A New Approach
to
Record Clustering for Large Databases

Raffi H. Makhoulian
B.Sc., Haigazian University College

THESIS

Submitted in partial fulfillment of the requirements for the degree of
Masters of Science in Computer Science
at the Lebanese American University
June 1997

[redacted]

Dr. Issam Moughrabi (Supervisor)
Assistant Professor of Computer Science
Lebanese American University

[redacted]

Dr. May Abboud
Assistant Professor of Computer Science
Lebanese American University

[redacted]

Dr. Nashaat Mansour
Assistant Professor of Computer Science
Lebanese American University

Table of Contents

Acknowledgments

I thank Dr. I.A.R. Moghrabi for his supervision, guidance, and help in researching and writing this thesis.

I further acknowledge my debt to my mother for her continuos financial and moral support throughout my study years.

Finally, I thank Dr. May Abboud and Dr. Nashaat Mansour for their assessment in finalizing this work.

Abstract

This work introduces a new approach to record clustering where a hybrid algorithm is presented that clusters records based upon threshold values and the query patterns made to a particular database. We study the space density of a file and how it affects retrieval time before and after clustering. The hamming distance of a file is used as a measure of space density. The objective of the algorithm is to minimize the hamming distance of the file while attaching significance to the most frequent queries being asked. Simulation experiments conducted proved that a great reduction in response time is yielded after the restructuring of a file. Criteria, such as, block size, threshold value, percentage of records satisfying a given set of queries, etc..., which affect clustering and response time are also studied. Random statistical and graph theory are used to substantiate the experimental results. As a further means for predicting performance, regression analysis is employed and later compared to experimental figure.

Chapter One

Introduction

Record clustering is a strategy that groups similar or related items into common classes with the aim of minimizing retrieval time. It is a way of providing order among a collection of stored records. The idea of clustering is based on the hypothesis that closely related records tend to be relevant to the same queries, and thus should be close to one another in their physical location.

The main aim of clustering is to reduce retrieval time and consequently response time. In a clustered file records similar to one another or related to each other for some reason or another are stored in adjacent locations so that a single file access makes available the maximum number of relevant records. Since retrieval is a main consideration, then organization is required to group records in such a way that retrieval is accelerated.

A significant proportion of queries processed against a database concern the retrieval of all records with certain characteristics in common. Normally, queries are inefficiently handled by scanning every record in the file. To make the retrieval faster, conventional structuring techniques for storage, such as, multilist and inverted list systems are used. These suffer from the fact that, in general, records are scattered across the file space in no predetermined way and the number of blocks accessed can be as many as the number of records retrieved [5]. Before fetching from 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 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 fewer blocks retrieved for a query, the smaller the total response time.

Retrieval time of multiattribute records can be improved if similar records are grouped close together, in the file space. The idea behind this is that fewer page transfers are

required as the probability of two or more of the target records residing in the same page of storage is increased. Such clustering strategies and techniques are more commonly associated with document retrieval in library information systems [8].

Much has been carried out in this field, in different directions, with the aim of minimizing retrieval time, such as: query reformulation, the single link method [1], the cosine similarity coefficient with the dissimilarity matrix and centroids [8], adaptive clustering [11], etc.... All of these have proved certain improvements. Recent research utilizes concurrent file reorganization algorithms for record clustering that carry reorganization using the original storage space of the file [6]. But, for example, experiments have shown that clustering methods based on the calculation of a dissimilarity matrix would, despite having desirable theoretical properties, be impractical for use with large files of documents [1,10]. This attitude arises from the fact that these clustering methods require a computation time of at least $O(n^2)$ for n documents. Therefore, new strategies are needed that do not require large amounts of time and also promise near optimal results.

This project is a further step to improve the existing ideas and introduces a new strategy to record clustering where importance is given not only to the closeness of records to one another, but also checks their relevance to the most frequent queries and clusters accordingly.

The existing algorithm [9] is based on the specification of a threshold value. Insertion of incoming records occurs immediately once the threshold condition is satisfied. It ensures that records, which have similar properties according to the specified threshold value, are in close proximity to each other in the file space. However, if a threshold value of zero is input, then the algorithm degenerates to the old algorithm of Lowden [4] where the required number of comparisons is of the order of n^2.

The new algorithm is based on two things:

1- A history tree of most frequent queries

2- A given threshold value

The clustering algorithm first checks every record of the file trying to match each one to a

given query in the history tree of queries. If a record satisfies any of the queries, then it is assigned to the query with the highest frequency (or priority); if not, then it is inserted in the file according to the threshold value specified.

Thus, the resulting clustered file will be of two groups. The first part will contain clusters of records where each cluster consists of a set of records pertaining to the same query, and the second part of the file will consist of records that did not satisfy any query and hence are clustered according to the hamming distance and threshold value specified.

For the rest of this work, I refer to the existing algorithm as clustering algorithm 1 and to the new algorithm as clustering algorithm 2.

The thesis is divided into five parts. The first part introduces the concept of hamming distance and explains both algorithms in depth, the second part is a project description, the third part is the experimental analysis where curves, comparisons and interpretation of results have been made, and the fourth part is a statistical analysis where some formulas are been derived. Finally the fifth part uses regression analysis to derive formulas for future estimations.

Chapter Two

Clustering Algorithms

2.1- File System Description

The records of the files in this paper are assumed to be of fixed length and the keyterms are confined to particular fields within the records. In other words, changing the position of the value of an attribute is forbidden.

An example of a file system is an employee file, which is also used in [7,9], where each record represents an employee's details and has a fixed number of keyterms.

A key conversion table is used to convert keyterms into numerical values within some range. Thus, every attribute will be mapped into its equivalent numerical value, according to the data it holds in that record. Thus, there is a mapping of records such that each record can be viewed as a vector X in a geometrical space and X can be represented as:

$$X=(x_1,x_2,x_3,\ldots,x_n)^T \{\text{where n is the number of keyterms}\}$$

Moreover, all of these X vectors, assuming that each is unique, can be written as a linear combination of the vectors of the basis in a unique fashion.

Table 1, 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 1: Employee File System

Following this table, we can map every record of the file to its image in the new vector space.

2.2- Hamming Distance

Various measures and methods have been used for clustering records. Let the difference between the ith and kth records be denoted as:

$$d(i,k)$$

One measure that can be used to calculate the difference between records is the Hamming Distance which is also used in this paper. Let HD stand for Hamming Distance; then, the difference between two records is given by:

$$d(i,k)=\sum_{j=1}^{r} f(x_{ij}, x_{kj}) \{\text{where r is the number of attributes of a record}\}$$

$$\text{where } f(x_{ij},x_{kj}) = 0 \text{ if } x_{ij}=x_{kj}$$

$$\text{and } \quad f(x_{ij},x_{kj}) = 1 \text{ if } x_{ij}<>x_{kj}$$

The total hamming distance of the file is given by [3]:

$$HD=\sum_{i=1}^{N-1} d(i, i+1) \{\text{where N is the total number of records}\}$$

2.3- Clustering Algorithm 1

This algorithm is based on the specification of a threshold value. When the latter is satisfied, no further comparisons are made and an insertion of the record in the bucket that satisfied the threshold condition occurs, thus reducing the number of comparisons.

Given a file of size N and a dissimilarity index definition d, a complete weighted graph denoted by (N,d) may be constructed. Each of the N nodes in the graph represents a record in the file space of size N. The weight d(x,y) on each one of the N(N-1)/2 edges of the graph represents the dissimilarity between the 2 records x and y connected by the edge (x,y). The algorithm's goal is to construct a short spanning path through the nodes thereby reducing to *near* optimal the overall hamming distance.

Thus, the operational steps of the algorithm can be specified as follows: a subpath of the graph (N,d) is an ordered subset of the nodes in N. (P,m) denotes a path including a path P and an extra node m which is not in P. The path (P,m) is formed as follows:

Case 1: If P passes through more than one node then

(a) find an edge (x,y) between nodes of P such that the value
$[d(x,m) + d(m,y)] - (x,y) \leq h$ (some user-defined threshold value)

or is minimal if no such edge exists

(b) delete edge (x,y) and add edges (x,m) and (m,y) to form path (P,m)

Case 2: If P passes through exactly one node n then make path (P,m) as the edge (n,m) or (m,n).

The clustering process starts by retrieving the records in the order they were originally arranged and finding an insertion position among the already clustered records in the new file. The position that is sought for each incoming record is one leading to an increase in

hamming distance of the new file only by the threshold value specified or less. Thus, the incoming record starts at the front of the already clustered 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 were found, the insertion would take place at the position yielding the smallest increase in hamming distance even though it is greater than the specified threshold value, which had been kept track of during the search.

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 when the record is placed between two records it does not have a single common keyterm with and the two records also do not have any common keyterm.

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 minus two since the first two records do not need any comparisons. With such a threshold value, no clustering 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. Thus, a wise choice of the threshold value is essential.

2.4- Clustering Algorithm 2

This clustering algorithm, before doing any physical clustering, starts matching each record of the given file to one of the nodes of the query tree. In other words, it tries to assign each record to the most frequent query that it satisfies. It is a single pass process which takes each record of the file, traverses through the tree and assigns it to the query with the highest frequency (or priority) that it satisfies. It stops when the file is exhausted.

Once the first phase is over, the second phase of clustering starts which is the real physical clustering of the file. It starts reading the file in sequence and puts the records that satisfy the same query in contiguous physical positions. Those records that did not satisfy any of the queries of the query tree are clustered according to a specified threshold value following the steps of algorithm 1. In other words, while the first part of the resultant file consists of buckets of records that belong to the same query, the second part consists of a sequence of records that are clustered according to a dissimilarity index which in our case is the hamming distance.

Thus, if a record was assigned to a query during the first phase of the algorithm, then its position in the new file will be at the end of the current sequence of records that satisfy that query. If a record did not satisfy any of the queries, then the clustering algorithm will search for it a position starting from the first record of the first bucket of the second part of the file trying to place it in a position where the increase in hamming distance is less or equal to the threshold value specified.

Those records that are assigned to the same query will be put in proximity to one another in the same bucket (and consecutive ones if they need more than one bucket). The clustering is based on the assumption that since these queries are asked very frequently, and consequently their probability of being asked again is very high, then by putting records that satisfy the same query in contiguous locations will result in the retrieval of the minimum number of blocks.

Before moving to the formal presentation of the record matching algorithm, we introduce in the next section the history tree of queries.

2.4.1- History Tree of Queries

Given that we have a key conversion system, then keyterms of queries could also be converted into numerical values. Furthermore, we can read the keyterms of a query as a single numerical value. For example, 1 2 3 1 as one thousand two hundred thirty one. This observation can be used to construct a dynamic query tree which has the properties of a binary search tree.

The nodes of this tree consist of an attribute that contains the keyterms of the query, an attribute that contains its equivalent numerical value, and an attribute that contains the frequency of the query. The binary search tree is constructed according to the numerical value of the query. For each query issued to the database, the query is traversed through the tree and inserted in its proper position if it does not exist. If it exists then its frequency is added by one. This way we will have the set of most frequent queries.

After some prescribed time interval or when the tree contains a certain load, one may decide to cluster the main file. Before proceeding with clustering, some queries of the tree may turn out to be subsets of other queries with respect to their keyterm values. For example, if we have a query that requests all records with a salary greater than 1000, then, according to our key conversion system, its converted form will be:

Starting Date	Position	Salary	Marital Status
0	0	3	0

Another query may request all records with a salary greater than 1000 and 'operator' position. The query is given as:

Starting Date	Position	Salary	Marital Status
0	1	3	0

where obviously the first query is a subset of the second query with respect to the keyterm values, and, thus, its frequency cannot be disintegrated from the frequency of the second query. We note here that the hamming distance of the resulting file will decrease by increasing the number of records that satisfy the second query.

Another major issue is the problem of queries with low frequencies. Just before the matching process starts, a certain threshold value can be specified such that those queries that have a frequency value less than the specified frequency threshold are deleted from the query tree. Thus, these insignificant queries do not affect the clustering process. Deleting those queries will also result in a smaller query tree which will increase the matching process.

When clustering starts the first step is to try to match every record of the file to some query by traversing it through the query tree. If it satisfies more than one query, then we assign it to the one with the highest frequency.

But what's the advantage of having a Binary Search Tree? The advantage is that when the numerical value of the keyterms of the compared record is smaller than the current compared node of the tree, then we discard the right sibling of that node and its successors, and just take the left path. In this way we chop our search tree from that node and on into half. Thus, our choice of the Binary Search Tree is due to its good performance which in the best case is of the order of N x lg(k) (where N is the number of records), and in the worst case is of the order of N x K which is still of a rectangular time and is acceptable.

2.4.2- The Matching Algorithm

In this section we introduce the algorithm of matching the records of the file to the query tree:

```
Convert the attributes of each record of the file according to the conversion table

While not end of file do
    read a record
    while the numerical value of the record < numerical value of current node do
                Take left path of the tree
    end;{while}
    If the tree is not exhausted then
        compare the record with every remaining node of the tree
        assign it to the one with the highest frequency that it satisfies
    end;{if}
end;{while}
```

If we cluster the file only according to the hamming distance threshold value, then we may face the following problem:

Consider the following set of records, clustered according to algorithm 1 with minimum threshold:

1 2 3 4 5

2 2 3 4 1

3 3 3 3 1

1 3 3 3 5

2 3 3 3 2

2 2 2 2 2

1 2 2 2 5

If we have the most frequent query as 1 0 0 0 5, then this sequencing for a block size of three will cause the retrieval of the maximum number of blocks, although it satisfies the property of minimum hamming distance.

Thus, the new strategy tries to make a compromise between the most frequent queries and the minimum hamming distance.

Finally, the clustering time to sequence a given file with this algorithm depends on both the percentage of records satisfying one of the queries of the query tree, and the threshold value specified. The higher the percentage of the records that are assigned to a query, the less the number of records that must be clustered according to the threshold value, which results in less number of comparisons to be made to find a position for a given record, and consequently less number of blocks are retrieved.

Thus, the new algorithm promises to make some enhancements over the old one with respect to clustering effort and also attaching some importance to the pattern of queries asked.

Moreover, if a database administrator has a prior knowledge of the pattern of queries that are asked to his database, then he could construct a query tree of his own and assign

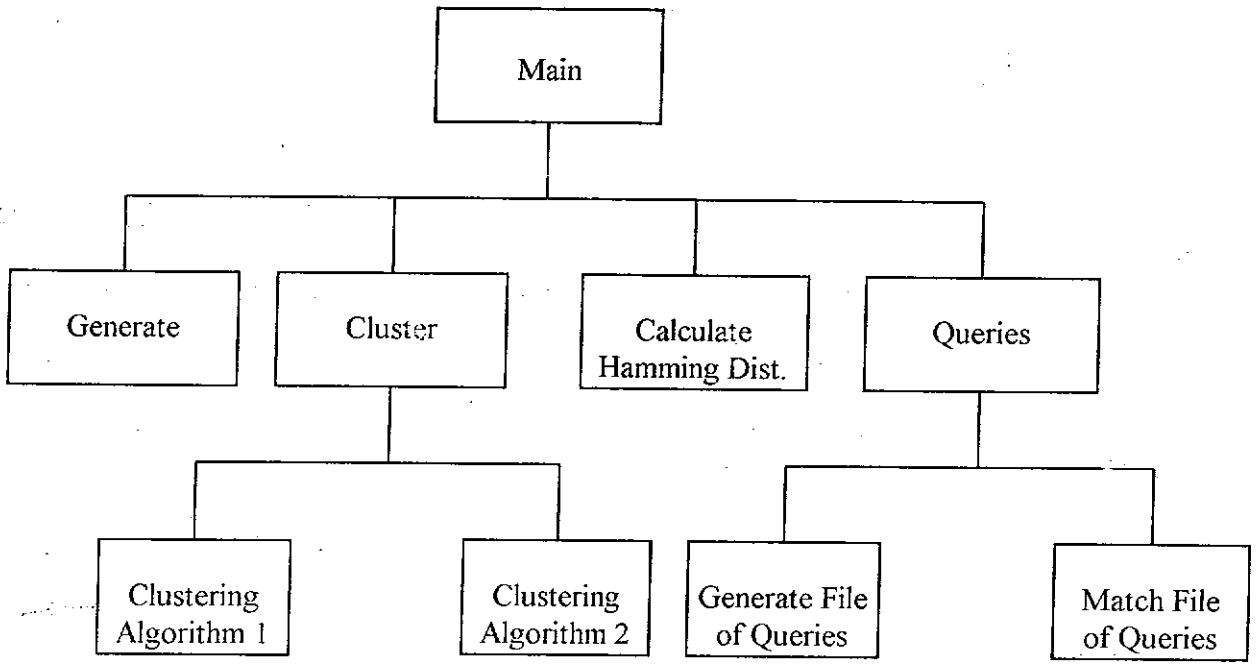
priority values to the queries instead of the frequencies, and then have the file clustered according to this query tree. Thus, the new algorithm is more flexible in the sense that the clustering of the file can be directed by the database administrator.

Chapter Three

Project Description

3.1- Program Design:

Below is the structure chart of the program of the simulation experiment. A detailed description of each program, its interface with the user, as well as its output are presented in Appendix A.



3.2- Simulation Experiments

The set of experiments to gather statistical data followed the following sequence of steps:

1- Generate a file with a given size, number of attributes, generated set of queries, range of keyterms, and percentage of the records satisfying the generated set of queries. Run Generate.

2- Cluster the generated file with different threshold values, using clustering algorithm 1. Run ClusterT.

3- Cluster the generated file with different threshold values, using clustering algorithm 2. Run ClusterQ.

4- Calculate the hamming distance of all the clustered files and the main file. Run CalcHD.

5- Generate a set of query files with varying number of keyterms. Run GenQry.

6- Match the records of all the generated query files against the records of the clustered files and the main file. Run MatchTo.

All the interesting variables such as number of blocks retrieved, number of comparisons done, hamming distance, threshold value, and percentage of records satisfying the queries are tabulated and graphically shown for further interpretation, derivation of formulas, and reaching to statistical conclusions.

Chapter Four

Experimental Analysis

The objective of this chapter is to simulate the algorithms described in the previous sections, compare the results, analyze, and demonstrate the enhancements in performance and retrieval.

The naming format of the files is explained in Appendix B.

4.1- Numerical Comparison of the Two Algorithms (The two algorithms are compared in terms of number of blocks retrieved, comparisons made, and hamming distance)

To compare the performance and results of both algorithms for the same input file the following steps were taken (note that the data presented here is one candidate set taken from the different experiments carried):

- GENERATE was run to generate a random file with the following specifications:
  - File name : "T_8_6_40"
  - File Size : 1000 records
  - Record Size : 8 attributes
  - Attribute Range : 6
  - Query File Name : "T_8_6_40.Q1H"
  - Query Information File Name : "T_8_6_40.I1H"
  - Query File Size : 100 queries
  - Percentage of records satisfying the queries: 40%
- CLUSTERT was run to cluster the generated file with threshold values from 0 to 8.
- CLUSTERQ was run to cluster the generated file with threshold values from 0 to 8.
- CALCHD was run to calculate the hamming distance of all the clustered files and the main file.

The results of clustering algorithm one are tabulated in Table 4.1 and of clustering algorithm two in Table 4.2:

File Name	Threshold	Hamming Distance	Average HD	Comparisons	Blocks Retrieved
T_8_6_40		6,620	6.62		
T_8_6_40.T0	0	2,733	2.73	462,374	120,476
T_8_6_40.T1	1	2,772	2.77	418,415	109,031
T_8_6_40.T2	2	2,926	2.93	373,522	95,858
T_8_6_40.T3	3	3,093	3.09	287,509	73,960
T_8_6_40.T4	4	3,586	3.59	119,647	31,215
T_8_6_40.T5	5	4,463	4.46	20,382	6,337
T_8_6_40.T6	6	5,372	5.37	4,914	2,557
T_8_6_40.T7	7	6,180	6.18	1,601	2,008
T_8_6_40.T8	8	6,620	6.62	998	1,990

Table 4.1: Results of Clustering Algorithm 1

Besides the number of comparisons listed in the table below, also there is the number of comparisons made when matching the records of the main file to the nodes of the generated query tree. The number of comparisons made during the traversal of the tree for this sample data was 59,433. Note that this matching is done only once and is independent of any threshold value or algorithm.

File Name	Threshold	Hamming Distance	Average HD	Comparisons	Blocks Retrieved
T_8_6_40		6,620	6.62		
T_8_6_40.Q0	0	3,348	3.35	180,153	46,380
T_8_6_40.Q1	1	3,348	3.35	178,707	46,012
T_8_6_40.Q2	2	3,356	3.36	168,210	44,285
T_8_6_40.Q3	3	3,378	3.38	128,517	34,252
T_8_6_40.Q4	4	3,580	3.58	60,589	16,744
T_8_6_40.Q5	5	4,037	4.04	12,434	4,269
T_8_6_40.Q6	6	4,530	4.53	2,721	1,891
T_8_6_40.Q7	7	4,992	4.99	975	1,595
T_8_6_40.Q8	8	5,240	5.24	598	1,584

Table 4.2: Results of Clustering Algorithm 2

What we observe from the data of both tables at first sight is that when the threshold value gets close to the number of attributes in a record, little clustering is done. With the first few comparisons an acceptable position satisfying the specified threshold value is found and the incoming record is inserted there. For a threshold value of 8, in the case of clustering algorithm 1, the number of comparisons made is 998, and in the case of clustering algorithm 2 is 598 (which is equal to 1000-(40%x1000)-2), since the first two incoming records do not need any comparison and the other remaining records are accepted at the first comparison made. This is due to the fact that the threshold value specified is equal to the maximum hamming distance. Thus, the resulting file from algorithm 1 is just the inverted version of the old file with no clustering done. For algorithm 2, the first part of the resulting file is clustered according to the queries the records belong to, and for the case of the second part no clustering is done at all for the same reasons as in the case of clustering algorithm 1.

On the other hand, as the threshold value decreases the hamming distance is decreased but at the cost of more number of comparisons, blocks retrieved, and consequently more amount of time. Hence, an enormous amount of time is needed for large document bases or databases if the clustering is going to be done with low threshold values. Thus, there must be a compromise between the time needed to cluster and the threshold value that is going to result in a file of closely interrelated records with respect to their positions. The issue that is raised here is whether it is worth to spend a large amount of time and resources to cluster using a low threshold value or it is enough to cluster at a threshold value that takes little time but results in a file that is near optimal.

The data in both tables lead to conjecture that by using an average threshold value we will attain an acceptable total hamming distance with the saving of enormous amount of work and time with respect to the comparisons needed to be made, blocks retrieved, time spent, and resources utilized.

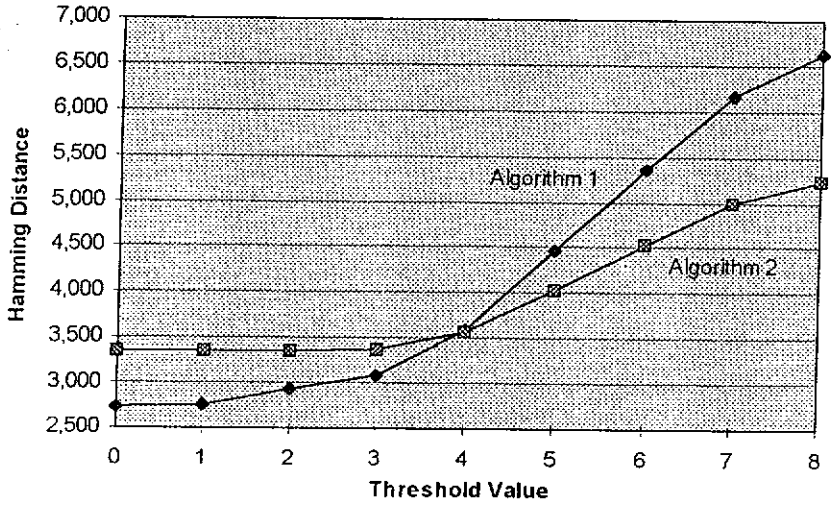
Comparing the results of both tables we realize that although for low threshold values clustering algorithm 1 results in lower hamming distance values, the difference between the hamming distance values resulting from both algorithms for these low threshold values is not very large specially when compared to the difference in the number of comparisons

made and blocks retrieved. It is obvious that clustering algorithm 2, while saving more than half of the work needed by clustering algorithm 1, attains results that are far more than acceptable and comparable with the results of algorithm 1.

Moreover, for threshold values above 4, we realize that the results obtained from algorithm 2 are better than algorithm 1 in all respects. Of course, the difference in hamming distance is due to the number of keyterms in the queries. As this number increases, the total hamming distance between the records satisfying the same query decreases, and thus, the total hamming distance of the first part of the file that consists of the records that satisfy the given set of queries decreases. Therefore, for clustering algorithm 2 to attain better results, the number of keyterms in the queries should be increased, and the percentage of records satisfying those queries should be more. These criteria will result in a file with a small hamming distance value with less amount of work and resources.

The results in both tables are depicted graphically for the different parameters:

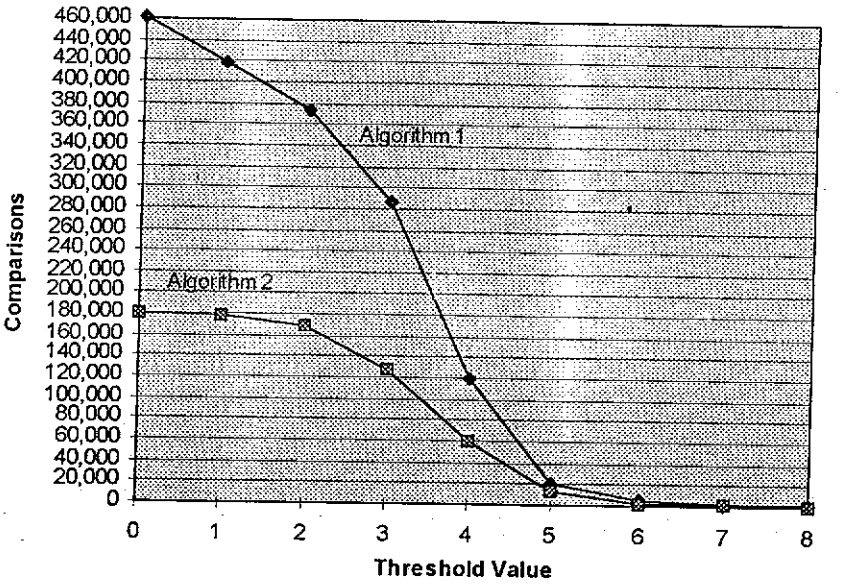
Figure 4.1:Threshold Value Vs. Hamming Distance
Clustering of the file using both algorithms



File Size=1000 records Record Size=8 attributes QueryTree=100 nodes

Figure 4.2:Threshold Value Vs. Comparisons

Clustering of the file using both algorithms



File Size=1000 records Record Size=8 attributes Query Tree Size=100 nodes

Figure 4.3: Threshold Value Vs. Retrieved Blocks

Clustering of the file using both algorithms

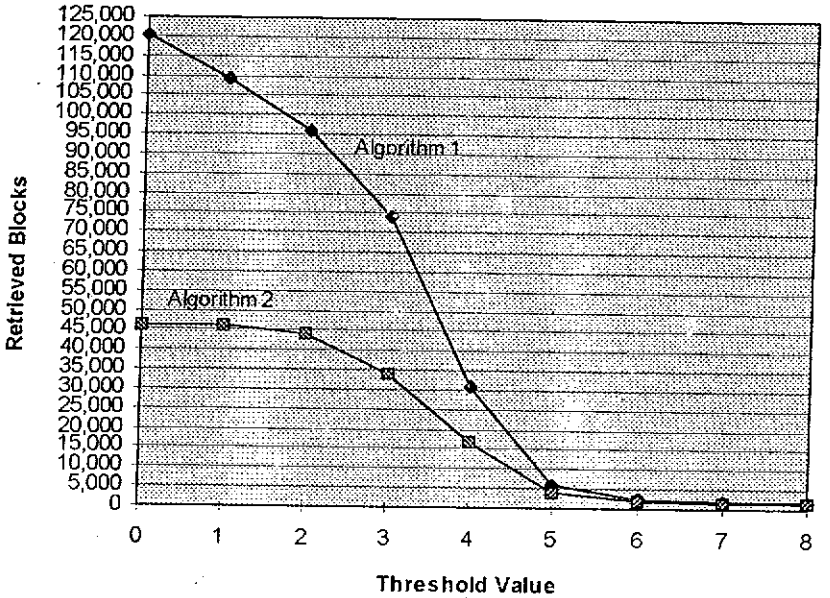
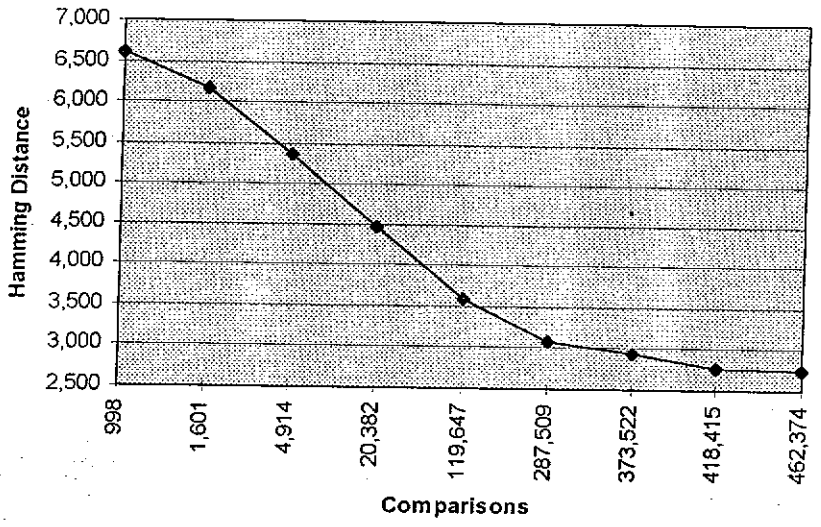


Figure 4.4: Comparisons Vs. Hamming Distance

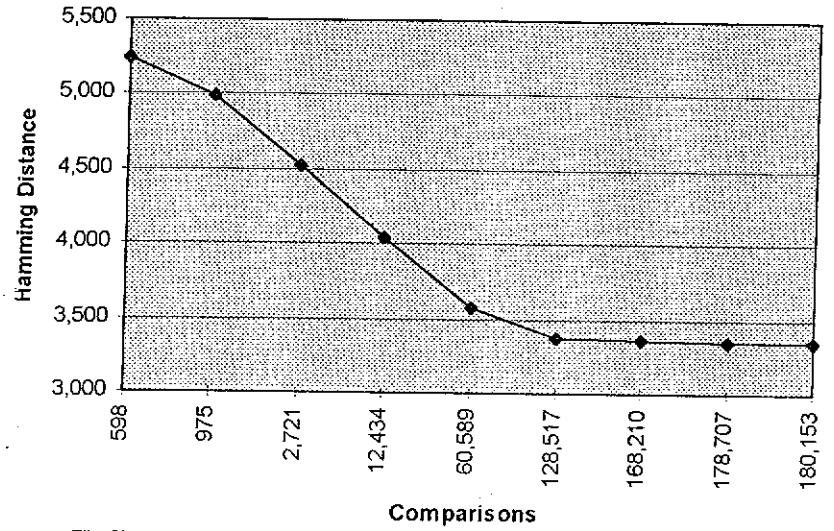
Clustering of the file using Algorithm 1



File Size=1000 records Record Size=8 attributes

Figure 4.5: Comparisons Vs. Hamming Distance

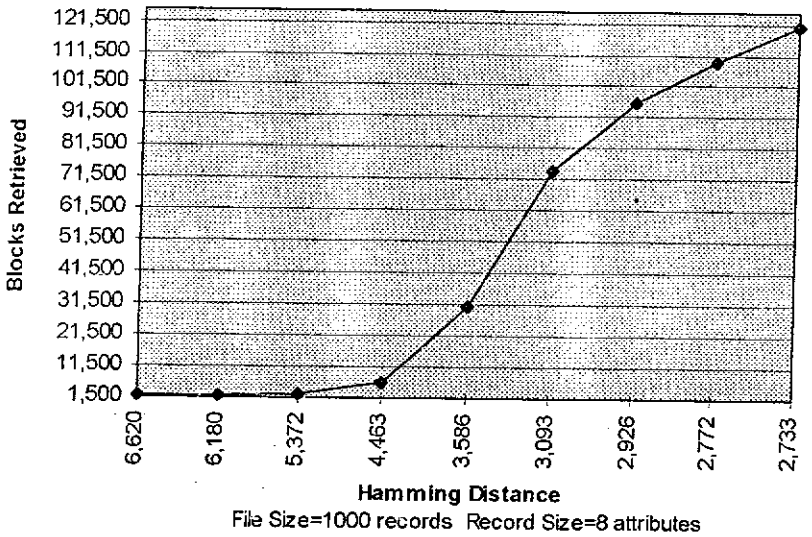
Clustering of the file using Algorithm 2



File Size=1000 records Record Size=8 attributes Query Tree Size=100 nodes

Figure 4.6: Hamming Distance Vs. Blocks Retrieved

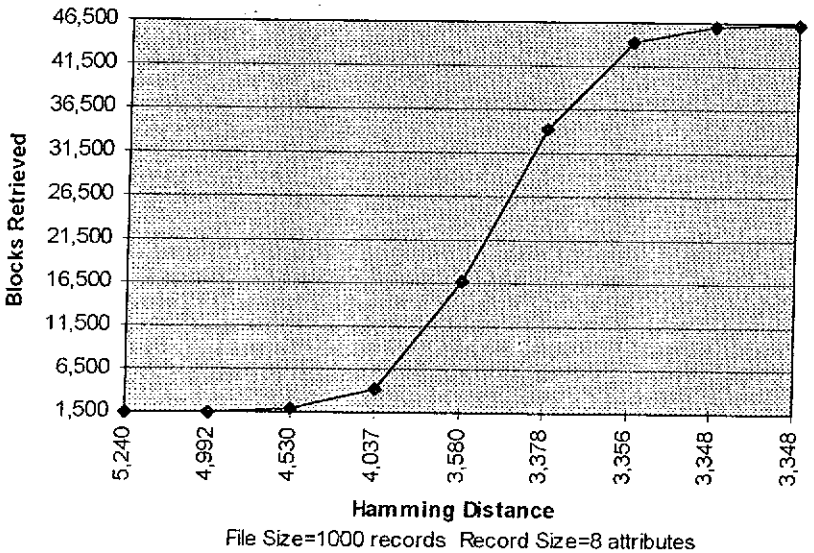
Clustering of the file using Algorithm 1



File Size=1000 records Record Size=8 attributes

Figure 4.7: Hamming Distance Vs. Blocks Retrieved

Clustering of the file using Algorithm 2



File Size=1000 records Record Size=8 attributes

4.2- Reduction in the Number of Retrieved Blocks

To compare the impact of clustering of the generated file by both algorithms on the reduction of retrieved blocks when a given set of queries is issued, the following experiments were carried:

- For the query file that contains the queries of the query tree, MatchTo was run against all the clustered files to match the records that satisfy the queries. The number of retrieved blocks for all the files is tabulated in Table 4.3.

File Name	Threshold	Hamming Distance	# of Retrieved Blocks	% Reduction in Ret. Blocks ($\|B_{before}-B_{after}\|/ B_{before}$)x100
T_8_6_40		6,620	387	0
T_8_6_40.T0	0	2,733	177	54.2
T_8_6_40.T1	1	2,772	177	54.2
T_8_6_40.T2	2	2,926	179	53.7
T_8_6_40.T3	3	3,093	185	52.2
T_8_6_40.T4	4	3,586	234	39.5
T_8_6_40.T5	5	4,463	310	19.9
T_8_6_40.T6	6	5,372	361	6.7
T_8_6_40.T7	7	6,180	387	0
T_8_6_40.T8	8	6,620	387	0
All the files that are clustered by algorithm 2			96	75.2
Number of matching records: 409				

Table 4.3: Results of Matching the Query Tree

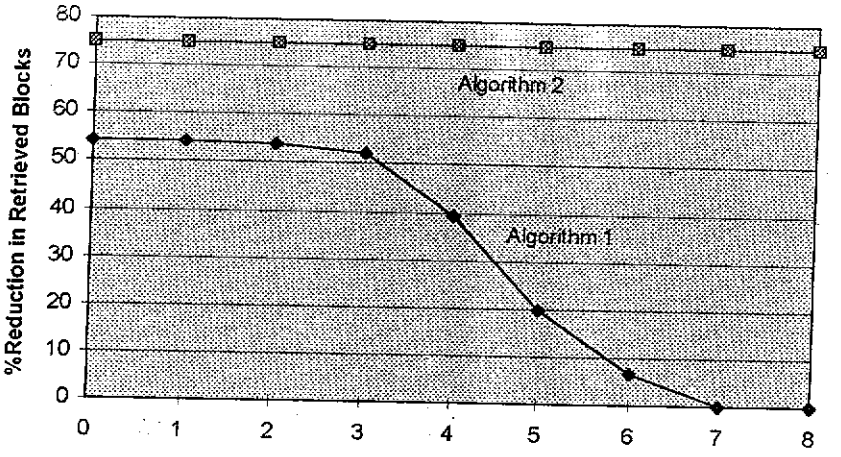
We observe from this test results that for the set of most frequent queries of the query tree, the clustered files by algorithm 2 gave the best retrieval results as expected. Clustering of the file by this algorithm reduced the number of retrieved blocks for the set

of most frequent queries by 75 percent, while clustering of the file by algorithm 1 for a threshold value of zero resulted in a reduction of 54 percent.

The results of Table 4.3 are depicted graphically as:

Figure 4.8: Threshold Value Vs. %Reduction in Retrieved Blocks

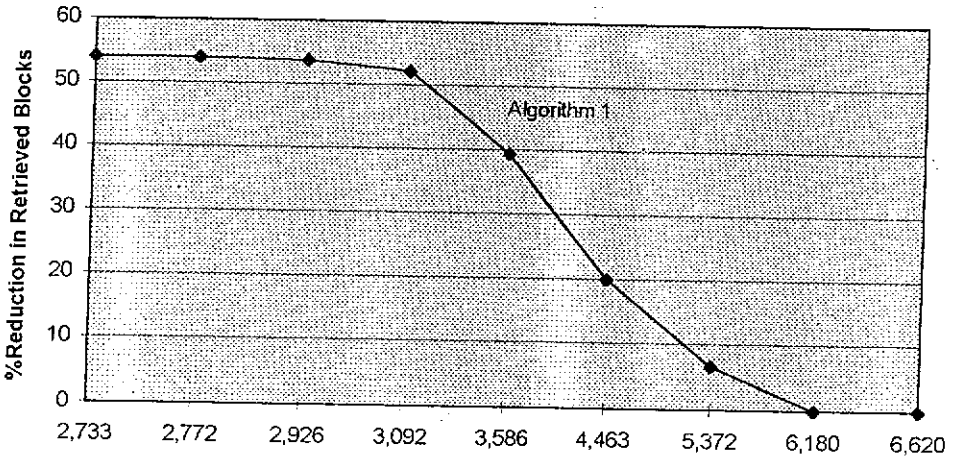
Clustering of the file using both algorithms



File Size=1000 records Record Size=8 attributes Query File Size=100 queries

Figure 4.9:Hamming Distance Vs. %Reduction in Retrieved Blocks

Clustering of the file using Algorithm 1



File Size=1000 records Record Size=8 attributes Query File Size=100 queries

-A second experiment that was carried for the same purpose was generating various query files of 100 query entries, each file consisting of queries with various fixed number of query keyterms, and then running MatchTo for those different query files against all the clustered files. The steps followed to accomplish this experiment were:

-GenQry was run to generate random query files with the following specifications:

- Query File Name: Query1, Query2, Query3, Query4
- Query File Size : 100 queries
- Keyterms : 1, 2, 3 and 4 respectively
- Attribute Range : 6

- MatchTo was run to match the queries of the generated query files against the records of all the clustered files.

The results of this experiment are tabulated in Tables 4.4 to 4.7.

Table of results obtained from matching the queries of QUERY1:

Threshold	File Name	Retrieved Blocks	% Reduction	File Name	Retrieved Blocks	% Reduction
-	T_8_6_40	5727	0			
0	T_8_6_40.T0	3497	38.9	T_8_6_40.Q0	3826	33.2
1	T_8_6_40.T1	3498	38.9	T_8_6_40.Q1	3826	33.2
2	T_8_6_40.T2	3583	37.4	T_8_6_40.Q2	3818	33.3
3	T_8_6_40.T3	3688	35.6	T_8_6_40.Q3	3838	32.9
4	T_8_6_40.T4	3974	30.6	T_8_6_40.Q4	3933	31.3
5	T_8_6_40.T5	4538	20.8	T_8_6_40.Q5	4239	25.9
6	T_8_6_40.T6	5150	10.1	T_8_6_40.Q6	4577	20.1
7	T_8_6_40.T7	5623	1.8	T_8_6_40.Q7	4826	15.7
8	T_8_6_40.T8	5727	0	T_8_6_40.Q8	4906	14.3
Number of Matching Records: 7999						

Table 4.4: Matching QUERY1 to the Clustered Files

Table of results obtained from matching the queries of QUERY2:

Threshold	File Name	Retrieved Blocks	% Reduction	File Name	Retrieved Blocks	% Reduction
-	T_8_6_40	952	0			
0	T_8_6_40.T0	679	28.7	T_8_6_40.Q0	712	25.2
1	T_8_6_40.T1	678	28.8	T_8_6_40.Q1	712	25.2
2	T_8_6_40.T2	672	29.4	T_8_6_40.Q2	718	24.6
3	T_8_6_40.T3	703	26.2	T_8_6_40.Q3	706	25.8
4	T_8_6_40.T4	756	20.6	T_8_6_40.Q4	738	22.5
5	T_8_6_40.T5	830	12.8	T_8_6_40.Q5	786	17.4
6	T_8_6_40.T6	906	4.8	T_8_6_40.Q6	833	12.5
7	T_8_6_40.T7	955	0.3	T_8_6_40.Q7	856	10.1
8	T_8_6_40.T8	952	0	T_8_6_40.Q8	862	9.5
Number of Matching Records: 1012						

Table 4.5: Matching QUERY2 to the Clustered Files

Table of results obtained from matching the queries of QUERY3:

Threshold	File Name	Retrieved Blocks	% Reduction	File Name	Retrieved Blocks	% Reduction
-	T_8_6_40	157	0			
0	T_8_6_40.T0	123	21.7	T_8_6_40.Q0	138	12.1
1	T_8_6_40.T1	123	21.7	T_8_6_40.Q1	138	12.1
2	T_8_6_40.T2	125	20.4	T_8_6_40.Q2	137	12.7
3	T_8_6_40.T3	133	15.3	T_8_6_40.Q3	137	12.7
4	T_8_6_40.T4	140	10.8	T_8_6_40.Q4	143	8.9
5	T_8_6_40.T5	149	5.1	T_8_6_40.Q5	147	6.4
6	T_8_6_40.T6	156	0.6	T_8_6_40.Q6	149	5.1
7	T_8_6_40.T7	157	0	T_8_6_40.Q7	149	5.1
8	T_8_6_40.T8	157	0	T_8_6_40.Q8	150	4.5
Number of Matching Records: 158						

Table 4.6: Matching QUERY3 to the Clustered Files

Table of results obtained from matching the queries of QUERY4:

Threshold	File Name	Retrieved Blocks	% Reduction	File Name	Retrieved Blocks	% Reduction
-	T_8_6_40	16	0			
0	T_8_6_40.T0	16	0	T_8_6_40.Q0	16	0
1	T_8_6_40.T1	16	0	T_8_6_40.Q1	16	0
2	T_8_6_40.T2	16	0	T_8_6_40.Q2	16	0
3	T_8_6_40.T3	16	0	T_8_6_40.Q3	16	0
4	T_8_6_40.T4	16	0	T_8_6_40.Q4	16	0
5	T_8_6_40.T5	16	0	T_8_6_40.Q5	16	0
6	T_8_6_40.T6	16	0	T_8_6_40.Q6	16	0
7	T_8_6_40.T7	16	0	T_8_6_40.Q7	16	0
8	T_8_6_40.T8	16	0	T_8_6_40.Q8	16	0
Number of Matching Records: 16						

Table 4.7: Matching QUERY4 to the Clustered Files

The results of these tables are depicted graphically for the different parameters:

Figure 4.10: Threshold Value Vs. Retrieved Blocks

Matching the QUERY1 file against the clustered files

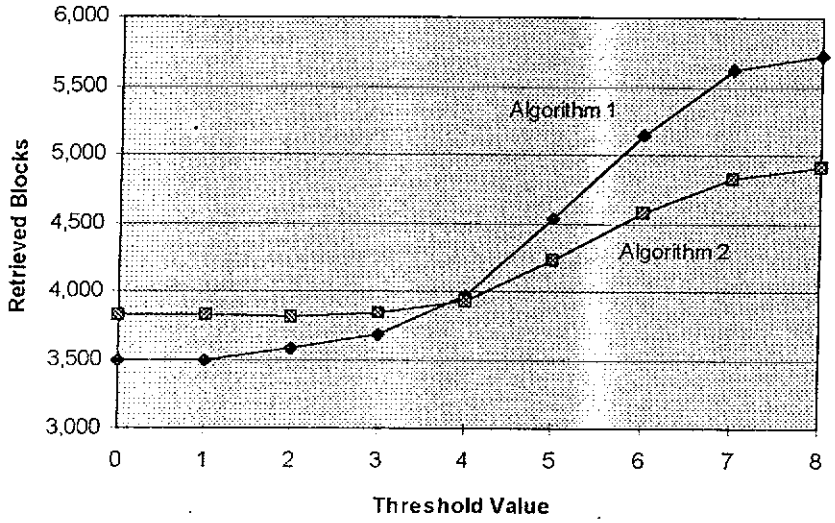


Figure 4.11: Threshold Value Vs. %Reduction in Retrieved Blocks

Matching the QUERY1file against the clustered files

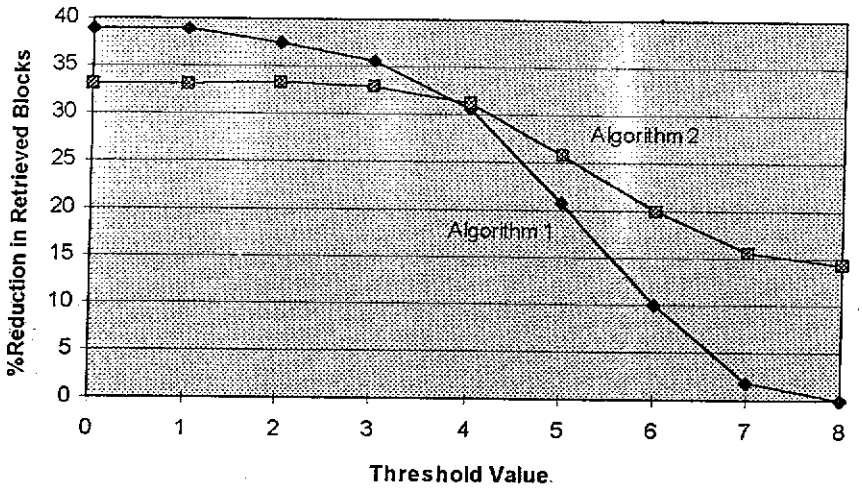


Figure 4.12: Threshold Value Vs. Retrieved Blocks

Matching the QUERY2 file against the clustered files

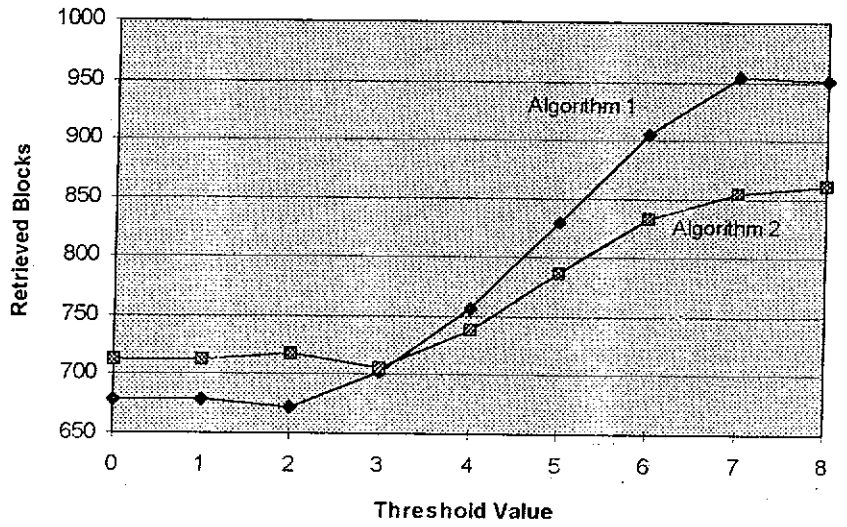


Figure 4.13: Threshold Value Vs. %Reduction in Retrieved Blocks

Matching the QUERY2 file against the clustered files

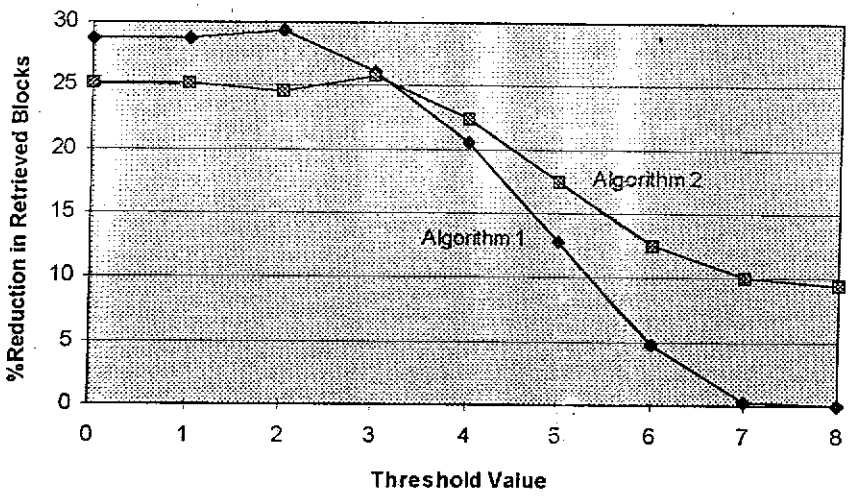


Figure 4.14: Threshold Value Vs. Retrieved Blocks

Matching the QUERY3 file against the clustered files

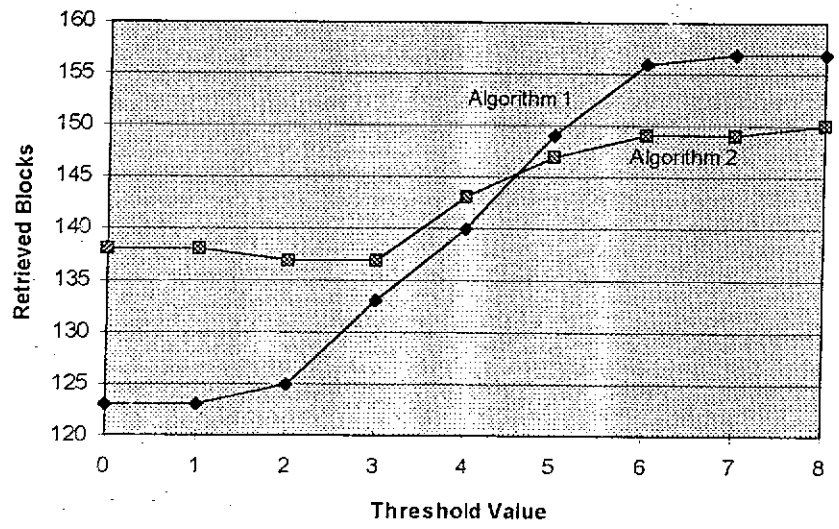
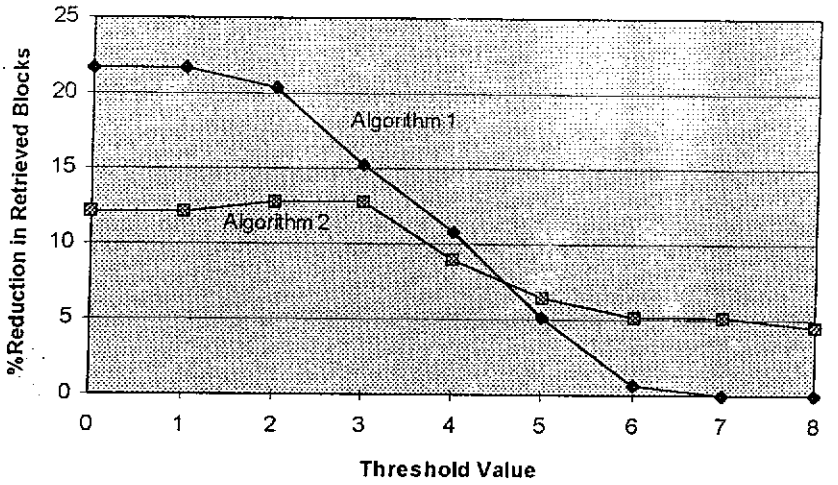


Figure 4.15: Threshold Value Vs. %Reduction in Retrieved Blocks

Matching the QUERY3 file against the clustered files



Chapter Five

Statistical Analysis of The Clustering Algorithms

This chapter derives statistical formulas for best case analysis and performance prediction of the two clustering algorithms of interest in this work. The formulas obtained here are based on Random Statistical theory.

5.1- The Expected Hamming Distance for a Non Clustered File:

Let

r be the number of attributes in a data record

x be the number of similar attributes between two consecutive records

then, r-x is the hamming distance between these two records.

To obtain the expected hamming distance we have to sum over all possible values of x, the product of the hamming distance r-x and its probability of occurring P(x). The domain of x is 0<=x<=r.

For the entire file of N records then the total hamming distance is:

$$E(hd)=(N-1)\times\sum_{k=1}^{N-1}(r-x_k) \qquad (1)$$

Therefore, the mathematical expected hamming distance between two consecutive records is:

$$E(hd)=\sum_{x=0}^{r}(r-x)\times P(x) \qquad (2)$$

If the file contains N records, then each incoming record will have the same expected

hamming distance between itself and the records it is placed next to. Thus, the expected total hamming distance can be estimated by :

$$\text{E(Total\_HD)}=(N-1)\times\sum_{x=0}^{r}(r-x)\times P(x) \tag{3}$$

which is a measure of number of mismatches between consecutive records.

Then, the average hamming distance is equal to:

$$\text{Av(Hamming\_Distance)}=\frac{E(Total\_HD)}{N-1} \tag{4}$$

The density of the file is given by:

$$\text{E(Total\_File\_Density)}=(N-1)\times\sum_{x=0}^{r}x\times P(x) \tag{5}$$

The problem of having x attributes in a record that match the corresponding attributes in another record is a binomial experiment [4]. The success probability, p, for attribute j of a record to match the attribute j of its neighbouring record is constant and equal to 1/d {where d is the attribute range}.

Thus, the probability mass function of X is:

$$b(x;r,P)={}^{r}C_{x}\times P^{x}\times(1-P)^{r-x} \quad \text{where } P=\frac{1}{d}$$

Thus, plugging this formula in (1) and (2) for P(x), we get:

$$E(HD)=\sum_{x=0}^{r}(r-x)\times{}^{r}C_{x}\times P^{x}\times(1-P)^{r-x} \tag{6}$$

and

$$E(Total\_HD)=(N-1)\times E(HD) \tag{7}$$

5.2- Expected Hamming Distance for an Algorithm I Clustered File:

Since clustering algorithm one clusters according to a threshold value k, then the hamming distance between adjacent records is less than or equal to k. Thus, the expected hamming distance between any two consecutive records is:

$$E(HD) = \sum_{x=0}^{k} (k-x) \times {}^{k}C_{x} \times P^{x} \times (1-P)^{k-x} \text{ where } P = \frac{1}{d} \quad (8)$$

5.3- Expected Hamming Distance for an Algorithm II Clustered File:

The file generated by this algorithm consists of two components primarily, one of records clustered according to the queries in the query tree, and the second of records clustered according to the specified threshold value should they not be popular records. Thus, the expected hamming distance can be estimated as:

$$E(HD) = E(HD1) + E(HD2) + \cdots + E(HDk) + E(HDT) + \Delta \quad (9)$$

where $E(HDk)$ is the expected hamming distance between the records satisfying query k, $E(HDT)$ is the expected hamming distance between the records clustered according to the specified threshold value, and Δ is the expected hamming distance between consecutive queries.

Thus, $E(HDk) = (m-1) \times \sum_{x=0}^{zk} (zk-x) \times {}^{zk}C_{x} \times P^{x} \times (1-P)^{zk-x}$ (10)

where, $P = \frac{1}{d}$

zk is the number of zero keyterms in query k

and m is the number of records in the file clustered according to query k.

$E(HDT)$ is calculated according to (8) and $\Delta = E'(HDk) + \sum_{i=1}^{k-1} \delta_i$ where,

$E'(HDk) = (k-1) \times \sum_{x=0}^{zk} (zk-x) \times {}^{zk}C_{x} \times P^{x} \times (1-P)^{zk-x}$ {k is the number of queries}

δ_i is the hamming distance for the nonzero and nonequal keyterms between query i and query i+1.

5.4- Statistical Evaluation of Percentage Reduction in Retrieved Blocks:

Before clustering the file, the number of blocks retrieved when a query is issued is more than the number of blocks retrieved after clustering the file. This section deals with the derivation of formulas that estimate the reduction in the number of retrieved blocks after clustering the file.

5.4.1- Probability of a record satisfying a query:

Let

x be the number of keyterms in a query

d be the attribute range that the keyterms of the query can take

p be the probability that a record satisfies a query

then the probability that a randomly selected record satisfies a given query is obtained by:

$$P = \left(\frac{1}{d}\right)^x \tag{11}$$

5.4.2- The Probability of Y Records Out of N Records Satisfying a Query:

Let

p be the probability that a record satisfies a query. Then (1-p) is the probability that the record does not satisfy that query.

N be the total number of records of a file

Y be the number of records, out of these N records, that satisfy a query

Then, the probability that the Y records of the file satisfy the query can be obtained by:

$$P(Y) = {}^{N}C_{Y} \times p^{Y} \times (1-p)^{N-Y} \tag{12}$$

since the above experiment is binomial with identical, independent trials of constant probability which is obtained by (11), where y takes values from the range 0 to N.

5.4.3- Estimation of the Number of Blocks Retrieved Before Clustering:

Let Y be the number of records retrieved satisfying a given query. Then, since the records are stored in no particular fashion, the number of blocks retrieved could be any of the following values:

$$\left\lceil \frac{Y}{Blocksize} \right\rceil, \left\lceil \frac{Y}{Blocksize} \right\rceil + 1, \left\lceil \frac{Y}{Blocksiz} \right\rceil + 2, \ldots\ldots, \left\lceil \frac{Y}{Blocksiz} \right\rceil + \left\lceil \frac{Y}{Blocksize} \right\rceil \times (Blocksize) - 1, Y$$

where blocksize is the number of records in a block.

The latter term in the sequence could be refined such that it becomes $\left\lceil \frac{Y}{Blocksize} \right\rceil + Y - 1$, since the number 1 compared with blocksize is negligible.

If we assume each of the possibility of each of these values to occur, then the probability of the occurrence of each is constant and equal to :

$$\frac{1}{Y - \left\lceil \frac{Y}{Blocksize} \right\rceil + 1} \approx \frac{Blocksize}{Y \times (Blocksize - 1)} \approx \frac{1}{Y}$$

Then, the estimate of the number of blocks retrieved is the sum of the product of the possible number of blocks retrieved with their probability of occurrence. This will yield us through the following steps to the following formula:

$$E(Num\_of\_Blocks\_Ret) = \frac{1}{Y} \times \left[\left\lceil \frac{Y}{Blocksize} \right\rceil + \left\lceil \frac{Y}{Blocksize} \right\rceil + 1 + \left\lceil \frac{Y}{Blocksize} \right\rceil + 2 + \cdots + \left\lceil \frac{Y}{Blocksize} \right\rceil + Y - 1 + Y \right]$$

$$= \frac{1}{Y} \times \left[Y \times \left\lceil \frac{Y}{Blocksize} \right\rceil + \frac{Y \times (Y+1)}{2} \right]$$

$$= \left\lceil \frac{Y}{Blocksize} \right\rceil + \frac{Y+1}{2} \qquad (13)$$

5.4.4- Estimation of the Number of Blocks Retrieved After Clustering:

In the case of a clustered file, it is expected that similar records are put in contiguous physical locations such that a single bucket access will retrieve the maximum number of related records. Therefore, the main criteria that affect the number of blocks retrieved are: the number of records satisfying the corresponding query, the number of records that a block can contain, and the threshold value according to which the file was clustered.

Let

Y be the number of records that satisfy a given query

B be the number of records that a block can contain

then, to estimate the number of retrieved blocks we have the following two cases:

A- The issued query is a subset of, or a superset of, or one of the queries of the query tree which was considered while clustering the file:

1- If the issued query is one of the queries of the query tree then,

a- If Y<=B then the expected number of retrieved blocks is 1

b- If Y>B then the expected number of retrieved blocks is exactly $\left\lceil \frac{Y}{B} \right\rceil$

2- If the issued query is a subset or superset of one or more of the queries of the query tree, then the expected number of retrieved blocks is:

$$\left(\sum_{1}^{j} \left\lceil \frac{Y_{q^n}}{B} \right\rceil \right) + \left(\left\lceil \frac{Y'}{B} \right\rceil \right)$$

where,

j is the number of queries in the query tree which are subsets or supersets of the issued query

Y_{q^n} is the number of records that are clustered according to query q^n

n can take values between 1 and total number of queries

Y' is the number of remaining records that satsify the issued query and are clustered according to a specified threshold value

B- The issued query does not belong to the set of queries of the query tree:

1- If Y<=B then the expected number of retrieved blocks is 1

2- If Y>B then the expected number of retrieved blocks is $\left\lceil \frac{Y}{B} \right\rceil$

5.4.5- Expected Percentage Reduction in Retrieved Blocks:

Let

B_{before} be the expected number of blocks retrieved before clustering

B_{after} be the expected number of blocks retrieved after clustering

then, the percentage reduction in retrieved blocks is:

$$Percent\_Reduction\_in\_Blocks\_Ret = \frac{B_{before} - B_{after}}{B_{before}} \quad (14)$$

Plugging the different formulas of B_{after} and B_{before} obtained in the previous section, we can derive the formulas for the various cases.

Multiplying those formulas with the formula obtained in (12), we can estimate the expected percentage reduction in retrieved blocks for the various cases:

$$E(Percent\_Reduction\_in\_Blocks\_Ret) = \sum_{Y=0}^{N} \frac{B_{before} - B_{after}}{B_{before}} \times {}^{N}C_{Y} \times p^{Y} \times (1-p)^{N-Y} \quad (15)$$

Chapter Six

Regression Analysis

In the previous chapter we derived statistical formulas to estimate the expected hamming distance and percentage reduction in blocks retrieved for a given file. These formulas were for cases where we were sure that our algorithm would always find for every incoming record a position that will yield an increase in hamming distance that is less or equal to the specified threshold value. But, in reality this is not necessarily the case. What happens many times is that for the record in concern the algorithm will fail to find a position that will satisfy the threshold condition and, instead, it will insert the incoming record in a position that will yield the minimum increase in hamming distance. In this chapter, we will employ regression analysis to check whether we can use it on a large scale in estimating the hamming distance of a file before and after clustering and the percentage reduction in blocks retrieved given different parameters. All the calculations were done using the "STATISTICA" package under Windows 3.1.

6.1- Multiple Regression

In this section we will introduce multiple regression in brief. For more detailed explanation, you can refer to the Statistics reference provided in the list of references.

Multiple regression is a general statistical technique through which one can analyze the relationship between a dependent or criterion variable and a set of independent or predicted variables. It may be viewed as a descriptive tool by which the dependence of one variable on others is summarized. One of its most important uses is to determine the best linear prediction equation and evaluate its prediction accuracy.

The linear prediction equation, which is referred to as the regression line, is an estimation of the population parameter based on the sample observations. It is a linear function of the form:

$$Y_i = ß_0 + ß_1 x_{1i} + ß_2 x_{2i} + \ldots + ß_p x_{pi} + E_i$$

where, Y_i = dependant variable of the i^{th} sample

x_{ji} = j^{th} independent variable of the i^{th} sample

$ß_i$ = coefficient of the i^{th} term

E_i = error of estimation of the i^{th} sample

p = number of independent variables

Thus, given a set of sample observations, it is possible to find the best set of the $ß_i$'s such that the error of estimation is minimized. With this set of $ß_i$'s it would then be possible to predict the value of Y_i given a set of x_{ji}.

Finding the best prediction equation is not enough. What we have to do is to justify that our prediction equation of regressor results in accurate estimates and that the linear association between the dependent variable and the set of independent variables is strong.

A natural measure of prediction accuracy and the strength of linear association is the ratio of explained variation in the dependent variable Y to the total variation in Y. It is referred to as the coefficient of determination and equal to:

$$R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(\hat{y}_i - \bar{y})^2 + \sum(y_i - \hat{y}_i)^2}$$

where $\hat{y}_i$ is the estimated value of the i^{th} sample.

The regression line will have high prediction accuracy if R^2 is close to 1.

In order to prove that there exists a linear relationship between the dependent and independent variables, and that this relationship is not by chance, we have to carry another test that is called the significance test of regression coefficients. This significance can be evaluated by the following ratio:

$$R^2 = \frac{\left(\sum(\hat{y}_i - \bar{y})^2\right) \div p}{\left(\sum(y_i - \hat{y}_i)^2\right) \div (n - p - 1)}$$

A parameter Significant F (Sig F) is considered which is the level of significance of the hypothesis H_0: $ß_i=0$ for all i, and H_1: $ß_i<>0$ for one or more i. If Sig F is small, then the H_0 hypothesis is rejected and this means that there is a linear relationship between the dependent variable and the set of independent variables.

6.2- Estimate of Hamming Distance of a Non-clustered File:

The hamming distance of a file depends on whether the file has been clustered or not. There are also other factors that determine the exact value of the hamming distance. This is the number of records in the file and the number of attributes in a record. For a random file, the hamming distance will depend solely on the size of the file and the random range of attributes.

Therefore, we have as dependent variable the initial hamming distance of the file, and as independent variables the record size, file size, and range of attributes.

Using the data in Table 6.1, the regression model obtained from it is:

HD = 0.541 x record_size + 0.803 x file_size + 0.144 x attribute_range

The R_squared coefficient of determination is equal to 0.958 and F is equal to 541.571.

File Size	Record Size	Attribute Range	Initial HD	Final HD	Threshold
1000	4	4	2840	574	1
1000	4	6	3307	1822	2
1000	4	8	3509	2747	3
1000	5	4	3698	658	0
1000	5	6	4178	1890	2
1000	5	8	4360	3673	4
1000	6	4	4469	2652	3
1000	6	6	4993	2011	1
1000	6	8	5226	2450	0
1000	7	4	5204	4997	6

1000	7	6	5801	4569	5
1000	7	8	6116	3241	3
1000	8	4	6074	2473	0
1000	8	6	6604	3249	2
1000	8	8	6955	4031	4
600	4	4	1696	368	1
600	4	6	1977	1078	2
600	4	8	2096	1659	3
600	5	4	2188	463	0
600	5	6	2505	1175	2
600	5	8	2657	2197	4
600	6	4	2699	1595	3
600	6	6	2989	1322	1
600	6	8	3127	1628	0
600	7	4	3192	3023	6
600	7	6	3493	2730	5
600	7	8	3678	2073	3
600	8	4	3571	1487	0
600	8	6	4038	2141	2
600	8	8	4171	2501	4
800	4	4	2339	446	1
800	4	6	2646	1446	2
800	4	8	2750	2188	3
800	5	4	2999	621	0
800	5	6	3315	1535	2
800	5	8	3507	2941	4
800	6	4	3568	2131	3
800	6	6	4017	1679	1
800	6	8	4132	2033	0
800	7	4	4196	3954	6
800	7	6	4660	3704	5

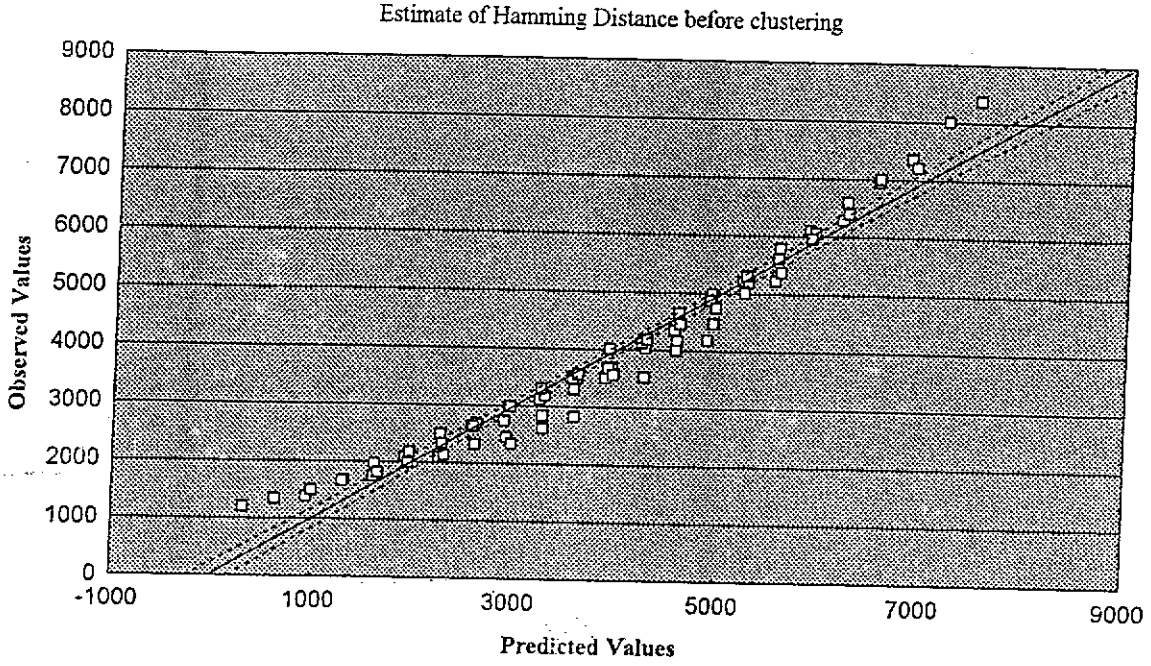
800	7	8	4872	2670	3
800	8	4	4753	1941	0
800	8	6	5315	2752	2
800	8	8	5618	3321	4
1200	4	4	3533	625	1
1200	4	6	4005	2144	2
1200	4	8	4190	3295	3
1200	5	4	4482	777	0
1200	5	6	5016	2254	2
1200	5	8	5236	4436	4
1200	6	4	5375	3175	3
1200	6	6	5967	2322	1
1200	6	8	6297	2878	0
1200	7	4	6410	6033	6
1200	7	6	7017	5508	5
1200	7	8	7362	3858	3
1200	8	4	7224	2680	0
1200	8	6	8024	3876	2
1200	8	8	8386	4867	4
400	4	4	1214	293	1
400	4	6	1349	734	2
400	4	8	1391	1107	3
400	5	4	1503	432	0
400	5	6	1665	787	2
400	5	8	1753	1463	4
400	6	4	1825	1078	3
400	6	6	2004	957	1
400	6	8	2095	1105	0
400	7	4	2143	2021	6
400	7	6	2326	1842	5
400	7	8	2462	1432	3

400	8	4	2350	1087	0
400	8	6	2621	1476	2
400	8	8	2828	1739	4

Table 6.1: Data of Multiple Regression of Hamming Distance

A graph of the observed values against the predicted values plotted by the Statistica package is:

Figure 6.1: Predicted vs. Observed Values



6.3- Estimate of Hamming Distance of an Algorithm I Clustered File:

If a file is clustered using algorithm 1, then the threshold value specified becomes a factor that determines the final hamming distance of that file. Therefore, it becomes one of the independent variables.

Therefore, we have as dependent variable the final hamming distance of a file and as independent variables the record size, file size, range of attributes, and threshold specified.

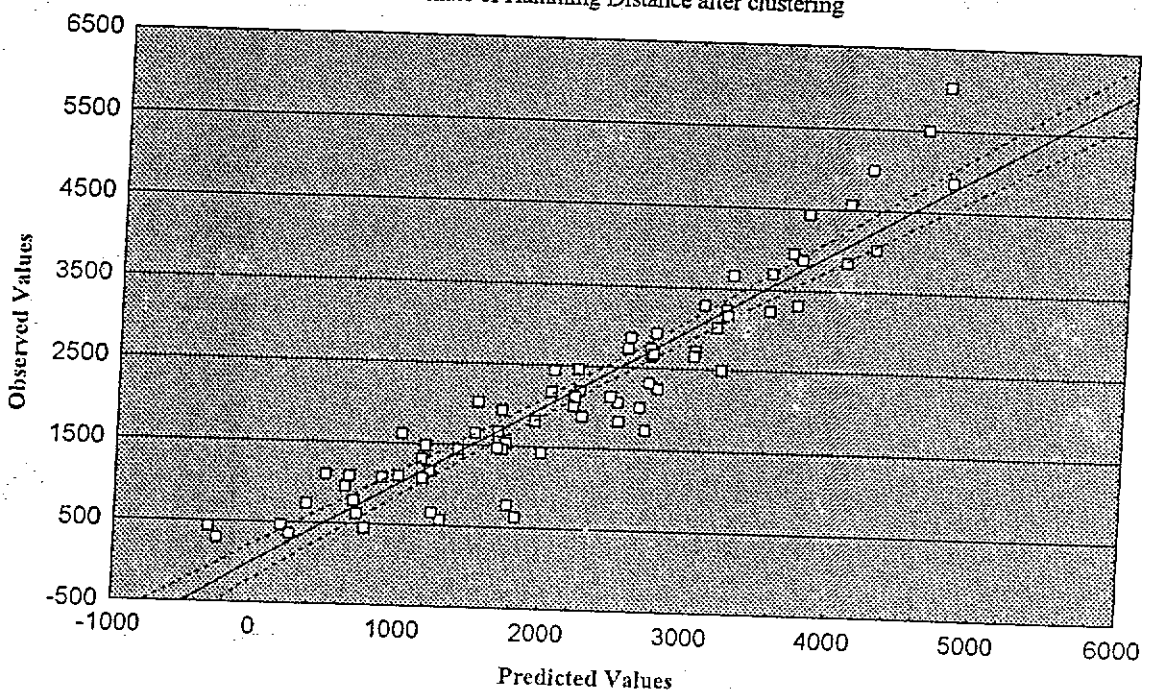
The data of Table 6.1 is used to calculate the regression coefficients of the variables and obtain the regression model. The model obtained is:

$$HDA = 0.360 \times record\_size + 0.572 \times file\_size + 0.155 \times attribute\_range + 0.536 \times threshold\_Value$$

The R_squared coefficient of determination is equal to 0.885 and F is equal to 129.176.

A graph of the observed values against the predicted values plotted by the Statistica package is:

Figure 6.2: Predicted vs. Observed Values



6.4- Estimate of Hamming Distance of an Algorithm II Clustered File:

By using the new algorithm for clustering a given file, a new factor adds to the list of factors that determines the final hamming distance of a given clustered file. That factor is the percentage of the records of the file satisfying the queries of the query tree. Thus, the new set of independent variables contains the record size, file size, range of attributes,

threshold specified, and the percentage of the records of the file satisfying the queries of the query tree.

To check whether a multiple regressor is suitable for the estimation of the final hamming distance of an algorithm II clustered file a new set of data shown below in Table 6.2 was used.

The regression model obtained is:

$$HDA = 0.512 \times record\_size + 0.671 \times file\_size + 0.321 \times attribute\_range + 0.121 \times threshold\_Value - 0.24 \times \%\_of\_records\ satisfying\ the\ tree$$

The R_squared coefficient of determination is equal to 0.916 and F is equal to 65.769.

File Size	Record Size	Attribute Range	Initial HD	Final HD	Threshold	% of Recs. Satisfying the query tree
1200	6	4	5335	1982	1	40
1200	7	6	6963	3227	2	30
1200	8	7	8115	4011	3	60
1200	6	5	5708	2268	0	40
1200	7	4	6245	2670	2	50
1200	8	8	8285	4247	4	70
1200	6	4	5171	2996	3	30
1200	7	5	6688	2932	1	40
1200	8	7	8086	3924	0	50
1000	6	4	4496	1707	1	40
1000	7	6	5788	2876	2	30
1000	8	7	6756	3514	3	60
1000	6	5	4756	2060	0	40
1000	7	4	5262	2267	2	50
1000	8	8	6917	3559	4	70

1000	6	4	4466	2591	3	30
1000	7	5	5608	2815	1	40
1000	8	7	6753	3554	0	50
800	6	4	3547	1375	1	40
800	7	6	4654	2263	2	30
800	8	7	5376	2730	3	60
800	6	5	3813	1751	0	40
800	7	4	4116	1830	2	50
800	8	8	5436	2700	4	70
800	6	4	3583	2065	3	30
800	7	5	4472	2014	1	40
800	8	7	5341	3150	0	50
600	6	4	2692	1047	1	40
600	7	6	3487	1844	2	30
600	8	7	3966	2186	3	60
600	6	5	2835	1401	0	40
600	7	4	3115	1514	2	50
600	8	8	4050	2157	4	70
600	6	4	2670	1554	3	30
600	7	5	3346	1776	1	40
600	8	7	3990	2144	0	50

Table 6.2: Data of Multiple Regression of Hamming Distance for Algorithm II

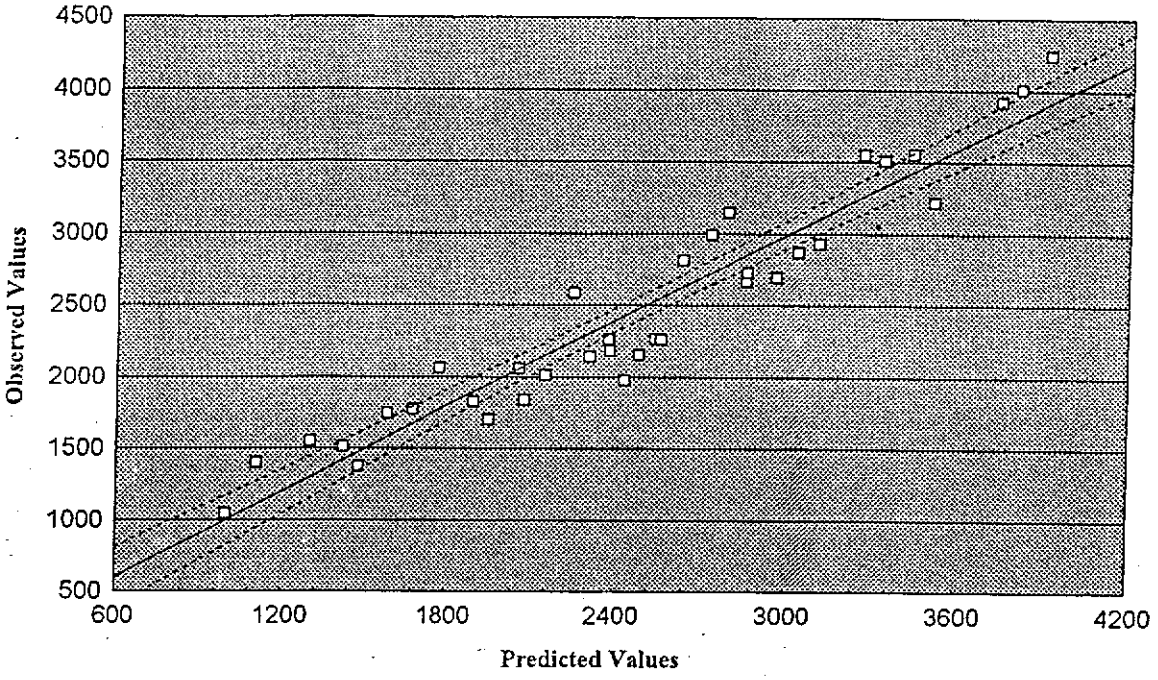
A graph of the observed values against the predicted values plotted by the Statistica package is shown on the next page.

6.5- Estimate of Percentage Reduction in Blocks Retrieved:

It is very important for us to know the percentage reduction in blocks retrieved after clustering a given file. By considering as dependent variable the percentage reduction in blocks retrieved and as independent variable the threshold value, record size, file size, block size, attribute range, number of keyterms in a query, and percentage of records

Figure 6.3: Predicted vs. Observed Values

Estimate of Hamming Distance after clustering



satisfying the queries of the query tree, and using the data of Table 6.3 we can obtain a multiple regressor and know whether regression analysis is suitable for that purpose.

The regression model obtained is:

%Reduction_in_retrieved_blocks = -0.184 x record_size + 0.410 x file_size - 0.102 x attribute_range - 0.296 x threshold _Value + 0.447 x %_of_records satisfying the tree - 0.099 block_size – 0.447 x Query_Terms

The R_squared coefficient of determination is equal to 0.784 and F is equal to 11.93.

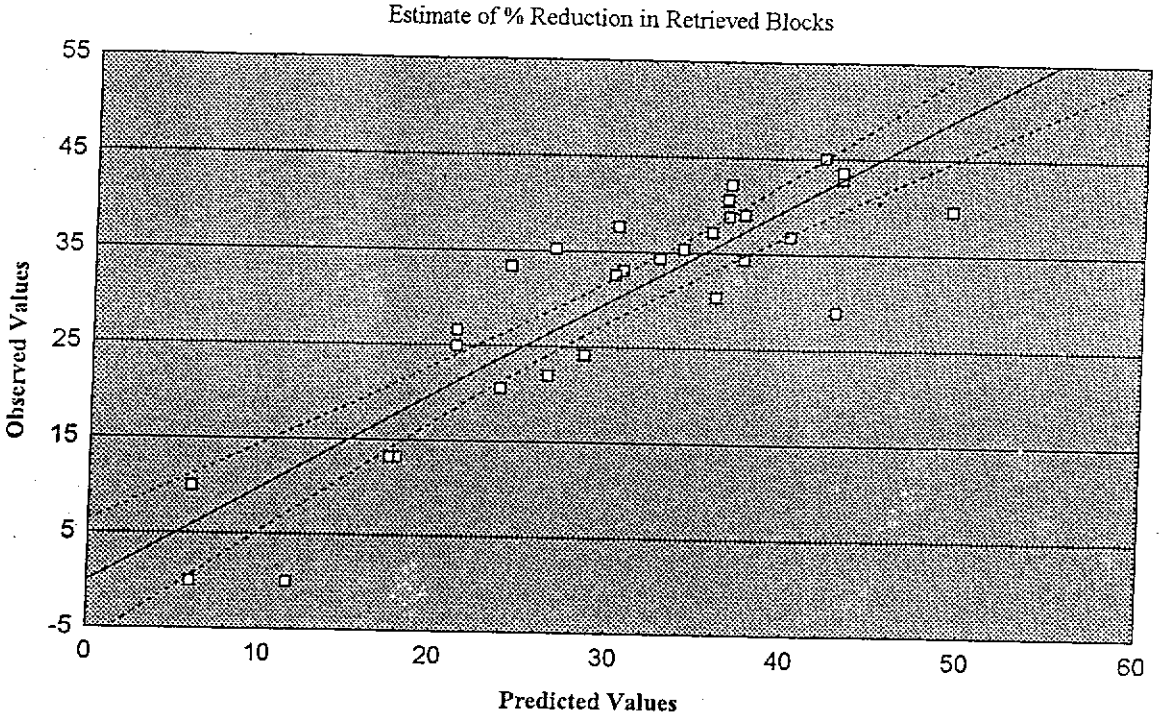
File Size	Rec. Size	Attrib. Range	% of Recs.	Query Terms	Block Size	Thre_ shold	Blocks Before	Blocks After	%Reduc. in Blocks Retrieved
1200	6	4	40	1	5	1	1785	1076	39.7
1200	6	4	40	2	10	1	567	324	42.9
1200	6	4	40	3	15	1	185	107	42.2

1200	6	4	40	2	10	1	550	310	43.6
1200	6	4	40	4	20	1	53	33	37.7
1000	8	7	60	1	5	3	1024	727	29.0
1000	8	7	60	2	10	3	220	131	40.5
1000	8	7	60	2	10	3	214	127	40.7
800	7	6	30	1	5	2	898	601	33.1
800	7	6	30	2	10	2	218	145	33.4
800	7	6	30	3	15	2	30	26	13.3
800	7	6	30	4	20	2	5	5	0.0
600	6	5	40	1	5	0	813	497	38.9
600	6	5	40	2	10	0	193	130	32.6
600	6	5	40	3	15	0	34	27	20.6
600	6	5	40	4	20	0	15	13	13.3
1000	7	4	50	2	5	2	1484	938	36.8
1000	7	4	50	1	20	2	350	192	45.1
1000	7	4	50	3	15	2	131	86	34.4
1000	7	4	50	4	10	2	33	25	24.2
800	8	8	70	2	5	4	809	508	37.2
800	8	8	70	1	20	4	90	59	34.4
800	8	8	70	1	20	4	87	53	39.1
1200	7	6	30	2	5	2	1388	897	35.4
1200	7	6	30	1	20	2	230	160	30.4
1200	7	6	30	3	15	2	50	39	22.0
1200	7	6	30	4	15	2	16	12	25.0
1200	7	6	30	4	15	2	15	11	26.7
600	8	7	50	3	10	0	9	9	0
600	8	7	50	1	10	0	487	315	35.3
600	8	7	50	3	10	0	10	9	1.0

Table 6.3: Data of Multiple Regression of Reduction in Retrieved Blocks

A graph of the observed values against the predicted values plotted by the Statistica package is:

Figure 6.4: Predicted vs. Observed Values



We realize that the above regression lines all produced large R-squared values and their Sig F's are zero. It therefore shows that their prediction accuracy and significance of regression coefficients are satisfactory. The graphs produced by the Statistica package further confirm that the predicted and observed values are very close to one another.

Thus, this illustrates that, within the context of concern in this work, regression analysis is a reliable tool in predicting the hamming distance of a clustered and non-clustered file, and the resulting reduction in retrieved blocks.

Chapter Seven

Conclusion

Clustering of a file using the new algorithm reduces the number of blocks needed to be retrieved and consequently improves the response time of a system. Choice of a threshold value and pattern of the most frequent queries are critical in obtaining an efficient improvement. We have used statistical formulas for estimating the behavior of the clustering algorithms and their impact that can be used to obtain information prior to any actual clustering of the file. Further research may concentrate on finding a model that predicts the best combination of the parameters that govern the practical response time. To this end, regression techniques are under investigation. A further improvement for the presented algorithm, we could check the correlation of the queries with each other so that records that satisfy overlapping frequent queries be put in close physical locations on storage media.

Appendix A

Program Design

Generate

This program generates a random history tree of queries by making use of a random number generator; it saves the queries of the tree in a file. It assigns random frequencies to every query. It then generates a main random file where the number of records satisfying the queries of the query tree is based on a specified percentage by the user. Then, the program starts the record matching process by reading the records of the main file traversing through the tree and assigning each to the most frequent query it satisfies. It saves all the information in a query information file. It outputs the number of comaprisons done during the matching process. The input required by the program is:

- Name of the main file
- Name of the query file
- Name of the query information file
- Percentage of records of the main file satisfying the queries
- Record size
- Block size
- Attribute range
- File size

ClusterT

This program clusters the main file using algorithm 1 based on a specified threshold value. After clustering the file, it outputs the total number of comparisons made during clustering and the total number of retrieved blocks. The input required by the program is:

- Name of the file to be clustered
- Name of the new clustered file
- Hamming distance threshold value

ClusterQ

This program clusters the main file using clustering algorithm 2 based on a specified threshold value and according to the query a record is assigned to. If a record is not assigned to a query it is clustered according to the threshold value, else it is put at the end of the current sequence of records satisfying the same query. After clustering the file, it outputs the total number of comparisons made during clustering and the total number of retrieved blocks. The input required by the program is:

- Name of the file to be clustered
- Name of the query information file
- Name of the new clustered file
- Hamming distance threshold value

CalcHD

This program calculates the total hamming distance of a file. It requires the name of the file as input.

GenQry

This program generates a random query file. It requires the following input:

- Name of the query file
- Size of the query file
- Number of keyterms in a query
- Attribute range

MatchTo

This program matches the queries of a query file to the records of a given file and outputs the total number of retrieved blocks and the number of matching records. It requires the following input:

- Name of the query file
- Name of the main file

Appendix B

File Naming Format

A specific format for naming the files during the experiments was followed. The files that were generated during the simulation experiments were:

- Main random file
- Query file of query tree
- Query information file
- Query file to match against the records of clustered files
- Clustered files by both algorithms

Main Random File

e.g. T_8_6_40

T : stands for thousand records in the main file

8 : stands for 8 attributes per record

6 : stands for attribute range between 1 and 6

40: stands for 40% of the records satisfy the generated query tree

Query File of Query Tree

e.g. T_8_6_40.Q1H

T_8_6_40: is the name of the main random file

Q : stands for query

1H: stands for the generated query tree has 100 nodes

Query Information File of Query Tree

e.g. T_8_6_40.I1H

T_8_6_40: is the name of the main random file

I : stands for information

1H: stands for the generated query tree has 100 nodes

Query File to Match Against the records of the Clustered Files

e.g. Query1.H

1 : stands for 1 keyterm per query

H: stands for hundred queries in the file

Clustered File with Algorithm 1

e.g. T_8_6_40.T1

T_8_6_40: is the name of the main random file

T: means that clustering was done purely based on the treshold value

1: stands for the specified threshold value for clustering

Clustered File with Algorithm 2

e.g. T_8_6_40.Q1

T_8_6_40: is the name of the main random file

Q: means that clustering was done based on the query tree and the threshold value

1 : stands for the specified threshold value for clustering

References

1. Croft, B., **Clustering Large Files of Documents Using the Single Link Method**, Journal of the American Society for Information Science, November 1977.

2. Devoe, J.L., **Probability and Statistics for Engineering and the Sciences**, Third Edition, Duxbury Press, 1991.

3. Kang, A.N.C., et al., **Storage Reduction Through Minimal Spanning Trees and Spanning Forests**, I.E.E.E. Transactions on Computers, 1977.

4. Lowden, B.G.T., **An Approach to Multikey Sequencing in an Equiprobable Keyterm Retrieval Simulation**, Proceeding of the 8th ACM SIGIR Conf. on Research and Development in Information Retrieval, 92-96, Montreal, 1985.

5. Lowden, B.G.T. and Kitsopanidis, A., **Enhancing Database Retrieval Performance Using Record Clustering**, Essex, UK, University of Essex, 1993.

6. Omiecinski, E., et al., **Performance Analysis of a Concurrent File Reorganization Algorithm for Record Clustering**, IEEE Transactions on Knowledge and Data Engineering, Vol.6, No.2, 248-257, 1994.

7. Safar, M.A., **Clustering Records in Information Retrieval Systems**, M.Sc. Dissertation, Beirut, Lebanese American University, 1995.

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

9. Moghrabi, I.A.R., **Analysis of Algorithms for Clustering Records in Document Databases**, Lebanese Scientific Research Reports, Vol.1, No.2, 33-44, 1996.

10. Williamson, R.E., **Realtime Document Retrieval**, Ph.D. Thesis, Ithaca, NY, Cornell University, 1974.

11. Yu, C.T., et al., **Adaptive Record Clustering**, ACM Transactions on Database Systems, Vol.10, No.2, 180-204, 1985.