Data Mining Using the Crossing Minimization Paradigm

A Thesis

Submitted to the

University of Stirling

for the Degree of
Doctor of Philosophy

by

Ahsan Abdullah

Department of Computing Science and Mathematics

University of Stirling

Stirling. FK9 4LA

Scotland

2007

Declaration

I, Ahsan Abdullah, hereby declare that this work has not been submitted for any other degree at this University or any other institution and that, except where reference is made to the work of other authors, the material presented is original.

Ahsan Abdullah

Abstract

Our ability and capacity to generate, record and store multi-dimensional, apparently unstructured data is increasing rapidly, while the cost of data storage is going down. The data recorded is not perfect, as noise gets introduced in it from different sources. Some of the basic forms of noise are incorrect recording of values and missing values. The formal study of discovering useful hidden information in the data is called Data Mining. Because of the size, and complexity of the problem, practical data mining problems are best attempted using automatic means.

Data Mining can be categorized into two types i.e. supervised learning or classification and unsupervised learning or clustering. Clustering only the records in a database (or data matrix) gives a global view of the data and is called one-way clustering. For a detailed analysis or a local view, biclustering or co-clustering or two-way clustering is required involving the simultaneous clustering of the records and the attributes.

In this dissertation, a novel fast and white noise tolerant data mining solution is proposed based on the Crossing Minimization (CM) paradigm; the solution works for one-way as well as two-way clustering for discovering overlapping biclusters. For decades the CM paradigm has traditionally been used for graph drawing and VLSI (Very Large Scale Integration) circuit design for reducing wire length and congestion. The utility of the proposed technique is demonstrated by comparing it with other biclustering techniques using simulated noisy, as well as real data from Agriculture, Biology and other domains.

Two other interesting and hard problems also addressed in this dissertation are (i) the Minimum Attribute Subset Selection (MASS) problem and (ii) Bandwidth Minimization (BWM) problem of sparse matrices. The proposed CM technique is demonstrated to provide very convincing results while attempting to solve the said problems using real public domain data.

Pakistan is the fourth largest supplier of cotton in the world. An apparent anomaly has been observed during 1989-97 between cotton yield and pesticide consumption in Pakistan showing unexpected periods of negative correlation. By applying the indigenous CM technique for one-way clustering to real Agro-Met data (2001-2002), a possible explanation of the anomaly has been presented in this thesis.

Acknowledgements

The humble accomplishment of this thesis would not have been possible without the contribution of many individuals, to whom I express my appreciation and gratitude.

Firstly, I am deeply indebted to my principal supervisor Dr. Amir Hussain who guided me every step of the way and was a source of inspiration. I am grateful to Dr. Amir Muhmmed the Rector of my University for his guidance, encouragement and support, especially for the Agriculture related work. My sincere gratitude to Dr. Ijaz Pervaiz Director General Pest Warning for providing the necessary pest scouting data and holding many useful discussions. I am also thankful to Dr. Aftab Maroof, the Director of my campus for providing a conducive work environment.

I am profoundly grateful for all the support of my father and my (late) mother. This thesis is dedicated to her. I owe my loving thanks to my wife who stood by my side through the years, and my children who have lost a lot due to my research work.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	**Meaning**
ATM:	Automated Teller Machine
BCH:	BaryCenter Heuristic
BC:	BiCluster
BEAM:	Bayesian Estimation of Array Measurement
BOT:	Bottom partition
BW:	Band Width
BWM:	Band Width Minimization
C&C:	Cheng and Church
CAST:	Cluster Affinity Search Technique
CFS:	Correlation based Feature Selection
CLCV:	Cotton Leaf Curl Virus
CM:	Cuthill McKee
CMH:	Crossing Minimization Heuristic
CPU:	Central Processing Unit
DAG:	Directed Acyclic Graph
DNA:	Deoxyribo Nucleic Acid
E:	Entropy
EFD:	Equal Frequency Discretization
ETL:	Economic Threshold Level
EWD:	Equal Width Discretization
FAO:	Food and Agriculture Organization

FoM:	Figure of Merit
GA:	Genetic Algorithm
GRASP:	Greedy Randomized Adaptive Search Procedure
IG:	Information Gain
KDD:	Knowledge Discovery in Databases
KDDM:	Knowledge Discovery by Data Mining
LNCS:	Lecture Notes in Computer Science
MASS:	Minimum Attribute Subset Selection
MH:	Median Heuristic
MS:	MinSort Heuristic
NP:	Nondeterministic Polynomial
OLTP:	On Line Transaction Processing
OPSM:	Order Preserving Sub Matrix
P:	Purity
PC:	Principal Component
RCM:	Reverse Cuthill-McKee
SAMBA:	Statistical Algorithmic Method for Bicluster Analysis
SBH:	Sequencing By Hybridization
SBW:	Spotted Boll Worm
SD:	Standard Deviation
SNR:	Signal to Noise Ratio
SOM:	Self Organizing Maps
SQL:	Structured Query Language

TOP:	Top partition
VLSI:	Very Large Scale Integration

Chapter 1

Introduction

Our ability and capacity to generate, record and store multi-dimensional, apparently unstructured data is increasing rapidly. Different Online Transaction Processing (OLTP) Systems, sensors, etc. are constantly recording data in a diverse set of application domains such as agro-informatics, bio-informatics, e-commerce, etc. A call made by phone, an email sent/received, money drawn from an ATM machine, web browsing, a payment made by a credit card, using electrical appliances, etc., all result in data recording or transaction logging. Therefore, knowing that one should be looking for a needle in a haystack is a simpler problem, as compared to when it is not known in advance what one should be looking for; such as useful hidden patterns and relationships in data.

The formal study of such problems is called Data Mining. Because of the size and complexity of the problem, practical data mining problems are best attempted using automatic means. Data Mining can be categorized into two types i.e. supervised learning or classification and unsupervised learning or clustering. In classification the number of classes or groups is known in advance along with their characteristics. However, in clustering neither the number of classes nor their characteristics are known in advance.

Clustering is the organization or grouping of a collection of patterns (usually represented as a vector of measurements, or a point in a multi-dimensional space) into clusters with similar properties. Applications of clustering can be found in most fields of natural, social and life sciences, consequently the clustering literature is enormous and diverse. Data mining through clustering is well suited for large databases. The notion of 'large' has varied (and will continue to do so) along with changes in technology (e.g., memory cost vs. capacity). In the 1960's, 'large' meant several thousand patterns; today this number has grown to be in the millions.

Clustering only the records (or the attributes) in a database (or a data matrix) gives a global view of the data and is called one-way clustering. For a detailed analysis or a local view, biclustering or co-clustering or two-way clustering is required involving the simultaneous clustering of the records and the attributes.

Example-1.1

Consider the analysis of height-weight relationship for a group of people. The natural clustering occurs on gender basis, as males tend to be taller and heavier than females, as shown in Figure 1.1(a). Consider assigning keywords to web pages for fast web searching. There is no visible grouping as shown in Fig 1.1(b), though there is natural clustering i.e. keywords of a certain type (say sports) occurring in typical web pages, and others too. Such groups can only become identifiable, after application of an appropriate clustering algorithm that rearranges the data points (or records). For data sets comparable in size to the screen resolution, the resulting clusters can be displayed on the monitor screen; however, for larger data sets clusters are extracted without completely displaying them on the screen. In this study, we will consider the latter case, where using k-means directly on the un-clustered data input may not yield any interesting results.

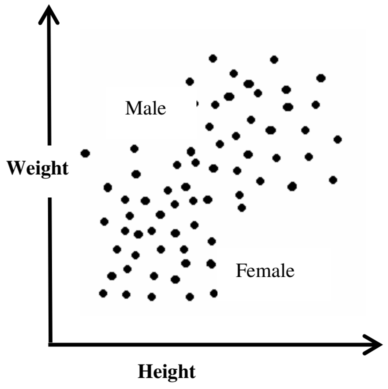


Fig-1.1(a): Height vs. Weight

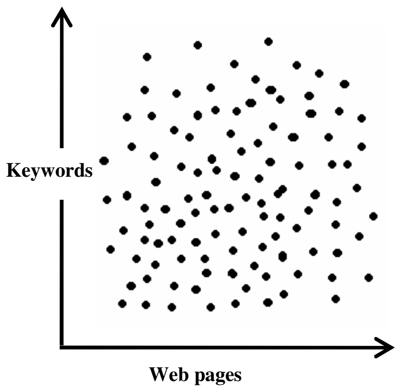


Fig-1.1(b): Keywords vs. Web pages

Figure 1.1: Comparison of nature of clusters

Section (1.1) presents the motivation for the Thesis. In section (1.2) the aims of the research are stated whilst in section (1.3), the main objectives of the research are outlined. The original contributions of the thesis are presented in section (1.4). A list of publication resulting from this research is given in section (1.5). Finally, the outline of the thesis is presented in section (1.6).

1.1 Motivation for the Thesis

The data recorded in a database (or a data matrix) has traditionally been analyzed from two perspectives, namely (i) with reference to the records; or (ii), with reference to the columns or attributes, which is termed one-way clustering or simply clustering. The purpose is the discovery of hidden, yet interesting patterns (or clusters), by comparing either the rows of the data matrix or the columns of the data matrix.

The application of traditional one-way clustering techniques for pattern discovery and data analysis poses significant problems. Consider the case of hundreds and thousands of items being sold at a supermarket. Typically, the items procured by men will have a high level of mutual exclusivity as compared to items procured by women. Discovering such similar local grouping of attributes (or items) may be the key to uncovering many interesting and useful purchase patterns that are not otherwise apparent. It is, therefore, highly desirable to move beyond the traditional one-way clustering paradigm, and instead develop algorithmic techniques capable of discovering local patterns in data matrices, which is termed biclustering, co-clustering or two-way clustering i.e. simultaneous grouping of rows and columns.

1.1.1 One-Way Clustering

In one-way clustering each record in a cluster is selected; using all the attributes and each attribute in a cluster is selected using all the records. Thus one-way clustering methods give a global view while biclustering algorithms give a local view of the hidden relationships in data (details in Chapter 5). In one-way clustering, evaluating each record in a given cluster using all of the attributes may actually distort the cluster, since all attributes may not be contributing towards that cluster. Similarly, each attribute in an attribute cluster is usually characterized by the effect of all the records, which may not always be true. For ease of comprehension, a colour coded correlations based similarity matrix obtained from one-way clustering of the clustered output of Figure 1.2(b) is shown in Figure 1.2(a) -- observe the square clusters. Note that in Fig 1.2(a) negative correlations have been assigned shades of red, while positive correlations assigned shades of green.

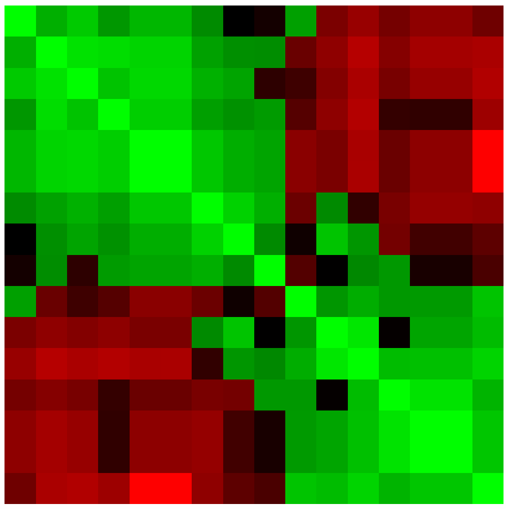


Fig 1.2(a): One-way clustering

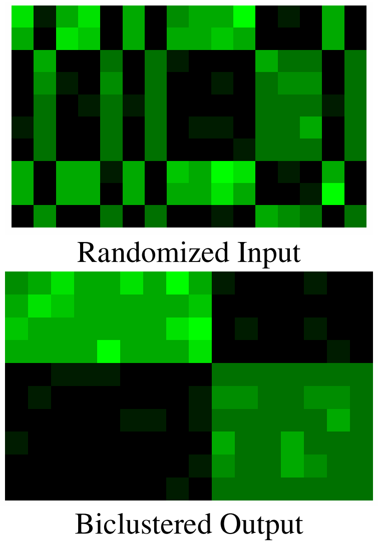


Fig 1.2(b): Two-way or biclustering

Figure 1.2: Comparison of one-way and two-way clustering

1.1.2 Two-way Clustering

As opposed to one-way clustering, each record in a bicluster is selected; using only a subset of the attributes and each attribute in a bicluster is selected using a subset of the records [13]. The complexity of the biclustering problem depends on its particular formulation. Our proposed solution is based on the binary representation of the bipartite graph corresponding to the data matrix (details in Chapter 2). Such a formulation of the problem is known to be NP-Complete in nature [71]. As a result, instead of devising exact algorithms, a pragmatic approach would be to come up with heuristics or approximation algorithms. Figure 1.2(b) shows a visualization of two-way clustering results consisting of two simulated biclusters with white noise-- observe the rectangular shape of the biclusters.

1.1.3 Bandwidth Minimization (BWM) Problem

Consider a $n \times n$ square symmetric matrix $A = \{a_{ij}\}$ the matrix Bandwidth Minimization (BWM) problem is to find a permutation of rows and columns of A so as to bring all of the non-zero elements of A to reside in a band that is as close as possible to the main diagonal, that is to $Minimize\{max\{|i - j| : a_{ij} \neq 0\}\}$. A sparse matrix has very few non-zeros that can be processed efficiently by exploiting the structure; typically, sparse matrices have O(n) non- zeros with a bounded number of non-zeros in each column or row. In scientific computing, the order of the elements in a sparse matrix often affects the performance of numerical algorithms. Sparse matrices can be factored with less work and storage than the dense matrices, provided that suitable algorithms are used. Algorithms for factorization use more complicated data structures and are relatively difficult to program; so they may also use heuristics to reduce both time and storage complexity.

1.1.4 Minimum Attribute Subset Selection (MASS) Problem

The Minimum Attribute Subset Selection (MASS) problem has been defined in a number of ways by different researchers. However, though differing in expression, the definitions resemble each other in intuition. In [18] four definitions have been listed that are conceptually different and cover a wide range. They are as follows:

a. Idealized: Find the minimally sized feature subset that is necessary and sufficient to the target concept.

b. Classical: Select a subset of F features from a set of A features, $F < A$, such that the value of a criterion function is optimized over all subsets of size F.

c. Improving Prediction accuracy: The aim of feature selection is to choose a subset of features for improving prediction accuracy or decreasing the size of the structure without significantly decreasing the prediction accuracy of the classifier built using only the selected features.

d. Approximating original class distribution: The goal of feature selection is to select a small subset such that the resulting class distribution, given only the values for the selected features, is as close as possible to the original class distribution given all feature values.

1.2 Aims of the Research

The fundamental aim of this thesis is to study and develop a new robust biclustering (two-way clustering) technique that can also be used for one-way clustering for performing the data mining of multidimensional data, with the other aim being the identification of additional applications of the proposed technique to address hard, yet interesting problems.

1.3 Objectives of the Research

Clustering only the records in a database (or data matrix) gives a global view of the data. For a detailed analysis or a local view, biclustering or co-clustering is required, involving the clustering of the records and the attributes simultaneously. The following are the fundamental objectives of this work:

a. To develop a new biclustering technique based on the crossing minimization paradigm.

b. To test the developed technique using simulated data and a white noise model.

c. To demonstrate the strength of the developed technique using published and publicly available real data and compare it with the work of other researchers.

d. To solve a real-life Agriculture problem.

e. To demonstrate the application of the developed technique to address other NP-Complete problems.

1.4 Original Contributions of the Thesis

A new, graph drawing-based biclustering technique is presented based on the crossing minimization paradigm, and is shown to work for asymmetric overlapping biclusters in the presence of white noise, using real as well as simulated data.

Other than biclustering, the technique presented is demonstrated to effectively work for the following classical NP-complete problems:

a. MASS problem.
b. Matrix BWM problem.
c. One-way clustering problem.

1.5 Publications Arising

The following papers have resulted from the research presented in this thesis:

1.5.1 Journal Papers Published

a. A. Abdullah & A. Hussain, *A New Biclustering Technique Based On Crossing Minimization*, Neurocomputing Journal, **69** (2006), 1882-1896.

b. A. Abdullah & A. Hussain, *Data Mining A New Pilot Agriculture Extension Data Warehouse,* Journal of Research and Practice in Information Technology, **38** no. 3 (2006), 229-249.

1.5.2 Book Chapters Published

a. A. Abdullah & A. Hussain, *Using Biclustering for Automatic Attribute Selection to Enhance Global Visualization*, Springer Verlag Lecture Notes in Computer Science, **4370** (2007), 35-47.

b. A. Abdullah & A. Hussain, *Biclustering Gene Expression Data in the Presence of Noise*, Springer Verlag Lecture Notes in Computer Science, **3696**, (2005), 611.

1.5.3 Refereed International Conference Proceeding Publications

a. A. Abdullah & A. Hussain, *Heuristics and Meta-Heuristics for Bandwidth Minimization of Sparse Matrices*, proceedings of IEEE ICES'06 Conference, Islamabad, 2006.

b. A. Abdullah & A. Hussain, *Analysis of Unsupervised Clustering by Crossing Minimization*, BICS'04 Conference, Scotland, UK, 2004.

c. A. Abdullah & A. Hussain, *A New Biclustering Technique Based on Crossing Minimization*, BICS'04 Conference, Scotland, UK, 2004.

1.5.4 Poster

A. Abdullah, A. Hussain and A. Ordys, *Biclustering Noisy Gene Expression Data*, International Conference on Bioinformatics and its Applications (ICBA'04), Nova Southeastern University, Florida, USA, 2004.

1.6 Submitted for publication

A. Abdullah & A. Hussain, *Heuristics and Meta-Heuristics for Bandwidth Minimization of Sparse Matrices*, IEE Letters Journal.

1.7 Outline of the Thesis

The thesis is organized as follows:

Chapter 2: This chapter will cover the background of the problem with a brief overview of clustering and graph drawing; this will be followed by model formulation, the development of the white noise model and a discussion of different crossing minimization techniques.

Chapter 3: This chapter will cover the proposed biclustering technique based on the crossing minimization paradigm and its comparison with contemporary techniques using public domain real data.

Chapter 4: This chapter will cover noise tolerance of the proposed technique using public domain real gene expression data as well as simulated data and comparison with another contemporary technique, including a proof of optimality of the proposed technique.

Chapter 5: This chapter will cover the application of the crossing minimization paradigm to solve the MASS problem. Using simulated as well as real data the strength of the proposed technique will be demonstrated.

Chapter 6: This chapter will cover the application of the crossing minimization paradigm to solve the BWM problem. Using simulated as well as real data and comparing the results with traditional techniques, the strength of the proposed technique will be demonstrated. The one-way clustering application of the proposed technique to address the pesticide usage anomaly using real Agro-Met data will also be demonstrated.

Chapter 7: Presents some concluding remarks, limitations of the proposed technique and several suggestions for future work and research directions.

Appendix A: An outline of different cluster quantification methods is briefly discussed.

Chapter 2

Background

The first step towards algorithmically solving a problem is its correct and most appropriate modeling, followed by the selection of a data structure which lends itself to effective programming. A graph-theoretic approach has been adopted in this work to model the problem; and a graph drawing approach has been adopted to solve the problem of biclustering, the data structure being used is a matrix. In a nut-shell a table is considered to be a matrix data structure representation of a bipartite graph corresponding to a data matrix. The said table is subsequently discretized to generate the drawing of a bipartite graph. Finally crossings are minimized in the bipartite graph under consideration to achieve biclustering.

2.5 Introduction

The purpose of this chapter is to develop the necessary background by: (i) presenting a brief overview of clustering and graph drawing, (ii) presenting the necessary definitions and mathematical notations used throughout the write-up and (iii) formulating the model (including the white noise model); and (iv) provide a brief overview of the working of some selected crossing minimization heuristics followed by a simple example.

The input data for a clustering problem is typically given in one of two forms as described by [36]:

a. Data matrix (or an *object-by-variable structure*) S is an $n \times m$ matrix, where for each of the n objects or rows, there are corresponding m variables, also called measurements, columns or attributes. Usually $n >> m$.

b. Similarity (or dissimilarity) matrix (or an *object-by-object structure*) is an $n \times n$ symmetric matrix which contains pair-wise similarity (or dissimilarity) that is usually computed from m for all pairs of n objects or rows.

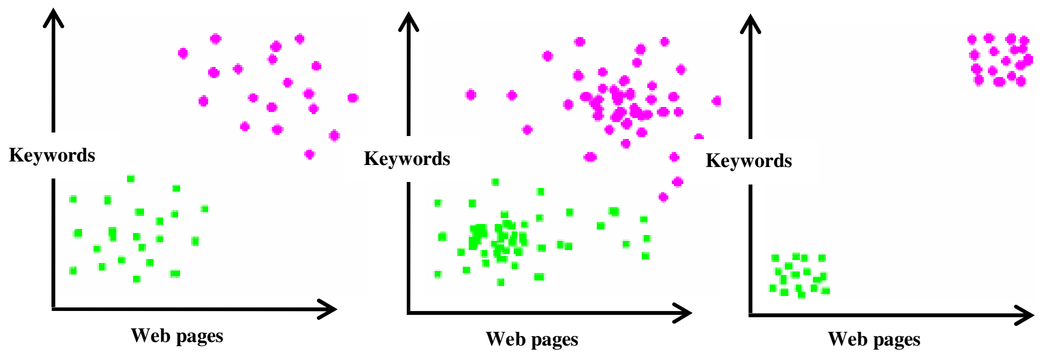
Due to the nature of the problem, only the data matrix form will be considered in this chapter.

2.1.1 Clustering and Graph Drawing

In this section we will briefly touch upon the basics of clustering along with some issues. This will be followed by the basics of graph drawing and its aesthetics.

2.1.1.1 Clustering

A cluster is a collection of elements that are similar to one another within the same cluster, but dissimilar and "away" from the elements in other clusters. One of the key elements of clustering for the ease of visual comprehension are homogeneity and separation as shown in Figure 2.1. Fig 2.1(a) shows the outcome of clustering that is well-separated, but not homogenous. Fig 2.1(b) shows homogenous clustering that is not well-separated. Fig 2.1(c) is the ideal situation.



(a) **(b)** **(c)**

Figure 2.1: Comparison of clustering objectives

The two objectives of a good cluster analysis technique - homogeneity and separation- can be defined in several different ways, resulting in principle different formulations of the underlying mathematical problem [37]. However, in this thesis, the quantifiable measures of cluster quality used are Purity (P) and Entropy (E). The variety of techniques for representing data, measuring proximity (similarity) between data elements and grouping data elements has produced a rich assortment of clustering methods. Details of some cluster quantification measures are given in Appendix A.

2.1.1.2 Issues of Clustering

When dealing with a cluster analysis problem the following issues need to be studied [29]:

a. What is the criterion used for the quantification of results? (e.g. Entropy vs. Purity).
b. What type of clustering should be considered? (one-way or two-way)
c. How difficult is it to perform the clustering? (clustering noisy data).

d. How should the clustering be done? (issue of algorithmic complexity).
e. Is the resulting clustering meaningful? (i.e. how should the results be interpreted)

Clustering is useful in several exploratory pattern-analyses, grouping of data, decision-support and machine-learning applications, including data mining, document retrieval, image segmentation, pattern classification, etc. However, in many such problems, there is little prior information available about the number and type of clusters present in the data and the decision-maker can't make more than a few assumptions before proceeding further. It is under these restrictions that clustering methodology is particularly appropriate for the exploration of interrelationships among the data points to make an assessment (perhaps preliminary) of their structure.

2.1.1.3 Traditional Clustering Techniques

Several clustering techniques and methods used in data mining have been discussed by Berkhin in [2], some of which are listed below. However, in this thesis only graph-theoretic and graph-drawing based methods will be considered.

a. **Hierarchical methods** [42]**:** Finds successive clusters using previously established clusters. Can be agglomerative or "bottom up" or divisive or "top down".

b. **Density based partitioning methods** [26]**:** Try to discover dense connected components of data, which are flexible in terms of their shape

c. **Evolutionary Methods:** Techniques such as Simulated Annealing [4], Genetic Algorithm [16], Tabu Search [18] etc.

2.1.2 Graph Drawing

In this work a graph drawing approach has been adopted. Before going any further, let's define what a graph drawing is. A *graph* is a collection of entities and their relationships. We refer to the entities as the *vertices* (or nodes) of the graph, and to their relationships as *edges* (or links). Each edge pairs two vertices, which we call its *endpoints*. A simple graph is denoted by **G**(**V**,**E**) where the vertex and edge sets are **V** and **E,** respectively.

In the simplest case, the vertices and edges have no further information associated with them. For example, one can describe the complete bipartite graph $K_{2,2}$ by enumerating its vertices **V = {1, 2, 3, 4}** to get the bipartite edge set **E = {(1, 3), (1, 4), (2, 3), (2, 4)}**. Here, the vertex names are just placeholders, since the vertices are indistinguishable except where the topology of the graph breaks symmetry. Figure 2.2 shows the drawing of $K_{2,2}$.

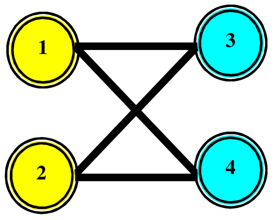


Figure 2.2: A bipartite clique $K_{2,2}$

A *graph drawing* is defined as the transformation of a graph into a visual representation of the graph, which is called a *drawing*, as shown in Figure 2.3. In a typical drawing, the vertices are mapped to boxes or circles on a subset of the plane and edges are mapped to lines connecting the boxes that represent their endpoints. For the simple example of Fig

2.3, there can be 36 possible drawings. Imagine the problem complexity when the graph has hundreds and thousands of vertices.

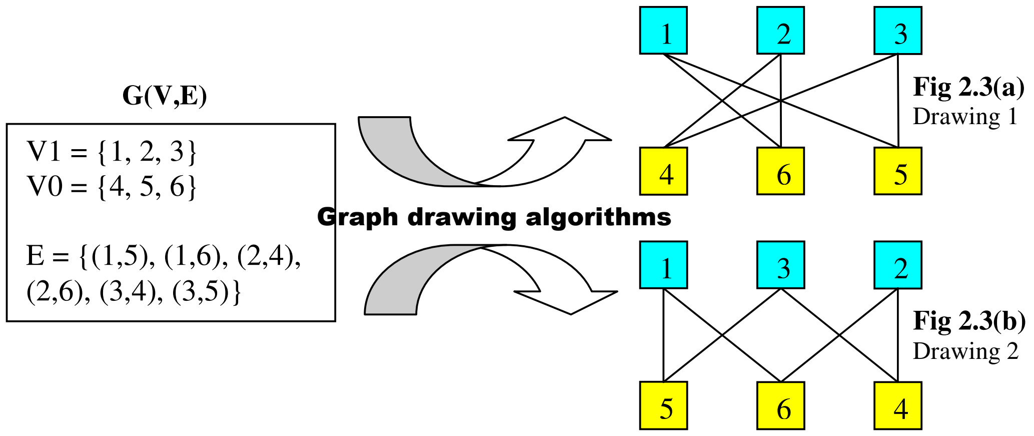


Figure 2.3: Different graph drawings of graph G(V,E)

2.1.3 Graph Drawing Requirements

Di Battista et al. [6] break down graph drawing requirements into three basic concepts: (i) drawing conventions, (ii) aesthetics and (iii) constraints. There are several aesthetics of a graph drawing that have a profound impact on the performance and the area of VLSI circuits and systems for sub-micron technology [72]. In this thesis however, the emphasis is only on drawing aesthetics, in which a single drawing convention is used (straight line drawing), while the constraints of graph drawing are not considered, as they are beyond the scope of the current work.

Figure 2.4 shows different graph drawing conventions, such as Fig 2.4(a) is a poly line drawing, Fig 2.4(b) is a straight-line drawing and Fig 2.4(c) is an orthogonal drawing, Fig 2.4(d) is a planar graph drawing while Fig 2.4(e) is an upward graph drawing.

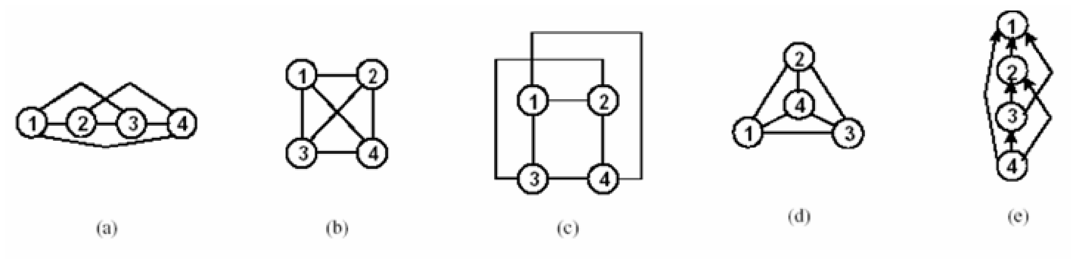


Figure 2.4: Different graph drawing conventions

Di Battista et al. list the following widely used preferences which they call "aesthetics":

a. *Edge Crossings*: minimization of the number of edge crossings.
b. *Area*: minimization of the drawing area which is measured using either the convex hull or the bounding rectangle. It is only meaningful when the drawing conventions prevent the drawing from being arbitrarily scaled down.
c. *Total Edge Length*: minimization of the sums of the lengths of edges. It is only meaningful when the drawing conventions prevent the drawing from being arbitrarily scaled down.
d. *Maximum Edge Length*: minimization of the maximum lengths of an edge. It is only meaningful when the drawing conventions prevent the drawing from being arbitrarily scaled down.
e. *Uniform Edge Length*: minimization of the variance in edge length. It is only meaningful when the drawing conventions prevent the drawing from being arbitrarily scaled down.
f. *Total Bends*: minimization of the total number of edge bends in a polyline drawing and orthogonal drawings.

g. *Maximum Bends*: minimization of the maximum number of edge bends per edge in a polyline drawing.

h. *Uniform Bends*: minimization of the variance in the number of edge bends in a polyline drawing.

i. *Angular Resolution*: maximization of the minimum angle between edges incident to the same vertex in a polyline (especially straight-line) drawing.

j. *Aspect Ratio*: minimization of the ratio between the larger and smaller dimensions of the drawing area.

k. *Symmetry*: displaying symmetries of the graph with geometric symmetries.

No.	**Aesthetic**	**Fig 2.3(a)**	**Fig 2.3(b)**
1	Edge crossings	6	2
2	Total edge length	12	10
3	Maximum edge length (Manhattan)	3	2
4	Uniform edge length (Manhattan)	0.8	0.266
5	Displaying of symmetries	No	Yes

Table 2.1: Comparison of graph drawing aesthetics for Figure 2.3

Figure 2.5 demonstrates a dilemma of aesthetic graph drawings. The first graph drawing aesthetic discussed in section 2.1.3 is edge crossing minimization, while the last one being the display of symmetries. Figure 2.5(a) shows a drawing with zero crossings. However, the same graph, when drawn differently (Fig 2.5(b)), shows a clique i.e. symmetry, but with an additional crossing.

Observe that in order to achieve these two aesthetics, two different drawing conventions were used i.e. planar drawing and straight-line drawing. But, despite this, while trying to achieve one aesthetic, it resulted in compromising another aesthetic. In this work a technique is proposed that reduces crossings and brings out symmetries in bipartite graphs simultaneously. The drawing convention being used is straight line drawing.

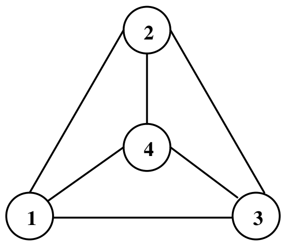


Fig 2.5(a): Crossings minimized but symmetry is hidden.

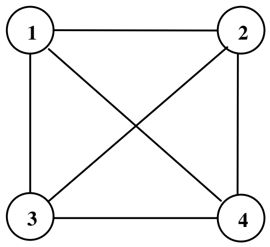


Fig 2.5(b): Symmetry is displayed but crossings are not minimized

Figure 2.5: The dilemma of aesthetic graph drawing

2.1.4 The Graph Drawing Model

A graph consists of a set of vertices *V* and a set of edges *E* denoted by *G(V, E)*. A bipartite graph is a special graph which can be partitioned into two, such that there are no edges that are present in both partitions. The bipartite graph can be either weighted with a weight assigned to each edge or binary i.e. edge weights of 0 (missing edge) and 1. This work is primarily focused around asymmetric binary bipartite graphs and the use of a matrix data structure for storing the graph.

Let S correspond to the matrix representation of a weighted bipartite graph, such that each row of the data matrix corresponds to a vertex of the bipartite graph in one partition and each column of the data matrix corresponds to a vertex of the bipartite graph in the other partition. Note that S is unlikely to be symmetric. To get a simple graph from a weighted graph, edge weights are discretized. This consequently converts S into a binary data matrix denoted by S_B. Note that the transformation of S to S_B is like a double-edged sword, as this may result in the removal of weak and uninteresting or even false relationships, including noise. However, this can also result in the loss of actual data as it is very difficult to differentiate between noise and data.

Let the bipartite graph (bi-graph) corresponding to S_B be denoted by G_B. It will be most likely that G_B will not consist of pure clusters and zero noise. However, to facilitate the formulation of the model, it is first assumed that the data matrix consists of pure disjoint clusters and zero noise, with a perfect grouping of rows and columns corresponding to the clusters. Such a data matrix is denoted by S^*_B and the corresponding bi-graph is denoted by $G^*_B(V_0, V_1, E)$, where G^*_B will be a union of overlapping bipartite graph cliques i.e. $K_{i,j}$, and V_0, V_1 are the bipartition of vertices of G^*_B such that $V_0 \cap V_1 = \varnothing$. E is the edge set such that $e = |E|$. As two-way clustering (or biclustering) is being considered in this work, namely the simultaneous clustering of rows and columns of S, therefore, $i \geq j$ for $K_{i,j}$ and $n = |V_1| + |V_0|$. Density of G_B is denoted by σ i.e. $\sigma(G_B) = e/(|V_1| \times |V_0|)$.

Now, if we let a bi-graph drawing (or layout) of G^*_B to be obtained by placing the vertices of V_0 and V_1 at distinct locations on two horizontal lines $y = 1$ representing TOP (top partition) and $y = 0$ representing BOT (bottom partition) in the *XY*-plane, respectively. The vertices of every clique are located at consecutive x-coordinates for TOP and BOT partitions i.e. in the same neighborhood. Now, if we draw each edge with one straight-line segment which connects the points with $y = 0$ and $y = 1$ where the end vertices of the edges were placed. This will result in a bi-graph drawing in which only those edges intersect that belongs to the same clique. Let φ^* be the bi-graph drawing corresponding to G^*_B and the order of vertices in the bipartitions V_0 and V_1 being denoted by π^*_0 and π^*_1, respectively. Note that for $K_{i,\,j}$ (as a convention) it will be assumed that the vertices in the bi-partition *i* will be placed in TOP and the vertices in *j* will be placed in BOT. For more on graph drawing, the reader is referred to [6].

2.1.5 Optimum Biclustering Graph Drawing Problem

Next, let the order of vertices in the bipartitions V_0 and V_1 of G_B to be denoted by π_0 and π_1, respectively. And if we let $\Phi(G)$ denote the set of all possible bi-graph drawings of the bi-graph G_B, then the optimum graph drawing problem can be stated as follows:

Optimum biclustering graph drawing problem: *Given a bi-graph G_B (V_0, V_1, E) find φ^* among $\Phi(G)$ with a <u>relative</u> permutation of vertices π^*_0 and π^*_1.*

Relative permutation of vertices means that vertices of a bicluster remain in their bicluster neighborhood or vicinity and do not cross bicluster boundaries. For example, the vertices labeled {3, 6, 10, 11} and {7, 9, 12, 13} (in TOP) correspond to two biclusters BC_1 and BC_2; the relative ordering of vertices within a bicluster is unimportant. However, bicluster quality deteriorates when noise causes (say) vertex 6 to move from BC_1 to BC_2.

Figure 2.6(a) shows a G^*_B that consists of $K_{4,8} \cup K_{4,4} \cup K_{8,2}$ with *n* = 16 (TOP) + 14 (BOT) and *e* = 64. Figure 2.7(a) shows the corresponding S^*_B.

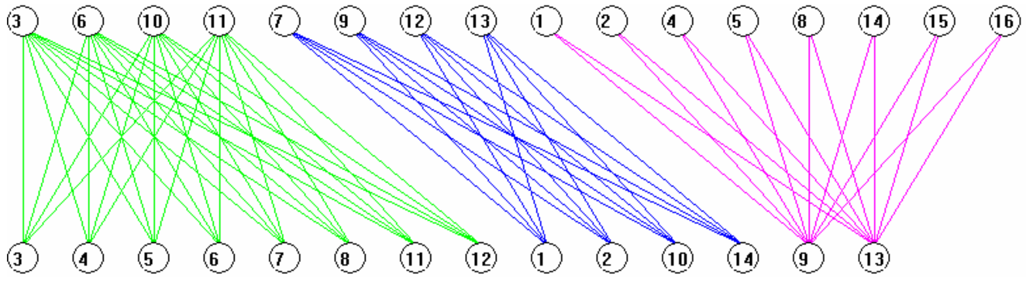


Figure 2.6(a): $G^*_B = K_{4,8} \cup K_{4,4} \cup K_{8,2}$ with 232 crossings

Finally, the Biclustering problem can be defined as follows: *"Given a* G^*_B *with a set of vertices V and a set of edges E, identify a set of biclusters* $B_k = (V_k; E_k)$ *where* $V_k \in V$ *and* $E_k \in E$ *such that each bicluster* B_k *satisfies some specific characteristics of quantification".*

2.3 White Noise Model

Real data-sets will always have noise and disorder. Therefore, as a first step towards modeling non-ideal data-sets, the vertices in each bipartition of G^*_B are randomly permuted. Figure 2.6(b) shows the G^*_B of Fig 2.6(a) with vertices randomly permuted. Figure 2.7(b) shows the corresponding S_B.

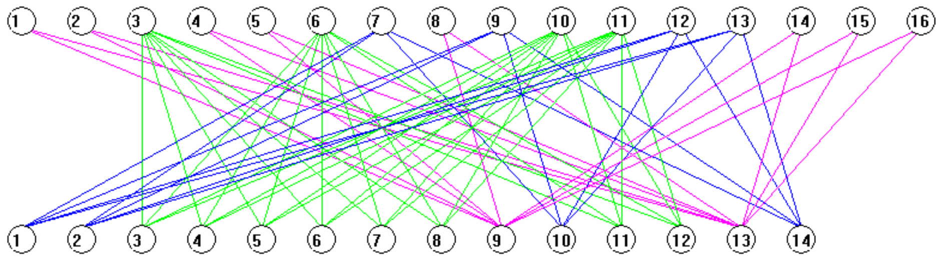


Figure 2.6(b): G_B obtained after randomly permuting G^*_B resulting in 894 crossings

Now, the second step towards non-ideal datasets i.e. "contamination" of G_B, is obtained by randomly adding edges (white noise) between $v \in V_0$ (i.e. TOP) and $u \in V_1$ (i.e. BOT) with a probability of $\alpha_E < ½$. Similarly, edges are also randomly removed (white noise) from each of $K_{i,j}$ with a probability of $\alpha_I < ½$, resulting in G_B such that $\alpha_E = \alpha_I$. The effect of these operations on the S_B would be the replacement of original 1's by 0's for α_E and the replacement of original (and not converted) 0's by 1's for α_I.

The effect of white noise is demonstrated in Figure 2.7(c) which shows the input S^*_B of Figure 2.7(a) with $\alpha_E = 0$ and $\alpha_I = 0.25$ (noting that the corresponding effect on G_B will be the removal of edges). Figure 2.7(d) shows the S^*_B of Figure 2.7(a) with $\alpha_E = 0.25$ and $\alpha_I = 0$ (note that the corresponding effect on G_B will be the addition of edges across the cliques). The collective effect of $\alpha_E = \alpha_I = 0.25$ is shown in Figure 2.7(e) which represents an S^*_B with noise. Permuting the vertices of the bi-graph corresponding to Fig 2.7(e) generates an isomorphic graph drawing and the corresponding data matrix is shown in Figure 2.7(f). Note that Figure 2.7(f), without color or grey coding (showing cluster classification), would correspond to real data. The objective of this work is to demonstrate the working of a technique that takes an S_B similar to Fig 2.7(f) as input and, using a CMH, reorders the rows to generate an output similar to Fig 2.7(e) i.e. S^*_B.

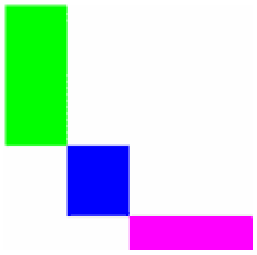


Fig 2.7(a) S^*_B with $\alpha_E = \alpha_I = 0$

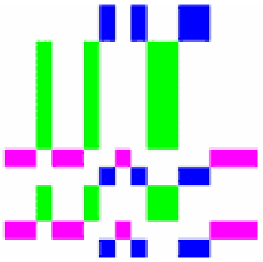


Fig 2.7(b) S_B with $\alpha_E = \alpha_I = 0$

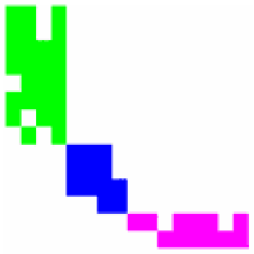


Fig 2.7(c) S^*_B with $\alpha_E = 0$ and $\alpha_I = 0.25$

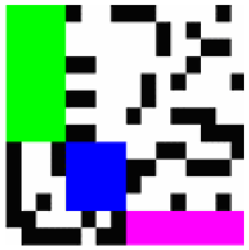


Fig 2.7(d) S^*_B with $\alpha_E = 0.25$ and $\alpha_I = 0$

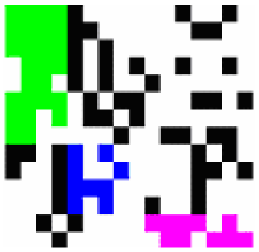


Fig 2.7(e) S_B with $\alpha_E = \alpha_I = 0.25$

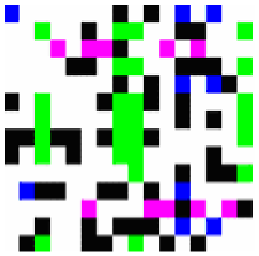


Fig 2.7(f) S_B with $\alpha_E = \alpha_I = 0.25$

Figure 2.7: Different types of noise in the data matrix

The effect of white noise in the data matrix (corresponding to S) manifests itself in the discretized data matrix (corresponding to S_B) by 1's replaced by 0's and 0's replaced by 1's. Assuming median based column discretization, a 1 replaced by a 0 in S_B corresponds to noise in S that reduces the actual value to less than the median of the column considered. Similarly a 0 replaced by a 1 in S_B corresponds to noise in S that increases the actual value to more than the median of the column considered.

Using the definition of noise by Hartigan [33] a *perfect* constant bicluster is a sub-matrix (I, J) where all values within the bicluster are equal for $i \in I$ and all $j \in J : s_{ij} = \mu$. Now, in view of the prior discussion the values s_{ij} that can be considered a constant bicluster, are presented as $\mu \pm \eta_{ij}$, where η_{ij} is the noise associated with the real value μ of s_{ij}.

2.4 Review of Various Crossing Minimization Techniques

The aim of this section is to briefly discuss the background of the crossing minimization paradigm and its application to biclustering, and to demonstrate the working of a particular crossing minimization heuristic (BaryCenter) using a simple example.

The crossing minimization problem has been studied for over two decades in the context of VLSI physical design, and its two variants namely, the one-layer (or one-partition) and two layers (or two-partitions) are known to be NP-Complete problems [32]. There are basically three types of Crossing Minimization Heuristics (CMH). The first type is the classical ones that do not count the crossings and, as a result, are very fast and yet give good results. Examples include the BaryCenter Heuristic (BCH) proposed by Sugiyama et al [77] and the Median Heuristic (MH) proposed by Eades and Wormald [27]. Other than the simple heuristics in which only the sorting part has a non-linear time complexity, there are also number of other methods that permute the vertices in the bipartitions. These methods have a higher time complexity (typically due to crossing-counting); therefore, the same will be only briefly mentioned here. Recognize that in the approached pursued in this work, the ordering of the vertices is repeated a number of times, therefore, the heuristic used must be efficient i.e. of lower time complexity. The higher time complexity approaches include Simulated Annealing [63], Genetic Algorithms [64], Tabu Search [54], Branch-and-Bound [82, 44], and Stochastic

Hill-Climbing [66]. Lastly, there are the so called meta-heuristics that make use of classical heuristics to generate the initial arrangement of vertices and then improve upon it by using other heuristics (Chapter 6).

Next, the MaxSort Heuristic, Median Heuristic and the BaryCenter Heuritic are briefly discussed. For a detailed review of CMH, see Marti and Laguna [59] and Stallmann [73].

2.3.1 MaxSort Heuristic

This heuristic reorders the vertices on the changeable layer (or partition) according to the MaxSort weight [1] or a representative value defined as follows:

$$MaxSort(v) = \begin{cases} Max(P(w)) & \text{if } |\,\text{adj(v)}\,| \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

where v is a vertex on the changeable layer (or partition), *adj*(v) is the set of neighbors of v on the fixed layer (or partition), and P(w) is the position of w on the fixed layer.

2.3.2 Median Heuristic (MH)

MH reorders the vertices on the changeable layer (BOT if TOP is fixed or vice-versa) according to the median weight.

$$Median\ (v) = \begin{cases} \frac{1}{2}(P(w_{\lceil \frac{d}{2} \rceil}) + P(w_{\lfloor \frac{d}{2} \rfloor})) & \text{if } d \geq 1 \\ 0.0 & \text{otherwise} \end{cases}$$

where v is a vertex on the changeable layer, d is the number of vertices adjacent to vertex v, and $w_1, \ldots w_d$ is the sequence of vertices adjacent to v on the fixed layer ordered according to the position *P*. We have slightly modified MH to ensure integer medians for fast (average case linear time) sorting.

2.3.3 BaryCenter Heuristic (BCH)

This heuristic reorders the vertices on the changeable layer (or partition) according to the BaryCenter weight [77] or a representative value defined as follows:

$$BaryCenter(v) = \begin{cases} Avg(P(w)) & \text{if } |\,\text{adj(v)}\,| \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

where *v* is a vertex on the changeable layer (or partition), *adj*(*v*) is the set of neighbors of *v* on the fixed layer (or partition), and P(*w*) is the position of *w* on the fixed layer.

Example 2.1

This example will help to explain the working of the CMH using a simple bi-graph consisting of 6 vertices and 5 edges, as shown in Fig 2.8(a).

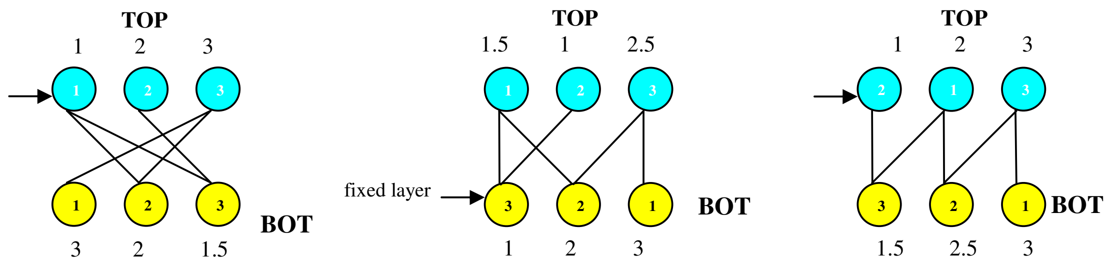


Fig 2.8(a): Input **Fig 2.8(b): Intermediate result** **Fig 2.8(c): Output**

Figure 2.8: Crossing Minimization Using the BaryCenter Heuristic

Figure 2.8(a) shows a bi-graph with 5 crossings where the vertices in the TOP partition are sequentially numbered (labeled) i.e. 1, 2, and 3. These numbers also become the labels of all the edges incident on each of these vertices. Subsequently the labels for the vertices in the BOT (bottom partition) are generated by taking the average of the edge labels incident on them. For example, vertex number 2 in BOT has two edges incident on it with edge label 1 (coming from vertex 1) and edge label 3 (coming from vertex 3). The average of the two edge labels is 2, and therefore, this is the label that gets assigned to vertex 2 in the BOT partition. This is repeated for the other vertices in the BOT partition which are then sorted based on their vertex labels, resulting in Fig 2.8(b). Next, the vertices in the BOT partition of Fig 2.8(b) are sequentially numbered (labeled) linearly, and the labels for vertices in the TOP partition are generated. Following this, the vertices in the TOP partition are sorted based on their vertex labels, resulting in Fig 2.8(c). This process is repeated until the ordering of the vertices in TOP and BOT ceases to change. The final optimal output is shown in Figure 2.8(c) which can be seen to have zero crossings.

Observe that the final outcome or the permutation of vertices is dependent on the initial permutation of vertices, along with the choice of selecting the partition to begin sequential numbering. In example 2.1, using MinSort and starting from TOP partition results in two crossings, while starting from the BOT partition results in zero crossings.

2.4 Conclusions

Storing a graph in a matrix has several advantages, the most significant being a direct relationship between the table values and the graph drawing. The other advantage being the easy visualization of the data matrix, thus providing a powerful analysis capability for the end user. The benefits of the crossing minimization approach for data mining will become clear when the proposed crossing minimization technique for biclustering is discussed in-depth in Chapter (3).

Chapter 3

Biclustering Using Crossing Minimization

3.1 Introduction

Biclustering of the data matrix was originally introduced in the seventies by Hartigan [33], who used the *Block Clustering* method based on evaluation of variance of the matrix splits. He demonstrated his technique by using the data of Republican vote for the President of the United States, and voting data consisting of the UN votes in 1969–1970. However, analysis of gene expression data by simultaneous clustering of genes and samples is a relatively new field of research, and its first usage was done by Cheng and Church [13]. Researchers have used different criteria and techniques for biclustering, which are discussed by Madeira and Oliveira [61]. As a consequence it is difficult to compare all these techniques and indeed such a comparison is outside the scope of this thesis. For comparative analysis, in this thesis the focus is on a selected set of algorithms which are (i) graph theoretic in nature, (ii) use discretized data matrix (iii) are popular in terms of citations and usage (iv) can be easily implemented for actual comparative analysis (v) and which are representative of a particular problem solving approach..

3.2 The Proposed Technique

Let G^{+}_{B} denote the isomorphic graph resulting from the application of the Crossing

Minimization Heuristic (CMH) on G_B and the corresponding bi-graph drawing be φ^+. S^+_B is a visualization of the clustering solution generated after running CMH on G^+_B. Note that for *n* (number of nodes) and *e* (number of edges) in hundreds φ (similar to Figure 2.1(b)) is visually meaningless and that only S^+_B makes sense. However, when *n* and *e* are in thousands, screen resolutions prohibit displaying S^+_B completely (similar to Figure 2.7(b)), and only results extracted using appropriate statistical means are meaningful (which are covered in section 3.8).

3.2.1 3-Step Procedure

For ease of understanding, the proposed solution is first presented below in a rather oversimplified form, as a three-step procedure. This will be followed by details of the main steps i.e. (b) and (c). The basic three steps are as follows:

a. Discretize S to obtain S_B.
b. Run a CMH on G_B corresponding to S_B to get G^+_B
c. Extract biclusters from $G^+_{B.}$

Step-a i.e. discretization is discussed in detail in section 3.4. Now step-b i.e. CMH is explained in more detail using the pseudo code as follows:

```
Procedure MinSort
Begin
  $\pi_{1New}$ := $\pi_{2New}$ := $\phi$ {new values of $\pi_1$ and $\pi_2$}
  R1 = TOP {row i.e. y = 1};
  R2 = BOTTOM {row i.e. y = 0};
  $\pi_1$ = Sequence of nodes in R1;
  $\pi_2$= Sequence of nodes in R2;
  top_ok := bot_ok := 0; {Flags for noting optimal permutations}

        While (top_ok = 0 AND bot_ok = 0) do
              Begin
              Label nodes in R1;
              Find Minimum of each node in R2;
              Sort nodes in R2 w.r.t minimum;
              $\pi_{2New}$ = Sequence of sorted nodes in R2;
              if $\pi_{2New}$ = $\pi_2$ then bot_ok := 1
                    Else $\pi_2$ := $\pi_{2New}$;

              Label nodes in R2;
              Find Minimum of each node in R1;
              Sort nodes in R1 w.r.t minimum;
              $\pi_{1New}$ = Sequence of sorted nodes in R1;
              if $\pi_{1New}$ = $\pi_1$ then top_ok := 1
                    Else $\pi_1$ := $\pi_{1New}$;
        end {of while};
        Return $\pi_1$ and $\pi_2$; {Optimum permutations}
end {of procedure}.
```

3.2.2 Bicluster Extraction

Step-c i.e. bicluster extraction is now explained in more detail, but before this a few words on the density of the data matrix. Let S_B be a (n x m) binary data matrix having *n* rows and *m* columns. Density (d) of the discretized data matrix is given as the ratio of the total number of 1's present to the maximum number of 1's possible i.e. *nxm*. The bicluster extraction algorithm consists of the main algorithm, which includes three sub-algorithms (i) Expansion along x-axis (ii) Expansion along y-axis and (iii) Refinement of boundaries. All three are explained now.

```
Algorithm BiclusterExtract
Inputs
$S_B$: binary matrix
s : minimum cluster size
d: threshold density
Steps
Step 1- Make an initial square of size s x s having coordinates (x1, y1, x2, y2)
Step 2-Calculate density (d') of square
Step 3-If d' >= d
    Mark square as cluster 'C' and expand it further
    Continue cluster expansion along x-axis and y-axis until
    density of expanded portion becomes less than threshold.
    Refine cluster boundaries
    Remove rows and columns belonging to extracted cluster.
Repeat step 1 to 3 on remaining matrix to extract other clusters.

Expansion along X-axis
    Expand cluster boundary along x-axis temporarily
    Find density (d'') of expanded portion having coordinates (x2, y1, x2+s, y2)
    If d'' >= d
        Make expansion permanent
    Else
        Move backward and fix cluster boundary along x-axis.

Expansion along Y-axis
    Expand cluster boundary along y-axis temporarily
    Find density (d'') of expanded portion having coordinates (x1, y2, x2, y2+s)
    If d'' >= d
        Make expansion permanent
    Else
        Move backward and fix cluster boundary along y-axis.

Refine Cluster Boundaries
    Refine boundaries of cluster along x-axis by moving one column backward at-a-
    time and identify columns whose density is less than threshold. Remove those
    columns from cluster and update cluster boundary.

    Refine boundaries of cluster along y-axis by moving one row backward at-a-
    time and identify rows whose density is less than threshold. Remove those rows
    from cluster and update cluster boundary.
```

Note that the bicluster extraction method is independent of the position of the biclusters within the data matrix. The next section discusses the need for discretization of a given data matrix and the different methods of performing discretization.

3.3 Review of Previous Work

In this section two types of biclustering techniques will be discussed i.e. graph theoretic techniques and non-graph theoretic biclustering techniques.

3.3.1 Graph theoretic approaches

The approach used in this thesis is graph theoretic, because usually a large variety of real-life problems can be formulated in terms of graphs [29, 83]. For example, computer networks as graphs of good edge connectivity and vertex connectivity, network analysis in telecommunications engineering, any interconnection system, VLSI physical design and many other problems. VLSI physical design process consists of seven main stages. Most of the problems in VLSI physical design are modeled with graphs [72]. For example, Leighton [50] has shown that the crossing number of a graph can be used to obtain a lower bound on the amount of chip area required by that graph for a VLSI circuit layout. Similarly, Chang et al. [14] proposed an algorithm for minimizing the number of *vias* (or holes) in a multi-layer VLSI circuit, which in-fact is a transformation of the minimal crossing number.

Cheng and Church [13] (C&C) were the first to use biclustering for gene expression data. The time complexity of their technique is *O(mn)* and *O(mlog(n))*, where *n* and *m* respectively, are the number of rows and columns in the data matrix. However, C&C did not evaluate the performance of their technique in the presence of noise whereas in this thesis we consider biclustering of specifically noisy data. C&C's technique is also not purely unsupervised biclustering, as the user has to provide a value of mean residue score δ as well as the number of biclusters to be extracted. On the other hand, as will be demonstrated in this thesis, our proposed technique does not require any form of similar user input for its working. Using C&Cs technique biclustering is performed by simultaneously taking into account row and column coherence for a sub-matrix; such a coherence termed the mean residue score or δ. The algorithm begins by greedily removing rows and columns from the data matrix so as to reduce the overall value of δ, this continues till the given value of δ is reached. In the second phase rows and columns are added using the same scoring scheme. This continues as long as the matrix size grows without crossing the thresholdδ. After a bicluster is extracted, the values of bicluster found are randomized and the process is repeated. There are a number of problematic issues associated with their approach including (i) how to ascertain the right value of δ (ii) the possibility of an exponential growth in the number of sub-matrices (iii) the approach of deleting rows and columns from the data matrix (in order to improve δ) can land into a local minima [88] and (iv) the technique requires a large number of iterations which translates to a slow solution.

Tanay et al. [80] model the data matrix as a bipartite graph and use statistical models to solve the problem by identifying bicliques using the so called Statistical-Algorithmic Method for Bicluster Analysis (SAMBA). The data matrix is modeled as a bipartite graph with rows corresponding to vertices in one bipartition, and columns corresponding to vertices in the second bipartition. Depending upon whether a gene is up-regulated or down regulated, the corresponding edges are assigned a weight. An edge with weight 1 corresponds to an edge present, and an edge with weight 0 corresponds to an edge absent. Biclusters correspond to heavy sub-graphs, which in turn correspond to the sum of the weights of gene-condition (row-column) pairs in it including edges and non-edges. A merit function is used to evaluate the quality of a computed bicluster using SAMBA by computing its weight. SAMBA performs simultaneous bicluster identification using exhaustive bicluster enumeration, and avoids exponential runtime by restricting the number of rows the biclusters may have by assuming that row vertices have *d*-bounded degree. The similarity between our work and [80] is that in both techniques the data matrix is represented as a bipartite graph and the biclustering corresponds to identifying bicliques. Another similar technique was reported by Dhillon [20] who identifies subsets of words and subsets of documents strongly related to each other by modeling the data matrix as a bipartite graph. The time and space complexity of Tanay et al.'s technique [80] is $O(n2^d)$ where n is the number of vertices. It is interesting to note that using SAMBA for graphs with arbitrary degrees will result in a non-polynomial time complexity. The technique proposed in this thesis works for graphs of arbitrary degrees, is non exhaustive in nature and is based on the crossing minimization paradigm from the domain of graph drawing, which is not used by Tanay et al. [80]. To accommodate noise Tanay et al. search for subgraphs that are not necessarily complete. Our notion of noise is more generic and comprehensive, in the sense that corruption of data values is assumed to result in insertion of edges in the bipartite graph, and missing data is considered to be represented by missing edges.

The so called Order Preserving Sub Matrix (OPSM) technique of Ben Dor et. al [5] assumes a probabilistic model of the data matrix. This biclustering technique incorporates a greedy algorithm for discovering a fixed pattern of rows in a data-set, one at a time. OPSM defines its biclusters as a sub matrix for which there exists a permutation of columns under which the sequence of values in every row is strictly increasing. The algorithm starts by evaluating all (1, 1) partial models which are m(m-1) and keeps the best ℓ of them. This is followed by expansion to (m-2) partial models of size (2, 1) and retaining the best ℓ of them. Subsequently it expands them to (2, 2) models, (3, 2) models and so on until it gets ℓ ([s/2], [s/2]) models, which are complete models, and where *s* is the size of the model. The algorithm finally outputs the best model. The time complexity of this technique is $O(nm^3\ell)$, where n is the number of rows of the data matrix, m is the number of columns and ℓ is the number of partial models. Note that even for a single partial model, the OPSM technique is $O(m^2)$ times slower as compared to our proposed technique. The OPSM approach of making partial models and then growing them to get a complete model makes a rather strict restriction on the number of columns in the data-set. This leads to a hit and trial method of discovering a bicluster, since the number of columns in the best bicluster is not known in advance.

Liu and Yang [55] have recently defined a bicluster as an Order Preserving Cluster (OP-Cluster) and used exhaustive bicluster enumeration to perform simultaneous bicluster identification. Essentially, the biclustering problem is transformed into a problem of finding longest common subsequences. The length of subsequences must be at least n_c and these subsequences must be common in at least n_r sequences. Each row in the input data matrix is arranged as a non decreasing sequence of columns. These rows are then stored in a tree and simultaneously, common subsequences and number of rows supporting these subsequences are determined by doing operations on the tree. Given an $n \times m$ data matrix, in the worst case, the algorithm has to visit every possible node. The time complexity for the insertion operation is $\Omega(nm^2)$ and there are many such operations, with the space complexity being $\Omega(nm^2)$. As compared to our proposed technique, the OP-Cluster approach uses a tree data structure, is exhaustive in nature, requires a threshold for grouping and is slower by an order of magnitude. Furthermore, the authors did not discuss the effect of noise on the bicluster quality.

Murali and Kasif [57] have proposed a representation for gene expression data called conserved gene expression motifs or xMOTIFs (biclusters). A gene's expression level is conserved across a set of samples if the gene is expressed with the same abundance in all the samples. A conserved gene expression motif is a subset of genes that is simultaneously conserved across a subset of samples. Note that this approach is similar to the one followed by Ben-Dor et al. [5]. Murali and Kasif [57] assume that a gene could be in a fixed number of states. These states can simply be up-regulated and down-regulated when only two states are considered, in this way there is a similarity with our proposed solution in the context of binary data matrix. They also assumed that data may contain several xMOTIFs (biclusters), and the goal is to find the largest xMOTIF i.e. the one that contains the maximum number of conserved rows by evaluating a merit function. An xMOTIF must also adhere to the properties of maximality and size i.e. it must contain at least α_fraction of all the columns in the data matrix, and for every row not belonging to the xMOTIF, the row must be conserved only in a β-fraction of the columns in it. Using their technique the largest Xmotif can be computed with probability greater than 0.5 in time $mn^{O(log(1/\alpha)/log(1/\beta))}$ where $\beta < 1$ and $\alpha > 0$. Observe that the Murali and Kasif algorithm can be *O(m)* times slower as compared to the theoretical upper bound of our proposed technique. Furthermore, our technique is completely unsupervised as it does not require any a priori knowledge or user input. The technique of Murali and Kaif on the other hand is supervised, as the algorithm uses information about which class each sample belongs to in order to compute the set of informative genes. Their technique is also more tuned towards mutually exclusive biclusters evident from the choice of data sets used, while our technique works for overlapping biclusters.

In other related work, Koyuturk et al. [47] consider the problem of finding unusually dense submatrices in a binary matrix by modeling the biclustering problem as an optimization problem. Their algorithm uses sparse matrix-vector multiplication, which can be performed in *O(K)* time, where *K* is the number of 1's in the data matrix. They assume that the data matrix is generated by a memory-less source, with probability that can be estimated by the density of the data matrix. For an arbitrary submatrix they assume that the 1’s have a binomial distribution. They proposed an objective function which is to be maximized, but the solution can take a long time to converge, and that too, to possible local maximas since the initialization is random. To overcome this problem, several runs of the *O(K)* algorithm are made to identify and rank promising biclusters, or to find a single bicluster that has maximum significance. Our graph drawing based proposed technique though uses sparse matrices but is not linked with any specific distribution of 1’s. One iteration of our technique will be shown to take *O(K)* time, but terminates in a few iterations, as it is not search based. Furthermore, the constant of proportionality is likely to be very small as compared to finding dense submatrices, as there is no evaluation of an objective function and requirement of multiple iterations to overcome the problem of local maxima.

Prelic et al. [67] proposed a divide-and-conquer based approach for biclustering called the Binary inclusion-maximal biclustering algorithm (Bimax), but does not uses crossing minimization. *Bimax* is based on discrete binary matrix $E^{n \times m}$ here *n* is the number of rows and *m* the number of columns. *Bimax* tries to identify areas of E that contain only 0s and therefore can be excluded from further inspection. This strategy may be beneficial for sparse E, depending on the cutoff threshold, but limits the application area of Bimax which works for constant biclusters. The technique requires memory resources $O(nm\beta \min \{n, m\})$, while providing a worst-case time complexity for matrices containing disjoint biclusters to be $O(nm\beta)$ and for arbitrary matrices $O(nm\beta \min\{n, m\})$, where β is the number of all inclusion-maximal biclusters. Assuming $m \leqslant n$, the upper bound on β is exponential in *m*, thus resulting in performance problems i.e. resource crunch for medium to large data sets. Our proposed technique is at least β times faster than *Bimax*.

Mäkinen and Siirtola [62] have proposed the use of the barycenter heuristic to order the rows and columns of the reorderable matrix. They have showed that other decision problems related to the ordering of the reorderable matrix such as Rectilinear Picture Compression and Trie Compaction are also NP-complete, however they have not established the connection between biclustering and the reorderable matrix, their work was focused on displaying the discrete reorderable matrix. Our work not only uses the reorderable matrix (or matrix permutation) concept for one-way as well as two-way clustering but also performs cluster extraction i.e. an end-to-end solution. We also demonstrate the strength of the Median crossing minimization Heuristic for clustering noisy data and the weakness of the said heuristic when used for sparse matrices.

The Biclustering Analysis Toolbox (BicAT) of Barkowl et al. [3] is a software platform for clustering-based data analysis that integrates various biclustering and clustering techniques in a common graphical user interface. BicAT provides a number of functionalities, which among other includes five biclustering algorithms and two traditional clustering algorithms. This toolbox could be useful for bicluster comparison.

3.3.2 Non-Graph theoretic approaches

Researchers have adopted a number of non-graph theoretic approaches for biclustering, such as using Genetic Algorithms [16], Simulated Annealing [4], Gibbs Sampling [76], Spectral Biclustering [45] to name a few.

Chakraborty et al. in [16] have used a Genetic Algorithm based approach to address the biclustering problem by proposing two genetic algorithms that uses greedy algorithm as a local search procedure using the mean squared residue score given by C&C. However, to avoid random interference they use an algorithm that starts with very tight co-regulated sub-matrices found using k-means. These sub-matrices form the initial population. For chromosome representation, biclusters are encoded using bit representation with a single bit string of length number of rows plus the number of columns (nrows + ncols). Two fitness functions are used; one is the mean squared residue score of C&C divided by the bicluster size, the other being the z-score, which is a measure of how unlikely it is for tightly co-regulated sub-matrices to be generated by random. Each iteration of GA main loop produces a new generation of biclusters based on the current population. The biclusters are selected for mating using the roulette wheel selection. For crossover bits are selected at random, and a certain fraction of members are chosen at random for mutation.

Bryan et al. in [4] have adopted the Simulated Annealing approach to address the biclustering problem using the greedy search technique of C&C for node deletion. It is anticipated that the stochastic search technique might produce better results as it will expand the search space by accepting those changes that do not immediately improve the fitness, but allow exploring other areas outside the locality for better solutions. The annealing schedule proposed uses the temperature at each stage which is approximately 0.9 times of the previous temperature. How long the search spends at each temperature is measured in terms how often the system is perturbed (*attempts*) and how many times the perturbance is accepted (*successes*). The success and attempts were set such that for a data set of 1,000 genes, between 10,000 and 100,000 changes would be made to the system at each temperature. The initial temperature was set so to allow 80% of reversals to be accepted, and the minimum size of the result was set to 10x10.

Sheng et al. [76] have presented a solution for the biclustering problem based on a simple frequency model for the expression pattern of a bicluster using Gibbs sampling for parameter estimation. The biclustering problem is considered in the Bayesian framework with biclusters statistically contrasted with noisy background. The proposed technique discovers genes and conditions of a bicluster, and furthermore represents the pattern of a bicluster as a probabilistic model explained by the posterior frequency of every discretized expression level discovered. Gibbs sampling is a Markov chain Monte Carlo method. The Monte Carlo property comes from the way by which Gibbs sampling evaluates statistics, such as mean and variance, of the target marginal distribution. Gibbs sampling does not suffer from the problem of local minima that often characterizes Expectation–Maximization. The technique works by discretization of the data matrix into γ bins ($\gamma = 3$ used in the paper). Once the micro-array data is discretized, it resembles the problem of finding subsequences sharing similar alphabetic expressions in sequence data. The biclusters discovered are masked and the algorithm is rerun on the rest of the data. For simulated data, Sheng at al. have reported that for a 100x30 data matrix with an embedded bicluster of size 25x30, they performed 500 iterations of their algorithm for extracting the 25x30 bicluster.

Spectral biclustering approach uses techniques from linear algebra to identify bicluster structures in the data matrix. In the model proposed by Kluger et al. [45] it is assumed that the gene expression matrix has a hidden checkerboard-like structure which is required to be discovered using eigenvector computations. Kluger et al. considered a data matrix M with a checkerboard-like structure; and pointed out that eigenvalues of M^TM and MM^T are same. These eigenvector pairs can be computed by Singular Value Decomposition (SVD) of M, which will express the real matrix M as $M = X\Delta Y^T$, where Δ is a diagonal matrix and X and Y are orthonormal matrices. The columns of X and Y are the eigenvectors of M^TM and MM^T respectively. To find the biclusters, SVD of M is performed and the eigenvectors of M^TM and MM^T analyzed. A hidden bicluster will be discovered by the existence of a pair of eigenvectors with the same eigenvalue, which are approximately piecewise constant. This can be determined by sorting the vectors. To discover hard to find hidden checkerboard-like structure due to difference of their mean expression levels, Kluger et al. have proposed normalizing the gene expression matrix.

3.4 Benefits of Data Discretization

According to Fayyad and Uthurusamy [28] algorithms with quadratic time complexities are unacceptable for most Knowledge Discovery by Data Mining (KDDM) applications. Recognize that a weighted graph corresponding to S will always be a clique with a quadratic number of edges. Working with S will force the consideration of each and every edge of the un-discretized graph, resulting in a highly undesirable $\Omega(n^2)$ time complexity. Hence, a viable way out is the discretization of S, leading to S_B.

Furthermore, binary matrices can naturally arise from quantization of gene expression data [79] or more comprehensive datasets such as gene-feature matrices.

As discussed in Chapter (2), another benefit of discretization is noise tolerance, which causes elimination of the effect of certain types of white noise, resulting in robust results. The third important benefit of discretization is, the ability to bicluster non-constant valued biclusters thus increasing the application domain. Discretization causes non-constant valued biclusters to inherently become constant valued biclusters i.e. consisting of 1's and 0's.

As this work deals with unsupervised clustering, conventional supervised discretization methods such as those based on Fuzzy discretization and Entropy Minimization will not be considered. Instead, unsupervised discretization methods will be considered that make use of information about the distribution of attribute values without class information. Some of the discretization methods applicable to our problem of interest are briefly discussed next (for a detailed review, see [87]).

3.4.1 Equal Width Discretization (EWD)

EWD divides the number line between v_{min} and v_{max} into k intervals of equal width. Thus the intervals have width $\varpi = (v_{max} - v_{min})/k$ and the cut points are at $v_{min} + w; v_{min} + 2\varpi \ldots v_{min} + (k - 1)\varpi$. Note that k is a user predefined parameter. Here v_{min} and v_{max} are minimum and maximum values, respectively.

3.4.2 Equal Frequency Discretization (EFD)

EFD divides the sorted values into k intervals so that each interval contains approximately the same number of instances. Thus each interval contains $n=k$ (possibly duplicated) adjacent values. Note that k is a user predefined parameter.

3.4.3 Threshold-based Discretization

A threshold is used to convert continuous values into discrete values, such that a value of 1 is assigned to those instances of the attributes for which the value crosses a certain threshold. The threshold could be the average value or any other value, depending on the nature of the data. Note that EWD and EFD, when used with $k = 2$, also result in threshold-based discretization.

Note that for small values of k more information about the original data is ignored. On the other hand, large values of k will result in too few points per interval to get a reasonable estimate of the frequency of each interval. Both EWD and EFD are deemed simplistic, and potentially suffer from critical information loss due to the formation of inappropriate interval boundaries, since k is determined without reference to the properties of the given data [87]. Furthermore both techniques are vulnerable to outliers that may drastically skew the range.

Our technique is not linked with any specific discretization method; it is generic enough to be used with any of the above discretization methods, and probably others too. Actually the choice of the discretization method depends upon the properties of the data, which can be ascertained by data profiling. In the next section, the application of crossing minimization heuristics for biclustering are discussed, and one of the three examples is used to illustrate the application of a particular discretization technique as part of our proposed technique.

3.5 Use of CMH to Achieve Biclustering

The aim of this section is to briefly discuss how crossing minimization results in biclustering, and why certain crossing minimization heuristics work better than others for biclustering. As a result of applying CMH, the edge lengths are decreased, and consequently the corresponding vertices come closer. If arbitrary vertices come close, then edge lengths are not likely to decrease, which will actually increase the number of crossings. On the other hand when edge lengths are reduced, vertices with high interconnectivity come together in their cluster neighborhood, hence enhancing clustering. The effect of noise is to break links of vertices with their neighbours (shown by white spots in Fig 3.3a in Section 3.7) and to make links with non-neighbours (shown by black spots in Fig 3.3a). As a result it becomes computationally intensive to "move" vertices into the "correct" cluster neighborhood, hence the need to have robust estimators of position. Although Median is a robust estimator, and works well too, the corresponding MH is slow. Thus the performance deteriorates for large problems.

Put in another way, for a G_B numbering the vertices in the TOP and subsequently sorting vertices in the BOT, and then repeating this process for the BOT, has the effect of two forced orderings that are alternated until an equilibrium state is reached. An equilibrium state corresponds to the final output that does not change with further application of the CMH.

The attractiveness of the proposed technique and the reason why it is extremely fast is that it is not a merit function minimization per se or a greedy technique. The problem with evaluating merit functions is that they take time and the higher the number of iterations, the greater the time that is taken. For example, a naïve approach to establish how many edge crossings there are would require $O(e^2)$ time i.e. checking pair-wise edge crossings (where e is the number of edges in the bi-graph). For a detailed comparative discussion on CMH see Marti and Laguna [59].

The proposed technique adopts a rather "natural" approach, where the connectivity of the vertices with other vertices results in clustering. This will be further demonstrated through complexity analysis in Chapter (4) .

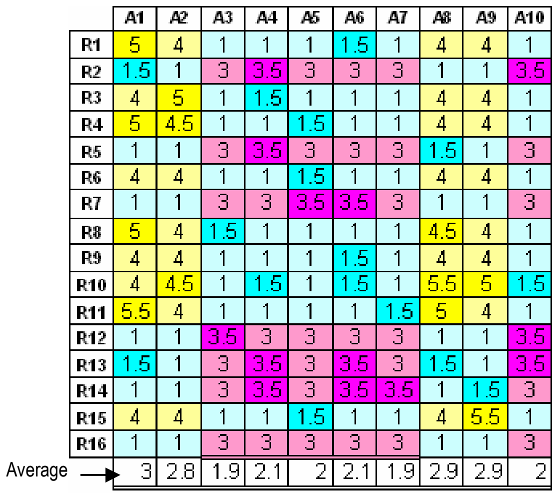
3.6 Complexity Analysis of the Proposed Technique

The purpose of this section is to perform a detailed complexity analysis of the proposed technique. Worst-case time complexity occurs for the MH, as finding the median is intrinsically slow as compared to finding the average or minimum (or maximum) value. To find the median of vertices in BOT, the worst-case linear time selection algorithm is used. The BaryCenter mean can be found in linear time, so is the minimum value for MinSort (MS). For each vertex i, the time to find the median will be $O(d_i)$, where d_i is the degree of vertex i. Since $\sum_{i=1}^{n} d_i = 2e$, hence the time complexity to find the median of all vertices in BOT will be *O(e)*. To ensure that the medians are always integer, the ceiling of the n/2 value is taken to be the median. Note that for the bi-graph model being considered, the median will always be ≤ *n*; and using this property counting sort is used to sort the median values in *O(n)* time. Thus the time complexity of one iteration of the MH (Median Heuristic) turns out to be *O(max{e, n})* = *O(e)*. The number of iterations taken to termination is *O(1)*, resulting in time complexity *O(e)*. Note that the number of iterations taken is also dependent upon the amount of noise. For $\alpha_E = \alpha_I = 0\%$, only a single iteration of MS is taken to give the optimum result. Note that the proof of optimality of the proposed biclustering technique under noiseless conditions is similar to the proof for one-way clustering as discussed in Chapter (4).

3.7 Examples

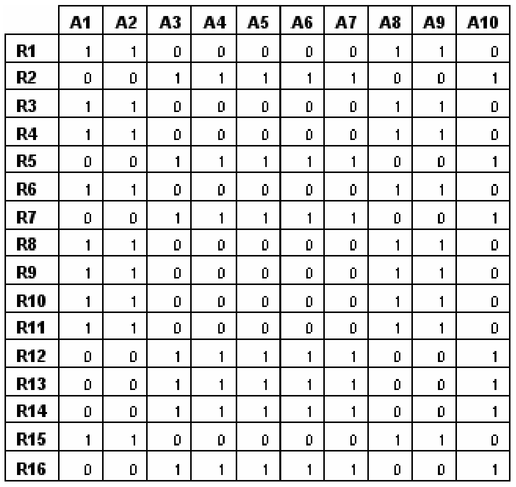
Example 3.1

In this example the working of the proposed solution is demonstrated using a variant of MaxSort CMH [1], namely the MinSort (MS). Figure 3.1(a) shows noisy input data matrix S, illustrated in the form of a table consisting of 16 rows (R1 to R16) and 10 columns or variables (A1 to A10). Note that this is the same dataset for which one-way and two-way clustering results were shown in Fig 1.2 Chapter (1). The last row in Fig 3.1(a) shows the average value of each column. For ease of comprehension non-overlapping biclusters are used in the example, but as demonstrated in the next example and section 3.8.2, the proposed technique works equally well for generic overlapping biclusters too.



	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
R1	5	4	1	1	1	1.5	1	4	4	1
R2	1.5	1	3	3.5	3	3	3	1	1	3.5
R3	4	5	1	1.5	1	1	1	4	4	1
R4	5	4.5	1	1	1.5	1	1	4	4	1
R5	1	1	3	3.5	3	3	3	1.5	1	3
R6	4	4	1	1	1.5	1	1	4	4	1
R7	1	1	3	3	3.5	3.5	3	1	1	3
R8	5	4	1.5	1	1	1	1	4.5	4	1
R9	4	4	1	1	1	1.5	1	4	4	1
R10	4	4.5	1	1.5	1	1.5	1	5.5	5	1.5
R11	5.5	4	1	1	1	1	1.5	5	4	1
R12	1	1	3.5	3	3	3	3	1	1	3.5
R13	1.5	1	3	3.5	3	3.5	3	1.5	1	3.5
R14	1	1	3	3.5	3	3.5	3.5	1	1.5	3
R15	4	4	1	1	1.5	1	1	4	5.5	1
R16	1	1	3	3	3	3	3	1	1	3
Average →	3	2.8	1.9	2.1	2	2.1	1.9	2.9	2.9	2

Fig 3.1(a): Input i.e. S



	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
R1	1	1	0	0	0	0	0	1	1	0
R2	0	0	1	1	1	1	1	0	0	1
R3	1	1	0	0	0	0	0	1	1	0
R4	1	1	0	0	0	0	0	1	1	0
R5	0	0	1	1	1	1	1	0	0	1
R6	1	1	0	0	0	0	0	1	1	0
R7	0	0	1	1	1	1	1	0	0	1
R8	1	1	0	0	0	0	0	1	1	0
R9	1	1	0	0	0	0	0	1	1	0
R10	1	1	0	0	0	0	0	1	1	0
R11	1	1	0	0	0	0	0	1	1	0
R12	0	0	1	1	1	1	1	0	0	1
R13	0	0	1	1	1	1	1	0	0	1
R14	0	0	1	1	1	1	1	0	0	1
R15	1	1	0	0	0	0	0	1	1	0
R16	0	0	1	1	1	1	1	0	0	1

Fig 3.1(b): S_B **based on average value**

Figure 3.1(b) shows S_B obtained after applying a threshold-based discretization to the average value of each column (or row) of Fig 3.1(a) by replacing all values greater than the average by 1 and others by 0. The corresponding bipartite graph drawing is shown in Figure 3.1(c).

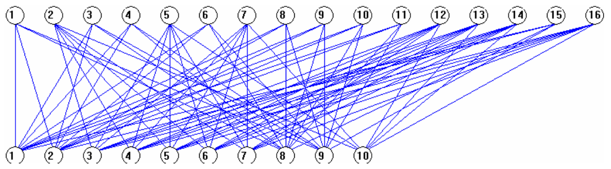


Fig 3.1(c): Bipartite graph drawing corresponding to Fig 3.1(b)

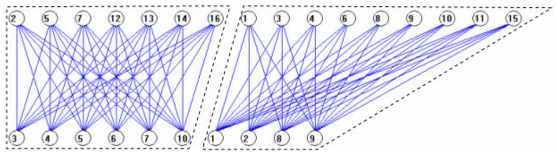


Fig 3.1(d): Bipartite graph drawing obtained after running MS on Fig 3.1(c)

Figure 3.1(d) shows an isomorphic graph drawing of G_B obtained after running the MS (MinSort) Heuristic on Figure 3.1(c). Note the simultaneous minimization of crossings and enhancing of symmetries, however, this goal is very hard to achieve when noise is introduced. Figure 3.1(d) also shows two extracted biclusters BC_1, and BC_2 shown by heavy lines. Figure 3.1(e) is the S^+_B corresponding to Fig 3.1(d), while Fig 3.1(f) is the final solution with clusters identified by bold boxes.

	A3	A4	A5	A6	A7	A10	A1	A2	A8	A9
R2	1	1	1	1	1	1	0	0	0	0
R5	1	1	1	1	1	1	0	0	0	0
R7	1	1	1	1	1	1	0	0	0	0
R12	1	1	1	1	1	1	0	0	0	0
R13	1	1	1	1	1	1	0	0	0	0
R14	1	1	1	1	1	1	0	0	0	0
R16	1	1	1	1	1	1	0	0	0	0
R1	0	0	0	0	0	0	1	1	1	1
R3	0	0	0	0	0	0	1	1	1	1
R4	0	0	0	0	0	0	1	1	1	1
R6	0	0	0	0	0	0	1	1	1	1
R8	0	0	0	0	0	0	1	1	1	1
R9	0	0	0	0	0	0	1	1	1	1
R10	0	0	0	0	0	0	1	1	1	1
R11	0	0	0	0	0	0	1	1	1	1
R15	0	0	0	0	0	0	1	1	1	1

Fig 3.1(e): Discretized output

	A3	A4	A5	A6	A7	A10	A1	A2	A8	A9
R2	3	3.5	3	3	3	3.5	1.5	1	1	1
R5	3	3.5	3	3	3	3	1	1	1.5	1
R7	3	3	3.5	3.5	3	3	1	1	1	1
R12	3.5	3	3	3	3	3.5	1	1	1	1
R13	3	3.5	3	3.5	3	3.5	1.5	1	1.5	1
R14	3	3.5	3	3.5	3.5	3	1	1	1	1.5
R16	3	3	3	3	3	3	1	1	1	1
R1	1	1	1	1.5	1	1	5	4	4	4
R3	1	1.5	1	1	1	1	4	5	4	4
R4	1	1	1.5	1	1	1	5	4.5	4	4
R6	1	1	1.5	1	1	1	4	4	4	4
R8	1.5	1	1	1	1	1	5	4	4.5	4
R9	1	1	1	1.5	1	1	4	4	4	4
R10	1	1.5	1	1.5	1	1.5	4	4.5	5.5	5
R11	1	1	1	1	1.5	1	5.5	4	5	4
R15	1	1	1.5	1	1	1	4	4	4	5.5

Fig 3.1(f): Final output

The biclusters in Figure 3.1(f) were extracted based on the combination of properties of the rows and columns, and are optimal. Subsequently, the corresponding data can be ranked using additional processing/coding; which will help discover stripe-patterns.

Observe the other benefit of discretization i.e. noise tolerance for white noise as discussed in sections 2.2 and 3.4. The effect of noise is changing the attribute values. However, discretization resulted in the same binary matrix as would have been obtained for constant values i.e. discretization nullified the effect of noise as demonstrated in this example. Thus discretization not only reduced the complexity of the problem, but also made the technique noise tolerant.

Also observe that as a consequence of discretization, all numeric values get treated as a constant value, which includes coherent values as well as a coherent evolution. In the best case even if discretization causes removal of all noise, and the biclusters are captured intact, the coherency within the biclusters can not be ensured. This is visible in Fig 3.1(f), where the values within the rows and columns are not ordered.

Example 3.2

For ease of understanding, a rather simple case of mutually exclusive biclusters was described in example 3.1. However, the proposed technique is powerful enough to bicluster overlapping biclusters involving the overlap of either rows or columns or both. Furthermore, it is not necessary for the biclusters to exist along the diagonal of the data matrix. In this example Figure 3.2(a) shows a data matrix rotated by 90^o that consists of asymmetric and overlapping biclusters consisting of $K_{60,38}$, $K_{162,24}$ and $K_{44,18}$ (the bounding size of the data set being 266x80) with 2% noise added. Figure 3.2(b) shows the input randomly permuted. Figure 3.2(c) shows the permuted input almost perfectly biclustered using the proposed technique and overlapping biclusters extracted, shown by red rectangles. Note that the amount of overlap among the biclusters is also asymmetric. Almost perfect biclustering was found to be achieved by the proposed technique in the presence of high noise levels of up to 5%.

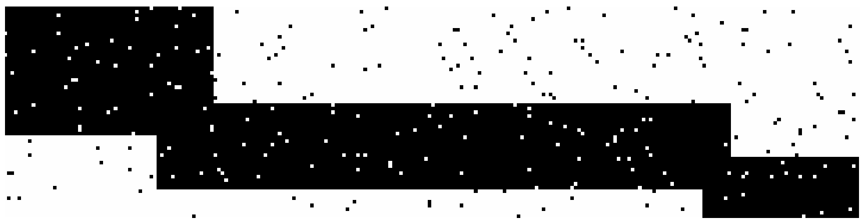


Figure 3.2(a): Input with overlapping biclusters and 2% white noise

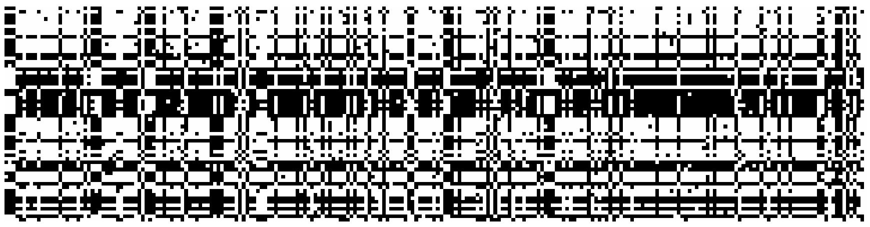


Figure 3.2(b): Permuted input of overlapping biclusters of Fig 3.2(a)

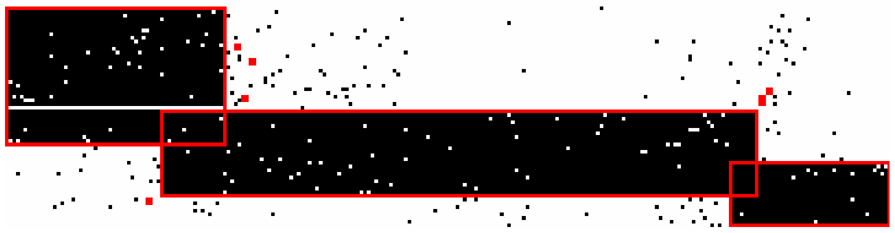


Figure 3.2(c): Biclustered output of Fig 3.2(b)

For comparative purposes, the randomly permuted data matrix shown in Fig 3.2(b) was used as input to the conventional SAMBA [78]. Three partial biclusters were extracted with sizes 44×11, 37×14 and 43×9 (with *Bic scores* of 409.82, 447.23 and 232.01 respectively). Note that the simulated data set actually consisted of biclusters of sizes 60×38, 162×24 and 44×18. Hence, as a result of the unimpressive performance of SAMBA on this relatively simple noisy data set it will not be further used for comparison purposes in this thesis.

3.8 Results & Comparison With Other Techniques

The aim of this section is to compare the quality of the results produced by the proposed technique with two other techniques namely Cheng & Church [13] and OPSM [5] using simulated as well as real data-sets in the presence of noise.

3.8.1 Comparative Performance Analysis Using Simulated Data

For a comparison with C&C a simulated data-set is used consisting of $K_{50,50} \cup K_{100,20} \cup K_{100,40}$ (the bounding size of the data set being 250x110). Figure 3.3(a) shows the actual data with $\alpha = 5\%$; for ease of comprehension the clusters are color-coded.

For quantifying the quality of biclusters, two standard measures are used namely Purity and Entropy. These are with reference to the original biclusters that exist in the simulated data and are known in advance. The value of Purity (P) and Entropy (E) will be between 0 and 1. The perfect bicluster has Purity of 1 and Entropy of 0. For a biclustered solution, the P and E are a weighted sum taking into account the size of the biclusters. For purposes of comparison, the C&C technique was used with appropriate data-set that resulted in good biclustering. The data set used with C&C was then discretized and the corresponding data matrix used as input to our proposed technique. A similar approach was adopted for comparing with the OPSM technique.

Figure 3.3(b) shows the randomly permuted input while Fig 3.3(c) shows the output obtained using the C&C technique. Interestingly, for $\alpha = 0\%$ C&C technique was found to require several iterations before termination, while our technique terminated in just a single iteration. Figure 3.3(c) shows the biclustering achieved using the C&C technique with 5% additive white noise which can be seen to result in the disintegration of biclusters (taking 2.75 CPU seconds), while biclustering using crossing minimization resulted in perfect biclustering and took 0.06 CPU seconds. In fact even for $\alpha = 20\%$

almost perfect biclustering was achieved using the proposed technique as shown in Figure 3.3(d).

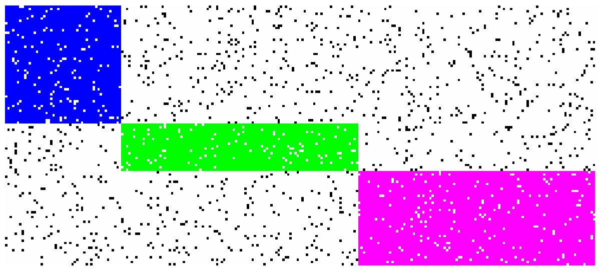


Fig 3.3(a): Input data matrix with α = 5%

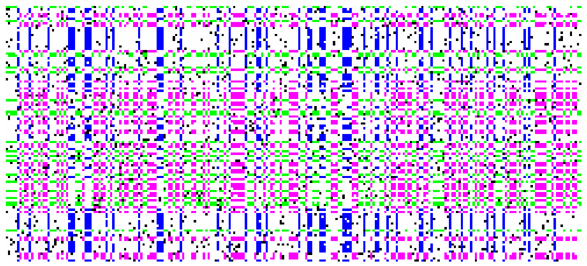


Fig 3.3(b): Randomly permuted input

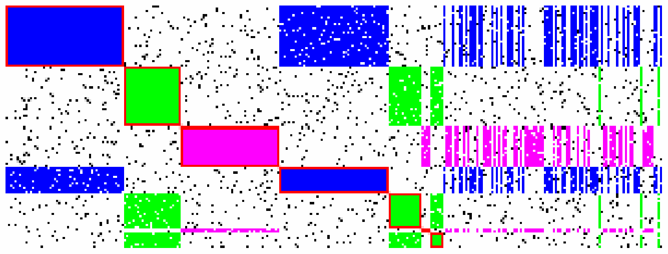


Fig 3.3(c): Disintegrated bi-clustered output obtained via C&C with α = 5%, P = 0.37, E = 0.

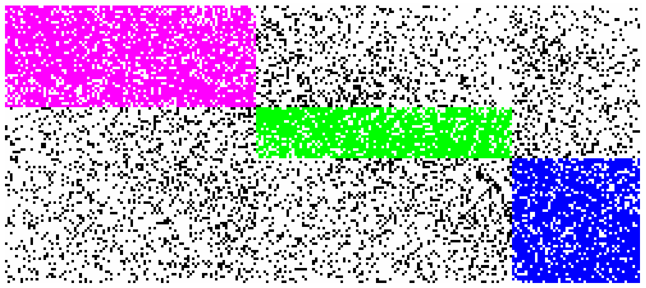


Fig 3.3(d): Almost perfect bi-clustered output using crossing minimization with α = 20%

Figure 3.3: Clustering of noisy data and comparison with C&C Technique

Observe that if the data matrix being considered has biclusters in certain order embedded with other data or noise, those biclusters may not be recovered with the same order. Compare Fig 3.3(a) with Fig 3.3(d), the order of biclusters has changed.

3.8.2 Visual (Qualitative) Comparison Using Real Data

The aim of this section is to compare the biclustering performance results of two techniques namely Cheng & Church and OPSM with our proposed technique using real data. The common data-set used for comparison purposes is publicly available at the University of California at Irvine (UCI) USA [8]. The dataset is the result of a chemical analysis of wines grown in the same region in Italy, but derived from three different cultivars or vineyards. The dataset consists of 178 records, 13 attributes and three classes (or clusters).

Before objectively quantifying the biclusters obtained via crossing minimization, we carry out visual data mining. For this purpose the data matrix is first normalized (based on the maximum value of each attribute) so that attributes with large values do not overshadow attributes with smaller values. The normalized color coded data matrix rotated by 90^o is shown in Fig 3.4(a). Here, bright green pixels corresponds to 1 and black to 0, with in-between values shown by shades of green. Observe that the shades of green are not very discriminating visually; therefore, the data matrix is discretized based on the median value of each column. The discretized version of the same data matrix is shown in Fig 3.4(b) which is more discriminating visually. Note that in Figures 3.4(b through e) red rectangles represent the biclusters extracted. It is actually within the control of the end user to decide what size of biclusters to extract, and which of the small biclusters (say of size 1x1) should be dropped from further processing.

For ease of comprehension, a small data-set is used that can be completely displayed on a single screen of 600×800 resolution. However, the biclustering and bicluster extraction methods both work for large data-sets consisting of several thousand rows and hundreds of columns. Such large data-sets cannot be completely displayed on a single screen, but can be processed in the main memory. Once all of the tasks are performed, the biclustering tool writes the details of extracted biclusters (rows, columns, size, etc.) in text files for further analysis/processing.

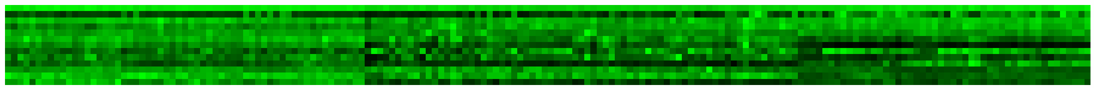


Figure 3.4(a): Input wine dataset: Color-coded, normalized data matrix

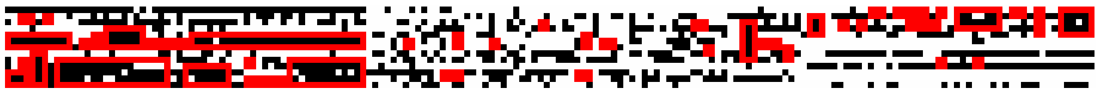


Figure 3.4(b): Input wine dataset: Biclusters extracted in discretized data matrix

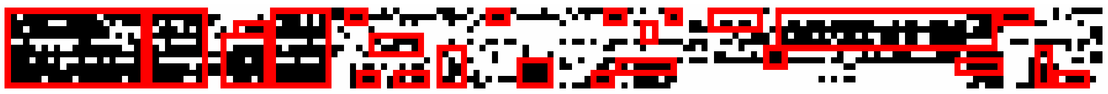


Figure 3.4(c): Output wine dataset: Biclusters extracted using <u>OPSM</u>

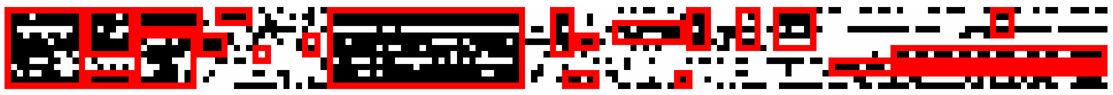


Figure 3.4(d): Output wine dataset: Biclusters extracted using <u>Cheng & Church</u>

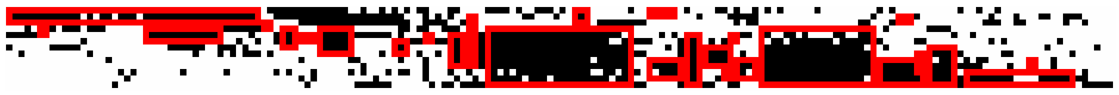


Figure 3.4(e): Output wine dataset: Biclusters extracted using <u>Crossing Minimization</u>

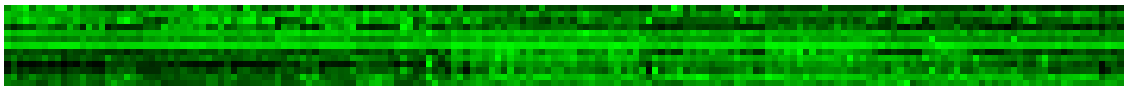


Figure 3.4(f): Output wine dataset: Color-coded, normalized biclustered data matrix obtained using crossing minimization

Given a data matrix consisting of objects (rows) and attributes (columns), the goal of a typical biclustering technique is to order the rows and columns, so as to discover 'dense' regions of the data matrix i.e. groups of rows or records with close similarity in the same subset of attributes or columns. The biclustering results of the three techniques are next evaluated through visual inspection. This visual inspection will quickly identify the biclusters of interest, as the tool based on the proposed technique can display both color coded data matrix, as well as binary data matrix (with zoom_in and zoom_out facility).

Fig 3.4(c) was obtained using OPSM with $s = 8$ (number of columns of biclusters), ℓ = 40% (representing pool size of partial models) and the solution took 458.63 CPU seconds. Comparing the original Fig 3.4(b) with Fig 3.4(c) a slight grouping of black dots or 'dense' regions are observed indicating a slight enhancement of biclustering. Note that this is the best result obtained after experimenting with several different values and conditions of s and ℓ.

Fig 3.4(d) is the output obtained using the C&C technique with $n = 3$, $\delta = 0.04$ (mean residue score), and the solution took 0.11 CPU seconds. Comparing Fig 3.4(b) with Fig 3.4(d) a modest grouping of black dots is again observed i.e. some enhancement of biclustering. Note that this is the best result obtained after experimenting with several different values and conditions of *n* and δ.

Fig 3.4(e) shows the output obtained after using the proposed crossing minimization technique with three main biclusters extracted. Note that there were no user supplied input parameters, and the result was obtained in less than 0.01 CPU seconds without the need for any trial and error. Comparing Fig 3.4(b) with Fig 3.4(e) a significant grouping of black dots or 'dense' regions are observed demonstrating significantly enhanced biclustering. Fig 3.4(f) shows the color coded normalized biclustered data matrix corresponding to Figure 3.4(e).

In conclusion, it can be seen from the above discussion that the C&C and the OPSM techniques did not yield any significant biclustering ('dense' regions). Therefore, further analysis of their output will not be performed. In the next section a qualitative analysis of the 'dense' regions created using our crossing minimization technique is presented.

3.8.3 Quantitative Analysis of Results Using Real Data

The aim of this section is to firstly give an overview of the biclustering results obtained using the crossing minimization technique with a real wine data set, and secondly to carry out an analysis of those results to ascertain the quality of biclustering achieved (via crossing minimization) and finally report new findings.

For this purpose the wine data-set used is first discussed in more detail. It consists of 14 features (including class information) and 178 records. Note that for actual biclustering process the class information was not used. Necessary details of the data set used are as follows:

Parameters (number in parenthesis is attribute ID)**:**

a. **Chemical properties:** (1) Alcohol, (2) Malic Acid, (3) Ash, (4) Alkalinity of Ash, (5) Magnesium, (6) Total Phenols, (7) Flavonoids, (8) Non-Flavonoid Phenols, (9) Proanthocyanin, (12) OD280/OD315 of Diluted Wines, (13) Proline

b. **Optical properties:** (10) Color, (11) Hue

c. **Goal parameter:** Vineyard (classes 1, 2, and 3)

A quick overview of the biclustering results obtained will now be presented. The biclustering using crossing minimization resulted in 27 biclusters (BC), out of which only those biclusters will be considered for further analysis that consist of more than three records and attributes. That leaves us with 11 biclusters, with details given in Table 3.1.

		Attributes (columns)													**Class**			
BC	**Records**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**1**	**2**	**3**	
1	41		x		x				x						0	8	33	Class_4 (new)
8	13	x		x		x					x				0	0	13	Class_3
10	6	x		x		x					x			x	0	0	6	
11	24	x		x		x	x	x		x	x	x	x		24	0	0	Class_1
12	19	x		x		x	x	x		x	x	x	x		17	0	2	
19	5	x					x	x						x	4	1	0	
20	5	x					x	x		x		x	x	x	5	0	0	
22	6						x	x		x			x		1	5	0	Class_2
23	4						x	x		x			x		0	4	0	
24	8						x	x		x		x	x		0	8	0	
27	18									x		x	x		0	18	0	

Table 3.1: Bicluster details with class information

From a cursory view of Table 3.1 it is obvious that the number of class members discovered as a result of biclustering is less than the total records in the wine data set. The reason for this is that the classes (or one-way clusters) in the original wine data set are based on all 13 attributes, whereas the biclusters discovered are based on a fewer number of highly similar attributes and records.

The class members discovered due to biclustering are as follows: For class_1: 50 members discovered, for class_2: 35 members discovered and for class_3: 19 members discovered. Before proceeding further with the analysis, it would be prudent to ascertain the "goodness" or the quality of the biclusters extracted. This will be done by taking a weighted ratio of the class members within a bicluster with the total number of records present in that bicluster. For example, the class accuracy of BC_11 (i.e. bicluster no. 11) consisting of 24 records of class_1 is 100%, as is the class accuracy of BC_27 that consists of 18 records of class_2.

For the un-clustered input, 44 biclusters were discovered (as shown in Fig 3.4b). Several instances of class_1 were discovered having biclusters with overlapping rows; with the largest bicluster corresponding to class_1 comprising 29 records. Not a single instance of class_2 was discovered in the un-clustered input, while 26 instances of class_3 records were discovered spread across multiple biclusters.

Next, the biclusters shown in Table 3.1 are qualitatively analyzed and interpreted. From Table 3.1 it is observed that BC_11, 12, 19 and 20 correspond to class_1, and BC_22, 23, 24 and 27 correspond to class_2, whereas BC_8 and BC_10 correspond to class_3. Note that, over here a class corresponds to a vineyard. This means that there is a strong connection between biclusters obtained and the vineyards (or classes). BC_1 however is a special case, as it consists of members of class_2 and class_3 with a very unique group of attributes. Out of 41 records in BC_1, 8 belong to class_2, while the remaining belongs to class_3. This apparent anomaly can only be ascertained after further investigation. In the entire wine dataset, for class_2 the average value of these three attributes (2, 4, 8) Malic acid, Alcalinity of ash and Nonflavanoids are 1.92, 20.2 and 0.36, respectively. Similarly, for class_3 the average values of these three attributes were found to be 3.37, 21.48 and 0.45, respectively. Since the average values of three common attributes (between class_2 and class_3) are quite similar for BC_1, it may be concluded that there could be a new class_4 present comprising of these attributes subset. A stacked line plot of normalized values of these three attributes (2, 4, 8) for BC_1 is shown in Fig 3.5 with highly similar behavior that further reinforces our conclusion.

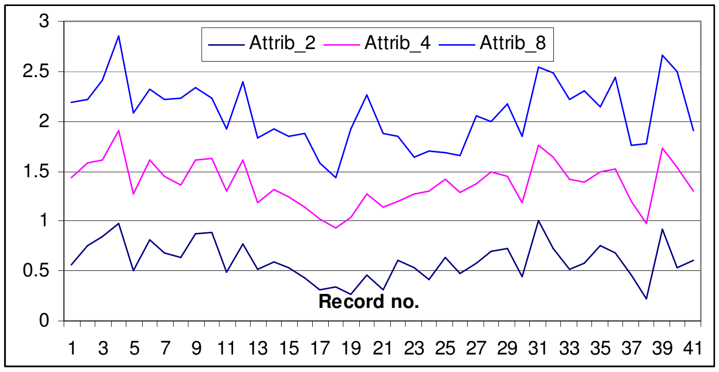


Figure 3.5: Stacked line plot of attributes 2, 4, and 8 in BC_1 (class_4)

From Table 3.1 it can also be observed that there is also a connection between the biclustered attributes and vineyards (or classes). The common attributes existing in all biclusters corresponding to a particular class are as follows:

a. For class_1 the common attributes are (1, 6, 7).
b. For class_2 the common attributes are (9, 12).
c. For class_3 the common attributes are (1, 3, 5, 10).

Thus it can be concluded from Table 3.1 that class_1 and class_2 have nothing in common based on attribute 1 (i.e. alcohol), although there is some partial overlap between other attributes among the two classes. It is a known fact that grape juice begins fermenting naturally within 6-12 hours. Once fermentation begins, this normally continues until all of the sugar is converted to alcohol and a dry wine is produced. The resulting level of alcohol in wine will vary from one locale (vineyard) to the next, due to the total sugar content of the juice. An alcohol level of 10% in cool climates versus a high of 15% in warmer areas is considered normal [41] i.e. the alcohol content varies from vineyard to vineyard (classes). This can also be further confirmed through a simple scatter plot of normalized values of alcohol attribute for the two classes, see Fig 3.6.

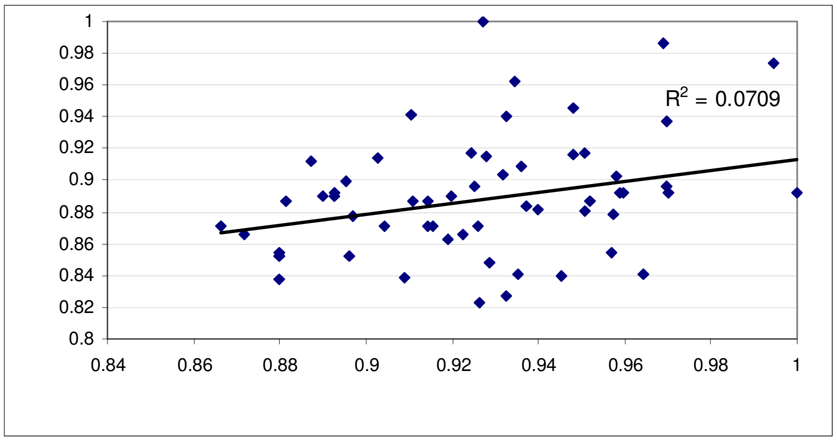


Figure 3.6: Scatter plot for class_1 and class_2 using the alcohol attribute

The scatter plot shows that based on the alcohol attribute both classes are indeed correctly identified to be mutually exclusive because of insignificant R^2 value.

3.9 Conclusions

In this chapter it was successfully demonstrated that the proposed technique does perform biclustering, and its results are better than other contemporary traditional techniques. The quality of the results were verified by biclustering a public domain wine data set, in which some new sub-classes of clusters were also identified. One of the challenges of biclustering is the amount of noise present in the data. In the next chapter an attempt will be made to compare two biclustering techniques on the basis of noise present in the gene expression data using simulated as well as real data.

Chapter 4

Robustness to Noise

4.1 Introduction

In Chapter 3 a new biclustering technique based on the crossing minimization paradigm was presented. In this chapter the proposed technique will be compared with an Order Preserving Sub Matrix (OPSM) technique [5] with specific reference to the quality of results for noisy simulated data as well as gene expression data. A brief overview will also be given about the Micro-array technology with specific reference to noise.

4.2 Background

The developments in DNA arrays enable simultaneous measurements of the expression level of thousands of genes. These methodologies have led to an explosion in the rate at which gene expression data is accumulated. One of the challenges in this regard is the inherent high level of noise present. Unfortunately, the noise often hides the patterns of interest – for example, the data often contains 'technical' and 'biological' noise [48]. Several potential sources of measurement of noise are discussed by Dror in [21].

Analyzing the DNA micro-array data involves some form of 'grouping' that is biologically significant. Due to certain well-known limitations of traditional (one-way) clustering techniques, several (two-way) clustering or *biclustering* techniques have been recently proposed. Biclustering techniques aim at finding local patterns in data sets where the effect of noise is more profound. Instead of finding a similarity in genes under all conditions, biclustering discovers subspaces (subset of genes and subset of conditions), which are coherent in some way or other.

4.2.1 Micro-Array Technology

Micro-array is a slide of glass or some other substrate on which thousands of DNA molecules are attached at fixed locations (spots). Each spot corresponds to a specific gene in a cell. RNA from the sample of interest is taken and hybridized with this array. The amount of hybridized RNA is measured and this gives the expression level of a particular gene. Various steps involved in micro-array experiments are summarized as follows (for details interested readers are referred to [19]):

a. DNA micro-array production
b. Sample preparation
c. Hybridization between mRNA and array
d. Signal Analysis
e. Extraction of hybridization

4.2.2 Noise In Gene Expression Data

Micro-array technology is most commonly available in two forms i.e. GeneChip®, a trademark used by Affymetrix Inc, that relies on DNA oligonucleotide for chip manufacturing and Spotted Arrays developed at Stanford University, which uses a glass slide on which DNA is immobilized using a robot arm.

These technologies enable the measurement of the expression levels of thousands of genes in a cell simultaneously under one experimental condition, or of a particular tissue type and give a global view of the cell. However, DNA micro-array technology is not without shortcomings. The production of a gene chip involves a large number of error-prone steps that lead to a high level of noise in the data. Three forms of noise or limitations of micro-arrays are highlighted and studied.

4.2.2.1 Noisy Nature of Data

The actual gene expression value changed, this change is due to ‘experimental’ (technical) noise or by ‘biological’ noise [48]. Experimental noise is caused by the different stages of micro-array experiments listed above. A careful design and selection of the immobilized DNA and planning for the hybridization of array with samples is required. Biological variations are due to real differences, impurities or a misclassification between the different cell types and tissues being compared.

4.2.2.2 Missing Values

Gene expression data sets often contain missing values for various reasons. For example, the background and the signal may have similar intensities, image corruption, or the surface of the chip may not be planar; there may be dust or scratches on the slides or the probe may not be properly fixed on the chip or washed properly, the hybridization step may not work properly. Furthermore, systematically missing values could be due to the robotic method for the generation of gene expression profiles. Researchers have adopted several approaches to deal with the missing values, such as (i) a costly and time consuming method of the repetition of identical experiments to validate downstream micro-array analysis or (ii) simply omitting the expression profile vector with missing values, and padding them with zeros or row averages. Although these methods are widely used by the biologists, their disadvantages are obvious: omitting the profile vector results in losing useful information; padding them with zeros and row averages do not provide proper missing value estimation [85]. Researchers have proposed several algorithmic approaches, but their discussion is beyond the scope of this chapter.

4.2.2.3 Hybridization Noise

DNA sequencing-by-hybridization (SBH) was proposed about two decades ago [10] as a potentially powerful alternative to the costly and time-consuming, yet effective, electrophoresis techniques. SBH consists of two fundamental steps. The first step involves the acquisition of all sets of sub-sequences (of a selected pattern) also called the *sequence spectrum* of a given *unknown target sequence* using the complementary hybridization process on a complete library of probes, this step being biochemical in nature. The second step is, *combinatorial* in nature i.e. the *algorithmic reconstruction* of the original target sequence from its sequence spectrum. Although SBH is elegant in concept, serious difficulties in the fields of biochemistry (the use of a “universal” base in gapped probes) and combinatorial algorithms have prevented SBH from being widely operational [70].

In the real-world application of DNA sequencing, random hybridization errors (hybridization noise) occur in the form of false positives (false-hits) and false negatives (misses). Intuitively, a false positive is not as detrimental as a false negative. In the simulation of SBH, the approach to error control requires a presupposition of an error model, i.e. a formalization of the random process which produces hybridization errors [15]. Unfortunately, knowledge of the hybridization process is currently not available for a precise quantification of the hybridization model. Therefore, the studies by the researchers about the graceful degradation in the efficiency of the reconstruction process are based on the following error process (modified standard model):

a. Any spectrum probe can be suppressed with a fixed probability (false negatives);
b. Any probe at Hamming Distance 1 from a correct spectrum can be added to the spectrum with a fixed probability (false positives);
c. Hybridization noise is expressed in terms of error rates for false negatives and positives.

4.2.2.4 Irrelevant Genes

DNA arrays provide a global view of the cell by enabling the measurement of the expression levels of thousands of genes simultaneously. Among the expression levels of these thousands of genes, a majority of the genes are irrelevant to the class distinction. A class can be discriminated through only a small subset of genes whose expression levels strongly correlate with the class distinction. These genes are called *informative genes*. The remaining genes in the gene expression matrix are irrelevant to the division of samples of interest and thus are regarded as noise in the data set [38]. Therefore, the combined effect of a large number of irrelevant genes can potentially obscure the contribution of the relevant genes.

These irrelevant data points or genes have an undesirable effect on clustering. For example, Self-Organizing Maps (SOM) will produce an output in which this type of data will populate the vast majority of clusters [39]. For a case such as SOM it is not effective because most of the interesting patterns may get diffused into only one or two clusters and cannot be identified. The amount of noise is less than 10%, but still when other classical clustering technicians such as K-means and Hierarchical Clustering (HC) are used when considering all genes as features, undesirable results might be obtained.

Example 4.1

In this example the effect of noise will be demonstrated for simulated gene expression data consisting of three clusters. Consider Figure 4.1 which shows examples of some of the types of noises discussed using the model proposed in the previous chapters. Figure 4.1(a) shows simulated gene expression data with 20% missing values from the clusters, and 20% hybridization noise. The resultant output, with clusters extracted successfully is shown in Fig 4.1(b). In another experiment, the amount of noise is increased to 35% i.e. 35% missing data and at the same time, 35% hybridization noise as shown in Fig 4.1(c). The resultant output, with only one cluster extracted successfully, is shown in Fig 4.1(d). Hence we see that with the increase of noise, cluster quality deteriorates.

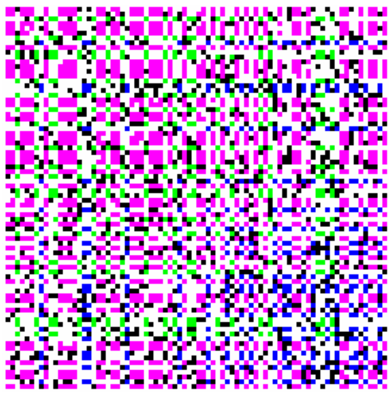


Fig 4.1(a): 20% missing values, 20% hybridization noise

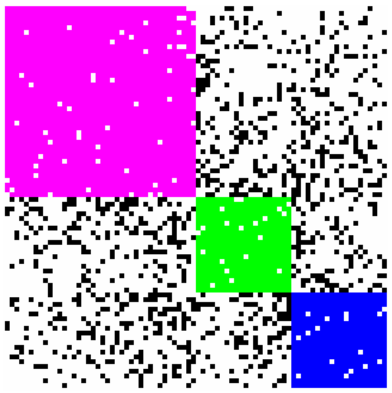


Fig 4.1(b): Resultant successful clustering and extraction

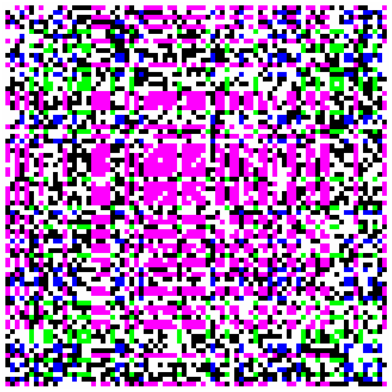


Fig 4.1(c): 35% missing values, 35% hybridization noise

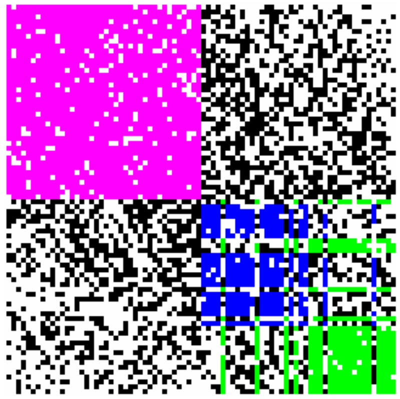


Fig 4.1(d): Resultant unsuccessful cluster extraction

Figure 4.1: Simulated data with missing values and hybridization noise

Researchers have worked in all three dimensions and various noise models have been proposed for dis-associating noise from actual value, estimating missing values and devising techniques for the selection of relevant genes. A study of those techniques is beyond the scope of this chapter, but the point being made here is that noise in gene expression data is more of a rule than an exception.

4.3 Review of Previous Work

This section provides an overview of the related work done in the field of biclustering and the noise models proposed and used.

Ben-Dor et al. [5] defined a bicluster as an order-preserving sub-matrix. According to their definition, a bicluster is a group of rows whose values induce a linear order across a subset of the columns. The given bicluster (Fig. 4.2) is an example of OPSM where columns are arranged in order c4 < c2< c3< c1 linear

c1	**c2**	**c3**	**c4**
70	13	19	10
69	40	49	35
40	20	27	15
90	15	20	12

(a) Input Data

c4	**c2**	**c3**	**c1**
10	13	19	70
35	40	49	69
15	20	27	40
12	15	20	90

(b) Output Data

Figure 4.2: An example of an OPSM bicluster

The data consists of $n \times m$ matrix D, where the rows correspond to genes and the columns to tissues (or, more generally, to conditions). OPSM finds an order-preserving sub-matrix for which there is a permutation of its columns, under which the sequence of values in every row is strictly increasing. The expression of data in a sub matrix is determined by a set of genes J and a set of tissues T such that, within the set of tissues T, the expression levels of all the genes J have the same order.

Teramoto et al. in [78] have proposed a graph-theoretic approximation algorithm for clustering gene expression data. The method classifies the data using a *p*-quasi complete graph. The experimental results showed that the proposed method could avoid the effects of noise or irrelevant elements and also give better biological results than the classical, but dated, hierarchical clustering method. The method was shown to be potentially useful in discovering the unknown disease classes in more detail with the results of clustering. However in order to get good results, a parameter setting is needed, which is only possible in the case that prior knowledge is available. Our proposed technique has been shown to work with white noise and performance compared with contemporary clustering techniques. Furthermore, no performance tuning is required based on prior knowledge.

Y. Tu et al. [74] have analyzed experimental noise quantitatively. They have decomposed total experimental noise into two parts: sample preparation (pre-hybridization) noise and hybridization noise. They found that noise from pre-hybridization does not depend on expression levels and is of constant nature whereas noise due to the hybridization step depends on expression level and varies depending on large and small expression values. They measured the noise by replicate experiments as the sample is divided into groups and each group is passed through different steps, multiple times independently.

Bayesian Estimation of Array Measurement (BEAM) [23] produces an estimate of expression level and a confidence level of this estimate. Given one or more gene array measurements, a statistical model of measurement noise and any available prior information about the transcript levels, BEAM produces a statistically optimal estimate of an expression level or expression level ratio as well as a measure of its uncertainty.

In [38] a probabilistic model for the measurement of noise in micro-array systems was presented. The model includes the inherent Poisson noise of the array as well as the systematic errors which are typically introduced during micro-array fabrication and detection processes. The model presented formulates not only the uncertainty of the measured expression levels, but also the contribution of each procedural step to the overall detection of the signal-to-noise ratio (SNR).

4.4 Optimality of the Proposed Technique

In section 4.4.1 we will formally prove the optimality of one of the CMH i.e. MH for pure clusters i.e. $\alpha_E = \alpha_I = 0$. In section 4.4.2 we will discuss the performance of CMH when both α_E and α_I are >0.

4.4.1 Noiseless Data

For biclustering the data matrix being considered is not likely to be square, but for notational convenience and with no loss of generality a square matrix is considered i.e. $|V_1| = |V_0| = n$ i.e. rows of data matrix and $p = q$. Although in this section we will discuss MH, similar reasoning can be used to show the optimality of BaryCenter Heuristic (BCH) and MS for a union of $K_{p,q}$ under noiseless conditions.

Theorem 1

For a $K_{p,q}$ the median of each vertex in the BOT will be $\left\lceil \frac{p}{2} \right\rceil$ and in TOP it will be $\left\lceil \frac{q}{2} \right\rceil$.

Theorem 2

For an optimal permutation of vertices for a union of $K_{p,q}$ i.e. G^*_B, the medians of vertices are in ascending order in both TOP and BOT partitions.

Theorem 3

For a union of $K_{p,q}$ if all the medians of vertices are in ascending order in TOP and BOT, then it is an optimal permutation.

Theorem 4

A permutation of union of $K_{p,q}$ is optimum $\Leftrightarrow$ All of the vertices of the union of $K_{p,q}$ are sorted in ascending order of their medians.

Theorem 5

MH terminates with an optimal permutation of a union of $K_{p,q}$.

Noiseless analysis applicable for noisy data when discretization eliminates noise.

4.4.2 Noisy Data

Consider G_B that is a union of cliques with both α_E and $\alpha_I > 0$, (section 2.2) resulting in corruption of the representative value of each vertex. The effect of noise can be minimized, (even eliminated), if the true representative value of each vertex can be estimated in the presence of noise. In other words, given a set of points in uni-dimensional space, find a point which best describes the set. When dealing with problems such as above, it is important to consider the issue of *robustness*: how much does the representative value change if some of the data is disturbed? An example of a non-robust estimator is the *mean*. If any point in the data set is placed at infinity, the *mean* will literally shift to infinity; the same is true for maximum and minimum values. This explains the relatively weak performance of BCH and MS under noisy conditions when used for clustering.

Conjecture

For asymptotically large *n*, MH redraws G_B into G^*_B with a very high probability, if the probability of white noise i.e. $\alpha < 0.5$ (i.e. $\alpha_E = \alpha_I < 0.5$).

From theorems 1 and 2 when $\alpha = 0$, a cluster corresponds to all vertices that have the same representative value; therefore, any CMH will group the vertices belonging to a clique. Permuting the vertices may change the representative value of every vertex; however, all the vertices corresponding to a clique will still have the same representative value, thus G^*_B is recovered in just two iterations of any CMH.

However, when $\alpha > 0$, the representative values do change and that change is dependent on the amount of noise and the robustness of the representative value used. For a sample of size n and uni-dimensional data set the breakdown point of the median is $\frac{n+1}{2n}$ for odd n or $\frac{1}{2}$ for even n because at least half of the points in the sample need to be replaced with sufficiently high or sufficiently low values before the median would be higher or lower than any bound. Intuitively, when an estimator of location has breakdown point ξ, then the estimator would still reveal location even if $\lfloor \xi n \rfloor$ data elements were corrupted.♦

4.5 Results Using Simulated Data

For quantifying the quality of biclusters two measures are used i.e. Purity and Entropy. These are with reference to the original biclusters that exist in the simulated data and are known in advance. The value of the Purity (P) and Entropy (E) will be between 0 and 1. The purest bicluster has a Purity of 1 and Entropy of 0.

Purity of a bicluster is a measure of how pure the bicluster is. The Purity of a biclustering solution can be defined as the number of points (data elements) in that bicluster which also belong to the original bicluster. Entropy is a measure of how disordered a given bicluster is. The Entropy of a bicluster is defined as the number of points (data elements) of other biclusters that are included in that bicluster.

We consider a simulated data set consisting of $K_{15,5} \cup K_{30,5} \cup K_{55,5}$ (bounding size of the data set being 100x15) where $K_{m,n}$ represents a bipartite clique with *m* and *n* vertices in each bipartition. The simulated data set is intentionally kept small due to the execution time limitations of OPSM. The graphs of P and E versus varying white noise are shown in Figure 4.3. Observe that at 10% noise the proposed technique has a very high purity i.e. more than 60%, while the purity for OPSM has fallen to just 20%. As the noise increases, the quality of results using crossing minimization is consistently better. Similarly, better quality results are obtained for Entropy i.e. less than 10% using crossing minimization while very high Entropy of about 50% for OPSM.

Figure 4.4 shows the visualization of the same data set; where black dots represent noise, white spots (or dots) represent missing data or noise, while each cluster is represented by a different color. In Fig 4.4(d) the red rectangles represent the extracted clusters.

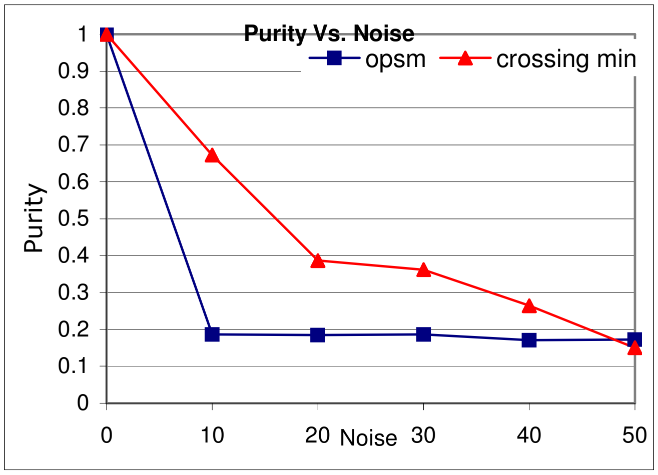


Figure 4.3(a): Effect of noise on the Purity of biclusters

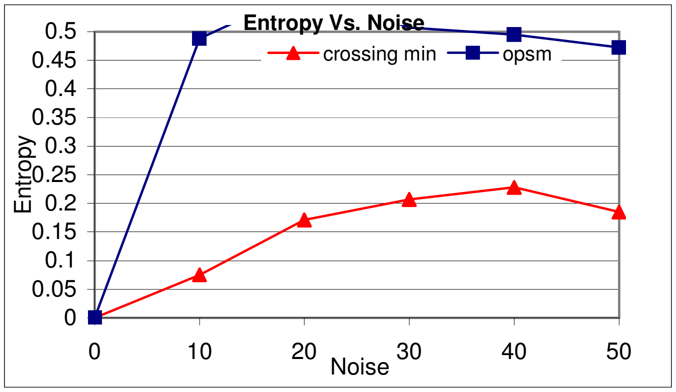


Figure 4.3(b): Effect of noise on the Entropy of biclusters

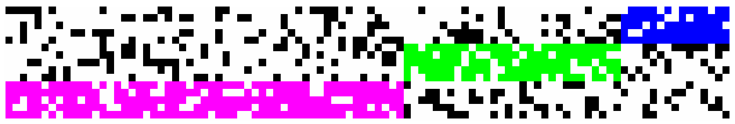


Figure 4.4(a): Input with 25% white noise

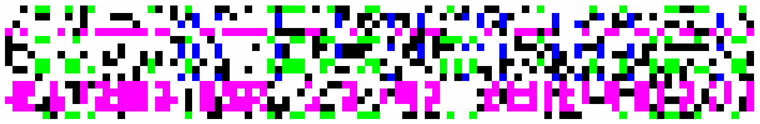


Figure 4.4(b): Randomly permuted Fig 4.4(a)

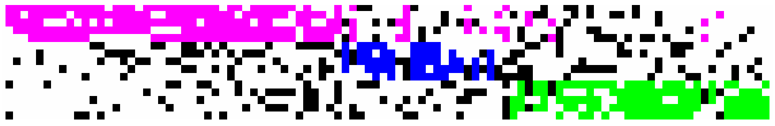


Figure 4.4(c): Output using crossing minimization

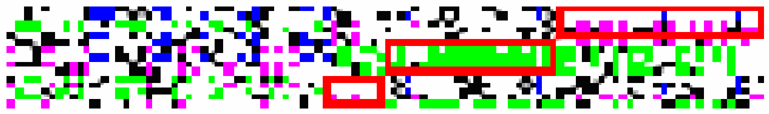


Figure 4.4(d): Output using OPSM

Figure 4.4: Effect of noise on biclustering using crossing minimization and OPSM

Observe that in Fig 4.4(c) the biclusters formed using crossing minimization are well defined, while in Fig 4.4(d) corresponding to biclusters formed using OPSM have disintegrated. This disintegration explains the relatively inferior performance of OPSM with reference to the proposed technique.

4.6 Results Using Real Data

Experiments were performed using real gene expression data set *gppca* (59x11) downloaded from http://ep.ebi.ac.uk/EP/EPCLUST/. Because of the inherent limitations of speed and exhaustive combinations made by OPSM a small data set was used i.e. consisting of 59 rows and 11 columns. For biclustering using OPSM the number of columns was varied from 2 to 11 and the time to completion was noted. The graph plotted between numbers of columns used vs. time is shown in Figure 4.5.

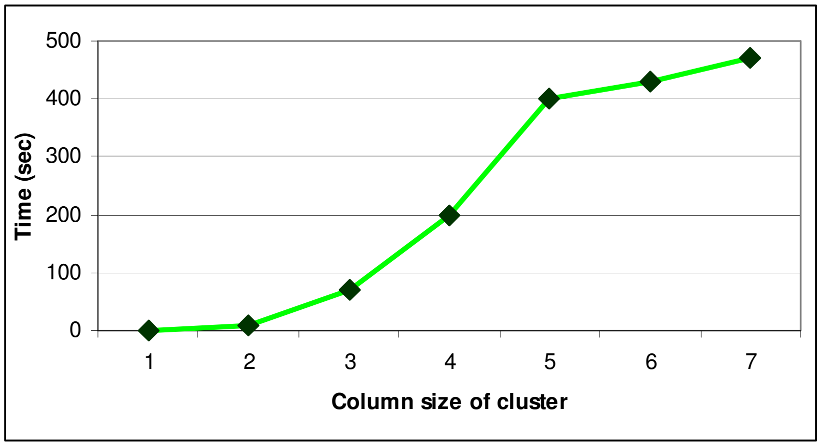


Figure 4.5: Effect of column size vs. time for OPSM

Note that for the given data set, biclustering using crossing minimization took less than a second.

For simplicity it is assumed, that the real data set is without any noise, therefore, before using it white noise is added to the data set. Using OPSM, two clusters were extracted with 24 and 21 rows, respectively. Note that the number of clusters and the number of columns (i.e. 4) within a cluster remained the same with 0% noise as well as 10% noise. The pool size for partial models maintained was 45% [5].

For biclustering by crossing minimization the numbers of columns are not required to be specified prior to extraction, hence resulting in more natural biclustering. Using this technique three major clusters were extracted as shown in Fig 4.6(c). With 10% noise added, the bicluster demographics changed, with the two larger biclusters collapsing into a single cluster consisting of 15 rows and a smaller cluster remaining more or less the same (i.e. 7 columns). Since the smaller bicluster is closer in demographics to the two biclusters extracted using OPSM we use it for comparison purposes.

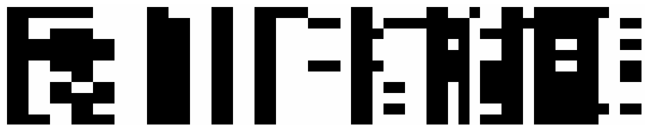


Figure 4.6(a): Input with 0% noise

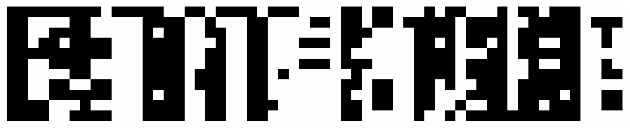


Figure 4.6(b): Input with 10% noise

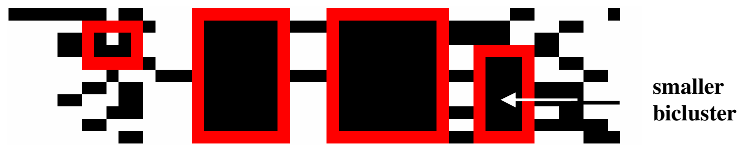


Figure 4.6(c): Output with 0% noise

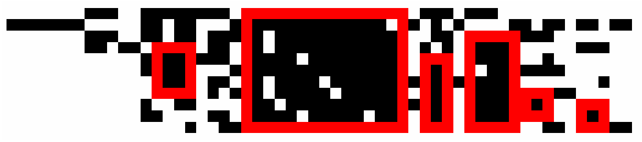


Figure 4.6(d): Output with 10% noise

Figure 4.6: Biclustering real data with simulated noise

As clustering as well as biclustering results in the grouping of similar values, one measure of the quality of the results could be the standard deviation (*Stdev*) of each column. The *Stdev* of the columns in the two biclusters extracted using OPSM and the smaller bicluster using crossing minimization are shown in Figure 4.7.

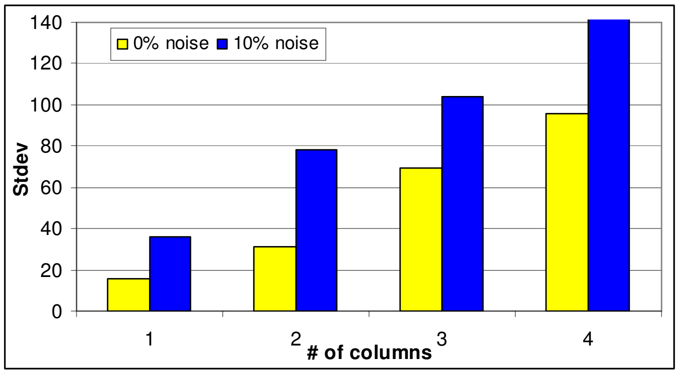


Figure 4.7(a): Biclustering results using OPSM

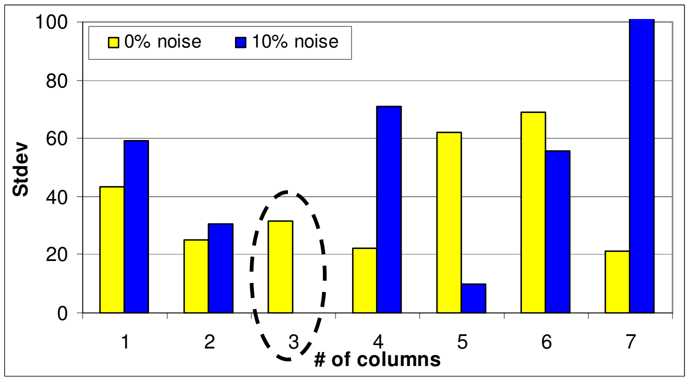


Figure 4.7(b): Biclustering results using Crossing Minimization

Figure 4.7: Quantification of biclustering results using OPSM and crossing minimization

Figure 4.7(a) shows that even with only four columns the *Stdev* using OPSM on an average is above 70 with and without noise. Figure 4.7(b) shows that for biclustering using crossing minimization, even with seven columns, the average value of *Stdev* is about 40. Thus noise had a profound effect on biclustering using OPSM as compared to biclustering by crossing minimization, even on real data with smaller biclusters.

It can be seen from Fig 4.7(b) that for 10% noise and for three columns the *Stdev* is zero. This result is attributed to the fact that the **gpcca** data set used has a certain column with six occurrences of 142. As a consequence in the absence of noise the corresponding rows clustered together along with other rows (with value other than 142) resulting in a large bicluster with a non-zero *Stdev* for that particular column. With the addition of 10% noise, the original bicluster (containing six occurrences of 142) was split into multiple biclusters, one of which is a smaller bicluster (with only three rows and columns), and all three rows having 142 present in the same column result in a *Stdev* of 0 for the said column.

4.7 Conclusions

Given the variety of available clustering algorithms, one of the problems faced by biologists is the selection of the algorithm that is the most appropriate for a given gene expression data set. However, there is no single best algorithm which is the winner in every aspect. Researchers typically select a few candidate algorithms and compare the clustering results. In this chapter a comparison is made between the proposed technique and OPSM using noisy data, this can help an end user make the best choice.

Chapter 5

Novel application of Crossing Minimization: Minimum Attribute Subset Selection (MASS)

In Chapter 3 a new biclustering technique based on the crossing minimization paradigm was presented, in this chapter another novel application of the crossing minimization paradigm will be shown to work effectively for an important NP-Complete problem i.e. the Minimum Attribute Subset Selection (MASS) problem. The problem will be discussed in detail in this chapter, and using a real public domain data set the effectiveness of the proposed techniques will be demonstrated qualitatively and quantitatively.

5.1 Introduction

Every day business managers, biologists, analysts, agriculture scientists etc. are confronted with large datasets to be explored and analyzed-effectively and quickly. The number and complexity of such datasets is growing by the day. Extracting knowledge from these datasets has two conflicting objectives (i) present the data using all the relevant attributes in an intuitive and easy to understand way for non technical users, such that (ii) the data retains all the important and relevant relationships. Thus there is a need for clustered visual representation of useful and relevant data for visual data mining.

Thus an important and critical issue in the data preprocessing step of the overall Knowledge Discovery in Databases (KDD) is the selection of useful and relevant attributes allowing a data mining algorithm to discover useful patterns effectively and quickly. It is typical of the data mining domain that the number of attributes is very large, and all of them may not be effective in the overall knowledge discovery process. A possible way is to use the domain expert's knowledge for the identification of most suitable attributes, and then run the data mining algorithms and hope that they give the most useful results with the selected attribute set. However, deciding 'suitability' is an issue, and only quantifying and relating it to some measure can make it automatic and objective.

5.2 Background

Among a number of factors affecting the success of a data mining algorithm is data quality. No matter how "intelligent" a data mining algorithm is, it will fail in discovering useful knowledge if applied to low-quality data [69]. The irrelevant, redundant, noisy and unreliable data makes the knowledge discovery task difficult. The MASS problem is the process of identifying and removing as much of the irrelevant and redundant information as possible [34]. A carefully chosen subset of attributes improves the performance and efficacy of a variety of algorithms [81]. A relevant feature is neither irrelevant nor redundant to the target concept; an irrelevant feature does not affect the target concept in any way, and a redundant feature does not add anything new to the target concept [22].

The subset selection problem is also known as dimensionality reduction, as it is the presentation of high dimensional patterns in a low dimensional subspace based on a transformation which optimizes a specified criterion in the subspace [24]. Dimensionality reduction is most useful when each of the *n* inputs carry information relevant to classification i.e. need a *global view* of data. On the other hand, subset selection is appropriate when it is possible to completely discard some attributes (i.e. irrelevant attributes) with low probability of misclassification [24] i.e. need a *local view* of data.

The MASS problem involves finding a 'good' set of attributes under some objective function that assigns numeric measure of quality to the patterns discovered by the data mining algorithm. It has long been proved that the classification accuracy of Machine learning algorithms is not monotonic ((i.e., a subset of features should not be better than any larger set that contains the subset [22]) with respect to the addition of features. Some possible objective functions for classification could be the prediction accuracy, misclassification cost, pattern complexity, minimal use of input attributes [43] etc. In this chapter, the objective function is the crossing minimization of the bipartite graph corresponding to the discretized data matrix. The clusters found are subsequently quantified using a Figure-of-Merit (*FoM*) which is the average of the standard deviation (Std-Dev) of the column values.

The attribute subset selection problem has been defined in a number of ways by different researchers. In [22] four definitions have been listed that are conceptually different and cover a wide range and are as follows:

a. *Idealized:* Find the minimally sized feature subset that is necessary and sufficient to the target concept.
b. *Classical:* Select a subset of F features from a set of A features, $F < A$, such that the value of a criterion function is optimized over all subsets of size F.
c. *Improving Prediction accuracy:* The aim of feature selection is to choose a subset of features for improving prediction accuracy or decreasing the size of the structure without significantly decreasing prediction accuracy of the classifier built using only the selected features.
d. *Approximating original class distribution:* The goal of feature selection is to select a small subset such that the resulting class distribution, given only the values for the selected features, is as close as possible to the original class distribution given all feature values.

5.2.1 Why MASS is important?

The importance of solving MASS lies in the benefits achieved by minimizing the attributes, some of which are as under [33, 34, 40]:

a. A reduction in the cost of acquisition of the data.
b. Reduction in the amount of processed data resulting into speedup of data mining.

c. Reducing the dimensionality reduces the size of hypothesis space and allows algorithms to operate faster and more effectively.

d. Reduction in the number of attributes in the discovered patterns, enhancing ease of understanding for the analyst.

e. Most data mining algorithms can get confused by the presence of irrelevant attributes, thus removing those attributes is expected to lead to better quality results.

5.3 Review of Previous Work

Data is typically stored in a table or a database, also called as a data matrix. From data matrix similarity (or dissimilarity) matrices are created. Clustering of the similarity (or dissimilarity) matrix is called one-way clustering, which gives a *global view* of the data. Simultaneous clustering of the rows and columns of the data matrix is called two-way clustering, which gives a *local view* of the data. For one-way clustering choice of columns (or attributes) is critical, as this may actually hide, or do not completely display the clusters present. There are techniques based on supervised [52] and unsupervised classification [49] for attribute selection, in this chapter; a new technique is proposed for the MASS using biclustering, with biclustering achieved through crossing minimization.

Some good work on reviewing the feature subset selection methods can be found in [22], [34]. The survey by [22] covers feature selection methods starting from the early 1970's to 1997. A framework is suggested that helps in finding the unexplored combinations of generation procedures and evaluation functions. Thus 16 representative feature selection methods based on the framework are explored for their strengths and weaknesses regarding issues like attribute types, data noise, computational time complexity etc.

The work by [22] presents a benchmark comparison of six major attribute selection methods: Information Gain (IG), Relief (RLF), Principal Components (PC), Correlation –based Feature Selection (CFS), Consistency (CNS) and Wrapper (WRP), on 14 well known and large benchmark datasets (two containing several hundreds of features and the third over a thousand features) for supervised classification. The benchmark shows that in general, attribute selection is beneficial for improving the performance of common learning algorithms and suggests some general recommendations.

Attribute selection algorithms can generally be classified into three categories based on whether or not attribute selection is done independently of the learning algorithm used to construct the classifier: filter, wrapper and embedded approaches. They can also be classified into three categories according to the search strategy used: exhaustive search, heuristic search and randomized search [9].

5.3.1 Filter Methods

These methods act as a filter in the data mining process, to remove possible irrelevant attributes before the application of the data mining algorithms. The methods work independently of the data mining algorithm (induction algorithm) [43]. In such methods, the goodness of an attribute subset can be assessed with respect to the intrinsic properties of data. The statistics literature proposes many measures for evaluating the goodness of a candidate attribute subset. Examples of statistical measures used are information gain and correlation [9] etc.

5.3.2 Wrapper Methods [43]

In this method the attribute subset selection algorithm exists as a wrapper around the data mining algorithm and the method that evaluates the results. The attribute subset selection algorithm conducts a search for a good subset using the mining algorithm, which itself is a part of the function that evaluates an attribute subset. In case of classification problems, the evaluation of the discovered classification model is used as an estimate of the accuracy through cross validation. The essential characteristic of the wrapper method is the usage of same data mining algorithm and the evaluation method that will be subsequently used in the data mining process. No knowledge of the induction algorithms is necessary, except testing the resulting patterns on the validation sets. Many algorithms have been suggested in the statistics literature for finding a good subset of attributes under various assumptions. However, these assumptions do not hold for most of the data mining algorithms, hence heuristic search is used mostly.

5.3.3 MASS as a Search Problem

MASS can be considered as an optimization problem which involves searching the space of possible attribute subsets and identifying the ones that are optimal or nearly optimal with respect to a performance measure [9]. The search space consists of $O(2^m)$ possible solutions, here m is the number of attributes in the data matrix. Many search techniques have been used for solving the MASS intelligently, but exhaustively searching the entire space is prohibitive especially when the number of attributes is large [24]. For example, randomized, evolutionary, population based search techniques and Genetic Algorithms (GAs) have long been used in the MASS solving process. GAs needs crossover and mutation operators to make the evolution possible, whose optimal selection is a hard problem.

5.3.4 Embedded Method [40]

In this approach the learning algorithm prefers some attributes instead of the others, and in the final classification model possibly not all of the available attributes are present. For example, some induction algorithms like partitioning and separate–and-conquer methods implicitly select attributes for inclusion in a branch or rule by giving preference to those attributes that appear less relevant, consequently some of the attributes never get selected. On the other hand, some induction algorithms like Naïve-Bayes include all the presented attributes in the model when no attribute selection method is executed. Conceptually the filter and the wrapper methods are located one abstraction level above the embedded approach, because they perform an attribute selection for the final classifier, apart from the embedded selection done by the learning algorithm itself.

5.4 Model Formulation

Let the data matrix be denoted by S and its discretized version be denoted by S_B. Let the bipartite graph corresponding to S_B be denoted by G_B, such that one bipartition of G_B i.e. BOT (bottom) consists of vertices that correspond to the rows (or records) and the second bipartition i.e. TOP (top) to consist of vertices that correspond to the columns (or attributes). Fig 5.1(a) shows a discretized data matrix i.e. S_B consisting of 16 rows and 7 attributes. There will be edges between those vertices of G_B, for which there is a non-zero value in S_B. Fig 5.1(b) shows the G_B corresponding to S_B. Discretization is critical to the proposed solution, as it reduces the complexity of the problem i.e. converts a weighted clique into a binary i.e. 1-0 graph of lower density. Discretization is achieved by partitioning continuous variables or attributes into discrete values or categories, typically using column median or average. Working with S will force considering each and every edge of the un-discretized graph, hence a highly undesirable $\Omega(n^2)$ time complexity. Working with S_B reduces the time complexity to $O(K)$, where K is the number of 1's in S_B. Thus, the viable way out is the discretization of S, leading to S_B. Note that even algorithms with quadratic time complexities are unacceptable for most KDDM (Knowledge Discovery by Data Mining) applications according to Fayyad and Uthurusamy [28].

Example 5.1

Figure 5.1(a) shows a 16×7 discretized S_B (with n records and m attributes) and Fig 5.1(b) shows the corresponding G_B. Attributes of the data matrix correspond to the TOP bipartition of the bipartite graph. After crossing minimization Fig 5.1(c) is obtained and the corresponding optimally arranged $\mathbf{S}^*_B$ is shown in Fig 5.1(d). The subset of attributes of interest have been identified as (1, 6), (3, 5, 7) and (2, 4) in Fig 5.1(c). For ease of comprehension an example of non-overlapping attributes is presented, but the proposed technique works for overlapping attributes also.

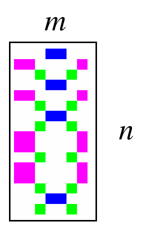


Figure 5.1(a) S_B

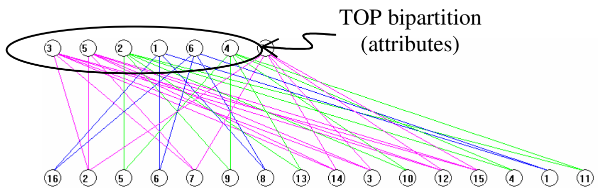


Figure 5.1(b): G_B without grouping of attributes (288 crossings)

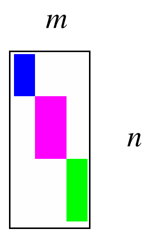


Figure 5.1(d) S^*_B

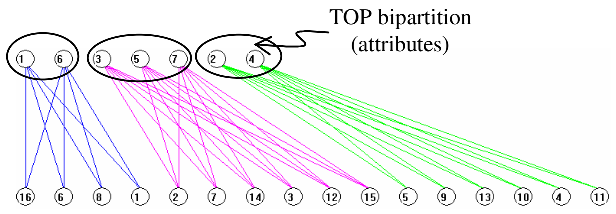


Figure 5.1(c): G^*_B with attributes grouped (66 crossings)

5.4.1 Crossing Minimization and MASS

One-way clustering is typically performed by creating a pair-wise similarity or dissimilarity matrix using measures such as Pearson's correlation. Assuming that the data matrix consists of n rows and m columns, creating the similarity matrix takes $O(n^2m)$ time. This is followed by the clustering algorithm, but the bottleneck is the time spent in creating the similarity matrix. Our proposed technique reduces the number of attributes (m) required for one-way clustering by spending $O(nm)$ time performing biclustering; prior to one-way clustering. The biclusters extracted are analyzed to identify the attributes, which should be used to perform clustering (Example 5.1).

5.5 Results Using Real Data

The aim of this section is to demonstrate the working of the proposed technique using two datasets i.e. the glass data set and the wine data set, both are publicly available [8].

In the first experiment the glass data set is used to demonstrate the concept of the proposed technique. The data set consist of 214 records, nine attributes (excluding the class attribute) with seven classes. In the said data set the records are already ordered by their class. A simple analysis would involve creating a pair wise similarity matrix of the given data set, and color coding it by assigning bright green color to the value of 1 and red color to the value of -1. Intermediate values are assigned shades of the corresponding color, with black color being assigned to a correlation of 0. The resulting color coded similarity matrix for the glass data set is shown in Fig 5.2(a), as can be seen that no clustering is visible.

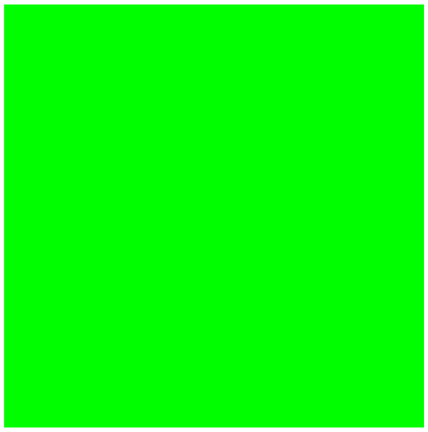


Fig 5.2(a): All nine attributes used

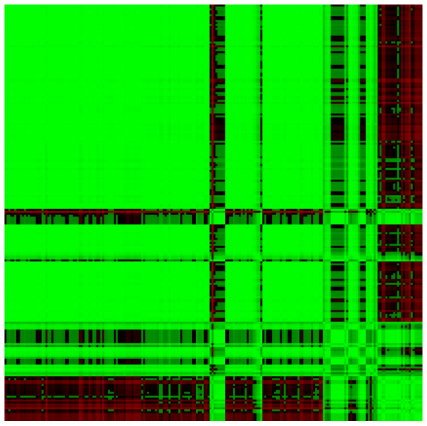


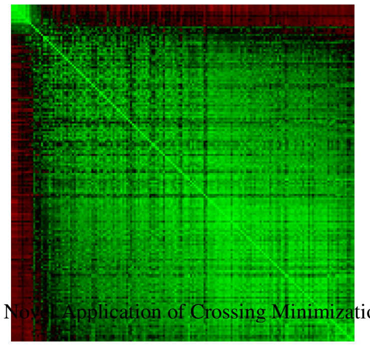
Fig 5.2(b): Attributes 3, 4, 8, 9 used

Figure 5.2: Glass dataset: Visualization with all attributes and subset of attributes

For selecting the attributes so as to enhance visualization, biclustering was performed on the glass data set. Based on the results of biclustering, the minimum attributes of significance were identified to be 3, 4, 8 and 9 i.e. Na, Mg, Ca and Ba. Using these attributes the similarity matrix was again generated, and the color coded matrix is shown in Figure 5.2(b). Observe the significantly enhanced visualization with lots of interesting relationships and clusters distinctly visible.

The wine data set used earlier in Chapter (3) is again used for a detailed experiment which will also demonstrate how biclustering is used for attribute selection and minimization. The data set consists of 178 rows, 13 attributes and three clusters. For the ease of understanding, a relatively small data set has been selected that can easily be displayed on a single screen of 600×800 pixel resolution. The 13 attributes of the dataset are (1) 'Alcohol', (2) 'Malic', (3) 'Ash', (4) 'Alcalinity', (5) 'Magnesium', (6) 'Phenols', (7) 'Flavanoids', (8) 'Nonflavanoids', (9) 'Proanthocyanins', (10) 'Color', (11) 'Hue', (12) 'Dilution', and (13) 'Proline'.

Fig 5.3(a) shows the corresponding color coded clustered similarity matrix, some clustering is visible, but visualization deteriorates when the real valued similarity matrix is discretized, as shown in Fig 5.3(b).



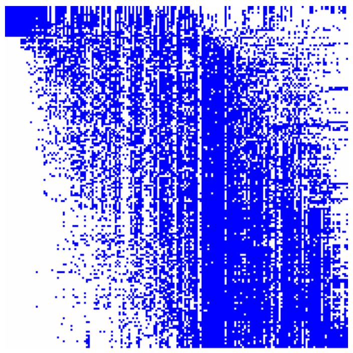


Fig 5.3(a): 178 × 178 clustered real valued similarity matrix using all 13 attributes

Fig 5.3(b): Binary valued version of Fig 5.3(a)

Figure 5.3: Wine dataset: One-way clustering

Firstly, the given data matrix is normalized and color coded it, such that bright green color is assigned to value of 1 and black to value of 0, intermediate values are assigned shades of green. The color coded un-clustered data matrix rotated by 90^o is shown in Fig 5.4(a). Secondly biclustering is performed using the discretized binary data matrix; the color coded biclustered data matrix is shown in Fig 5.4(b), while the discretized version of Fig 5.4(b) with biclusters extracted (red rectangles) is shown in Fig 5.4(c).

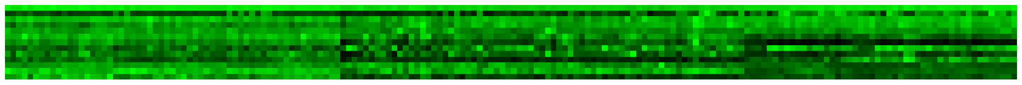


Fig 5.4(a): Color coded input data matrix of size 178 ×13 rotated by 90^o

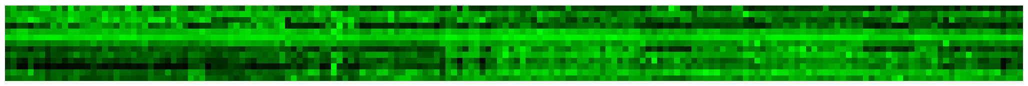


Fig 5.4(b): Color coded output biclustered data matrix

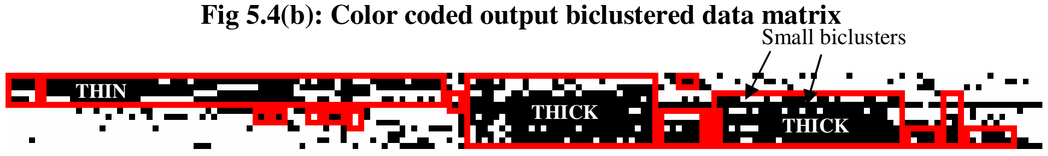


Fig 5.4(c): Binary biclustered data matrix with biclusters shown by red rectangles

Figure 5.4: Wine dataset: Two-way clustering for attribute selection

From Fig 5.4(c) three significant biclusters are identified titled “thin” of size 41×3 and “thick” of sizes 24×10 and 19×10 and some small biclusters that consist of three columns or attributes. The small biclusters share three columns or attributes with the “thick” biclusters, and no columns with the “thin” bicluster. Thus two sets of attributes are obtained through biclustering and subsequently used for one-way clustering i.e. set-1 consisting of attributes 2, 4, 8 and set-2 consisting of attributes 3, 5, 10.

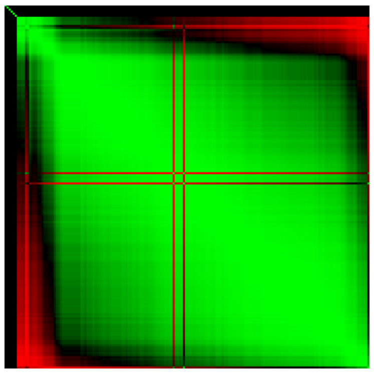


Figure 5.5(a): Input similarity matrix of size 178 × 178 using set-1

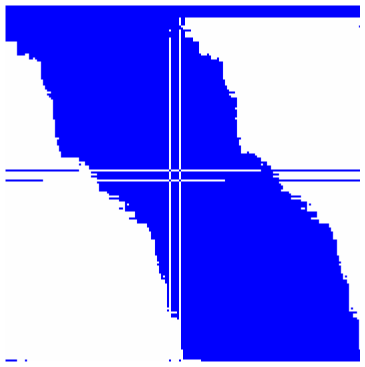


Figure 5.5(b): Binary version of Fig 5.5(a)

Figure 5.5: Wine dataset: One-way clustering with subset of attributes

Using set-1 one-way clustering of the wine dataset is performed, and the results are shown in Fig 5.5(a). Observe the enhanced visualization with negative correlations also visible and very well defined pixelized binary visualization as shown in Fig 5.5(b).

5.5.1 Quantitative Analysis of Results

The aim of this section is to quantitatively analysis the one-way clustering results with all the attributes used and with fewer attributes used. For the purpose of quantification, a simple yet effective Figure-of-Merit or *FoM* is defined. The *FoM* being the average of the standard deviation (Std_Dev) of the columns values of records or rows included in each cluster. Clusters with rows of highly similar values will have a low *FoM* and is desirable.

Quantitative analysis of the results is performed by noting the density and the *FoM* of each cluster. In the context of the binary data matrix, clusters with high density apparently seem to correspond to good clustering, but this may not always be true. The reason being, it hides the actual column values which may all be above the median value of the entire column of the input data matrix, but all different within the part of the column included in the corresponding cluster.

Table 5.1 shows the results of using all 13 attributes of the wine data set with high values of *FoM*. Table 5.2 shows the results of using set-1 and set-2 with attributes identified using biclustering through crossing minimization. Observe the significant reduction (or improvement) in the *FoM*.

All 13 attributes	**Cluster size**	**Density**	**FoM**
	178 × 178 (input)	0.50	26.17
	16 × 16	1.00	17.93
	40 × 40	0.90	17.6
	36 × 36	0.83	15.4

Table 5.1: Results of using all 13 attributes (wine dataset)

Three attributes of set-1 i.e. 2, 4 and 8	**Cluster size**	**Density**	**FoM**
	82 × 82	0.79	1.59
	89 × 89	0.87	1.42
Three attributes of set-2 i.e. 3, 5 and 10	81 × 81	0.76	4.72
	27 × 27	0.85	4.28
	44 × 44	0.93	3.82
	16 × 16	0.90	3.92

Table 5.2: Results of using subset of attributes i.e. set-1 and set-2 (wine data set)

A trivial way of improving *FoM* is to use attributes with low Std_Dev. This could be done by calculating the Std_Dev of each attribute in *O(m)* time and then sorting them based on their Std_Dev in *O(m log m)* time. Subsequently the attributes with low Std_Dev are selected to perform one-way clustering. This technique does not take into account any relationships between the attributes (as the proposed technique does) and attributes are treated as independent of each other. Based on this trivial approach the top three attributes in the wine dataset with ascending Std_Dev are 8, 11 and 3. Note the contradiction as attribute 3 belongs to set-1 while attribute 8 belongs to the set-2. Clustering using these attributes resulted in numerous negative correlations or red colored pixels visible in the clustered similarity matrix, as shown in Fig 5.6. The paradox is that while trying to naively improve the *FoM*, the trivial approach ends up using those attributes which don't collectively support clustering.

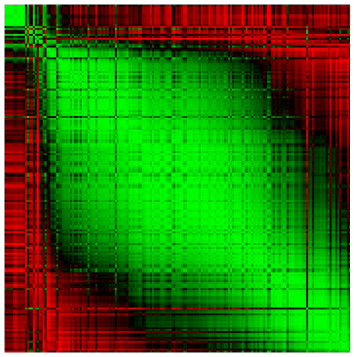


Figure 5.6: Input similarity matrix of size 178×178 using attributes 3, 8, and 11

5.6 Conclusions

The MASS problem has generated a lot of interest, because of the increase in the number of attributes in the datasets. It is observed that using the proposed technique not only visually better results are obtained, but the quality of those results was also better quantitatively. It is also observed that using a naïve approach of picking attributes with high similarity may actually deteriorate the cluster quality. The proposed technique takes into account the relationships between records and attributes, therefore, gives better results. In this thesis biclustering is used as a wrapper for one-way clustering.

Chapter 6

Novel Application of Crossing Minimization: Bandwidth Minimization (BWM)

In Chapter 3 and Chapter 5 the application of the crossing minimization paradigm was presented to address two proven NP-Complete problems i.e. (i) Biclustering (ii) and the MASS problem. In this chapter, the results obtained by using the crossing minimization paradigm to heuristically solve another hard problem i.e. the BWM problem using public domain real data will be discussed. Finally, the results of solving a real-life Agriculture problem through one-way clustering application of the proposed technique, using real Agro-Met data will also be discussed.

6.1 Introduction

Consider $n \times n$ square symmetric matrix A = $\{a_{ij}\}$ the BWM problem is to find a permutation of rows and columns of A so as to bring all the non-zero elements of A to reside in a band that is as close as possible to the main diagonal, that is to $Minimize\{max\{|i - j| : a_{ij} \neq 0\}\}$. A sparse matrix has a very few non zeros that can be processed efficiently by exploiting the structure; typically, sparse matrices have $O(n)$ non zeros with a bounded number of non zeros in each column or row. In scientific computing, the order of the elements in a sparse matrix often affects the performance of numerical algorithms. Sparse matrices can be factored with less work and storage than the dense matrices, provided suitable algorithms are used. Algorithms for factorization use more complicated data structures and are relatively difficult to program; so they may also use heuristics to reduce time and storage complexity.

There can be two ways of evaluating a new technique i.e. by comparing it with the work of other researchers, and then applying the technique on simulated data as well as real data and analyzing the results. This approach will be followed in this chapter; therefore, first the working of the proposed technique will be discussed for the BWM problem using simulated, as well as publicly available Harwell-Boeing Sparse Matrix real data.

6.2 Background

The Matrix Bandwidth Minimization (BWM) problem originates from the 1950's when structural engineers first analyzed steel frameworks using computers. Various algorithms exist for reordering the matrices to achieve certain properties that can yield better performance [30]. The bandwidth minimization problem was proved to be NP-complete by Papadimitriou [68]. Garey et al. [31] has shown that BWM is NP-complete even if the input graph is a tree whose maximum vertex degree is 3.

A dot plot is a technique for displaying relationships between elements in a data set. In the context of data mining, dot plots can correspond to discretized similarity (or dissimilarity) matrices used in one-way clustering i.e. clustering based on either rows or columns. Dot plots are commonly used to visually compare genes using their expression data and inspect the structure of sparse matrices in scientific applications [51].

The matrices under consideration for bandwidth minimization can be visualized using a dot plot, as shown in Figure 6.1(a), which shows a sparse discretized similarity matrix for Wisconsin cancer data set available at http://www.ics.uci.edu/~mlearn/. As can be seen in Figure 6.1(b) the enhancement of relationships in the data set also results in reduction of the bandwidth of the similarity matrix.

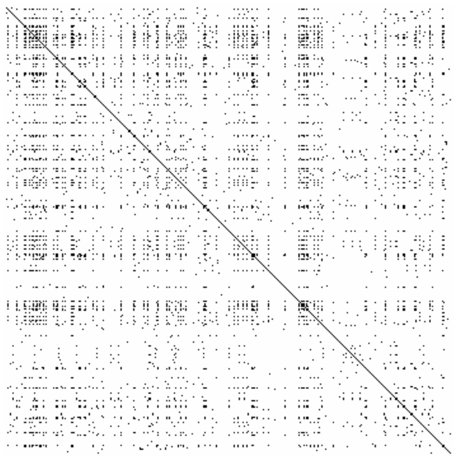


Fig 6.1(a): Input similarity matrix

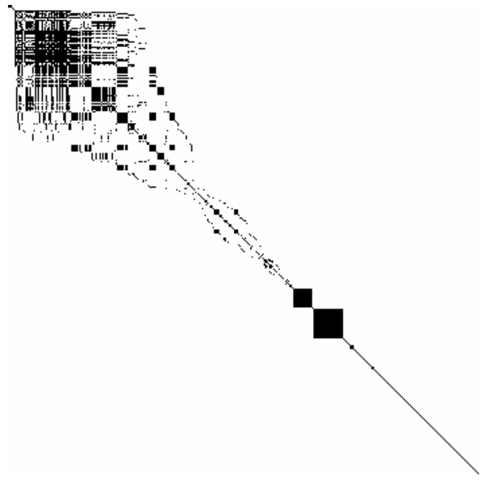


Fig 6.1(b): Output with bandwidth reduced

Figure 6.1: Bandwidth reduction of sparse matrix

The metrics used for quantification of the results of Fig 6.1 are shown in Table 6.1 which are (i) *Avg BW*: Average bandwidth (ii) *Med BW*: Median bandwidth (iii) *Max BW*: Maximum bandwidth and (iv) *SD BW*: Standard deviation of bandwidths.

	Avg BW	**Med BW**	**Max BW**	**SD BW**	**Crossings**
Before	416.92	542	688	273.83	92,908,711
After	74.35	40	240	81.11	47,569,404
% reduction	82.16	92.61	65.11	70.37	48.79

Table 6.1: Bandwidth reduction due to crossing minimization (cancer dataset)

Like the bandwidth minimization problem, the two-way crossing minimization problem has been studied for almost two decades and is also NP-Complete [31]. In this chapter, using extensive computational experiments, followed by application to real data the performance of four crossing minimization heuristics, four Meta CM heuristics, a recent clustering technique and two classical BWM heuristics is compared. The crossing minimization heuristics considered in this chapter are the Median Heuristic (MH) [27], BaryCenter Heuristic (BCH) [77], MaxSort Heuristic (MS) [1] and MS* a derivative of MS. The Meta-Heuristics considered are MS followed by MH and MS followed by BCH i.e. MS+MH and MS+BCH, respectively. Similarly MS* used instead of MS in the Meta heuristics. The clustering technique considered is called Cluster Affinity Search Technique (CAST) [11]. The traditional bandwidth minimization heuristics used are Reverse Cuthill-McKee (RCM) [17] and King's heuristic [30].

6.2.1 Why BWM is Important?

The bandwidth minimization problem has been found to be relevant to a wide range of applications. Matrices are the main data structures used in large-scale scientific and engineering applications for representing linear systems of equations. Linear systems typically have thousands of variables, but each individual variable is usually related to only a few other variables. This leads to equations where most of the coefficients are zero. Rather than allocating space for every element in a matrix i.e. $O(n^2)$ space, where n is the number of rows (or columns), sparse matrix data structures exploit this feature and attempt to minimize the amount of memory used, by only allocating memory for the non-zero elements and elements that are used directly by an algorithm.

Another example is Gaussian elimination. Gaussian elimination can be performed in $O(nb^2)$ time on matrices of bandwidth b, which is much faster than the normal $O(n^3)$ algorithm if $b << n$. Some other applications of bandwidth minimization are hierarchical graph drawing with bends [60] with applications in VLSI design, finite element methods for approximating solutions of partial differential equations, chemical kinetics, numerical geophysics, electro magnetics etc. Link analysis of the web is also a BWM problem, as the graph representing the relationship (as edges) between web pages (vertex) and keywords (vertex) is very sparse. Leading search engines indexes tens of billions of pages, but there are very few pages with hundreds or even dozens of hyperlinks.

6.3 Review of Previous Work

The aim of this section is to compare the proposed technique with related and similar techniques proposed by other researchers. A number of approximation methods have been proposed to solve the BWM problem since the 1960's. In 1969, the well-known CM algorithm of Cuthill and McKee [17] appeared which used Breadth-First Search to construct a level structure of the graph. By labeling the vertex in the graph according to a level structure, good results were achieved in a short time. George and Liu in [30] provide a comprehensive survey of algorithms for operating on symmetric positive definite matrices, which are a main class of matrices used in practice.

A hierarchical graph is generally a DAG (Directed Acyclic Graph) that displays hierarchy. Each edge in such a graph does not display its arrow since the parent-child relationship implies the direction. The use of the BaryCenter Heuristic (BCH) for the maximum edge length minimization problem was discussed by May and Mennecke [60] for planar straight line drawings of hierarchical graphs with bends. In the context of graph drawing, the proposed solution is not limited to hierarchical graphs, non planar graphs and drawings without bends.

Laguna and Martí [53] proposed a path re-linking procedure for arc crossing minimization in the context of Greedy Randomized Adaptive Search Procedure (GRASP). Subsequently Piñana et al. [25] proposed a two phased path re-linking solution using GRASP for the BWM problem. However, crossing minimization was not used for BWM. Marti et al. [58] used Tabu Search for the problem of BWM. They used a candidate list strategy to accelerate the selection of moves in the neighborhood of the current solution. Extensive experimentation showed that their Tabu Search outperforms the best-known contemporary algorithms in terms of solution quality in reasonable time.

Muller in [49] a graph theoretic probabilistic model is used to perform one-way clustering using the so called Cluster Affinity Search Technique (CAST), but BWM was not considered. Muller in [49] used BWM techniques for one-way clustering that included a comparison of RCM, Kings heuristic and a Modified Minimum Degree technique. Our work is the logical converse i.e. using crossing minimization one-way clustering techniques for BWM.

6.4 Model Formulation

The aim of this section is to briefly discuss the graph drawing model which forms the core of the solution presented. Let the binary sparse matrix S_B correspond to the matrix representation of a bipartite graph, such that the vertices correspond to the rows of the data matrix, and for each edge of the bipartite graph there is a corresponding 1 in the matrix. Let the bipartite graph (bi-graph) corresponding to S_B be denoted by G_B, most likely G_B will not have high level of diagonalization i.e. low bandwidth. To facilitate defining the model, it is first assumed that S_B consists of perfect diagonalization. Such a similarity matrix is denoted by S^*_B and the corresponding bi-graph by $G^*_B(V_0, V_1, E)$. G^*_B can be assumed to be a union of $K_{i,j}$ i.e. bipartite graph cliques, and V_0, V_1 is the bipartition of vertices such that $V_0 \cap V_1 = \varnothing$. E is the edge set such that $e = |E|$. As symmetric matrix is being considered, therefore, $i = j$ and $n = |V_1| = |V_0|$ and density of G_B denoted by σ i.e. $\sigma(G_B) = e/n^2$.

Let a bi-graph drawing (or layout) of G^*_B be obtained by placing the vertices of V_0 and V_1 on distinct locations on two horizontal lines $y = 1$ i.e. TOP (top bipartition) and $y = 0$ i.e. BOT (bottom bipartition) in the XY-plane, respectively. The vertices of every clique are located on consecutive and identical x-coordinates for TOP and BOT. Now drawing each edge with one straight-line segment which connects the points on $y = 0$ and $y = 1$ where the end vertices of the edges were placed. This will result in a bi-graph drawing, in which only those edges intersect, that belong to the same clique. Let φ^* be the corresponding bi-graph drawing and the order of vertices in the bipartitions V_0 and V_1 is denoted by π^*_0 and π^*_1, respectively. Note that for $K_{i,j}$ (as a convention) it is assumed that the vertices in bi-partition *i* will be placed in TOP and the vertices in *j* will be placed in BOT. To get a non-ideal input, the vertices in each bipartition of G_B are now randomly permuted. Let the current order of vertices in the bipartitions V_0 and V_1 of G_B be denoted by π_0 and π_1, respectively. Let $\Phi(G)$ denote the set of all possible bi-graph drawings of the bi-graph G_B. Now the BWM problem becomes: given a bi-graph G_B (V_0, V_1, E) find φ^* among Φ (G) with permutation of vertices π^*_0 and π^*_1.

Example 6.1

In this example the use of crossing minimization for bandwidth minimization will be demonstrated using a very small matrix. Figure 6.2(a) shows the input matrix, the corresponding bipartite graph drawing G_B is shown in Figure 6.2(b) with 62 crossings. When MaxSort is used on Fig 6.2(b) the bipartite graph drawing obtained is shown in Fig 6.2(c) with 26 crossings and the corresponding matrix is shown in Fig 6.2(d).

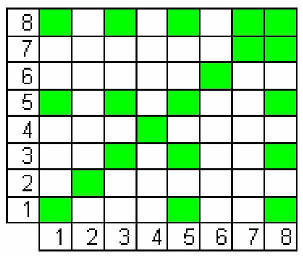


Fig 6.2(a): Input matrix

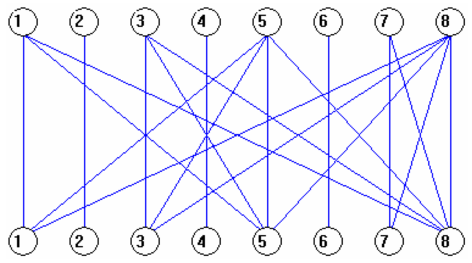


Fig 6.2(b) : G_B corresponding to Fig 6.2(a)

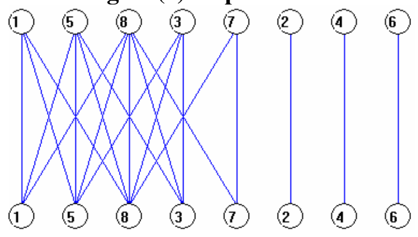


Fig 6.2(c) : Crossing Minimized G_B

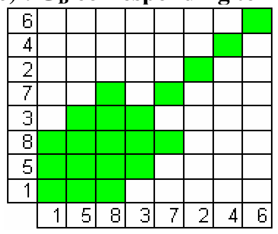


Fig 6.2(d): Output matrix corresponding to Fig 6.2(c)

Figure 6.2 – Bipartite graphs with their corresponding matrix representation

By comparing Figure 6.2(a) with Figure 6.2(d) a significant reduction in bandwidth is observed, which is summarized in Table 6.2.

	Avg BW	**Med BW**	**Max BW**	**SD BW**	**Crossings**
Before	3	2.5	7	3.19	62
After	1.25	2	2	1.03	25
% reduction	58.33	20	71.4	66.78	59.67

Table 6.2: Bandwidth minimization due to crossing minimization (simulated data)

6.4.1 Crossing Minimization and BWM

For a G_B numbering the vertices in the TOP bipartition and subsequently sorting vertices in the BOT bipartition, and then alternatively repeating this process for the two bipartitions, has the effect of two forced orderings that are alternated until an equilibrium state is reached. An equilibrium state corresponds to the graph drawing that does not change with further application of the CMH meaning vertices with high connectivity have same or similar locations on $y = 0$ and $y = 1$ i.e. forming a band close to the diagonal in the corresponding matrix. The attractiveness of the proposed technique and the reason why it is extremely fast is that it is not a merit function minimization per se or a greedy technique. The problem with evaluating merit functions is that they take additional time and iterations, resulting in a slow solution. For example, a naïve approach to establish how many edge crossings there are would require $O(e^2)$ time i.e. checking pair-wise edge crossings (where e is the number of edges in the bipartite graph).

6.5 Comparative Performance Analysis Using Simulated & Real Data

In this section a comparison is made of the performance of 10 heuristics and Meta heuristics through 750 experiments on simulated data, and using Harwell-Boeing real data.

6.5.1 Results Using Simulated Data

A set of experiments were performed on a bi-graph consisting of a union of four $K_{25,25}$ (with a bounding size 100x100). The density of the resulting graph i.e. $\sigma(\mathbf{G_B})$ was varied from 0.625% to 10% (i.e. 0.625%, 1.25%, 2.5%, 5% and 10%) and for each density 15 experiments were performed using randomly permuted matrices (or corresponding bipartite graph partitions) which were used as input for each of the 10 heuristics and Meta heuristic (BCH, MH, MS, MSS, MS+BCH, MS+MH, MSS+BCH, MSS+MH, King and RCM) for a total of 750 experiments. For each experiment the bandwidth of the resulting matrix was noted and used for subsequent analysis.

Table 6.3 shows the Standard Error (SE) for each combination of graph densities and heuristics and Meta heuristics calculated using the equation SE = $Stdev/\sqrt{n}$ where n is the sample size (n=15) and *Stdev* is the standard deviation. From Table 6.3 it can be seen that for sparse matrices the classical crossing minimization BCH gives inferior results similar to the classical crossing minimization MH. However, as the density of the matrix (or corresponding bipartite graph) increases, the performance of BCH becomes comparable to other heuristics. As can also be observed that the results for MH are an order of magnitude inferior as compared to other heuristics, and displaying them in the same scale skews the graph, therefore, the same have not been shown in Fig 6.3 (a, b).

	0.63%	1.25%	2.50%	5%	10%
BCH	0.341	1.318	2.420	1.363	0.206
MH	3.839	2.388	1.938	1.531	4.315
MS	0.131	0.215	0.476	0.350	0.153
MS+BCH	0.000	0.255	0.388	0.384	0.239
MS+MH	0.091	0.182	0.659	0.598	0.165
MSS	0.000	0.192	0.517	0.428	0.118
MSS+BCH	0.000	0.291	0.388	0.450	0.131
MSS+MH	0.000	0.206	0.636	0.736	0.165
King	0.182	0.280	0.476	0.411	0.182
RCM	0.182	0.368	0.588	0.363	0.290

Table 6.3: Comparison of heuristics and Meta heuristics on the basis of SE of Avg bandwidths

Fig 6.3(a) shows the values of Avg BW (Avg BW ± 1.96SE, where 1.96 represents the upper and lower 95% confidence limits assuming an underlying normal distribution of the data) plotted against different heuristics and Meta heuristics for each value of graph density. Except for the BCH all other heuristics can be seen to exhibit more or less similar BW error performance for all densities at the 95% confidence level. The BCH can also be seen to exhibit the worst performance for density values ranging from of 1.25% to 5% (at the 95% confidence level).

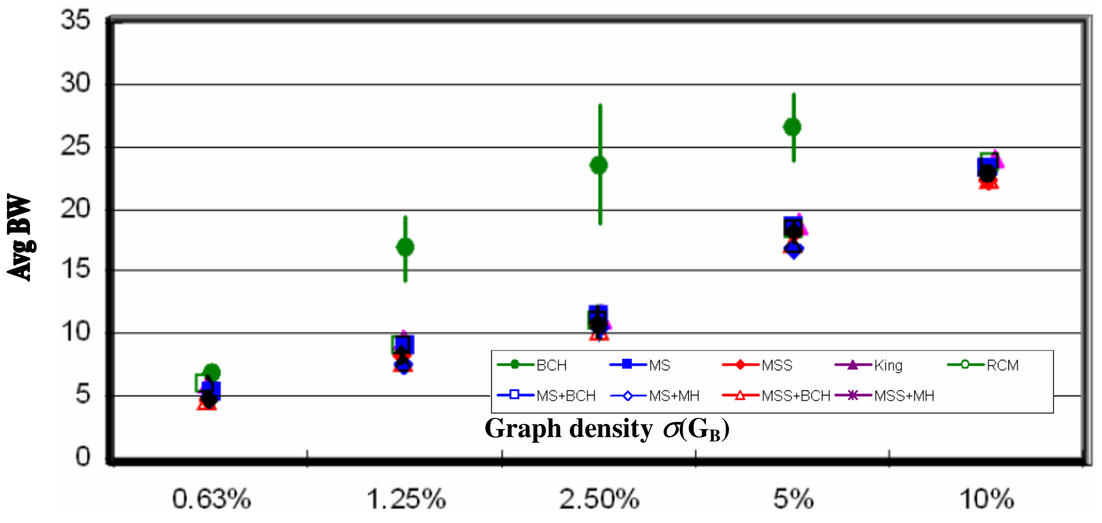


Figure 6.3(a): Comparison of all heuristics and Meta heuristics for each density (Avg BW ± 1.96SE)

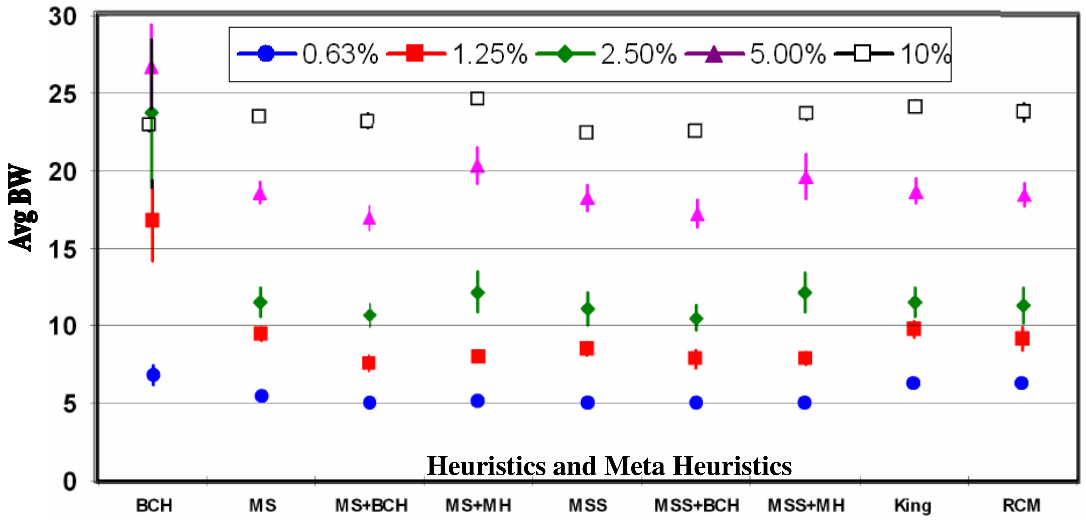


Figure 6.3(b): Comparison of all heuristics and Meta heuristics for all densities (Avg BW ± 1.96SE)

Fig 6.3(b) shows the effect of density on Avg BW error for each heuristic and Meta heuristic. For the lowest density value (0.63%) the proposed heuristics and Meta heuristics (MS, MSS and their Meta variants) have produced comparatively better results compared to the conventional heuristics (BCH, King and RCM). However, at the highest density value of 10% all heuristics and Meta heuristics can be seen to produce similar results.

6.5.2 Results Using Real Data

For comparing the 10 combinations of heuristics and Meta heuristics, standard test matrices data is used from the Harwell-Boeing Sparse Matrix Collection at http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/. The reason being these matrices represent a large spectrum of scientific and engineering applications from problems in linear systems, least squares, and eigenvalue calculations. Furthermore, these matrices have been used as test sets in recent experiments by other researchers, as they are representative of real-world data. For demonstration purposes the matrix LSHP1009 is used which is from a finite-element model problem and is a symmetric square 1009×1009 matrix with density 0.67%.

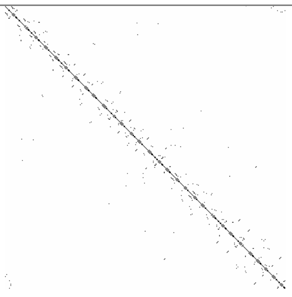
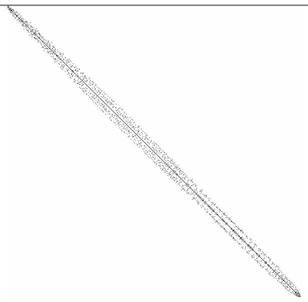
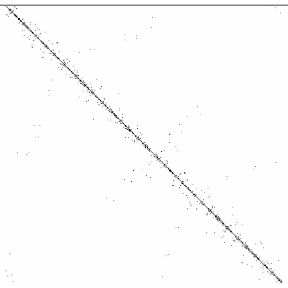
		
Input	**BCH**	**MH**

Figure 6.4(a): Dot plots of input and outputs of classical CMH for LSHP1009

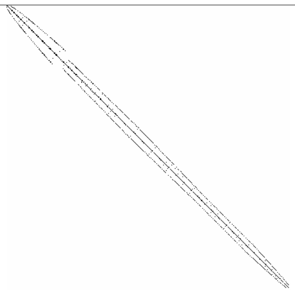
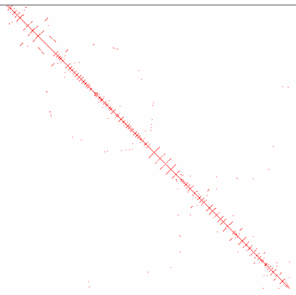
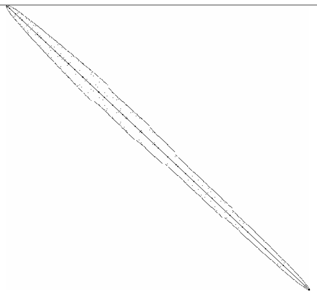
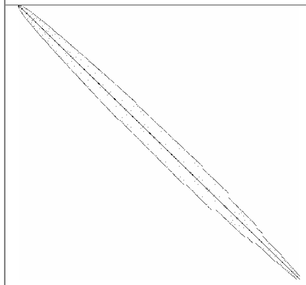
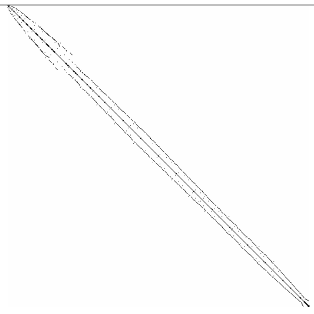
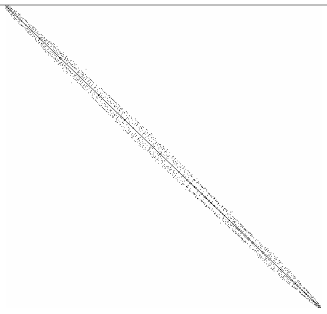
		
MS	**CAST**	**RCM**
		
King's	**MS+MH**	**MS+BCH**

Figure 6.4(b): Dot plots of proposed heuristics and Meta heuristics and classical BWM heuristics for LSHP1009

	Avg BW	Med BW	Max BW	SD BW	Time (CPU sec)
Input	106.99	43	994	193.25	
MH	93.83	30	998	186.66	0.08
BCH	52.12	54	108	15.16	1.763
MS	56.2	56	91	13.49	0.08
MS*	56.17	56	91	13.51	0.09
MS+MH	56.05	56	92	13.75	0.08+0.05 = 0.13
MS+BCH	49.97	51	102	14.59	0.08+0.871 = 0.95
MS*+MH	56.05	56	92	13.74	0.09+0.06 = 0.15
MS*+BCH	50.01	51	100	14.53	0.09+0.871 = 0.96
King	61.39	67	93	18.87	0.63
RCM	52.75	57	67	12.89	0.599
CAST	70.45	35	742	116.55	0.015

Table 6.4: Comparison of heuristics and Meta heuristics for real sparse matrix LSHP1009

Table 6.4 shows the results of using different heuristics and Meta heuristics for LSHP1009. The column on the extreme right gives the overall CPU time in seconds.

From Table 6.4 and Fig 6.4 MS is a clear winner over BCH and MH for Max BW, SD BW and CPU time. Furthermore, MS+BCH outperform both the King's and RCM traditional BWM heuristics for Avg BW and Med BW. The CPU time taken by MS and BCH independently is 0.08 and 1.763 seconds, respectively and adds up to 1.84 seconds. However, the Meta heuristics MS+BCH took about 50% overall less time and gave better results. Observe that the quality of the results of RCM depends on the starting vertex, while the quality of results and overall time taken by BCH depends on the initial permutation of vertices. Using MS before running BCH i.e. Meta heuristic generates a better initial permutation as compared to a random permutation, resulting in better output. Although finding the minimum value in MS and performing addition to get average in BCH takes *O(n)* time, but addition with carry propagation etc. is slower than comparisons done by MS to get the minimum. Furthermore, BCH is a pure crossing minimization heuristics, while MS tends to reduce crossings and is consequently much faster than BCH.

6.6 One-Way Clustering: Anomaly of Pesticide Usage

The aim of this section is to discuss the results of performing one-way clustering using pest scouting data obtained from the Agriculture Department, the Government of Pakistan. The objective of data mining is an attempt to provide a possible explanation about an apparent anomaly between pesticide consumption and cotton yield, as shown in Fig 6.5.

Pesticides are used as a means for increasing yield by controlling pest populations, thus a positive correlation is believed to exist between yield and pesticide consumption. However, existence of an undesirable, sometime even negative correlation between pesticide consumption and yield has been observed in Pakistan [46]. Figure 6.5 shows periods of marked decrease in yield while the pesticide consumption is on the rise, thus creating a complex situation and an apparent anomaly.

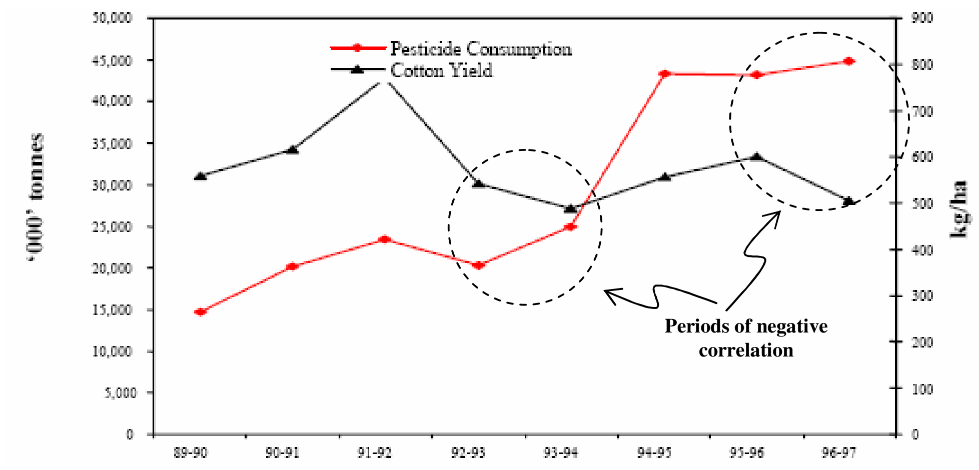


Figure 6.5: Pesticide consumption and cotton production [46]

Excessive use of pesticides is harmful in multiple ways. Farmers have to bear the additional cost of buying and applying pesticides, while increased pesticide usage develops immunity in pests, thus making them more damaging to the crops. Excessive usage of pesticides is also harmful for the environment and hazardous to humans, such as those who spray the pesticides, and the women cotton pickers.

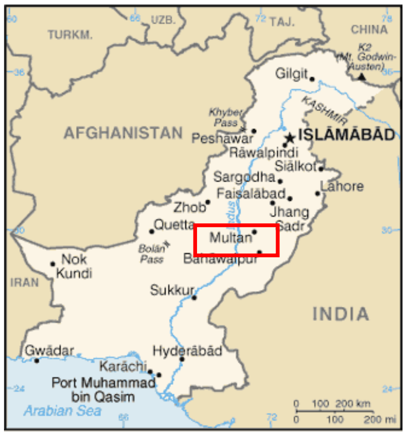
Reasons for the apparent anomaly of pesticide consumption and yield could be many. Because of the size and complexity of the data sets, automatically discovering such reasons by clustering seems to be the only viable way.

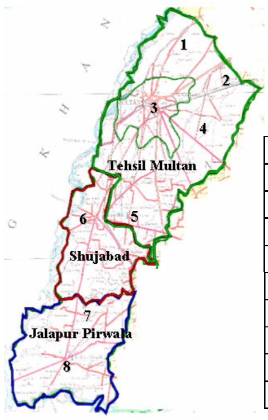
6.6.1 Problem Background

The aim of this section is to give a brief background about cotton grown in Pakistan, followed by a short introduction of entomology terminology used throughout the chapter. Pakistan is the 4th largest cotton supplier of the world. Almost 70% of world cotton is produced in China (Mainland), India, Pakistan, USA and Uzbekistan [12]. As textile exports comprise more than 60% of Pakistan's total exports, the success or failure of cotton crop has a direct bearing on the economy. Cotton production is the inherent comparative advantage of the textile sector of Pakistan [75]; with total textile industry exports amounting to US$ 7 billion and 68% share in export earnings.

Punjab is the breadbasket of Pakistan, and is administratively divided into eight divisions, including Multan division. Multan division is further divided into six districts, including district Multan. District Multan has three *Tehsils*. Within each *Tehsil* are central points or *Markaz*. This work is centered around district Multan (Figure 6.6(b)).

The area under study is shown in Figure 6.6.





Key	Markaz
1	Bosan
2	Qadirpurran
3	Multan
4	Makhdum Rashid
5	Mumtazabad
6	Shujabad
7	Hafizwala
8	Jalalpur Pirwala
9	Qasba Marral

Figure 6.6(a): Map of Pakistan (www.cia.gov)

Figure 6.6 (b): Area under study District Multan

Figure 6.6: Area under study for pesticide usage

In the context of this section, a pest is an insect that eats or damages the crop. Since cotton crop is being considered, so some of the pests considered are Jassid, Thrips, SBW (Spotted Boll Worm) etc. A predator is an insect that destroys the pests. Some of the cotton pest predators are ladybug beetles, spiders, ants, Assassin Bug etc. A sample of cotton virus, pest and predator are shown in Figure 6.7. Other than pests, the cotton crop is also effected by viruses, the predominant one being CLCV (Cotton Leaf Curl Virus). In this chapter, **field** would mean the cultivated land, with certain area and ownership.

Virus: Cotton Leaf Curl Virus [86]	**Pest:** Boll Worm [7]	**Predator:** Assassin Bug [7]

Figure 6.7: Virus, Pest and Predator present in the cotton fields

ETL: Economic Threshold Level in agriculture extension is that pest population beyond which the benefit of spraying a pesticide outweighs its cost. It is highly infeasible and expensive to eradicate all pests, therefore, pest control measure are employed, when pest populations cross a certain threshold. Thus the objective is to control pest population, not even minimize them. This control threshold varies from pest to pest, and from crop to crop and from region to region. Figure 6.8 shows the ETL by a dotted line, and undesired pest populations by humps above the said line. For "humps" below the dotted line, it is not feasible to spray.

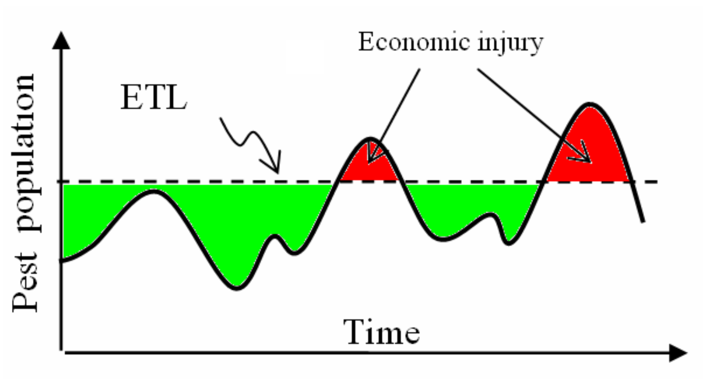


Figure 6.8: Agriculture Economic Threshold Level (ETL) and time.

6.6.2 Data Considered

For 2001 the available pest scouting data covered spray dates from 01 Jun. 2001 to 29 Oct. 2001, there were 948 spray records, using SQL Structured Query Language (SQL) with GROUP BY clause the records grouped into 94 groups of unique dates along with spray frequency for each date. For 2002 the scouting data available covered spray dates from 14 Jul. 2002 to 12 Oct. 2002. There were 1,014 spray records, using SQL the records grouped into 74 groups of unique dates along with spray frequency. For the remaining records there were no sprays. Using the spray dates as reference, T_{min} (minimum temperature of the day), T_{max} (maximum temperature of the day), % Humidity along with spray frequency were used to compute pair wise correlation similarity matrices for both years. The matrices were subsequently symmetrically discretized using the median value of each column. The discretized similarity matrices for both years but not in any particular order of rows (or columns) are shown in Figures-6.9(a) and 6.10(a).

6.6.3 Clustering of Agro-Met Data

Figure 6.9(a) shows the input similarity matrix for year 2001. The biclustering technique using the crossing minimization paradigm was modified to perform one-way clustering, resulting in Figure 6.9(b) which shows two clusters i.e. C1 and C2.

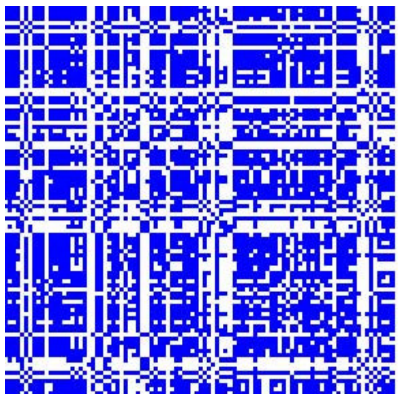


Figure 6.9(a): Input for 2001 (94 groups)

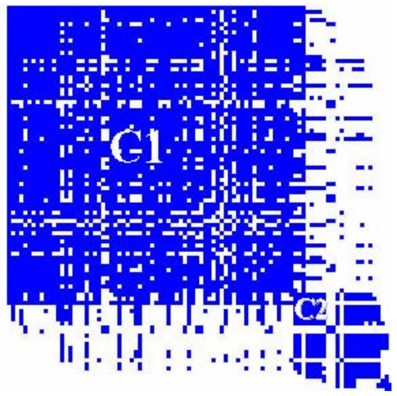


Figure 6.9(b): Clustered Output for 2001

After extracting clusters C1 and C2 for 2001, detailed scrutiny of the smaller cluster i.e. C2 turns out to be roughly the first unique 30 dates of records, for this date range the average sprays per day throughout the district Multan are 5, while for the larger cluster i.e. C1 average sprays per day are 13, which is as per the prevalent practice. As shown in Figure 6.10(b) no significant clustering occurred for year 2002.

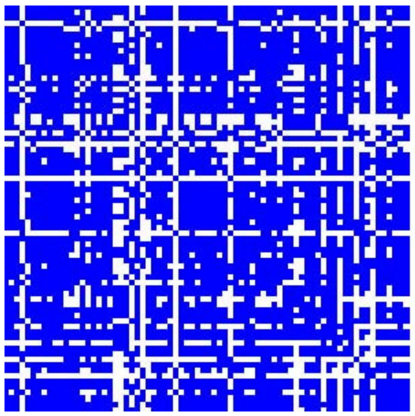


Figure 6.10(a): Input for 2002 (74 groups)

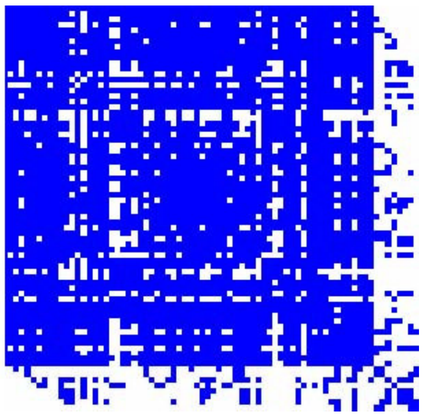


Figure 6.10(b): Clustered Output for 2002

6.6.4 Quantitative Analysis of Results

Pesticides are either sprayed in the early morning or by the end of the day i.e. avoiding high temperatures to prevent breakdown and evaporation; hence spraying at low temperatures are recommended i.e. negative correlation between temperature and frequency of spray. Similarly, sprays are advised in high humidity to prevent evaporation i.e. positive correlation between frequency of spray and % humidity.

Analysis of detailed data corresponding to C2 as shown in Figure 6.9(b) revealed **negative** correlation with humidity (i.e. -0.17), and **positive** correlation with minimum temperature (i.e. 0.32) i.e. opposite to the recommendations. The Pakistan Meteorological department records T_{min} and T_{max} for each day i.e. the minimum and maximum temperature of the day, respectively. Pesticide spraying is recommended to be done either before sunrise or around sunset, thus T_{min} is important. For a hot day, the T_{max} is high, so is T_{min}, thus positive correlation of sprays with T_{min} indicates that more sprays were done on relatively hotter days when T_{min} was relatively higher.

For the other large cluster i.e. C1 the correlation between frequency of sprays was found to be zero against both met elements i.e. humidity and minimum temperature T_{min}. Analysis of detailed data corresponding to Figure 6.10(b) i.e. year 2002 showed no distinct clustering, a desirable positive correlation with humidity i.e. 0.24 but an undesirable positive correlation with T_{min} i.e. 0.064. Again this does not guarantees maximum efficacy of pesticide, yet it is better than 2001.

After individualization of the cultivated fields the total cotton area scouted was established for 2001 and 2002. Based on this area, the average pesticide used per acre for the entire cotton season was calculated to be $\underline{23.96}$ ml/acre and $\underline{22.46}$ ml/acre for year 2001 and 2002, respectively. Thus in 2001 around 6% more pesticide was used as compared to 2002.

On further querying the data for specific pesticides used in 2001 and 2002 during roughly the same time period covered in C2, it was discovered that in 2001 the pesticide *Cypermethrin* was used during the early season at three times the dose as compared to the dosage used in 2002. *Cypermethrin* was also the top pesticide of choice, but during 2002 it was the 5th pesticide of choice. *Cypermethrin* is a systemic pesticide i.e. absorbed by foliage and translocated throughout the plant, and is recommended against sucking and chewing cotton pests i.e. bollworms. There are about a dozen brands of pesticides based on *Cypermethrin* in the market, some more focused towards bollworms (such as *Ripcord*), while others covering both sucking pests and bollworms (such as *Arrivo*). Unfortunately the scouting data did not specified which brand of *Cypermethrin* was used. However, as per the pesticides data available at www.nationalpak.com collected from the Central Cotton Research Institute, *Cypermethrin* is mainly recommended to control the population of the bollworm complex.

The next obvious, and interesting question is, "the effect of these malpractices on pest populations" The findings are given in Figure 6.11 that shows the number of records when ETL (Economic Threshold Level) was crossed.

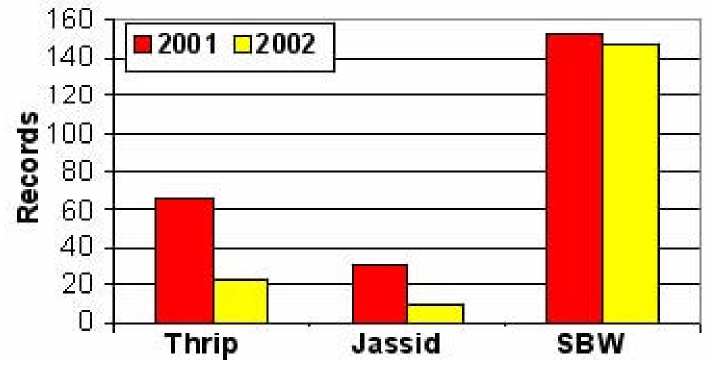


Figure 6.11: Comparison of pest incidence above ETL for 2001 and 2002

It can be seen that in year 2001 there are higher incidences of ETL (Economic Threshold Level) crossings for sucking pests as compared to 2002. Because of the attack of sucking pests, the plants lose their vigor, their growth is stunted and they are more susceptible to attack by bollworms at the later stage of growth. This is also evident by the higher number of ETL crossings for Spotted Boll Worm (SBW) in 2001 as compared to 2002.

In a nut-shell, in 2001 higher doses of *Cypermethrin* were used against metrological recommendations basically for controlling sucking pests, which is also not the best recommendation. As the most suitable pesticide was not used under lethal conditions, this could have resulted in developing resistance in the subsequent generations of the surviving pests, evident from comparatively many instances of ETL crossings in 2001, thus resulting in weak plants with low yields. This could be one of the possible reasons for low cotton yield during the 2000-2001 cotton season.

6.7 Conclusions

The BWM problem has been traditionally used in the domain of engineering, but has lately found applications in the domain of knowledge discovery vis-à-vis google.com with reference to link analysis. In this chapter, the crossing minimization paradigm was used to address the BWM problem and it was found that the proposed solution does significantly improve the BWM attributes of the problem being considered. For the pesticide usage anomaly a possible explanation is provided based on one-way clustering of pesticide usage and Meteorological data.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The aims of this research were to develop a new biclustering technique based on the crossing minimization paradigm, to study the use of the technique using simulated as well as real data and to identify and demonstrate the application of the said technique for other hard, yet interesting problems such as Minimum Attribute Subset Selection (MASS) problem and the sparse matrix Bandwidth Minimization (BWM) problem.

In Chapter (1) a brief introduction of both one-way and two-way clustering was presented along with a brief overview of the Minimum Attribute Subset Selection (MASS) problem, along with the BWM problem. The main focus of the chapter was about the aims, objectives and motivation of the research and its novelty. The novelty of the work presented in this thesis is established by two Journal publications, two book chapters in Springer Verlag Lecture Notes in Computer Science (LNCS) and a number of refereed International Conferences Publications, including IEEE.

In Chapter (2), necessary background material is presented regarding one-way and two-clustering methods with respect to unsupervised learning. Different aspects of graph drawing are discussed i.e. types of graph drawings, drawing aesthetics and the dilemma of generating quality drawings while meeting multiple-graph drawing aesthetics.

Necessary definitions, mathematical notation and the graph drawing model (including the white noise model) is also developed. Finally the working of a selected crossing minimization heuristic is explained using a simple example.

In Chapter (3) the proposed biclustering technique was discussed and was found to work well on simulated as well as real data sets as compared to the conventional biclustering techniques such as Cheng & Church [13], the OPSM approach [5] and SAMBA [80]. In contrast with these conventional approaches, the proposed technique does not require any a-priori knowledge or user input for performance tuning and is relatively significantly faster. It was observed that the quality of the results obtained using the proposed technique was dependent upon the initial permutation or the initialization of the vertices of the bipartite graph. Better results were obtained when the initial permutation was not random. For this purpose a meta-heuristic approach (running MH after MS) was used to improve the initial permutation and hence the overall results (details in Chapter 6).

One of the challenges of biclustering is the amount of noise present in the data. Noisy data is a reality rather than an exception. In Chapter (4) the noise tolerance of the proposed technique was discussed with reference to the problem faced by researchers i.e. selection of a clustering algorithm which is most appropriate while clustering noisy gene expression data. Researchers typically select a few candidate algorithms and compare the clustering results. However, there is no single “best” algorithm which is the “winner” in every respect. In this chapter an attempt was made to compare the proposed biclustering technique with OPSM [5] on the basis of noise present in the gene expression data using simulated as well as real data. It was observed that OPSM has low noise immunity as compared to biclustering by crossing minimization. For this study the cluster parameters in simulated data were deliberately kept the same due to the limitations of OPSM; biclustering by crossing minimization does not have such a limitation. Bipartite graph clustering is the grouping of highly connected vertices. Reordering the vertices within the cluster boundary does not change the cluster results, but is likely to change (or reduce) the number of crossings. Hence the objective is more of crossing reduction then crossing minimization, however, using the MS heuristic only good initial permutation of the vertices was achieved instead of good final solutions as compared to BC and MH.

In Chapter (5) the application of crossing minimization paradigm to the MASS was discussed. MASS problem has generated a lot of interest because of the increase in the number of attributes in datasets i.e. into hundreds. It was observed that using the proposed technique not only visually better results were obtained, but the quality of those results was also better quantitatively. It was also observed that when using a naïve approach of picking attributes with high similarity may actually deteriorate the cluster quality. The proposed technique takes into account the relationships between records and attributes and thus gives good results.

In Chapter (6) the application of crossing minimization paradigm to the BWM was discussed through an objective comparison of four crossing minimization heuristics, two classical bandwidth minimization heuristics along with two Meta combinations. Through extensive experiments it was observed that for very weak clusters i.e. corresponding sparse matrices, the performance of MH drastically deteriorates and the classical crossing minimization heuristics of Barycentre also fail to give good results for low density (sparse) matrices. However, using a variant of the MaxSort (MS) heuristic gave superior results and took less overall time, even when compared with the classical bandwidth minimization heuristics of King and RCM. The interesting observation was that using the Meta heuristic combination of MS with BC and MH gave an overall better performance and took less time too for real data.

Also in Chapter (6) the application of the crossing minimization paradigm for one-way clustering of Agro-Met data identified those spray dates when pesticide was sprayed against Met recommendations, furthermore the wrong pesticide was used in higher doses. This finding could be one of the possible explanations of negative correlation between pesticide consumption and cotton yield. Observe that cluster analysis can provide the rules for classification of similar datasets. For example, the analysis of the clustering results of Agro-met data for a certain year and geography can result in discovery of strong patterns, leading to identification of metrological conditions or rules conducive for pest infestation. Subsequently based on the weather forecast, these rules are used to classify periods of pest infestation next year for the same geography.

7.2 Limitations of the proposed technique

1. The final outcome or the permutation of vertices is dependent on the initial permutation of vertices, along with the choice of selecting the partition to begin

sequential numbering. In example 2.1, using MinSort and starting from TOP partition results in two crossings, while starting from the BOT partition results in zero crossings.

2. If the data matrix being considered has biclusters embedded in certain order with other data or noise, those biclusters may not be recovered with the same order. This could be a problem where actual order of recovery of biclusters is important. Compare Fig 3.3(a) with Fig 3.3(d), the order of biclusters has changed.

3. As a consequence of discretization, all numeric values get treated as a constant value, which includes coherent values as well as a coherent evolution. In the best case even if discretization causes removal of all noise, and the biclusters are captured intact, the coherency within the biclusters can not be ensured. This is visible in Fig 3.1(f), where the values within the rows and columns are not ordered.

4. In the context of MASS, a disadvantage of discretization is that all values are treated alike. As a consequence dense regions of 1's may not necessarily correspond to values of high similarity. The reason being, the 1's just indicate the values higher than the discretization criterion.

7.3 Future Work

As an extension to the work done so far in this thesis, possible future work proposals are given as follows:

a. In this thesis other than the indigenous MS crossing minimization technique the application of BC and MH was demonstrated for biclustering. It would be interesting to use other graph-theoretic crossing minimization techniques [65] for biclustering, such as sifting, split heuristic, greedy heuristic, greedy insert etc. and compare the performance as well as the quality of results obtained using these techniques with the work presented in this thesis.

b. The biclustering problem discussed in this thesis is an NP-Complete problem. As the proposed solution was shown to work with two other hard, yet interesting problems i.e. MASS and BWM problem, it would be worthwhile to identify other hard, yet interesting or NP-Complete problems for which the proposed solution could provide good results, quickly.

c. In this thesis the robustness of the proposed technique is evaluated using a white noise model. As discussed in Chapter (4) there are, however, at least three other major noise models used by the researchers. It would be interesting to evaluate and compare the performance of the proposed technique using these noise models.

d. In this thesis the biclustering solution was developed for a single processor; it would also be interesting to implement the proposed technique on distributed architectures or PC grids for one-way, as well as two-way clustering of very large data sets and observe the speed-up for cost-benefit analysis.

e. In this thesis a strong relationship was demonstrated between data mining and crossing minimization. It would be interesting to relate the crossing number to the metrics of the data mining so that the quality of the clustering results are estimated even before running the algorithm for very large data sets, thus saving substantial time and consequently improving the final solution. This approach could be further extended to developing bounds on the reduction of crossings and improvement of corresponding data mining results.

f. In this thesis it was shown that, due to the robustness of the estimation of position of the Median, the Median crossing minimization heuristic (MH) when used for data mining resulted in high noise tolerance. However, the problem with this approach is the high time complexity of calculating the median. One approach worth exploration would be to use median approximation algorithms to get quick results and do a performance-versus-quality analysis of the results.

g. Although multi-dimensional data sets were considered in this thesis, the results were displayed in a two-dimensional (2D) space, as it related with the bi-partite nature of the underlying graph. To enhance the information content of the visualizations, it would be interesting to explore representing the data using tri-partite graphs and displaying the resultant “matrix” in a 3D space.

h. It would be worth exploring the application of multi-level crossing minimization to data mining by taking into account the inherent relationships between attributes with respect to the Bayesian knowledge networks. These relationships to be represented as edges between different sets of attributes on different levels of a hierarchical graph.

Finally, the high time complexity approaches of permuting the vertices such as Simulated Annealing, Genetic Algorithm, Tabu Search etc. could be tried by starting from a “good” initial position of the solution space (wherever applicable) using the MS heuristic or its variation.

Appendix-A

Cluster quantification

For simulated data (and when actual clusters are known for real data), the clusters extracted can be quantified by comparing with actual clusters. There are several measures of cluster quantifications, the measures used in this thesis are Purity (P) and Entropy (E). When a clustering solution is unknown, then the clustering result can be statistically quantified using the measures of *Homogeneity* and *Separation.*

A.1 Purity

Purity measures how much of the extracted cluster belongs to a real cluster. The more a cluster falls in the extracted cluster, the better. Thus it is the max ratio between the elements of a cluster present in the extracted cluster, to the size of the extracted cluster. Formally purity is defined as follows:

The purity P of a cluster C_r, of size n_r is defined as $P(C_r) = \frac{1}{n_r} \max_i (n_r^i)$

where n_r^i is the number of members of the ith class that was assigned to the rth cluster. The overall purity P of the clustering solution is obtained as a weighted sum of the individual cluster purities and is given by

$$Purity = \sum_{r=1}^{k} \frac{n_r}{n} P(C_r)$$

In general the larger the values of purity, the better the clustering solution is.

A.2 Entropy

Entropy measures how distinct the extracted clusters are. Formally entropy is defined as follows:

Given a particular cluster C_r of size n_r, the entropy of this cluster is defined to be

$$E(C_r) = -\frac{1}{\log(q)} \sum_{i=1}^{q} \frac{n_r^i}{n_r} \log\left(\frac{n_r^i}{n_r}\right)$$

where q is the number of classes in the data set, and n_r^i is the number of members of the ith class that was assigned to the rth cluster.

The entropy E of the whole solution for k clusters is

$$Entropy = \sum_{r=1}^{k} \frac{n_r}{n} E(s_r)$$

A perfect clustering solution will be one that leads to the clusters that contains members from only a single class, in which case the entropy will be zero, in general the smaller the entropy values, the better the clustering solution is.

A.3 Homogeneity

It is the average similarity between each row and that of its cluster. Precisely, if *cl(u)* is the cluster of *u*, *F(X)* and *F(u)* are the columns of a cluster *X* and an element *u*, respectively, then

$$H_{Ave} = \frac{1}{|N|}\sum_{u\in N} S(F(u), F(cl(u)))$$

For a set of elements $K \subseteq N$, the *centroid* of *K* is the mean vector of the columns of the members of *K*. For two rows *x* and *y*, *S(x, y)* denotes their similarity.

A.4 Separation

Separation is evaluated by the weighted average similarity between cluster members. Therefore, for clusters $X_1, \ldots X_t$, the separation is defined as

$$S_{Ave} = \frac{1}{\sum_{i\neq j}|X_i||X_j|}\sum_{i\neq j}|X_i||X_j|S(F(X_i), F(X_j))$$

Related measures that take a worst case instead of average case approach are minimum homogeneity $H_{Min} = min_{u\in N}\ S(F(u),F(cl(u)))$ and maximum separation: $S_{max} = max_{i\neq j}\ S(F(Xi),F(Xj))$. Hence a solution improves if H_{Ave} or H_{min} increases, and if S_{Ave} or S_{max} decreases.

References

[1] A. Abdullah, *On placement of logic schematics*, Proceedings of IEEE TENCON'93, Beijing, China, (1993), 885-888.

[2] P. Berkhin, *Survey of Clustering Data Mining Techniques*, Technical Report, Accrue Software, San Jose, CA, 2002

[3] S. Barkow1, S. Bleuler, A. Prelic, P. Zimmermann, and E. Zitzler, *BicAT: a biclustering analysis toolbox*, Bioinformatics, **22** no. 10 (2006), 1282–1283

[4] K. Bryan, P. Cunningham and N. Bolshakova, *Biclustering of Expression Data Using Simulated Annealing*, Proceedings 18th IEEE Symposium on Computer-Based Medical Systems, 23-24 June 2005, 383-388.

[5] A. Ben-Dor, B. Chor, R. Karp, & Z. Yakhini, *Discovering local structure in gene expression data: The order-preserving submatrix problem*, Proceedings of the 6th International Conference on Computational Biology (RECOMB'02), (2002), 49–57.

[6] G. D. Battista, P. Eades, R. Tamassia, I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, (1999).

[7] G. T. Bohmfalk, R. E. Frisbie, W. L. Sterling, R. B. Metzer and A. E. Knutson *Identification, Biology And Sampling Of Cotton Insects*, Texas A&M University System, (1996).

[8] P. Baldi, D. Kibler et al., *UCI Machine Learning Repository*, retrieved on 2nd Oct. 2004 from UCI Machine Learning, Web Resource, http://www.ics.uci.edu/~mlearn/

[9] L. Boudjeloud and F. Poulet, *Attribute Selection for High Dimensional Data Clustering*, ESIEA Recherche, Parc Universitaire de Laval-Change, 38 Rue des Docteurs Calmette et Guerin, 53000 Laval, France, (2005).

[10] W. Baines, G. Smith, *A Novel method for DNA sequence determination*, J Theor Biol **135**, (1988), 330-307

[11] A. Ben-Dor, R. Shamir and Z. Yakhini, *Clustering Gene Expression Patterns*, Journal of Computational Biology, **6** (1999), 281-297.

[12] M. R. Chaudhry, *New Frontiers in Cotton Production*, International Cotton Advisory Committee, USA, (2000).

[13] Y. Cheng and G. M. Church, *Biclustering of expression data*, in Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB'00), (2000), 93–103.

[14] K. C. Chang and D. H. C. Du, *Efficient algorithms for layer assignment problem*, IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, **6** no. 1 (1987), 67-78

[15] Z. Cai, H. W. Leong, *Improving Performance of DNA Sequencing-by-Hybridization with Gapped-Probes via Domain Knowledge*, Proceedings of the BioTechnology and Bioinformatics Symposium (BIOT-2006), Provo, Utah, USA(2006), 74-82.

[16] A. Chakraborty and H. Maka, *Biclustering Gene Expression Data Using Genetic Algorithm*, Proceedings of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, 14-15 Nov. 2005, 1 – 8

[17] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, Proceedings of the 24th National Conference of the ACM, (1969).

[18] S. Chu and J. F. Roddick, *A Clustering Algorithm using the Tabu Search Approach with Simulated Annealing*, International conference on data mining Cambridge, Jul. 2000, 515-523

[19] N. A. Cambell, J. B. Reece, *Biology*, 6th edition, Pearson Education Inc., 2002.

[20] I. S. Dhillon, *Co-clustering documents and words using bipartite spectral graph partitioning*, in Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01), (2001), 269–274.

[21] [D01-2] R. Dror, *Noise models in gene array analysis*, Report in fulfillment of the area exam requirement in the MIT Department of Electrical Engineering and Computer Science, (2001).

[22] M. Dash, H. Liu, *Feature Selection for Classification*, Intelligent data Analysis, **1** no. 3, (1997).

[23] R.O. Dror, J.G. Murnick, N.J. Rinaldi, V. D. Marinescu, R.M. Rifkin, & R.A. Young, *A Bayesian approach to transcript estimation from gene array data: the*

BEAM technique, In Proceedings of the Sixth Annual International Conference on Research in Computational Molecular Biology (RECOMB), (2002).

[24] M. Dong, *A New Measure of Classifiability and Its Applications*, PhD Dissertation Department of Electrical & Computer Engineering and Computer Science of the College of Engineering, University of Cincinnati, (2001).

[25] E. Piñana, I. Plana, V. Campos and R. Martí, *GRASP and Path Relinking for the Matrix Bandwidth Minimization*, European Journal of Operations Research, **153** (2002).

[26] M. Ester, H. P. Kriege, J. Sander and X. Xu, *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*, Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR, (1996), 226-231.

[27] P. Eades, N. Wormald, *The Median Heuristic for Drawing 2-layers Networks*, Technical Report 69, Department of Computer Science, University of Queensland, Brisbane, Australia, (1986).

[28] U. Fayyad, and R. Uthurusamy, *Data Mining and Knowledge Discovery in Databases*, Comm. ACM, **39** no. 11, (1996), 24-27.

[29] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, 1980.

[30] A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Inc, Englewood Cliffs, NJ, (1981).

[31] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, (1979).

[32] M. R. Garey and D. S Johnson, *Crossing number is NP-Complete*, SIAM J. Algebraic Discrete Methods, **4** (1983), 312-316.

[33] J. A. Hartigan, *Direct clustering of a data matrix*. Journal of the American Statistical Association (JASA), **67** (1972), 123–129.

[34] M. A. Hall and G. Holmes, *Benchmarking Attribute Selection Techniques for Discrete Class Data Mining*, IEEE Trans. on knowledge discovery and Data Engineering, **15** no. 6, (2003), 1437-1447.

[35] P. Hansen, and B. Jaumard, *Cluster analysis and mathematical programming*, Mathematical Programming, **79** (1997), 191-215

[36] Han and Kamber, *Data Mining: Concepts and Techniques*, Pub Morgan Kaufmann Publishers, (2000).

[37] E. Hartuv and R. Shamir, *A clustering algorithm based on graph connectivity*, Information Processing Letter, **76** (2000), 175-181.

[38] A. Hassibi and H. Vikalo, *A probabilistic model for inherent noise and systematic errors of microarrays,* in Proc. IEEE Workshop on Genomic Signal Processing and Statistics (GENSIPS*)*, Newport, RI, (2005).

[39] J. Herrero, A. Valencia A. and J. Dopazo, *A hierarchical unsupervised growing neural network for clustering gene expression patterns*, Bioinformatics, **17** (2001), 126–136.

[40] I. Inza, P. Larranga, R. Etxeberria and B. Sierra, *Feature Subset Selection by Bayesian networks based optimization*, Artificial Intelligence, **123** no. 1, (2000), 157-184.

[41] *The wine making process*, retrieved 10th Oct. 2004 from The International Wine of the Month Club, Web Resource, **`http://www.winemonthclub.com/index.htm`**

[42] S. C. Johnson, *Hierarchical Clustering Schemes*, Psychometrika, 2 (1967), 241-254

[43] G. H. John, *Enhancements to the Data Mining Process*, PhD Dissertation, Stanford University, (1997).

[44] M. Jünger and P. Mutzel, *2-layer straight-line crossing minimization: performance of exact and heuristic algorithms*, J. Graph Algorithms Appl. **1** (1997), 1–25.

[45] Y. Kluger, R. Barsi, JT. Cheng, and M. Gerstein, *Spectral biclustering of microarray data: coclustering genes and conditions*, Genome Res., 13(4): (2003) 703–16.

[46] M. A. Khan, M. Iqbal, I. Ahmad and M. Soomro, *Economic Evaluation of Pesticide Use Externalities in the Cotton Zones of Punjab, Pakistan*, The Pakistan Development Review, 41:4 Part II (2002) 683–698

[47] M. Koyuturk, W. Szpankowski, A. Grama, *Biclustering gene-feature matrices for statistically significant dense patterns*, Proc. of the 8th International Conference on Research in Computational Molecular Biology, (2004), 480-484.

[48] A. Keller, M. Schummer, L Hood & W. Ruzzo, *Bayesian Classification of DNA Array Expression Data*, Technical Report UW-CSE-2000-08-01, (2000).

[49] Y. Kim, W. N. Street, and F. Menczer, *Evolutionary model selection in unsupervised learning*, Intelligent Data Analysis, **6** (2002), 531–556.

[50] F. T. Leighton, *New lower bound techniques for VLSI*, NASA ST/Recon Technical Report N, 83:19003, Aug. 1982

[51] T. Loos and R. Bramley, *Emily: A visualization utility for large matrices*, Technical Report 412, Indiana University, Bloomington, IN, (1994).

[52] H. Liu and H. Motoda, *Feature selection for knowledge discovery and data mining*, in Kluwer International Series in Engineering and Computer Science, (1998).

[53] M. Laguna and R. Martí, *GRASP and Path Relinking for 2-Layer straight line crossing minimization*, INFORMS Journal on Computing, **11** no. 1, (1999), 44–52.

[54] M. Laguna, R. Martí and V. Valls, *Arc crossing minimization in hierarchical digraphs with tabu search*, Comput. Oper. Res. 24, 2 (1997), 1175–1186.

[55] J. Liu, W. Wang, J. Yang, *Biclustering in gene expression data by tendency*, Proc. of the 3rd International IEEE Computer Society Computational Systems Bioinformatics Conference, (2004), 182-193.

[56] C. Muller, *Sparse Matrix Reordering Algorithms for Cluster Identification*, For I532, Machine Learning in Bioinformatics, (2004).

[57] T. M. Murali, S. Kasif, *Extracting conserved gene expression motifs from gene expression data*, Proc. of the Pacific Symposium on Biocomputing, **8** (2003), 77-88.

[58] R. Martí, M. Laguna, Fred Glover and Vicente Campos, *Reducing the Bandwidth of a Sparse Matrix with Tabu Search*, European Journal of Operational Research, 135 no. 2, (2001), 211-220

[59] R. Marti and M. Laguna, *Heuristics and Meta Heuristics for 2-layer Straight Line Crossing Minimization*, Discrete Applied Mathematics, **127** Issue 3, (2001), 665 – 678.

[60] M. May and P. Mennecke, *Layout of schematic drawings*, Syst. Anal. Modelling Simulation, **1** (1984), 307–338.

[61] S. C. Madeira & A. L. Oliveira, *Biclustering algorithms for biological data analysis: a survey*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, **1** no.1 (2004), 24-45.

[62] E. Mäkinen and H. Siirtola. *The Barycenter Heuristic and the Reorderable Matrix*, Informatica **29** (2005), 357–363

[63] M. May and K. Szkatula, *On the bipartite crossing number*, Control Cybern. 17 (1988), 85–98.

[64] E. Mäkinen and M. Sieranta, *Genetic algorithms for drawing bipartite graphs*, Int. J. Comput. Math. 53 (1994), 157–166.

[65] C. Matuszewski, R. Schönfeld and P. Molior, *Using sifting for k-layer straightline crossing minimization*. Proc. 7th Int. Symposium of Graph Drawing, Lecture Notes in Computer Science 1731(1999), pp. 217-224.

[66] M. Newton, O. Sýkora, and I. Vrt'o, *Two new heuristics for two-sided bipartite graph drawing*, Proc. 10th Int. Symposium of Graph Drawing. Lecture Notes in Computer Science 2528, (2002), 312–319.

[67] A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, et al. *A systematic comparison and evaluation of biclustering methods for gene expression data*, Bioinformatics **22** (2006), 1122–1129.

[68] C. H. Papadimitriou, *The NP-completeness of the bandwidth minimization problem*, Computing, **16** (1976), 263-270.

[69] G. L. Pappa, A. A. Freitas and C. A. A. Kaestner, *A multiobjective Genetic Algorithm for Attribute Selection*, Proc. 16th Brazilian Symp. on Artificial Intelligence (SBIA-2002), Springer-Verlag Lecture Notes in Artificial Intelligence **2507** (2002), 280-290.

[70] F. Preparata, J. Oliver, *DNA Sequencing by Hybridization Using Semi-degenerate Bases*, Comput Biol **11** no. 4, (2004), 753-765

[71] R. Peeters, *The maximum edge biclique problem is NP-complete*, Discrete Applied Mathematics, **131** no. 3 (2003), 651–654.

[72] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, Boston, 1993.

[73] M. Stallmann, F. Brglez, and D. Ghosh, Heuristics, *Experimental Subjects, and Treatment Evaluation in Bigraph Crossing Minimization*, Technical report 2000-TR@CBL-04-Stallmann, NCSU, (2000).

[74] Y. Tu, G. Stolovitzky, & U. Klein, *Quantitative noise analysis for gene expression microarray experiments*, Proceedings of the National Academy of Sciences, USA, **99** no. 22, (2002), 14031–14036.

[75] *Cotton and Ginning* retrieved on 20th Sept. 2003 from Small and Medium Enterprise Development Agency (SMEDA), Pakistan, Web Resource, http://pgbf.com.pk/market_briefs/Cotton%20&%20Ginning.pdf

[76] Q. Sheng, Y. Moreau, B. D. Moor, *Biclustering microarray data by Gibbs sampling*, Bioinformatics **19** no. 2, (2003), 196-205.

[77] K. Sugiyama, S. Tagawa and M. Toda, *Methods for Visual Understanding of Hierarchical Systems*. IEEE Trans. Syst. Man Cybern., SMC-11, **2** (1981), 109-125.

[78] S. Seno, R. Teramoto, Y. Takenak, and H. Matsuda, *A Method for Clustering Gene Expression Data Based on Graph Structure*, Genome Informatics **15** no. 2, (2004), 151–160

[79] I. Shmulevich and W. Zhang, *Binary analysis and optimization-based normalization of gene expression data*, Bioinformatics, **18** (2002), 555–565.

[80] A. Tanay, R. Sharan and R. Shamir, *Discovering statistically significant biclusters in gene expression data*, Bioinformatics, **18** (2002), 136-144.

[81] C. Traina, L. Wu, A. Traina, and C. Faloutsos, *Fast Feature Selection Using Fractal Dimension*, XV Brazilian Symposium on Databases (SBBD), Paraiba, Brazil, (2000).

[82] V. Valls, R. Martí and P. Lino, *A branch and bound algorithm for minimizing the number of crossing arcs in bipartite graphs*, Eur. J. Oper. Res. **90** (1996), 303–319.

[83] D. B. West, *Introduction to Graph Theory*, Prentice-Hall, Inc., 2003.

[84] C. Wu, Y. Fu, T. M. Murali, S. Kasif, *Gene expression module discovery using Gibbs sampling*, Genome Informatics **15** no. 1, (2004), 239-248.

[85] X. Wang, Ao Li, Z. Jiang and H. Feng, *Missing value estimation for DNA microarray gene expression data by Support Vector Regression imputation and orthogonal coding scheme*, BMC Bioinformatics, **7** no. 32 (2006), 1-10

[86] Z. Xiong, A. Nadeem, Z. Weng, and M. R. Nelson, *Cotton Leaf Curl Virus is Distinct from Cotton Leaf Crumple Virus*, Department of Plant Pathology, University of Arizona, (1997)

[87] Y. Yang and G. I. Webb, *A Comparative Study of Discretization Methods for Naive-Bayesian Classifiers*, Proceedings Pacific Rim Knowledge Acquisition Workshop, National Center of Sciences, Tokyo, Japan, (2002).

[88] J. Yang, H. Wang, W. Wang, P. Yu, *Enhanced biclustering on expression*, Proceedings IEEE Third Symposium on Bioinformatics and Bioengineering, (2003).