placed either in front or at the back of the 1st record and both positions give rise to the same increase in hamming distance. It is therefore placed at either position. The 3rd record will have 3 possible positions. If there are n records in the file the nth record will have n possible positions to try and hence n comparisons of total hamming distance increases to make.
Therefore the total number of comparisons made in sequencing a file of size n is

$$1+2+3+\cdots+n=\frac{n\times(n+1)}{2}=\frac{n^2}{2}+\frac{n}{2} \quad \in O(\frac{n^2}{2}) \quad .$$

This implies that the algorithm is of the order of n^2 for large n.
Reducing the number of comparisons made in sequencing using sequencing algorithm 1 results in the algorithm described in the next section.

2.4 Sequencing Algorithm 2

This algorithm's sequencing process is similar to that of sequencing algorithm 1 with the difference of it being based on the specification of a threshold value. When the latter is satisfied no further comparisons are made, thereby minimising the number of comparisons.

An incoming record starts at the front of the already sequenced list of records and tries one position after the other until a position is found where the increase in hamming distance is less than or equal to the specified threshold value. If no such position is found, the insertion would take place at the position yielding the smallest increase in hamming distance (even though it is greater than the threshold value specified) which had been kept track of during the search for the first position.

The increase in hamming distance ranges between 0 and r inclusive, r being the number of keyterms per record. It is 0 when the incoming record is placed adjacent to a record that it is identical to. The increment is r if a record is inserted in between two records with which it has no keyterms in common provided the two records initially had no keyterms in common.

The higher the threshold value, the fewer positions the incoming record needs to try before it finds itself a satisfactory position. A threshold value equal to r results in a total number of comparisons equal to the number of records in the file - 2 since the first incoming record to be sequenced is the third one in the unsequenced list and each incoming record makes exactly 1 comparison (which is always satisfactory). With such a threshold value no sequencing is done and the resultant file turns out to be a copy of the original file.

A low threshold value implies a higher number of positions tried and therefore a greater number of comparisons. For example, with a threshold value of 0, and if the records in the file are unique, no position tested will be satisfactory. Incoming records will all end up being placed at the position giving rise to the smallest increase in hamming distance. Hence sequencing algorithm 2 with a threshold level of 0 degenerates into sequencing algorithm1.

The number of comparisons made in sequencing algorithm 2 is always less than or equal to that of sequencing algorithm 1. It is equal in the case of a threshold value of 0.

A wise choice of the threshold value ensures that sequencing algorithm 2 could be far more efficient than sequencing algorithm 1, i.e. producing an almost fully sequenced file with far less comparisons made.

Chapter 3

Project Description

3.1 Program Design

The following tasks are each assigned to a specific program. A detailed description of each program, its interface with the user, as well as its interaction with other programs in the package are presented in appendix A.

- Simulation of a data file (a set of records made up of randomly generated attributes-integers within some range) which is assigned to a program named FILEB.C.
- Unsequencing of the randomly generated file that simulates a file in its worst case, i.e. with the largest possible hamming distance. (Assigned to FIRST_PRIME.C)
- Sequencing of the file using the first sequencing algorithm, so that the number of comparisons it requires to accomplish the sequencing task can be contrasted with the enhanced version of the algorithm. (Assigned ta 4Hof FIRST.C)
- Sequencing of the file using the second sequencing algorithm that minimizes on the number of comparisons required by the first sequencing algorithm and which relies on a wise choice of the threshold value. (Assigned to SECOND.C)
- Simulation of a query file which is a set of query records made up of randomly generated query terms. (Assigned to QUERY.C)
- Running of queries against the data file in order to determine the number of blocks that would be retrieved. By matching queries against the random,

unsequenced, and sequenced data files the percentage reduction in blocks retrieved after sequencing is demonstrated. The data files can be any of the following :

- A random data file (as generated by FILEB.C)
- An unsequenced data file (that is produced by FIRST_PRIME.C)
- A file sequenced with FIRST.C
- A file sequenced using SECOND.C ordered with a threshold value ranging from zero to the number of attributes. (Assigned to MATCHB.C)

3.2 Simulation Experiments

The set of experiments done to simulate a file, sequence it, and run a bunch of queries against it followed the sequence of steps that is specified below. The output of the program run at each step is also described.

- Generating a number of files of varying file size and record size and range of keyterms. The output at this step is the hamming distance of each file generated. (Running FILEB.C)
- Sequencing the file using sequencing algorithm 1. The output is listed below. (Running FIRST.C)
  - Hamming distance of the original file
  - Hamming distance of the sequenced file
  - Number of comparisons made in sequencing
  - Elapsed time
- Sequencing the file using sequencing algorithm 2 at various threshold values. (Running SECOND.C)
  - Hamming distance of the original file
  - Hamming distance of the sequenced file

- Number of comparisons made in sequencing
- Elapsed time

- Unsequencing the random file to illustrate the case when a file is in its worst unordered state. The output is listed below. (Running FIRST_PRIME.C)
  - Hamming distance of the original file
  - Hamming distance of the unsequenced file
  - Number of comparisons made in unsequencing

- Generating query files of varying parameters namely the number of query terms and the query file size. (Running QUERY.C)
- Matching one of the generated query files to one of the various main files or their sequenced versions at different threshold levels. For the multitude of combinations of query files and main files, the main file block size is varied. The output at this step is as listed below. (Running MATCHB.C)
  - Number of records in the main file that satisfied queries in the query file
  - Number of blocks retrieved for having contained any of the records that are meant to be read.

Of the experiments that took place, interesting tests chosen as representative data were recorded, and used in tabulating the results in chapter 4.

Chapter 4

Experimental Analysis

Simulation of information retrieval from a file is the objective of this chapter. In order to demonstrate the behavior of the system developed, selective results are revealed.
Before proceeding with the analysis, it is of importance to take a look at appendix A which describes the special format adopted in naming the files of our experiments.

4.1 Relationship between Hamming Distance and Blocks Retrieved

- FILEB.C was run to generate a random file with the following specifications :
  - File name : "5h_10_4.R"
  - File size : 500 records
  - Record size : 10 attributes
  - Attribute range : 4

- FIRST_PRIME.C was run to produce the unsequenced version of the random file.
  - File name : "5h_10_4.U"
  - File size : 500 records
  - Record size : 10 attributes
  - Attribute range : 4

- FIRST.C was run to produce the sequenced version of the random file.

- File name : "5h_10_4.RS1"
- File size : 500 records
- Record size : 10 attributes
- Attribute range : 4

- QUERY.C was run to generate a query file.
  - File name : "5h_10_4.Q02"
  - File size : 500 records
  - Record size : 10 attributes
  - Query keyterms : 2 attributes
  - Attribute range : 4

- MATCHB.C was run against the files mentioned above with the query file that was also specified above.
  - Bucket size : 5 records per block.

The results pertaining to this experiment were as tabulated below in Table1.

File Name	Type	Hamming Distance	Average Hamming Distance	Total Blocks Retrieved
5h_10_4.R	Random	3799	8	2753
5h_10_4.U	Unsequenced	4906	10	2792
5h_10_4.RS1	Sequenced	1845	4	1992

Table 1

Table 1 indicates that for the unsequenced file the hamming distance between adjacent records is 10. That is adjacent records have on average 0 attributes out of 10 in common. Similarly, the hamming distance between adjacent records for the random file is 8 implying adjacent records have on the average as many as 6 keyterms in common. For each of these 3 files, 3189 records were read for the 100 queries as obtained from MATCHB.C. This gives an average of 32 records

retrieved, out of the 500 records in the file, per query. With a bucket size of 5, the number of buckets in the file of 500 records is 100. Since there were 100 queries and a total of 2908 blocks retrieved for the unsequenced file, it implies that the unsequenced file had on the average 29 blocks out of the 100 blocks retrieved for each query.

In the case of the random file, with a total of 2837 blocks retrieved an average of 28 blocks were retrieved for each query. Finally the sequenced file, with 1992 blocks retrieved, retrieved on average 20 blocks out of the 100 blocks per query.

For the unsequenced file there is an average of 1.1 or approximately 1 record in each of the blocks that were retrieved. This value is almost the same for the random file. As for the sequenced file on average 2 records are read in each of the blocks retrieved which shows how sequencing reduces the number of blocks retrieved when a query is made to a file, as a result of having more of the records confined in a single block.

The results of Table 1 are depicted graphically in Figure1 below.

Figure 1: Hamming Distance Vs. Blocks Retrieved

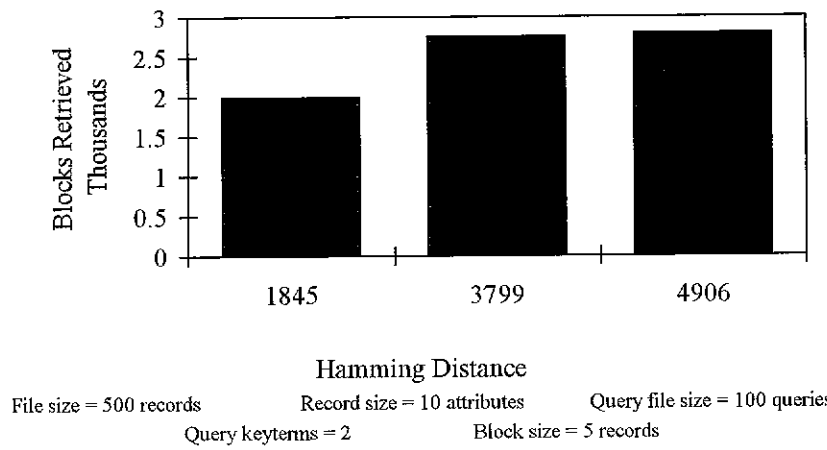


Figure 1 shows that as the hamming distance decreases the blocks retrieved when a query is made to a file also decreases.

4.2 Minimising the Hamming Distance

The unsequenced file "5h_10_4.U" is sequenced using Sequencing algorithm 1 and sequencing algorithm 2 with threshold values from 0 to 10.
Table 2 shows the hamming distance of each of the 12 resulting files, the no. of comparisons made as well as the elapsed time in undertaking the sequencing. We call the file resulting from sequencing algorithm 1 "5h_10_4. US1" and those from sequencing algorithm 2 "5h_10_4.2i" where *i* is the threshold value used. The same query file in the previous section namely "h_10_4.Q02" was run against those 12 files using a block size of 5.
The number of blocks retrieved is also specified in Table 2 .

File Name	Threshold	Hamming Distance	Average Hamming Distance	Elapsed Time	Comparisons	Blocks Retrieved
5h_10_4.US1	-	1865	3.74	2'52"	125,247	2020
5H_10_4.U20	0	1865	3.74	2'52"	124,395	2020
5H_10_4.U21	1	1866	3.74	2'36"	122,919	2020
5H_10_4.U22	2	1880	3.77	1'92"	113,151	2022
5H_10_4.U23	3	1891	3.79	1'15"	92,677	2024
5H_10_4.U24	4	1984	4.00	0'43"	56,705	2092
5H_10_4.U25	5	2301	4.61	0'43"	20,146	2232
5H_10_4.U26	6	2686	5.38	0'01"	5,212	2458
5H_10_4.U27	7	2973	5.96	0'05"	2,255	2622
5H_10_4.U28	8	3219	6.45	0'05"	1,461	2701
5H_10_4.U29	9	3499	7.01	0'05"	995	2766
5H_10_4.U2T	10	4903	9.83	0'05"	498	2908

Table 2

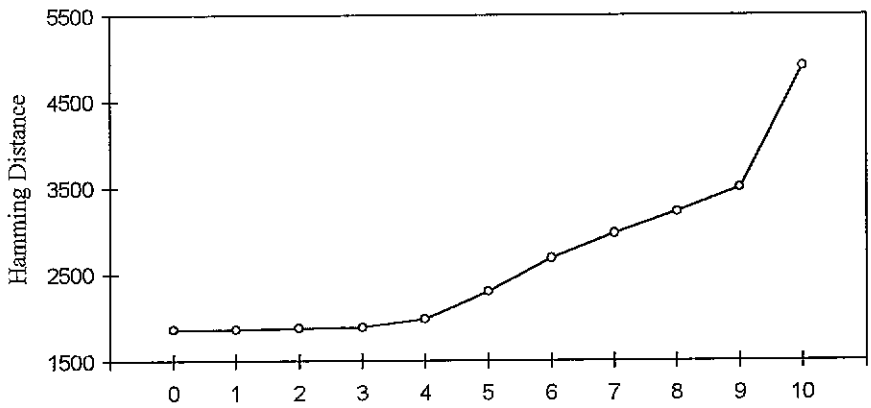
The observations that can be made here are that, with high threshold values such as 10, little sequencing is made. The first position checked is always acceptable. The number of comparisons 498 is in fact the number of records in the file minus 2 since the first 2 records in the file constitute the initial sequenced list in sequencing algorithm 2 as stated earlier. In other words, we start sequencing from the third record trying to find a place for it in the so far sequencenced list. The resulting file has therefore a hamming distance of 4903 which is almost the same as the hamming distance of the original unsequenced file (See Table 1).

Table 2 clearly demonstrates that as the threshold value is reduced, the hamming distance decreases at the cost of increasing the number of comparisons being made. Also notice that the same hamming distance results from the use of either sequencing algorithm 1 or sequencing algorithm 2 with a threshold value of 0, however with less comparisons made using the latter.

The results of Table 2 are depicted graphically as Figure 2.1.

Figure 2.1: Threshold Value Vs. Hamming Distance

Sequencing of files done using Sequencing Algorithm 2

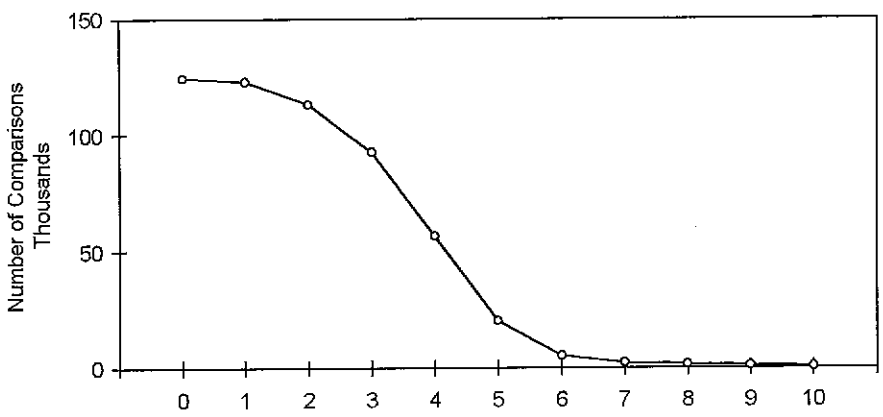


File size = 500 records Record size = 10 attributes Query file size = 100 queries Query keyterms = 2

A graphical representation of threshold value against the number of comparisons is as shown below in Figure 2.2.

Figure 2.2: Threshold Value Vs. Comparisons

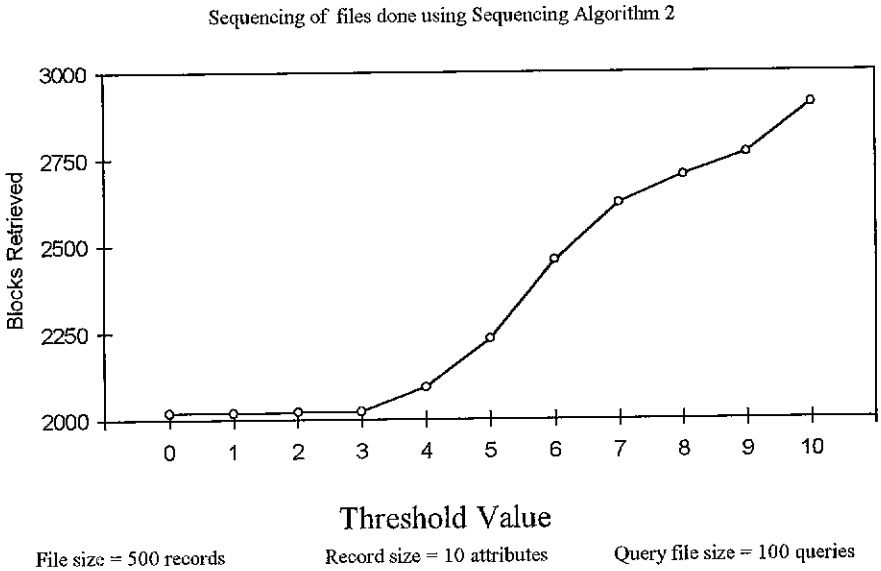
Sequencing of files done using Sequencing Algorithm 2



File size = 500 records Record size = 10 attributez Query file size = 100 queries

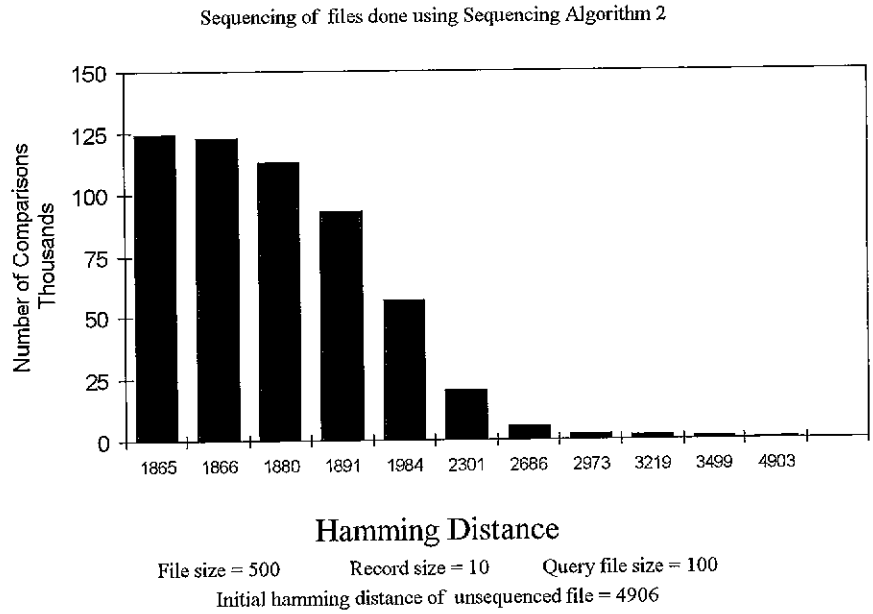
A graphical representation of threshold value against the total number of blocks retrieved is as shown below in Figure 2.3.

Figure 2.3: Threshold Value Vs. Blocks Retrieved



A graphical representation of hamming distance against the number of comparisons is as shown below in Figure 2.4.

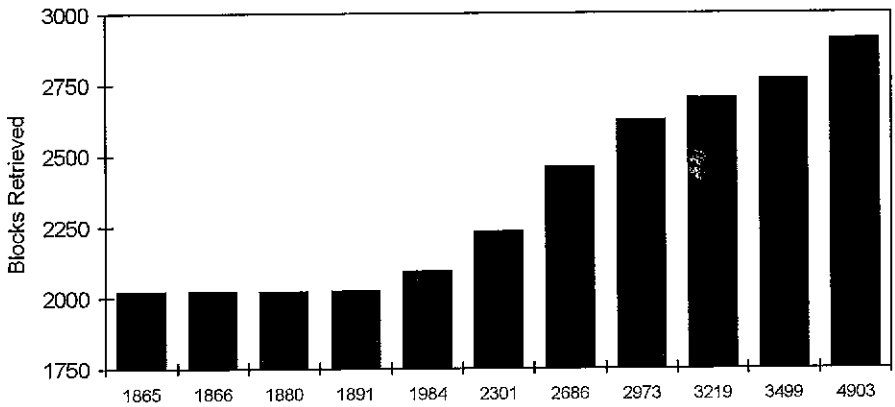
Figure 2.4: Hamming Distance Vs. Comparisons



A graphical representation of hamming distance against the total number blocks retrieved is as shown below in Figure 2.5.

Figure 2.5: Hamming Distance Vs. Blocks Retrieved

Sequencing of files done using Sequencing Algorithm 2



Hamming Distance

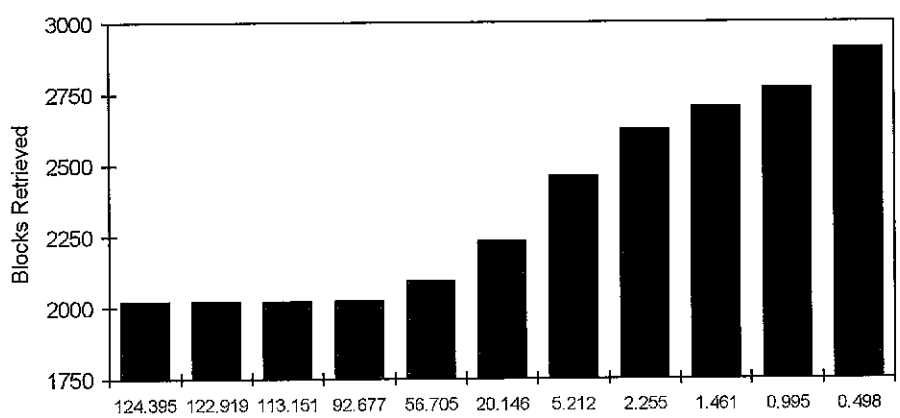
File size = 500 records Record size = 10 attributes Query file size = 100 queries

Initial hamming distance of unsequenced file = 4906

A graphical representation of comparisons against the total number blocks retrieved is as shown below in Figure 2.6.

Figure 2.6: Comparisons Vs. Blocks Retrieved

Sequencing of files done using Sequencing Algorithm 2



Comparisons

File size = 500 records Record size = 10 attributes Query file size = 100 queries

Initial hamming distance of unsequenced file = 4906

The random file "5h_10_4.R" is also sequenced using sequencing sequencing algorithm 1 and sequencing algorithm 2 with threshold values 0 to 10. MATCHB.C was run against the resultant files using a query file of 2 terms per record and a block size of 5 records. The results are tabulated below in Table 3.

File Name	Threshold	Hamming Distance	Average Hamming Distance	Elapsed Time (sec)	Comparisons	Blocks Retrieved
5h_10_4.RS1	-	1845	4	2.0	125247	2027
5H_10_4.R20	0	1845	4	2.58	124216	2027
5H_10_4.R21	1	1845	4	2.52	121491	2027
5H_10_4.R22	2	1853	4	2.30	112684	2037
5H_10_4.R23	3	1878	4	1.86	90514	2041
5H_10_4.R24	4	2002	4	1.15	54515	2105
5H_10_4.R25	5	2310	5	0.43	21185	2325
5H_10_4.R26	6	2701	5	0.16	5837	2478
5H_10_4.R27	7	3096	6	0.05	1897	2682
5H_10_4.R28	8	3451	7	0.05	852	2775
5H_10_4.R29	9	3710	7	0.05	558	2822
5H_10_4.R2T	10	3800	8	0.05	498	2837

Table 3

An observation that can be made from Tables 2 & 3 is that irrespective of the total hamming distance of original files a specified threshold value below the average threshold value of the files (the number of attributes per record / 2) in question gives new total hamming distances with small differences.

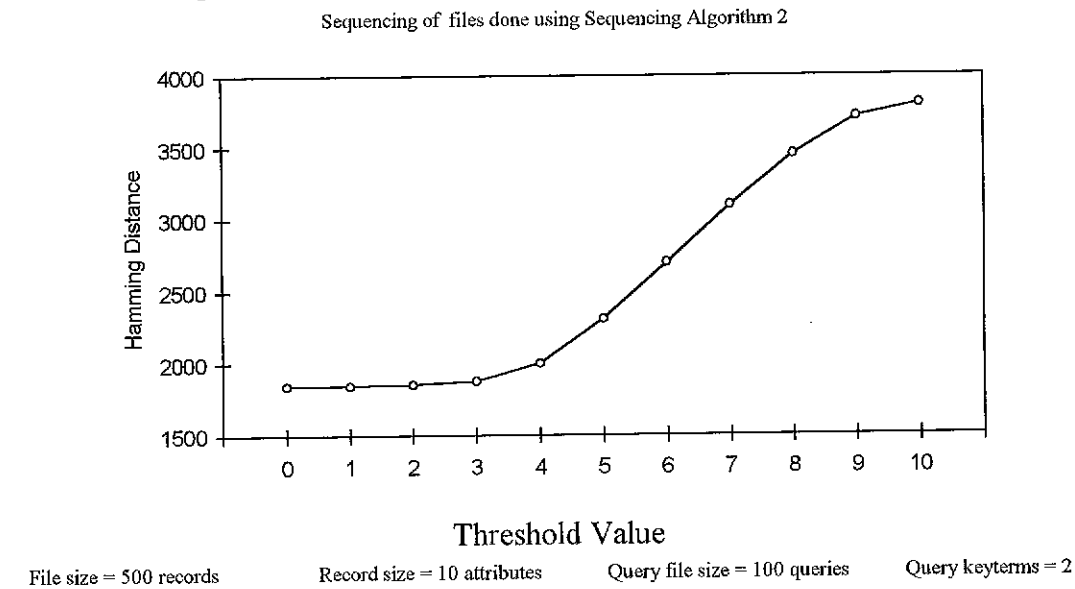
Using threshold value 2 for example in sequencing "5h_10_4.U" and "5h_10_4.R" gave a total hamming distance of 1857 and 1853 respectively, a difference of 4 which is very small considering that we are dealing with a file of 500 records and 10 attributes per record. For threshold value 5, the difference was 9. With threshold values greater than 5, the disparity is greater.

Another observation from Table 3 is that despite the fact that the hamming distance for "5h_10_4.R20" and "5h_10_4.R21" were the same (1845) the number of comparisons were different. Using threshold 0 required more comparisons than using threshold 1. This is basically because for incoming records that would increase the hamming distance by 1 at the best position, using threshold 1 would result in an insertion immediately once the position is found whereas with treshold 0, comparisons will continue till all positions have been tried before the incoming record is inserted at the best position encountered during the search.

Using threshold 2 produced a file with hamming distance close to that using threshold 0 and 1 (1853, a difference of 8). The difference in comparisons made between threshold 3 and threshold 0 is 33,702, which is quite large. From Table 3, if a choice is to be made for the best threshold value that reduces the hamming distance significantly at a low price (a reasonable number of comparisons), then threshold value 3 will be a likely candidate.

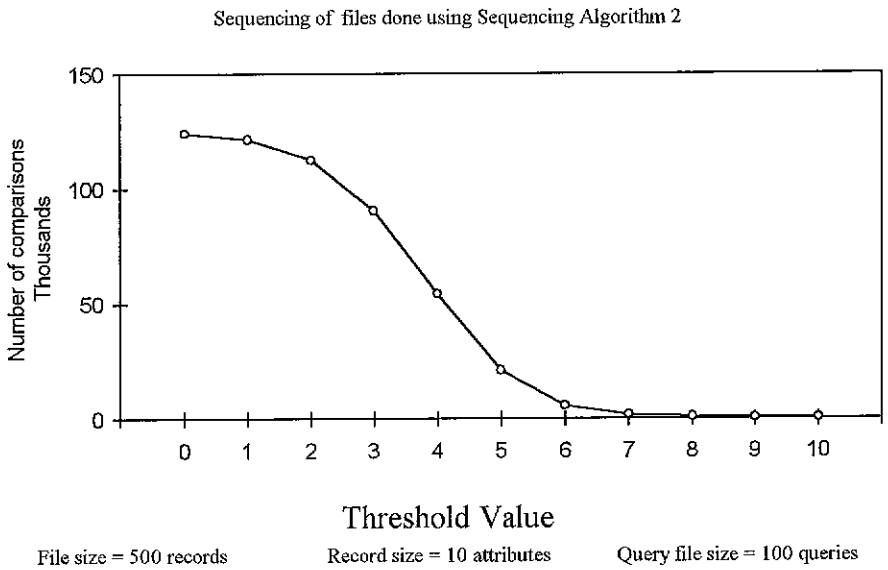
A graphical representation of threshold value against hamming distance is shown below in Figure 3.1.

Figure 3.1: Threshold Value Vs. Hamming Distance



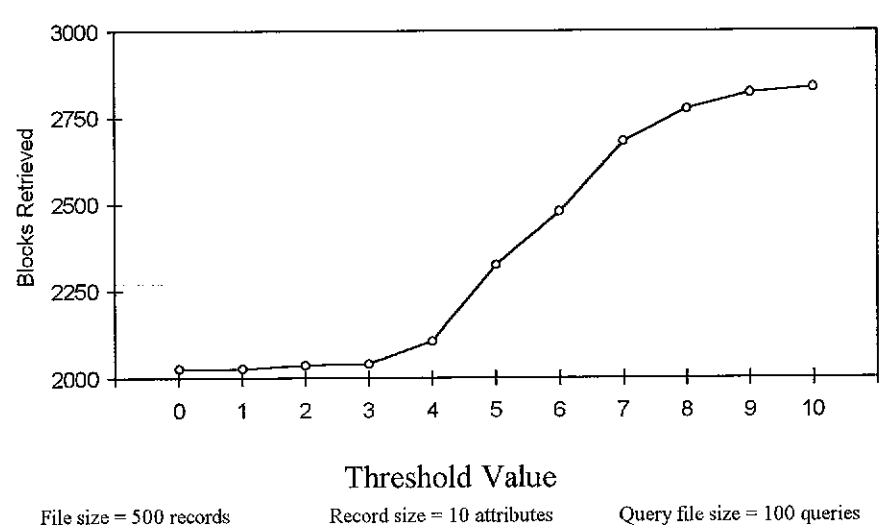
A graphical representation of threshold value against the number of comparisons is as shown below in Figure 3.2.

Figure 3.2: Threshold Value Vs. Comparisons



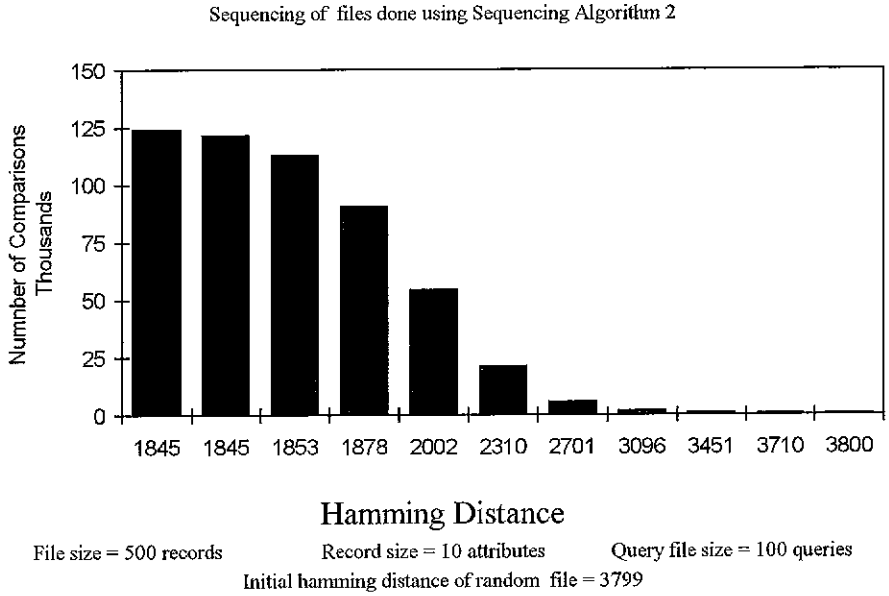
A graphical representation of threshold value against the total number of blocks retrieved is as shown below in Figure 3.3.

Figure 3.3: Threshold Value Vs. Blocks Retrieved



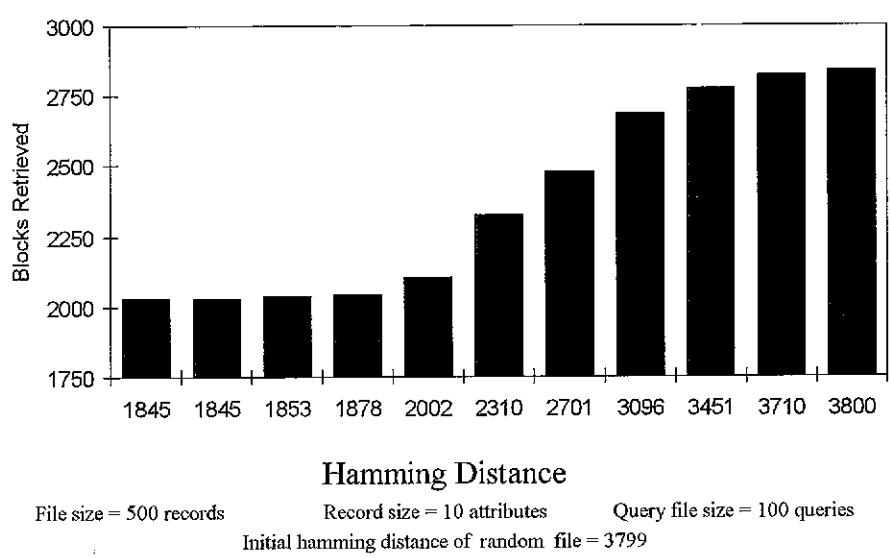
A graphical representation of hamming distance against the number of comparisons is as shown below in Figure 3.4.

Figure 3.4: Hamming Distance Vs. Comparisons



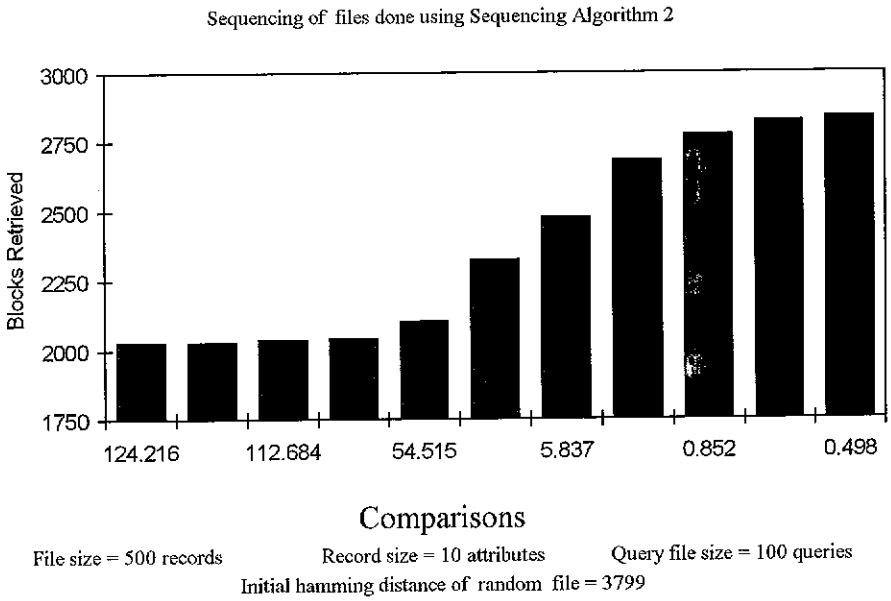
A graphical representation of hamming distance against the total number blocks retrieved is as shown below in Figure 3.5.

Figure 3.5: Hamming Distance Vs. Blocks Retrieved



A graphical representation of comparisons against the total number blocks retrieved is as shown below in Figure 3.6.

Figure 3.6: Comparisons Vs. Blocks Retrieved



4.3 Impact of Sequencing: Percentage Reduction in Blocks Retrieved

The percentage reduction in blocks retrieved is a sign of how work done in sequencing has improved the existing situation. It is the ratio of the reduction in blocks retrieved after sequencing and the blocks being retrieved before sequencing. Let B_{before} be the number of blocks retrieved before sequencing and B_{after} be the number of blocks retrieved after sequencing so that the percentage reduction could be represented as :

$$\frac{|B_{before} - B_{after}|}{B_{before}}.$$

The steps described below pertain to another simulation experiment that would help in illustrating the relationship between work done in sequencing and percentage reduction in blocks retrieved.

- FILEB.C was run to generate a random file with the following specifications :
  - File name : "T_10_4.R"
  - File size : 1000 records
  - Record size : 10 attributes
  - Attribute range : 4

- FIRST_PRIME.C was run to produce the unsequenced version of the random file.
  - File name : "T_10_4.U"
  - File size : 1000 records
  - Record size : 10 attributes
  - Attribute range : 4

- FIRST.C was run to produce the sequenced version of the random file.
  - File name : "T_10_4.RS1"
  - File size : 1000 records
  - Record size : 10 attributes
  - Attribute range : 4

- SECOND.C was run to produce the sequenced version of the random file at threshold values varying from 0 to 10 of course using sequencing algorithm 2.
  - File name : "T_10_4.R2i"
  - File size : 1000 records
  - Record size : 10 attributes
  - Attribute range : 4
  - Threshold value : *i*

- QUERY.C was run to generate a query file.
  - File name : "H_10_4.Q02"
  - File size : 100 query records

- Record size : 10 attributes
- Query Keyterms : 2
- Attribute range : 4

- MATCHB.C was run against the above mentioned data files with the query file that was just specified.
  - Bucket size : 8 records per block.

File Name	Thre-shold	Hamming Distance	Elapsed Time (sec)	Comparisons	Blocks Retrieved	Reduction in Blocks Retrieved
T_10_4.RS1	-	3316	10.05	500497	3339	0.3339
T_10_4.R20	0	3316	10.21	495048	3339	0.3339
T_10_4.R21	1	3316	9.94	482721	3339	0.3339
T_10_4.R22	2	3342	8.73	422665	3341	0.3335
T_10_4.R23	3	3392	6.42	312251	3304	0.3409
T_10_4.R24	4	3808	3.24	157608	3564	0.2890
T_10_4.R25	5	4560	0.87	45301	3993	0.2035
T_10_4.R26	6	5473	0.16	10787	4471	0.1081
T_10_4.R27	7	6257	0.05	3448	4815	0.0400
T_10_4.R28	8	6934	0	1724	4995	0.0040
T_10_4.R29	9	7371	0	1116	5011	0.0004
T_10_4.R2T	10	7508	0	998	5013	0
T_10_4.U	-	9925	10.05	500497	5107	-

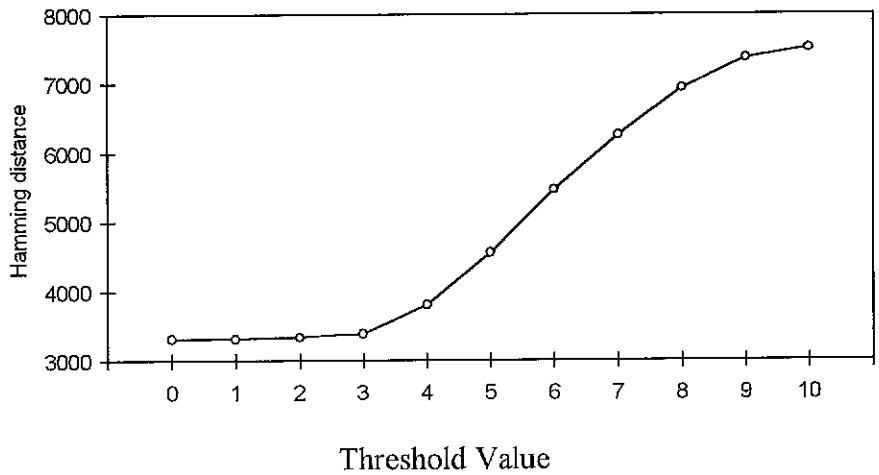
Table 4

Notice that a threshold value near 8 (the average hamming distance of the random file) produced no significant percentage reduction in blocks retrieved. Those below the average hamming distance of the original file had a significant impact on the number of blocks retrieved. It is also noticeable that the smaller the threshold value the greater the percentage reduction in blocks retrieved but at the expense of more work done during sequencing.

A graph of threshold value against hamming distance is as shown below in Figure 4.0.

Figure 4.0: Threshold Value Vs. Hamming Distance

Sequencing of files done using Sequencing Algorithm 2

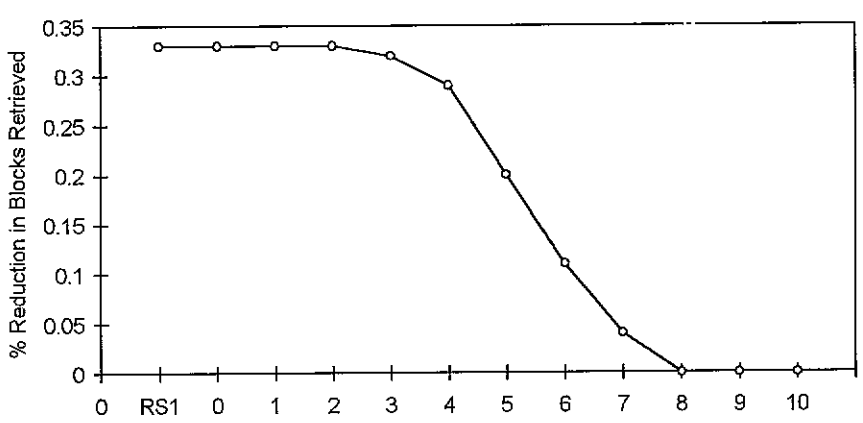


File size = 1000 records Record size = 10 attributes Query file size = 100 queries Query keyterms = 2

A graphical representation of threshold value against percentage reduction in blocks retrieved is as shown below in Figure 4.1.

Figure 4.1: Threshold Value Vs. % Reduction In Blocks Retrieved

Sequencing of files done using Sequencing Algorithm 2

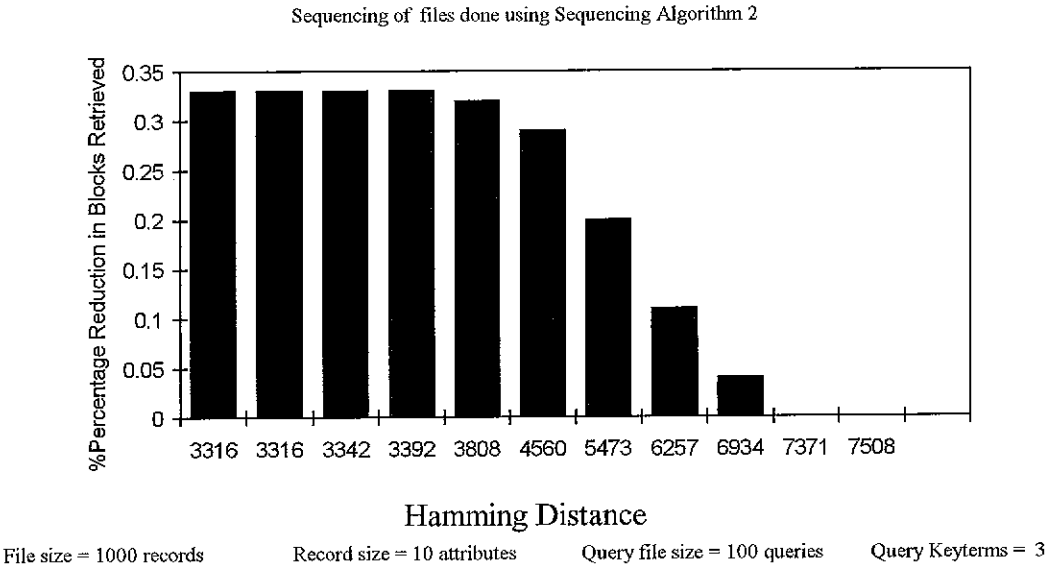


File size = 1000 records Record size = 10 attributes Query file size = 100 queries Query Keyterms = 2

RS1 represents the data file sequenced using Sequencing Algorithm 1.

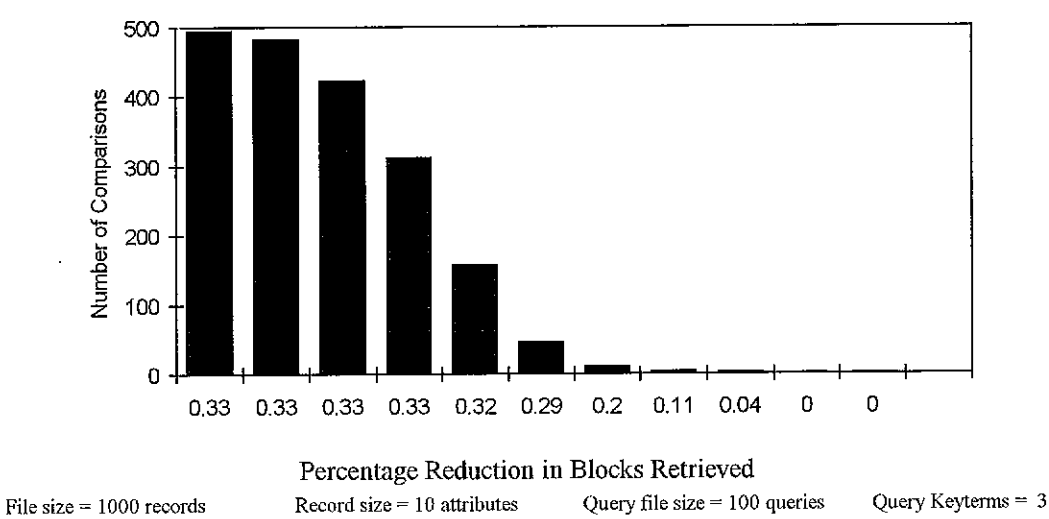
A graphical representation of hamming distance against percentage reduction in blocks retrieved is as shown below in Figure 4.2.

Figure 4.2: Hamming Distance Vs. % Reduction in Blocks Retrieved



A graphical representation of comparisons against percentage reduction in blocks retrieved is as shown below in Figure 4.3.

Figure 4.3: % Reduction In Blocks Reduction Vs. Comparisons



The steps described below pertain to a new simulation experiment that would further help in illustrating the relationship between work done in sequencing and percentage reduction in blocks retrieved.

- FILEB.C was run to generate a random file with the following specifications :
  - File name : "T_10_4.R"
  - File size : 1000 records
  - Record size : 10 attributes
  - Attribute range : 4
  - Hamming Distance : 7508
- FIRST_PRIME.C was run to produce the unsequenced version of the random file.
  - File name : "T_10_4.U"
  - File size : 1000 records
  - Record size : 10 attributes
  - Attribute range : 4
  - Hamming Distance : 9925
- FIRST.C was run to produce the sequenced version of the unsequenced file.
  - File name : "T_10_4.US1"
  - File size : 1000 records
  - Record size : 10 attributes
  - Attribute range : 4
  - Hamming Distance : 3275
- SECOND.C was run to produce the sequenced version of the unsequenced file at threshold values varying from 0 to 10 of course using sequencing algorithm 2.
  - File name : "T_10_4.U2i"
  - File size : 1000 records
  - Record size : 10 attributes
  - Attribute range : 4
  - Threshold value : *i*
- QUERY.C was run to generate a query file.

- File name : "H_10_4.Q02"
- File size : 100 query records
- Record size : 10 attributes
- Query Keyterms : 2
- Attribute range : 4

- MATCHB.C was run against the above mentioned data files with the query file that was just specified.
  - Bucket size : 8 records per block.

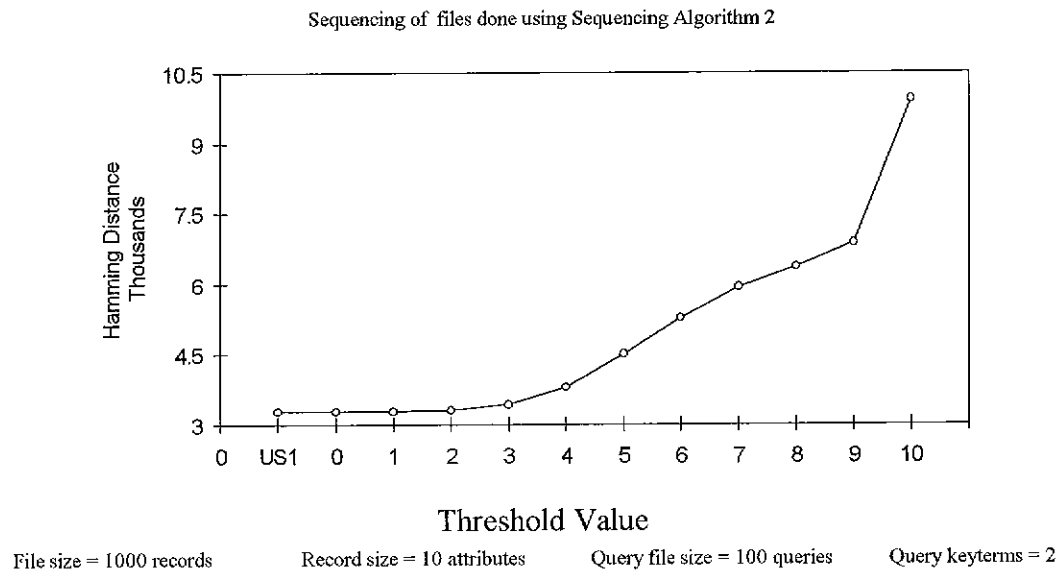
File Name	Thre-shold	Hamming Distance	Elapsed Time (sec)	Comparisons	Blocks Retrieved	Reduction in Blocks Retrieved
T_10_4.US1	-	3275	10.10	500497	3292	0.3554
T_10_4.U20	0	3275	10.10	491904	3292	0.3554
T_10_4.U21	1	3278	9.83	476627	3291	0.3556
T_10_4.U22	2	3308	8.73	422488	3293	0.3552
T_10_4.U23	3	3421	6.48	314650	3334	0.3472
T_10_4.U24	4	3803	3.29	160245	3575	0.3
T_10_4.U25	5	4511	0.87	45321	3984	0.2199
T_10_4.U26	6	5295	0.21	10478	4383	0.1418
T_10_4.U27	7	5927	0.05	4320	4696	0.0805
T_10_4.U28	8	6374	0.05	2838	4861	0.0482
T_10_4.U29	9	6875	0	2172	4928	0.0350
T_10_4.U2T	10	9922	0	998	5107	0
T_10_4.U	-	9925	10.10	500497	5107	0

Table 5

Note that in unsequencing "T_10_4.R" to obtain "T_10_4.U" and in sequencing it using FIRST.C to obtain "T_10_4.US1", the number of comparisons made was 500497.

A figure of threshold value against the number of comparisons is as shown below in Figure 5.0.

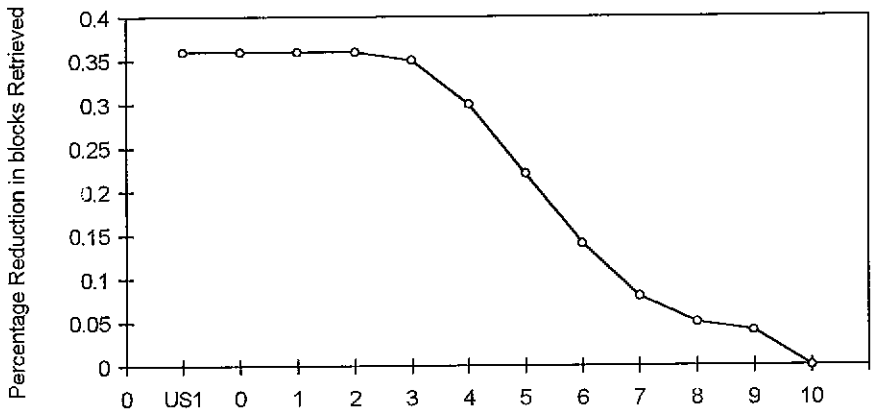
Figure 5.0: Threshold Value Vs. Hamming Distance



A graphical representation of threshold value against percentage reduction in blocks retrieved is as shown below in Figure 5.1.

Figure 5.1: Threshold Value Vs. % Reduction In Blocks Retrieved

Sequencing of files done using Sequencing Algorithm 2



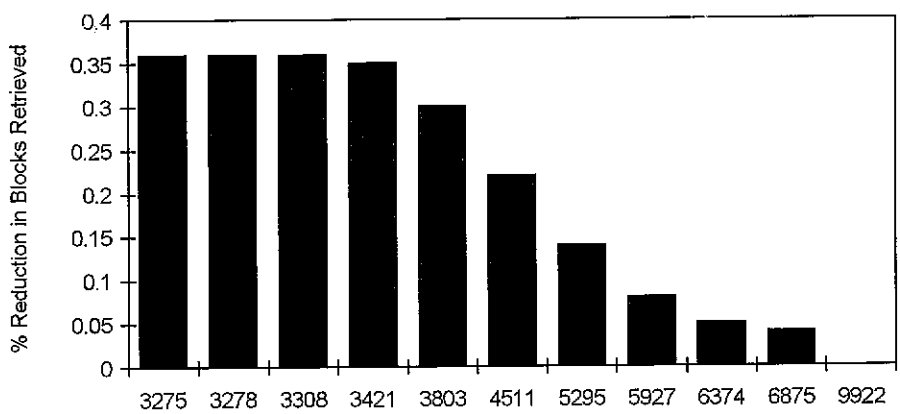
File size = 1000 records Record size = 10 attributes Query file size = 100 queries Query Keyterms = 2

US1 represents the data file sequenced using Sequencing Algorithm 1.

A graphical representation of hamming distance against percentage reduction in blocks retrieved is as shown below in Figure 5.2.

Figure 5.2: Hamming Distance Vs. % Reduction In Blocks Retrieved

Sequencing of files done using Sequencing Algorithm 2

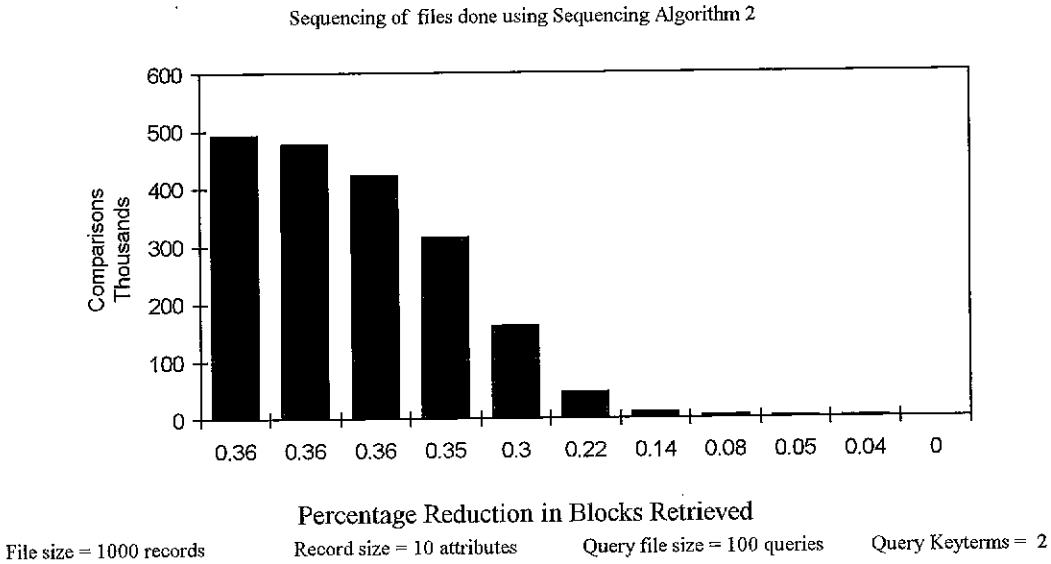


File size = 1000 records Record size = 10 attributes Query file size = 100 queries

Query Keyterms = 2 Block size = 8

A graphical representation of comparisons against percentage reduction in blocks retrieved is as shown below in Figure 5.3.

Figure 5.3: % Reduction In Blocks Reduction Vs. Comparisons



4.4 Impact of Number of Terms in a Query on Percentage Reduction in Blocks Retrieved

In this section, it is important to keep in mind that query keyterms are ANDed together. The number of query keyterms as well as bucket size do have an impact on the percentage reduction in blocks retrieved. Studying the correlation between these factors helps in the maximisation of the percentage sought.
For this purpose, the following experiment was carried out.

- FILEB.C was run to generate a random file with the following specifications :
  - File name : "T_10_4.R"
  - File size : 1000 records
  - Record size : 10 attributes
  - Attribute range : 4
  - Hamming Distance : 7508

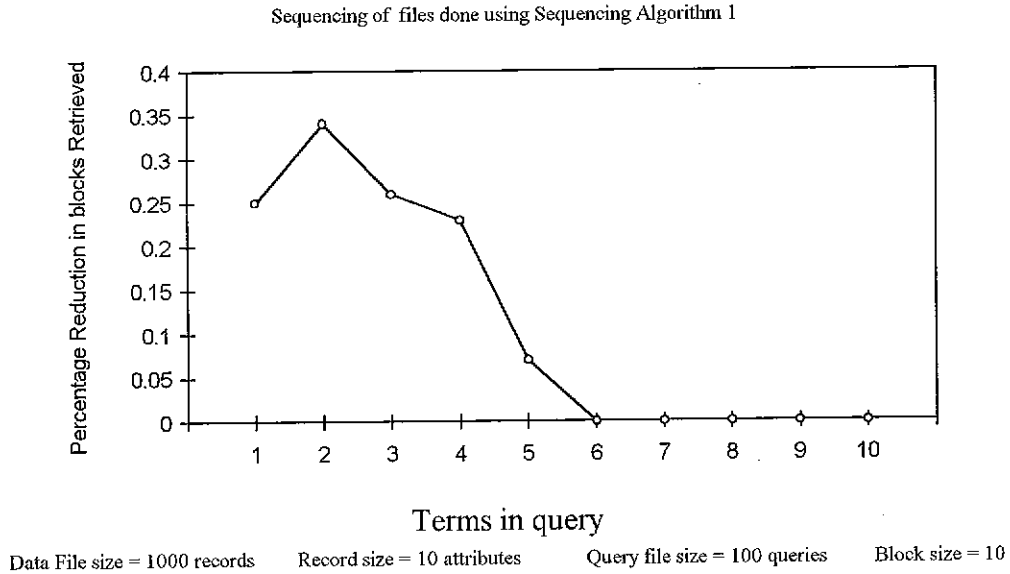
- FIRST.C was run to produce the sequenced version of the unsequenced file.
  - File name : "T_10_4.RS1"
  - File size : 1000 records
  - Record size : 10 attributes
  - Attribute range : 4
  - Hamming Distance : 3316
- QUERY.C was run 10 times to generate 10 query files.
  - File name : H_10_4.Q*i*
  - File size : 100 query records
  - Record size : 10 attributes
  - Query Keyterms : *i* where *i* varies from 1 to 10
  - Attribute range : 4
- MATCHB.C was run twice once with T_10_4.R and another time with T_10_4.RS1.
  - Bucket size : 8 records per block.
  - Query file : H_10_4.Q*i*

Name of Query File	Terms In Query	No. of Records Retrieved	Blocks Retrieved for T_10_4.R	Blocks Retrieved for T_10_4.RS1	Reduction in Blocks Retrieved
H_10_4.Q01	1	24998	9520	7166	0.25
H_10_4.Q02	2	6187	4748	3128	0.34
H_10_4.Q03	3	1604	1485	1095	0.26
H_10_4.Q04	4	376	370	284	0.23
H_10_4.Q05	5	94	94	87	0.07
H_10_4.Q06	6	25	25	25	0
H_10_4.Q07	7	7	7	7	0
H_10_4.Q08	8	4	4	4	0
H_10_4.Q09	9	0	0	0	0
H_10_4.Q10	10	1	1	1	0

Table 6

It is observed from Table 6 that the greatest percentage reduction in blocks retrieved is yielded when the number of query terms is two. When the number of query terms is smaller, say one, a larger number of records is retrieved and therefore whether the records are sequenced or not a greater number of records are retrieved therefore the arrangement of the records within the blocks does not matter as much as in the case when the number of query terms is for instance 2 When the number of terms in a query is larger (closer to the number of keyterms in the data file) less records are retrieved, 1 record could be read for each query, and the record read will occupy a single block. As a result, the number of blocks retrieved in this case would be the same as the number of records read. For example from Table 6 the query set of 100 queries of 6 keyterms retrieved the same number of blocks for both the random and the sequenced files.
A Figure of percentage reduction in blocks retrieved against number of query keyterms is as shown below in Figure 6.

Figure 5.1: Terms in Query Vs. % Reduction in Blocks Retrieved



Consider the following experiment which also illustrates the effect of number of terms in a query on the percentage reduction in blocks retrieved.

- FILEB.C was run to generate a random file with the following specifications :

- File name : "T_5_4.R"
- File size : 1000 records
- Record size : 5 attributes
- Attribute range : 4
- Hamming Distance : 3783

- FILEB.C was run a second time to generate a random file with the following specifications :
  - File name : "T_15_4.R"
  - File size : 1000 records
  - Record size : 15 attributes
  - Attribute range : 4
  - Hamming Distance : 11259
- FIRST.C was run to produce the sequenced version of the first random file.
  - File name : "T_5_4.RS1"
  - File size : 1000 records
  - Record size : 5 attributes
  - Attribute range : 4
  - Hamming Distance : 725
- FIRST.C was run a second time to produce the sequenced version of the second random file.
  - File name : "T_15_4.RS1"
  - File size : 1000 records
  - Record size : 15 attributes
  - Attribute range : 4
  - Hamming Distance : 6231
- QUERY.C was first run 5 times to generate 5 query files.
  - File name : H_5_4.Q*i*
  - File size : 100 query records
  - Record size : 5 attributes
  - Query Keyterms : *i* where *i* varies from 1 to 10
  - Attribute range : 4

- QUERY.C was second run 6 times to generate 6 query files.
  - File name : H_15_4.Q*i*
  - File size : 100 query records
  - Record size : 15 attributes
  - Query Keyterms : *i* where *i* varies from 1 to 6.
  - Attribute range : 4
- MATCHB.C was run 10 times with T_5_4.R and T_5_4.RS1.
  - Bucket size : 10 records per block.
  - Query file : H_5_4.Q*i*
- MATCHB.C was next run 24 times with T_15_4.R and T_15_4.RS1.
  - Bucket size : 10 records per block.
  - Query file : H_5_4.Q*i*

The resultant data is as tabulated below in Table 7 and Table 8.

Name of Query File	Terms In Query	No. of Records Retrieved	Blocks Retrieved for T_5_4.R	Blocks Retrieved for T_5_4.RS1	Reduction in Blocks Retrieved after sequencing
H_5_4.Q01	1	25283	9417	5314	0.44
H_5_4.Q02	2	6333	4783	2053	0.57
H_5_4.Q03	3	1537	1449	702	0.52
H_5_4.Q04	4	401	398	233	0.42
H_5_4.Q05	5	100	98	70	0.29

Table 7

Name of Query File	Terms In Query	No. of Records Retrieved	Blocks Retrieved for T_15_4.R	Blocks Retrieved for T_15_4.RS1	Reduction in Blocks Retrieved after sequencing
H_15_4.Q01	1	24955	9416	7722	0.18
H_15_4.Q02	2	6326	4799	3562	0.26

H_15_4.Q03	3	1591	1495	1212	0.19
H_15_4.Q04	4	394	384	349	0.09
H_15_4.Q05	5	74	74	72	0.03
H_15_4.Q06	6	22	22	22	0

Table 8

A graphical representation of the data recorded above is as shown below in Figure 7 and 8.

Figure 7: Terms In Query Vs. % Reduction In Blocks Retrieved

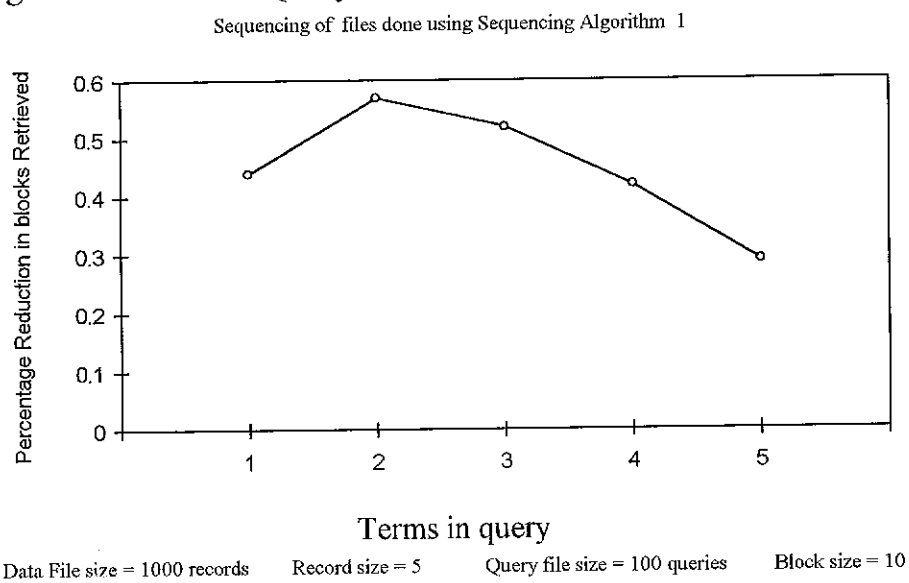
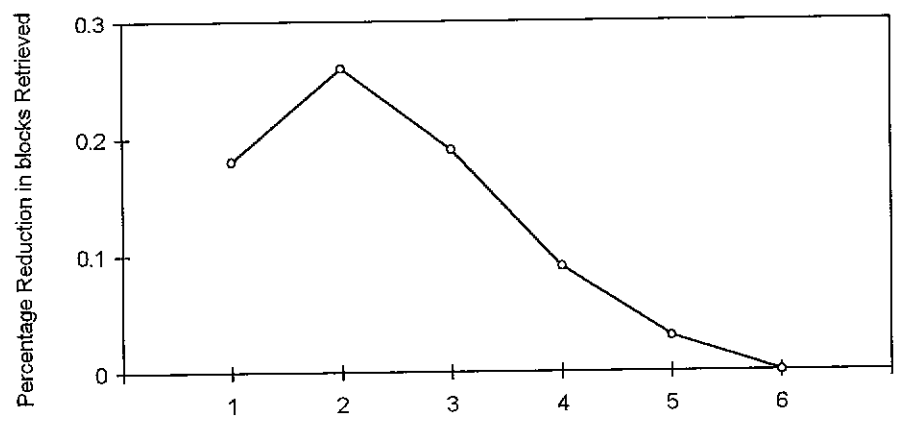


Figure 8: Terms In Query Vs. % Reduction In Blocks Retrieved



Terms in query

Data File size = 1000 records Record size = 5 Query file size = 100 queries Block size = 10

Notice that for a record size of 5, the greatest percentage reduction in blocks retrieved was obtained when the number of query terms is 2. In fact, we had a percentage reduction of 0.57. for a record size of 15, the greatest percentage reduction in blocks retrieved was also yielded with 2 as the number of query terms. a table of attributes in a record and terms giving the greatest reduction in blocks retrieved is shown below in Table 9. Ratio indicates the ratio of the values in the corresponding first 2 columns of the table.

Attributes in record	Terms in query yielding greatest percentage in blocks retrieved	RATIO
5	2	2/5 = 0.4
10	2	2/10 = 0.2
15	2	2/15 = 0.13

Table 9

It is observed from the Table 9 that between 0.13 and 0.4 of the number of keyterms in a record is the best value for the terms in a query in order to maximise the percentage reduction in blocks retrieved after sequencing.

4.5 Effect of Block Size on Percentage Reduction in Blocks Retrieved

- MATCHB.C was run 140 times with T_10_4.R and T_10_4.RS1.
  - Bucket size : *i* ,where *i* varies between 1 and 1000 records per block at varying intervals.
  - Query file : H_5_4.Q02

The resultant data is as tabulated below in Table 10, 11, 12, and 13.

Block Size	Blocks Retrieved for T_10_4.R	Blocks Retrieved for T_10_4.RS1	Reduction in Blocks Retrieved
1	6187	6187	0
2	6006	4872	0.1888
3	5830	4326	0.2580
4	5664	4009	0.2922
5	5491	3787	0.3103
6	5311	3584	0.3252
7	5181	3487	0.3270
8	5013	3339	0.334
9	4872	3264	0.33
10	4748	3128	0.3412
11	4613	3077	0.333
12	4472	2983	0.333
13	4338	2919	0.3271
14	4251	2871	0.3246
15	4165	2808	0.3258
16	4041	2737	0.3227
17	3904	2705	0.3071
18	3790	2628	0.3066
19	3697	2601	0.2965
20	3619	2550	0.2954
21	3538	2504	0.2923
22	3457	2472	0.2850
23	3386	2405	0.2897
24	3269	2366	0.2762
25	3197	2361	0.2615
26	3124	2303	0.2628

27	3054	2286	0.2515
28	2993	2253	0.2472
29	2923	2205	0.2456
30	2882	2144	0.2561

Table 10

Block Size	Blocks Retrieved for T_10_4.R	Blocks Retrieved for T_10_4.RS1	Reduction in Blocks Retrieved
5	5491	3787	0.3103
10	4748	3128	0.3412
15	4165	2808	0.3258
20	3619	2550	0.2954
25	3197	2361	0.2615
30	2882	2444	0.2561
35	2558	2026	0.2080
40	2294	1885	0.1783
45	2124	1772	0.1657
50	1910	1653	0.1346
55	1793	1550	0.1355
60	1656	1442	0.1292
65	1551	1387	0.1057
70	1452	1339	0.0778
75	1368	1271	0.0709
80	1283	1197	0.0670
85	1195	1133	0.0519
90	1150	1085	0.0565
95	1093	1055	0.0350
100	998	970	0.0281

Table 11

Block Size	Blocks Retrieved for T_10_4.R	Blocks Retrieved for T_10_4.RS1	Reduction in Blocks Retrieved
50	1910	1653	0.1346
100	998	970	0.0281
150	700	697	0.0043
200	500	499	0.002
250	400	400	0
300	400	399	0.0025
350	300	300	0
400	300	300	0
450	300	299	0.0033
500	200	200	0

Table 12

Block Size	Blocks Retrieved for T_10_4.R	Blocks Retrieved for T_10_4.RS1	Reduction in Blocks Retrieved
100	998	970	0.0281
200	500	499	0.002
300	400	399	0.0025
400	300	300	0
500	200	200	0
600	200	200	0
700	200	200	0
800	200	200	0
900	200	199	0.005
1000	100	100	0

Table 13

Chapter 5

Statistical Analysis

This chapter aims at the derivation of statistical formulae that can predict the numerical results obtained in Chapter 4. Comparisons of some of the results obtained statistically and those obtained by experimental simulation are made.

5.1 Statistical Evaluation of Hamming Distance *before* Sequencing

In the following statistical evaluation we assume that the records are initially randomly distributed within the file. Let:

- r denote the number of attributes in a data record.
- x denote the number of common attributes among 2 consecutive records.

Then the hamming distance is (r - x) denoted by hd.

In other words, let the first record selected be say, A and the second record B. If x is the number of similar keyterms between A and B then the hamming distance is (r - x). In order to obtain the expected hamming distance we sum over all possible values of x, the product of the hamming distance (r - x) and its probability of occurring P(x). The possible values of x are among 0 and r inclusive. The 2 extremes being the following:

- x = 0, when A and B are completely different and do not have any keyterm in common
- x = r, when A and B are identical.

Let U represent the hamming distance, that is U = r-x. Therefore the expected hamming distance can be obtained as

$$E(hd) = E(U) = \sum_{U=0}^{r} U \times P(U).$$

Now, P(U) or P(r - x) is the probability of obtaining a hamming distance of (r-x) which is the same as the probability of having x terms or (r-U) terms in common between records A and B. As a result,

$$E(hd) = \sum_{x=0}^{r} (r-x) \times P(x). \quad \textbf{(1)}$$

5.1.1 Finding the Probability of X Attributes in Common

The problem of having x attributes in B that match the corresponding attributes in A is a *binomial experiment*. The following 4 conditions being satisfied verify that the experiment is indeed binomial:

1) The experiment consists of r trials, r being the number of attributes in a data record that is fixed in advance of the experiment.
2) Trials are identical and each trial outcome is either a success or a failure.
3) Trials are independent.
4) The success probability denoted by p is constant from trial to trial. Furthermore p is the probability that attribute j of record B matches the corresponding attribute of record A. In other words p is the probability that attribute j between consecutive records is a match. So $p = \frac{1}{d}$ where d is the attribute range.

Therefore x is a *binomial random variable* based on r trials with success probability p [Devore, 1991].

$X \sim Bin(r, p)$.

The *probability mass function* (p.m.f) of X denoted by $b(x; r, p)$ is then

$$b(x; r, p) = \begin{cases} {}^{r}C_{x} \times p^{x} \times (1-p)^{r-x} & where \quad x = 0,1,2,..,r \\ 0 & otherwise \end{cases}$$

Plugging the formula obtained for the probability of x attributes in common into eqn (1) we obtain formula (2) below.

$$E(hd) = \sum_{x=0}^{r} (r-x) \times {}^{r}C_x \times p^{x} \times (1-p)^{r-x} \text{ where } p = \frac{1}{d}. \quad \textbf{(2)}$$

If there are N records in the file then each incoming record will have the same expected hamming distance between itself and the records it is placed next to. By summing the (N - 1) expected hamming distances ((N - 1) hamming distances are computed for a file of N records) we obtain the total hamming distance. The expected total hamming distance can be estimated as

$$E(total\_hd) = (N-1) \sum_{x=0}^{r} (r-x) \times P(x). \quad \textbf{(3)}$$

where as defined in (2), $P(x) = {}^{r}C_x \times p^{x} \times (1-p)^{r-x}$ where $p = \frac{1}{d}$.

Example 1:

Parameter	*Abbreviation*	*Value*
Attribute Range	d	5
Record Size	r	2
File Size	N	10

thus, $p = \frac{1}{d} = \frac{1}{5} = 0.2.$

The expected hamming distance is found using eqn (2) as

$$E(hd) = \sum_{x=0}^{2} (2-x) \times P(x) = 2P(0) + P(1).$$

$P(0) = {}^{2}C_0 \times (0.2)^{0} \times (1 - 0.2)^{2} = 0.64$

$P(1) = {}^{2}C_1 \times (0.2)^{1} \times (0.8)^{1} = 0.64$

$E(hd) = 2 \times 0.64 + 0.32 = 1.6 \cong 2.$

$E(hd) = (N-1) \times E(hd) = 9 \times 1.6 = 14.4 \cong 14.$

This is to say that on the average the records, made up of 2 keyterms, have (x=0) keyterms in common and hence a hamming distance of (r-x)= (2-0)=2 between them.

5.2 Statistical Evaluation of Hamming Distance *after* Sequencing

This section deals with estimates of hamming distance after the sequencing of a random file using sequencing algorithm 2. Using a threshold value of k, the possible hamming distance between adjacent records after sequencing is less than or equal to k . Therefore the hamming distance could be 0, 1, 2,..., or k.
Eqn. 2 is still valid if we only modify the way p is computed. In fact p is the probability that attribute j is a match between 2 consecutive records. Now if sequencing algorithm 2 hopefully finds an insertion position for the record at hand (B) where the increase in hamming distance is less than or equal to the threshold value specified (k), then the probability that attribute j of B matches the corresponding attribute of A which was 1/d (d being the attribute range) before sequencing becomes $\frac{r-k}{r}$. This is because the number of terms not in common between A and B is maximum K, making the number of terms in common between them minimum (r - k). The above explanation allows us to modify eqn 2 to become

$$E(hd) = \sum_{x=0}^{r} (r-x) \times {}^{r}C_{x} \times p^{x} \times (1-p)^{r-x} \text{ where } p = \frac{r-k}{r} \qquad \textbf{(4)}$$

Note that eqn. (3) for the expected value for the total hamming distance is still valid.

Example 2:

Parameter	*Abbreviation*	*Value*
Threshold Value	k	2
Record Size	r	5
File Size	N	500

thus, $p = \frac{r-k}{r} = \frac{5-2}{3} = 0.6.$

The expected hamming distance is found using eqn (4) as

$$E(hd) = \sum_{x=0}^{5} (5-x) \times P(x) = 5P(0) + 4P(1) + 3P(2) + 2P(3) + P(4)$$

$P(0) = {}^{5}C_{0} \times (0.6)^{0} \times (1 - 0.6)^{5} = 0.01$

$P(1) = {}^{5}C_{1} \times (0.6)^{1} \times (0.4)^{4} = 0.077$

$P(2) = {}^5C_2 \times (0.6)^2 \times (0.4)^3 = 0.23$

$P(3) = {}^5C_3 \times (0.6)^3 \times (0.4)^2 = 0.346$

$P(4) = {}^5C_4 \times (0.6)^4 \times (0.4)^1 = 0.259$

$E(hd) = 5 \times 0.01 + 4 \times 0.077 + 3 \times 0.23 + 2 \times 0.346 + 0.259 = 1.999 \cong 2.$

$E(total\_hd) = (N-1) \times E(hd) = 499 \times 1.999 = 997.5 \cong 998.$

5.3 Statistical Evaluation of Percentage Reduction in Blocks Retrieved After Sequencing

Let p be the probability that a record satisfies a query. That is (1 - p) is the probability that a record does not satisfy a query. If there are N records in the file, the probability that Y of these records satisfy the query follows a binomial distribution. The probability that Y records out of the N records in the file satisfy the query is given as ${}^N C_Y \times p^Y \times (1-p)^{N-Y}$ **(5)**

The reasons that justify why the above experiment is binomial are

1) The experiment consists of N trials, N being the file size is fixed in advance of the experiment.
2) Trials are identical and each trial outcome is either a success or a failure. Success means the record satisfies the query.
3) Trials are independent.
4) The success probability denoted by p is constant from trial to trial. Furthermore p is the probability that a record satisfies a query.

5.3.1 Finding The Probability That A Record Satisfies A Query

Let:

- q be the query size
- x be the number of data attributes that match the query record at hand

- r be the record size (of course $q \leq r$).

Let p' be the probability that a single data attribute matches its corresponding query attribute. If the attribute range is d then $p' = \frac{1}{d}$.

If there are q attributes in the query record the probability that x data attributes match these q attributes follows a binomial distribution. The probability that x attributes out of the q are a match is given as

$$^{q}C_{x} \times p'^{q} \times (1 - p')^{q-x} \quad where \quad p' = \frac{1}{d} \tag{6}$$

For a data record to satisfy a query record all attributes in the query record should match the attributes in the data record. That is to say x should be equal to q. Therefore eqn. (6) becomes

$$^{q}C_{q} \times p'^{q} \times (1 - p')^{q-q} = p'^{q} \quad where \quad p' = \frac{1}{d} \tag{7}$$

Estimate of the expected number of records to be retrieved is therefore

$$\sum_{Y=0}^{N} Y \times (^{N}C_{Y} \times p^{Y} \times (1-p)^{N-Y}) \tag{8}$$

5.3.2 Estimation of the Number of Blocks Retrieved in the case of a *Random* File

If Y is the number of records retrieved, for a completely random file these records will be randomly distributed in the file. We would therefore expect the records to be in 1, 2, 3,..., or Y blocks. If we assume each of these possibilities has an equal probability of occurring, then each has a probability of $\frac{1}{Y}$ of occurring.

The estimate of the blocks retrieved is the sum of the product of the possible number of blocks retrieved and their probabilities. This gives the estimate as

$$(1\times\frac{1}{Y})+(2\times\frac{1}{Y})+(3\times\frac{1}{Y})+...+(Y\times\frac{1}{Y})$$
$$=\frac{(1+2+3+...+Y)}{Y}=\frac{Y\times(Y+1)}{2Y}=\frac{Y+1}{2}.$$

5.3.3 Estimation of the Number of Blocks Retrieved in the case of a *Sequenced* File

If the file is sequenced, it is expected that all the records to be retrieved will be placed consecutively. The number of blocks retrieved will therefore depend on the value of Y and the block size (B).

- If Y < B then the expected number of blocks retrieved = 1
- If Y > B then the expected number of blocks retrieved $=\left\lceil\frac{Y}{B}\right\rceil$.

5.3.4 Percentage Reduction in Blocks Retrieved

Let:

- B_{before} be the number of blocks retrieved before sequencing
- B_{after} be the number of blocks retrieved after sequencing
- Y be the number of records to be retrieved
- B be the block size

Then the percentage reduction in blocks retrieved would be obtained

$$\frac{B_{before}-B_{after}}{B_{before}}=\frac{\frac{Y+1}{2}-\frac{Y}{B}}{\frac{Y+1}{2}}. \quad \textbf{(9)}$$

In general and summing over all possible values of Y the expected reduction in the number of blocks retrieved can be obtained as

$$\sum_{Y=0}^{N}\frac{\frac{Y+1}{2}-\frac{Y}{B}}{\frac{Y+1}{2}}\times({}^{N}C_{Y}\times p^{Y}\times(1-p)^{N-Y}) \quad \textbf{(10)}$$

Example 3 :

Parameter	*Abbreviation*	*Value*
Attribute Range	d	4
Record Size	r	5
File Size	N	10
Block Size	B	2
Query Record size	q	2

The number of records expected to be retrieved using eqn.(8) is obtained

$$\sum_{Y=0}^{10} Y \times ({}^{10}C_Y \times p^Y \times (1-p)^{10-Y})$$

$$= 0 \times ({}^{10}C_0 \times p^0 \times (1-p)^{10}) + 1 \times ({}^{10}C_1 \times p^1 \times (1-p)^9) + \ldots + 10 \times ({}^{10}C_{10} \times p^{10} \times (1-p)^0.$$

Now p is the probability that a record of size r is retrieved by a query of size q, which is given by eqn.(7) as

$$p = p'^q \quad where \quad p' = \frac{1}{d}.$$

$$p = \left(\frac{1}{4}\right)^2 = 0.06.$$

$p = 0.06$
$p^2 = 3.6E\text{-}3$
$p^3 = 2.16E\text{-}4$
$p^4 = 1.296E\text{-}5$
$p^5 = 7.776E\text{-}7$
$p^6 = 4.6656E\text{-}8$
$p^7 = 2.79936E\text{-}9$
$p^8 = 1.679616E\text{-}10$
$p^9 = 1.00777E\text{-}11$
$p^{10} = 6.04662E\text{-}13$

$(1 - p) = 0.94$
$(1 - p)^2 = 0.8836$
$(1 - p)^3 = 0.830584$
$(1 - p)^4 = 0.780749$
$(1 - p)^5 = 0.733904$
$(1 - p)^6 = 0.689870$
$(1 - p)^7 = 1.648478$
$(1 - p)^8 = 0.609569$
$(1 - p)^9 = 0.572995$
$(1 - p)^{10} = 0.538615$

$${}^{10}C_0 \times p^0 \times (1 - p)^{10} = 0.538615$$
$${}^{10}C_1 \times p^1 \times (1 - p)^9 = 0.343797$$
$${}^{10}C_2 \times p^2 \times (1 - p)^8 = 0.098750$$

$^{10}C_3 \times p^3 \times (1-p)^7 = 0.016809$
$^{10}C_4 \times p^4 \times (1-p)^6 = 1.87755E\text{-}3$
$^{10}C_5 \times p^5 \times (1-p)^5 = 1.438123E\text{-}4$
$^{10}C_6 \times p^6 \times (1-p)^4 = 9.79776E\text{-}6$
$^{10}C_7 \times p^7 \times (1-p)^3 = 3.359232E\text{-}7$
$^{10}C_8 \times p^8 \times (1-p)^2 = 7.558272E\text{-}9$
$^{10}C_9 \times p^9 \times (1-p)^1 = 9.47304E\text{-}11$
$^{10}C_{10} \times p^{10} \times (1-p)^0 = 6.04662E\text{-}13$

The number of records Y that would be retrieved is obtained as

$(0 \times 0.538615) + (1 \times 0.343797) + \ldots\ldots + (10 \times 6.04662E\text{-}13) = 0.6$

The number of blocks B_{before} expected to be retrieved for the random file is obtained as

$$B_{before} = \frac{0.6+1}{2} = 0.8.$$

Assuming the number of blocks in the file B is less than Y, the number of blocks B_{after} to be retrieved for the sequenced file

$$B_{after} = \frac{0.6}{2} = 0.3.$$

The expected percentage reduction in blocks retrieved is expected to be

$$\frac{0.8-0.3}{0.8} = 0.625.$$

By changing the values of q (the query size) used in calculating p (the probability that a record is retrieved, and also B (the block size), we could determine the effects of these parameters on the percentage reduction in blocks retrieved.

5.4 Statistical Confirmation of Experimental Results

Consider a file of record size r (r = 5) and of attribute range d (d = 10).
Using eqn. (2) to calulate the hamming distance yields :

$$E(hd) = \sum_{x=0}^{r}(r-x)\times P(x)$$

$$P(x)={}^{r}C_{x}\times p^{x}\times(1-p)^{r-x} \text{ where } \quad p=\frac{1}{d}=\frac{1}{10}=0.1.$$

$$E(hd) = \sum_{x=0}^{5}(5-x)\times P(x) = 5\times P(0)+4\times P(1)+3\times P(2)+2\times P(3)+P(4).$$

$P(0) = {}^{5}C_0 \times(0.1)^0 \times (0.9)^5 = 0.590$

$P(1) = {}^{5}C_1 \times(0.1)^1 \times (0.9)^4 = 0.328$

$P(2) = {}^{5}C_2 \times(0.1)^2 \times (0.9)^3 = 0.073$

$P(3) = {}^{5}C_3 \times(0.1)^3 \times (0.9)^2 = 0.008$

$P(4) = {}^{5}C_4 \times(0.1)^4 \times (0.9)^1 = 0$

$E(hd) = (5 \times 0.590) + (4 \times 0.328) + ...+ (1 \times 0) = 4.497$

For a large file of 1000 records (N = 1000), the expected total hamming distance can be obtained from eqn.(3) as

$E(total\_hd) = (N-1) \times E(hd)$

$= (1000 - 1) \times 4.497$

$= 4492.$

For a smaller file of 500 records (N = 500), the expected total hamming distance obtained from eqn.(3) is

$E(total\_hd) = (N-1) \times E(hd)$

$= (500 - 1) \times 4.497$

$= 2244.$

Verification of Statistical Results using Experimental Simulation

FILEB.C is first run to generate a file with the following specifications:

Parameter Name	*Parameter Value*
File Name	T_5_10.R
File Size	1000
Attribute Range	10
Record Size	5

FILEB.C outputs the following values:

Name	*Value*
Average Hamming Distance	4.486
Total Hamming Distance	4482

FILEB.C is next run to generate a file with the following specifications:

Parameter Name	*Parameter Value*
File Name	5H_5_10.R
File Size	1000
Attribute Range	10
Record Size	5

FILEB.C outputs the following values

Name	*Value*
Average Hamming Distance	4.497
Total Hamming Distance	2244

The data recorded above is tabulated below as Table 14.

	Expected Value	*Actual value*
Average Hamming Distance	4.497	4.497
Total Hamming Distance (file size = 500)	2244	2244
Total Hamming Distance (file size = 1000)	4492	4482

Table 14

Statistical Evaluation of the expected hamming distance after sequencing using eqn.(4) was done in Example 2 and resulted in

- E (Average hamming distance after sequencing) = 1.999
- E (Total hamming distance after sequencing)= $\begin{cases} 998 & file \quad size = 500 records. \\ 1997 & file \quad size = 1000 records. \end{cases}$

Verification of statistical results using experimental simulation

SECOND.C is first run to produce the sequenced version of "5h_5_4.R" which we call "5h_5_4.R22" with the following specifications:

Parameter Name	*Parameter Value*
File Name	5H_5_10.R22
File Size	500
Attribute Range	10
Record Size	5
Threshold Value	2

SECOND.C outputs the following results:

Average Hamming Distance After Sequencing	=2.35
Total Hamming Distance After Sequencing	=1175

SECOND.C is next run to produce the sequenced version of "T_5_4.R" which we call "T_5_4.R22" with the following specifications:

Parameter Name	*Parameter Value*
File Name	T_5_4.R22
File Size	1000
Attribute Range	10
Record Size	5
Threshold Value	2

SECOND.C outputs the following results:

Name	*Value*
Average Hamming Distance after sequencing	2.18
Total Hamming Distance after sequencing	2179

The data recorded above is tabulated below as Table 15 and 16.

	Expected Value	*Actual value*
Average Hamming Distance	1.999	2.35
Total Hamming Distance (file size = 500)	998	1175

Table 15

	Expected Value	*Actual value*
Average Hamming Distance	1.999	2.18
Total Hamming Distance (file size = 1000)	1997	2179

Table 16

It is observed from tables 15 and 16 that there is some discrepancy between expected values and actual values of the hamming distance after sequencing. The

reason behind this discrepancy is that the threshold condition imposed in sequencing algorithm 2 is unsatisfied. In other words, for some records in the file to be sequenced the insertion point was not at a position that yields an increase in hamming distance less than or equal to the threshold value. On the contrary, these few records are inserted at a position where they can contribute the least to the total hamming distance. Our estimation of the hamming distance after sequencing can therefore be considered as an optimum that may be reached only if the insertion position for all the records of the file at hand occurs at a position where the increase in hamming distance is less than or equal to the threshold value specified.

Chapter 6

Conclusion

Sequencing a file using sequencing algorithm 2 can improve response time to a query by reducing the number of blocks retrieved. Choice of a threshold value is critical in obtaining an efficient improvement. The number of keyterms in a query is also important in obtaining substantial improvement. Further research is believed to find a definite way of determining the best parameters to be used for each individual type of file in order to improve response time.

Appendix A

Program Design

FILEB.C

This program makes use of a random number generator to generate integers that represent the keyterms and requires the specification of the following parameters:

- File name : the name of the file to be generated.
- Record no : the number of records that constitute the file.
- Attribute no : The number of attributes per record.
- Attribute range : The range from 1 that the integers should be selected from.

FIRST.C

This program sequences the file using sequencing algorithm 1. The program keeps track of the smallest hamming distance and places the incoming record at that position after all possible positions have been tried. The original hamming distance, the number of comparisons made in sequencing the file, the final hamming distance, as well as the time taken to sequence the file are printed out. The required input for this program is listed below.

- Random File Name : The name of the random file to be sequenced
- Record No : The file size
- Attribute No : The number of attributes per record.
- Sequenced File Name : The name to be given to the sequenced version of the original file.

FIRST_PRIME.C

This program rearranges the records of a file to give the greatest possible hamming distance. The algorithm used for this program is similar to that of sequencing algorithm 1 except that the program keeps track of the greatest hamming distance and places the incoming record at that position after all posible positions have been tried. The original hamming distance, the number of comparisons made in sequencing the file, as well as the final hamming distance are printed out. The arguments to be specified when running this program are listed below.

- Random File Name : The name of the random file to be unsequenced
- Record No : The file size
- Attribute No : The number of attributes per record.
- Unsequenced File Name : The name to be given to the unsequenced version of the original file.

SECOND.C

This program sequences the file using sequencing algorithm 2. The program keeps track of the smallest hamming distance and places the incoming record at the first position that yields an increase in hamming distance which is less than or equal to the threshold value. If no such position is found then the record is inserted at the best position that has been kept track of. The original hamming distance, the number of comparisons made in sequencing the file, the final hamming distance, as well as the time taken to sequence the file are printed out. The required input to the program is listed below.

- Random File Name : The name of the random file to be sequenced
- Record No : The file size
- Attribute No : The number of attributes per record.

- Threshold value : The threshold value k which is the largest increase in hamming distance the record to be inserted is allowed to contribute to the total hamming distance.
- Sequenced File Name : The name to be given to the sequenced version of the original file.

QUERY.C

This program uses a random number generator to generate integers that represent the query terms and requires the inputting of the arguments listed below.

- Query File name : the name of the query file to be generated
- Record no : the number of query records constituting the file
- Attribute no : The number of attributes per record
- Query no : The number of nonzero query terms per record (query no. <= attribute no.)
- Attribute range : The range from 1 that the integers are to be selected from

MATCHB.C

This program is assigned the task of matching a specified query file to a data file. The main file is assumed to be divided into blocks. All the queries in the query file are matched against all the records in the data file. If all the nonzero query terms in a query record match the corresponding data attributes of a data record, then that data record is meant to be read. Any block that contains at least one record meant to be read must be retrieved. A count is made of the blocks that are to be retrieved and of the matching data records The program input is specified below.

- Data File name : the data file name

- Data record no : the data file size
- Attribute no : The data record size
- Query File name : the query file name
- Query record no : the query file size
- Bucket size : the number of records per block.

Appendix B

File Naming Format

A special format in naming our simulation files was adopted and is worth explaining so that the analysed experimental results can be better understood. Note however that it is not mandatory to use this naming format in testing the information retrieval system. In fact, any format you choose would do just fine.
As a reminder, there are three types of files used in the simulation carried out, namely

- Random file
- Unsequenced file
- Sequenced file, sequencing done using
    - Sequencing Algorithm 1
    - Sequencing Algorithm 2
- Query file

Random File

e.g. 5H_10_4.R

- 5H : 500 records is the file size, could be any 2 characters
- 10 : 10 attributes per record
- 4 : attribute range
- Extension .R : the file is a Random file as generated by FILEB.C

Unsequenced File

e.g. 5H_10_4.U

- 5H : 500 records is the file size, could be any 2 characters
- 10 : 10 attributes per record
- 4 : attribute range

Extension .U : the file is an Unsequenced file as produced by FIRST_PRIME.C

Sequenced File using Sequencing Algorithm 1

e.g. 5H_10_4.RS1

- 5H : file size
- 10 : record size
- 4 : attribute range

Extension .RS1 :

- R : The file used to be a random file
- S : The file now is sequenced
- 1 : Sequencing algorithm 1 was used

Extension .US1 :

- U : The file used to be unsequenced
- S : The file now is sequenced
- 1 : Sequencing algorithm 1 was used.

Sequenced File using sequencing algorithm 2

e.g. H_10_4.R2*i*

- H : file size
- 10 : record size

- 4 : attribute range

Extension .R2*i* :

- R : The file used to be a random file
- 2 : The file now is sequenced by using sequencing algorithm 2
- *i* : Threshold value used.

Extension .U2*i* :

- U : The file used to be unsequenced
- 2 : The file now is sequenced
- *i* : Threshold value used.

Query File

e.g. H_10_4.Q*i*

- H : Query file size
- 10 : record size
- 4 : attribute range

Extension .Q*i* :

- Q : The file is of a query file type
- *i* : The number of nonzero attributes in a query record.

References

B. G. T. Lowden 1985, *An Approach to Multikey Sequencing in an Equiprobable Keyterm Retrieval Simulation*, University of Essex.

A. K. Jain, R.C. Dubes 1983, *Algorithms for Clustering Data*, Prentice-Hall.

Jay L. Devore 1991, *Probability and Statistics for Engineering and the Sciences*, Third Edition, Duxbury Press.

Salton G., M. J. McGill 1983, *Introduction to Modern Information Retrieval, Mc-Graw Hill.*