Intelligent Optimized Mining Pattern For Genomic Database Via Clustered Indexing Method

S.Thabasu Kannan

ABSTRACT

Genome datasets have been growing exponentially in the past few years. The GenBank database, for example, will be doubled every 15 months. With this rapid growth, genome datasets and the associated access structures have become too larger to fit in the main memory of a computer. It leads to a large number of disk accesses. Therefore, slow response times occurred for homology searches and other queries. In the circumstances, we should take all possible efforts to develop proper tools to access the data and mine them efficiently. Otherwise, Data mine will be wasted that resulted in prolonging of search time and lack of efficiency.

This paper proposes a new architecture for fast searching of genomic databases efficiently and effectively. In this paper, projected clustering algorithm is used as a first and foremost component of proposed model for effective clustering. Because it supports heavily to cluster high dimensional data. However, most of the existing projected clustering algorithms depends always on some critical user parameters in determining the relevant attributes of each cluster. In case wrong parameter values are used, the clustering performance will be seriously degraded. It is unfortunate that correct parameter values are rarely known in real datasets. It does not depend on user inputs in determining relevant attributes. However, it responds to the clustering status and adjusts the internal thresholds dynamically. This algorithm displays a much higher usability than the other projected clustering algorithms and also works well with a gene expression dataset.

For the second component of proposed model, a new metric index, called M+-Tree is proposed and organized for large datasets in metric spaces. Because M+-tree takes a new concept called key dimension, which effectively reduces response time for similarity search. The main idea behind is to make the fan-out of tree larger by partitioning a subspace further into two subspaces, called twin-nodes. By utilizing the twin-nodes, the filtering effectiveness can be doubled. In addition, for ensuring high space utilization, data will be reallocated dynamically between the twin nodes.

Keywords: Cluster, Indexing, Searching, Genomic, DNA, Protein

Prof., Head, Dept of Computer Science, Sourashtra College, Madurai, Tamilnadu, India 625 004
thabasukannan@gmail.com

1. INTRODUCTION

Bioinformatics is fast becoming oft-uttered buzzwords these days. Bioinformatics is nothing but good, sound, regular biology appropriately dressed so that it can fit into a computer. Bioinformatics is about searching biological protein structures and more generally, asking biological questions with a computer.

Searching for sequence similarities

Searching [1] is mainly designed to examine and show how they are similar. This availability of public databases is designed to search the identity homologous sequences from unknown sequences. This task has to be performed in a particular sequence. The sequential data processing arranged in order and leads to a particular result. We can reach homology sequence as the basis of similarities calculated. From what we already known, we can find out indirectly homologous sequences. The clear inference is that, two DNA /Protein sequences are either homologous or not. Two sequences share significant similarity scores of 15% to 30% identity is considered as homologous. Even

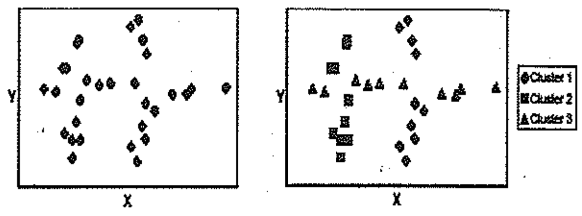
similarity sequences is less than the above 15% to 30%, we may not treat it as non homologous.

Local vs. Global

In the local alignment model [1], portions of two sequences should be aligned. But the portions should be similar; dissimilar portions should not be used in alignment. It is not very useful for near matching subsequences. But in Global alignment model, two sequences should be aligned sequence by sequence. Here complete and closely related sequences are compared from beginning to end. It is not very useful for more distantly related sequences.

Projected Clustering

Clusters are the group of things of the same type that grow and appear close together. But in projected clustering, clusters exist in subspaces of the input space defined by the *dimensions* of the dataset. The similarity between clusters can only be recognized in the specific subspaces. A dataset is possessing of a number of projected clusters. Each and every projected clusters takes form in subspaces. However, they are easily and distinctly seen.



(a) A set of 2D points. (b) 1-D projected clusters.

An example illustrating the idea of projected clustering

Stud	*MFC*	*FA*	*C++*	*PIT*
1	89	50	82	88
2	90	70	66	91
3	91	83	50	90
4	70	90	92	76
5	55	65	90	63
6	79	54	99	50

Stud	*1*	*2*	*3*	*4*	*5*
2	26				
3	46	21			
4	47	41	50		
5	46	51	63	32	
6	43	61	61	46	31

(a) Data records (b)Distance between different records Projected clusters[6] in a virtual examination score dataset

M+-TREE

M$^+$-Tree [2] is a dynamical paged and balance tree. It combines binary MVP-tree (Multiple Vantage Point) and M-tree. It significantly improves the partition of binary MVP tree and the node structure of M-Tree. Here node means a point at which two lines or two systems meet or crossed. In binary MVP-tree, a data space is partitioned into four subspaces and two vantage points. Here vantage point is a position from which we can watch something. In M+-tree, the partition is done through one vantage point and a key dimension. The main idea of key dimension is to make the fan-out of tree larger. Because there is no distance computation for partitioning data space by key dimension. Twin node is partitioning of a space further into two subspaces. Twin node is performing well in effective, optimal and efficient filtering.

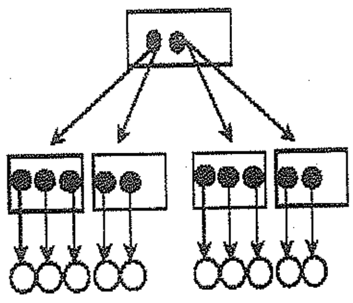


Figure1 : The M+ Tree Structure

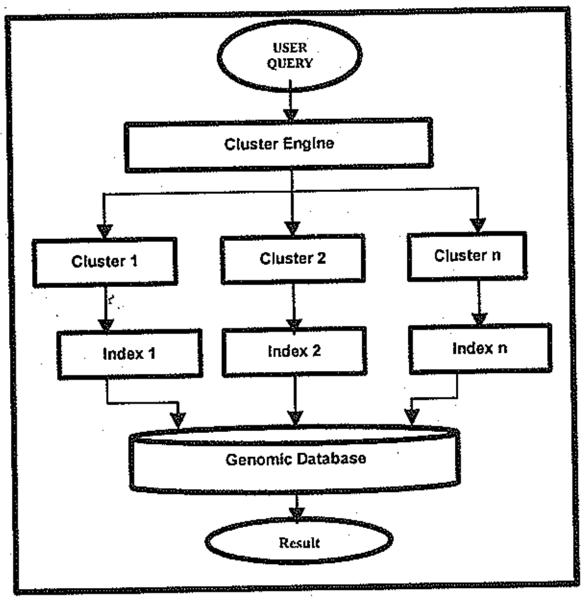


Figure2: The Architecture of the new model

The architecture of the proposed model draws on the structure of the four spaces. It works well for getting the efficiency and robustness query about data management. Query provided by Clustered Indexing Algorithm is always supporting for getting optimum level of relevant result. This model totally exists in a hybrid space formed by user influences, segregation, indexing and data. As shown in the figure of proposed model, user query normally comes to cluster engine. From which it gets segregated as n number for fast access after indexing only it reaches to Genomic databases for searching and finally the outcome of relevant result is displayed.

2 The Components Of The Shanuchi Model

A Genomic Database is storing data suitably in to the genomes. This is actually a Data Mine but not a data warehouse. Multidimensional Cluster engine, segregates all relevant data for fast access. It is considered for a way entering and reaching a place. Simply saying database gains access through a multidimensional cluster engines.

A multidimensional Cluster engine sends a query of user to individual Clusters. Individual Clusters after segregating all queries with multidimensions, send it the local Index. Here all clustered indexes are attached to the local individual clusters. This method is evolved to the need of user in data analysis. It is evolved simultaneously to the growth of user needs. It provides definitely executives, scientists, and analysts, a valuable information via "slice, dice and rotate" method. In complicated queries, this method access data queries efficiently by augmenting or replacing complications. This slice dice and rotate provides the following features:

- To ensure user consistency
- To access fastly to a wide variety of data.
- To organize data by key selection criteria.
- To match real dimensions of the modern organizations.
- To analyze multidimensional organizational data including complex calculations, modeling and simulation.
- To enable the performer adhoc analysis of data in multiple dimensions.
- To give user the insight and understanding for better decision-making.

2.1 Query:

Genomic Database consists of a set of nucleotide sequences. An important kind of queries is the local sequence alignment. The relevance of sequence alignment is derived from evolutionary relationship between sequences. The sequence database is to find for a given sequence *x*, all sequences *p*. In the database such that the alignment score between *x* and *p* is greater or equal to a given threshold σ [4][7]. Threshold is a level at which something starts to happen.

The most efficient algorithms for aligning two sequences with total length L need time O(L log L). The most existing tools exhaustively compare query sequences with each sequence in the database. This exhaustive search techniques become prohibitively expensive. Because more volume of data needed to process for each query [7].

2.2 Cluster Engine:

This is a highly accurate engine for homology searching. This engine is doing the task of homology searching in an exact way. Prior to searching, all queries are to be filtered to mask low complexity regions. This may reduce retrieval effectiveness. This cluster Engine provides access to the hybrid space of influences over data and segregations, indexing and displaying user defined results. Here we describe our new projected clustering algorithm HARP (a Hierarchical approach with Automatic Relevant dimension selection for Projected clustering).

HARP Algorithm

It is an agglomerative hierarchical clustering algorithm based on greedy merging of the most similar clusters. HARP is classified as a projected clustering algorithm and it determines object similarity based on their distance in the projected subspace. In other words, given a cluster of objects, the relevance of a dimension in the cluster is related to the average distance between the projected values of the member objects on the dimension. So relevance is directly measured by this average distance. This may not be appropriate if the input dimensions have different ranges of values.

The skeleton of the whole algorithm is shown below. At the beginning of the clustering process, each object forms a singleton cluster. The dimensionality and relevance thresholds d_{min} and R_{min} are initialized to their tightest values. For each cluster, the dimensions that satisfy the threshold requirements are selected. The merge score between each pair of clusters is then calculated. Only the merges that form a resulting cluster with d_{min} or more selected dimensions are qualified. The other merges are being ignored.

Algorithm HARP (k: target no. of clusters (default: 1))

```
1.  For step := 0 to d - 1 do {
2.  dmin := d - step
3.  Rmin := 1 - step/(d - 1)
4.  Foreach cluster CI
5.  SelectDim(CI , Rmin)
6.  BuildScoreCache(dmin, Rmin)
7.  While cache is not empty {
8.  // CI1 and CI2 are the clusters involved //in
    the best merge, which forms the //new cluster
    CI3
9.  CI3 := CI1 ∪ CI2
10. SelectDimNew(CI3 , Rmin)
11. UpdateScoreCache(CI3 , dmin, Rmin)
12. If clusters remained = k
13. Goto 16
14. }
15. }
16. ReassignObjects( )
End
```

The HARP algorithm.

```
Procedure buildScoreCache(dmin: dim.threshold,Rmin: rel.threshold)
1.  Foreach cluster pair CI1 , CI2 do {
2.  CI3 := CI1 ∪ CI2
3.  SelectDimNew(CI3, Rmin)
4.  If dI3 ≥ dmin
5.  Insert MS(CI1 ,CI2) into score cache
6.  }
End
```

The score cache building procedure.

The algorithm repeatedly performs the best merge according to the MS scores of the qualified merges. In order to efficiently determine the next best merge, merge scores are stored in a cache. After each merge, the scores related to the merged clusters are removed from the cache, and the best scores of the qualified merges that involve the new cluster are inserted back. The selected dimensions of the new cluster are determined by its members according to R_{min}.

```
Procedure SelectDim(CI : target cluster, Rmin: rel.threshold)
1.      Foreach dimension vj
2.      If RIj ≥ Rmin and ValidRel(CI , vj)
3.      Select vj for CI
    End
```

The dimension selection procedure for an existing cluster.

```
Procedure SelectDimNew(CI3 : target cluster, Rmin: rel.threshold)
1.      Foreach dimension vj {
2.      // CI3 is a potential cluster formed by merging
    CI1 and CI2
3.      If R_Ij ≥ Rmin and ValidRel(CI1 , vj) and
    ValidRel(CI2 , vj)
4.      Select vj for CI3
5.      }
    End
```

The dimension selection procedure for a new cluster.

```
Procedure ValidRel(CI : target cluster, vj : target dimension)
1.      lowv := max(xIj - 2σIj , minIj)
2.      highv := min(xIj + 2σIj , maxIj)
3.      If mean frequency of the bins covering [lowv,
        highv] < mean frequency of all bins
4.      return false
5.      Else
6.      return true
        End
```

The relevance index validation procedure.

```
Procedure UpdateScoreCache(CI3 : new cluster,dmin: dim.threshold, Rmin: rel.threshold)
1.      // CI3 is formed by merging CI1 and CI2
2.      Delete all entries involving CI1 and CI2 from
    cache
3.      Foreach cluster CI4 ≠ CI3 do {
4.      CI5 := CI3 ∪ CI4
5.      SelectDimNew(CI5 , Rmin)
6.      If dI5 ≥ dmin
7.      Insert MS(CI3 ,CI4) into score cache
8.      } End
    End
```

The score cache updating procedure.

According to the definition of R, if a dimension is originally not selected by both merging clusters, it must not be selected by the new cluster. However, if a dimension is originally selected by one or both of the merging clusters, it may or may not be selected by the new cluster.

We implemented three kinds of cache structures: priority queue similar to the one used in, quad tree, and Conga line.

- *Quad tree* is optimal in access time, which takes O(N) time per update, but it needs $O(N^2)$ space.

- *Conga line* is best for very large datasets as it takes only O(N) space, but it needs O(N logN) time per insertion and O(N log2 N) time per deletion.
- *Priority queue* takes worst case O(N^2) space, but due to the two thresholds of HARP, usually only a small fraction of the O(N2) cluster pairs are allowed to merge, so the actual space being used is much lower. The time for each update is O(N logN).

Depending on the memory available, HARP chooses the best cache structure to use and all structures give identical clustering results. Whenever the cache becomes empty, there are no more qualified merges at the current threshold level. The thresholds will be loosened linearly according to the formulas in lines 2 and 3 of the first algorithm. Further rounds of merging and threshold loosening will be carried out until a target number of clusters remains, or the thresholds reach their baseline values and no more qualified merges exist.

The whole algorithm requires no user parameters in guiding dimension selection or merge score calculation, so it can easily be used in real applications. The high usability is attributed to the dynamic threshold loosening mechanism, which relies on the hierarchical nature of HARP. The parameter k that specifies the target number of clusters is optional. Like other hierarchical clustering methods, k can be set to 1 and the whole clustering process can be logged as a dendrogram, which allows users to determine the cluster boundaries from a graphical representation. These advantages justify the use of the hierarchical approach in spite of its intrinsic high time complexity. HARP is especially suitable for applications where accuracy is the first priority and the datasets are of moderate sizes, such as gene expression profiles. In order to deal with very large datasets, we will discuss some speedup methods in the next section.

Finally, we describe the outlier handling mechanism of HARP. It contains two phases. *Phase one* is performed when the number of clusters remained reaches a fraction of the dataset size. Clusters with very few objects are removed. *Phase two* is performed near the end of clustering to prevent the merging of different real clusters due to the existence of noise clusters. The time to perform phase one outlier removal is critical. Performing too early may remove many non-outlier objects, while performing too late may have some outliers already merged into clusters. HARP performs phase one relatively earlier so that most outliers are removed, possibly together with some non-outlier objects. Before phase two starts, each removed object is filled back to the most similar cluster subject to the current threshold requirements. Due to the thresholds, real outliers are unlikely to be filled back.

2.3 Indexing:

In many applications today, however, the existence of an index is vital for retrieving information. This model investigates the indexing technique called M+ tree, to support fast query evaluation for the local alignment by transforming the alignment problem to a variant metric space neighborhood search problem. M^+-Tree is a tree dynamically organized for large datasets in metric spaces. It takes full advantages of M-tree, [3] with a new concept called *key dimension*, which effectively reduces response time for similarity search.

The main idea behind the key dimension is to make the fanout of tree larger by partitioning a subspace further into two subspaces, called twin-nodes. We can double the filtering effectiveness by utilizing the twin-nodes. In addition, for the purpose of ensuring high space utilization, and also conduct data reallocation between the twin nodes dynamically. This model use indexing to provide efficient access to the data. For the efficient indexing method we are using M+Tree [3] for our proposed model. Now we will explain about the construction of M+tree; Insertion and splitting.

Construction of an Index (M+-tree)

A new object is inserted into M+-tree in the following way. First, from the root, the appropriate node is found. If the twin nodes are all full, a split is needed. The split is performed in two steps. First is the splitting based on the distances among objects, and the other is the splitting based on the key dimension. If the node is full but its twin node is not full, then the entries of the twin nodes are reallocated. The process of reallocation is the same as that of the splitting with the key dimension. If the node is not full, the object is inserted directly.

Now, the tree construction and node split algorithms are given in the following. In the insert algorithm, entry(On) is an entry to be inserted and N is a node into which the entry is inserted.

Algorithm Insert (N.node.entry(On).M+-tree_entry)

```
Begin
// let N be the set of entries in Node N;
if N is Not Leaf   // if N is a internal node
      Node=ChooseSubtree(N);
      Insert(Node, entry(On) );
        else  // if N is a leaf node
      if N is Not Full
        StoreEntry(N, entry(On) );
      else
          if TwinNode(N) is Not Full
          Reallocate(N, TwinNode(N),entry(On));
          else
             Split(nEntry, entry(On) );
          end if
      end if
   end if
 end
```

M+-tree performs node split in a bottom-up fashion. It shares the promotion and partition mechanism with M-tree. At the same time, M+-tree performs node split by two steps: splitting with m-RAD-2 like in M-tree and splitting with key dimension. The split algorithm of M+-tree and its input parameters are described as follows.

Algorithm Split (N.nEntry.E.M+-tree_entry)

```
          begin
            Nt=entries(*N->leftTwinNode).
            entries(*N->rightTwinNode).{E};
           if N is not root
      let Op let the parent of N, stored in Np node;
           end if
            Allocates a new node N'
            promote(N,Op1,Op2);
            partition(N,Op1,Op2,N1,N2);
            keydimSplit(N, N1);
            keydimSplit(N',N2);
            if N is the current root
            Allocate a new root node: Np;
         Store entry(Op1) and entry(Op2) in Np.
               else
              Replace entry(Op) with entry(Op1);
                   if node Np is full
                      if the twin node of Np is full
         Split(NP,entry(Op2));//NP: entry to Np
                      else
               Allocate the entries of parent twins
                      else
                           store entry(Op2) in Np.
                      end if;
                  end if
               end if
           end
```

2.4 Databases

Sequence databases [8] are among the most important information repositories in molecular biology. Two types of sequences are found in sequence databases: nucleotide sequences with four-letter alphabet, and amino acid residue sequences with a twenty-letter alphabet.

Because of the high rate of data production and the need for researchers to have rapid access to new data, public databases have become the major medium through which genome sequence data are published. Public databases and the data services that support them are important resources in bioinformatics, and will be essential sources of information for all the molecular biosciences.

EMBL and GenBank [8] are the two major nucleotide databases. EMBL (European Molecular Biology Laboratory), is the European version and GenBank is the American. EMBL and GenBank collaborate and synchronize their databases so that the databases will contain the same information.

For molecular biologists it is thus extremely important to have access to public databases with the most recent information. The most widely known and used databases of

- DNA and RNA sequences [1] are GenBank and EMBL (European Molecular Biology Laboratory), DDBJ (Databank of Japan) and GSDB (Genome Sequence Database).
- ?Protein sequences [1] are SWISS-PROT and PIR (Protein Identification Resource)
- ?Macromolecular structures is PDB (Protein Databank)

In addition, there are numerous specialized databases covering a diverse array of areas. Taken together the databases provide valuable collections of organized data that is of broad benefit to the molecular biologists. However, the number and diversity of information resources makes efficient data resource discovery as important as effective data resource use. Thus, tools and systems to assist the researcher in navigating the biological data are increasingly important.

BLAST and **FASTA** are used for finding sequence similarities between a selected piece of DNA and the sequences contained in the databases. These report the hit in the database along with the estimated statistical significance of the hit.

According to *probability theory*, the similarity score required for a given level of statistical significance is proportional to the logarithm of the database size. Therefore, as the database grows, biologically significant matches of distantly related sequences may have smaller similarity scores than random matches, and may be lost in the noise. One approach to this problem is to simplify the database by eliminating redundant sequences or by reducing families of similar sequences to a single representative sequence in the database that can be used for the initial searching.

More recently the databases have shared and integrated information from a variety of databases to produce complex, comprehensive sites with access to information in most databases from a single site, and in some cases an integrated database.

3. Conclusion

This 21st century marks the apex of a new era in computer arena. We are really excited at the prospects of computer advancements. We are living in the golden age of computer databases. In this paper HARP is projected as a new clustering algorithm. HARP is deserving to be accepted as one of the best or most important of its kind. All the feaures are elegantly designed and exemplary one. It is very clear that there is a growing need for fast efficient searching of genomic databases. There are good opportunities available for utilizing computer databases, if we have the right skill or right algorithm. Clustered Indexing (CI) is a computer-based algorithm fully aims at solving major challenges of the projected clustering problems. In current databases billions of nucleotide base that are queried 10,000 of times per day. This rapid growth of databases cover increasing number of species and potentialities. We aim at achieving to reach trillions of nucleotides bases within the next few years.

CI is one of the good triumph of modern computer database algorithm. It is considered as an excellent example of how successfully fast searching can be done by this algorithm. Also it does not depend on user input in determining relevant dimensions of clusters. CI makes use of relevant index, histogram based validation and dynamic thresholds.

CI is dynamically adjusted the merging requirements of clusters according to the current clustering status.

In this paper, various problems of finding clusters in subspaces have been reviewed and some approaches to solving existing problems are proposed. In addition, clustering problems have been dealt with analytically and suggesting reasons for dependence of some existing projected clustering algorithms user's parameters.

CI makes databases very practically and also very useful where correct values of the parameters are unknown in databases. CI works efficiently well within the limit already existed.

CI can be extended to pattern based clustering and to produce nondisjoint clusters by adaptive mean-centering and post-clustering object assignment.

In near feature M+Tree may be replaced by some optimized algorithms like Ant Colony algorithm, Tabu Search, Genetic Algorithm, Particle Swarm Optimization, Memetic algorithm, because it may have so many advantageous features than the former.

This method definitely will prove a tool to optimize the output of all queries related to genomic databases. It may be very advantageous to solve even in complex queries. It will provide to ensure high dimensionality. We can surely be rely on the relevance since the occurrence of failures is found less in number by comparison of sequences in more detailed. The only drawback of this system noticed is that the time taken for searching the sequences is more. We can entirely rely on the output, which is very decisive.

References

1. S.Thabasu Kannan, Efficient and effective mining of bioinformatics database via Clustered Indexing, UGC-GRI proceeding.
2. S.Thabasu Kannan, Optimized mining of bioinformatics database via clustered indexing methods, IAENG, IMECS, Hongkong proceedings.
3. S.Thabasu Kannan, Intelligent Optimized Mining of Genomic Database Via Clustered Indexing Method, IKE conference, Las Vegas, USA
4. S.Thabasu Kannan, Effective querying method for Genomic Database, M.Phil dissertation.
5. S.Thabasu Kannan, Shift in Bioinformatics from large to small database, TCE Conference.
6. S.Thabasu Kannan, Biomine via Clustered Indexing, JMC Proceeding.
7. S.Thabasu Kannan, Knowledge Based Query Processing of Bioinformatics Database Via Clustered Indexing Method, NCKM&DM conference, PSNA proceeding.
8. S.Thabasu Kannan, Fixing Knowledge to Query Processing sequential analysis in Genomic Database Via Clustered Indexing Method, IADIS conference, Dublin, Ireland.

9. Anderson and A. Brass. Searching DNA databases for similarities to DNA sequences: when is a match significant?
10. Williams and Zobel: Indexing and Retrieval for Genomic Databases
11. T. Kahveci, A.K. Singh. An efficient index structure for string databases. In: VLDB'01, pp. 351–360. Morgan Kaufmann, San Francisco, Calif., 2001
12. H.E. Williams and J. Zobel. Indexing nucleotide databases for fast query evaluation. In *Proc. International Conference on Advances in Database Technology (EDBT).*
13. D.A.White and R.Jain. *"Similarity Indexing with the M+-tree,"* Proc.of the 12th Int.conf. on Data Engineering, New Orleans, USA, pp.516-523. T.Bozkaya,M. Ozsoyoglu. "*Distance-based indexing for high-dimensional metric spaces.*" Proc. The ACM SIGMOD International Conference on Management of Data,Tucson, Arizona, page 357-368.

Author's Biography

Thabasu S. Kannan, received the Master of Computer Applications in the year 1993 and took his M.Phil degree in 2003. He is doing his Ph.D degree in Bioinformatics. His current research interests include algorithms, data mining and information retrieval. He is interested in the use of data mining techniques for Bioinformatics, Commerce Applications, and GIS, which needs large data sets. He in recent years publishes more than 15 papers in various international / national journals / proceedings. His recent papers have been selected one for IMCECS 2006 conducted by IAENG, HongKong, and another for WCIC2006 conducted by MIT, Texas, USA. Under his guidance 8 scholars were awarded M.Phil degree, and 13 are doing. He has the credit of authorship of three text books in Computer Science. He serves as a referee for some Journals in Computer Science and Bioinformatics. He is Head of the Dept of Computer Science, Sourashtra College, Madurai, India. He is a member of Board of Studies of various Universities and Autonomous Colleges.