WEIGHTED BIPARTITE CROSSING MINIMIZATION APPLICATIONS ON BICLUSTERING AND GRAPH UNIONS

MELİH SÖZDİNLER

IŞIK UNIVERSITY
2009

M. SÖZDİNLER M.S. Thesis 2009

WEIGHTED BIPARTITE CROSSING MINIMIZATION APPLICATIONS ON BICLUSTERING AND GRAPH UNIONS

MELİH SÖZDİNLER

IŞIK UNIVERSITY
2009

WEIGHTED BIPARTITE CROSSING MINIMIZATION APPLICATIONS ON BICLUSTERING AND GRAPH UNIONS

MELİH SÖZDİNLER

B.S, Computer Engineering, Işık University, 2007

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Engineering

IŞIK UNIVERSITY
2009

IŞIK UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

WEIGHTED BIPARTITE CROSSING MINIMIZATION
APPLICATIONS ON
BICLUSTERING AND GRAPH UNIONS

MELİH SÖZDİNLER

APPROVED BY:

Assist. Prof. Cesim Erten (Kadir Has University) _____
(Thesis Supervisor)

Assist. Prof. Tankut Atan (Işık University) _____

Assist. Prof. Taner Eskil (Işık University) _____

APPROVAL DATE: 03/02/2009

WEIGHTED BIPARTITE CROSSING MINIMIZATION APPLICATIONS ON BICLUSTERING AND GRAPH UNIONS

Abstract

Biclustering gene expression data is the problem of extracting submatrices of genes and conditions exhibiting significant correlation across both the rows and the columns of a data matrix of expression values. We provide a method, LEB (Localize-and-Extract Biclusters) which reduces the search space into local neighborhoods within the matrix by first localizing correlated structures. The localization procedure takes its roots from effective use of graph-theoretical methods applied to problems exhibiting a similar structure to that of biclustering. Once interesting structures are localized the search space reduces to small neighborhoods and the biclusters are extracted from these localities. We evaluate the effectiveness of our method with extensive experiments both using artificial and real datasets. Finally, we also used our crossing minimization heuristics for graph visualization in a layered fashion.

AĞIRLIKLI İKİLİ ÇİZGELERİN
AYRIT KESIŞİMLERİ AZALTILMASININ
İKİLİ KÜMELEME VE ÇİZGELERİN GÖRSELLEŞTİRMESİ
PROBLEMLERİNE UYGULANMASI

Özet

Çift taraflı kümeleme problemi, biyol ile ilgili verilerin alt matrisler arasından belli oranda verinin sütun kısmında ve satır kısmında ilintili olanları elde etme problemidir. Biz adı LEB(Localize-and-Extract Biclusters) olan bir yöntem sunarak çift taraflı kümeleme problemini tüm veri üzerinde çalışması yerine, kendine yakın komşu olan alt matrisler üzerinde çalışmasını sağladık. Bu sayede tarama süreci, genelden yerel alt kümelere indirgenmiş oldu. Yerelleştirme probleminin temelini, *çizge* tabanlı teorik yöntem kullanarak çift taraflı kümeleme problemi ile ilintili oldugunu deney yaparak belirledik. Yerelleştirme metodundan sonra bu küçük alt yapıların birleştirilmesi içinde ayrı yöntem önerdik. Son olarakta biz öne sürdügümüz yöntemın performansını birçok deney yaparak hem gerçek hem de sanal veriler üzerinde denedik. Bunun yanısıra çizgeler için öne sürdüğmüz yöntemi, çizgelerin görselleştirmesi içinde kullandık. Bunu da ikinci kısımda ayrıntılı olarak inceledik.

Acknowledgements

I am glad to finish my master thesis.

First of all, thanks to my family since they are always behind me and they encouraged me to complete my graduate study. In addition, for my development at computer engineering, thanks to Asist. Prof. Cesim Erten, Prof. Dr. Selahattin Kuru, Asist. Prof. Ercan Solak, Asist. Prof. Olcay Taner Yıldız, Asist. Prof. Taner Eskil. Also, thanks other lecturers for my development in Computer Engineering. Additionally, Thanks to the professors that accept my invitatition for being a jury for my presentation. Finally, special thanks to all my Teaching Assistant friends and also thanks to our Graduate Studies secretary Sevgi Dikmen.

For this master thesis, thanks to my supervisor Assistant Profesor Cesim Erten and also my project partners Arda Çakıroglu and Ömer Karataş. We had to support ourselves and we did well by having a conference paper at *WEA'07* and a journal paper at *JDA'08*.

Furthermore, I have special thanks to TUBITAK(The Scientific and Technological Research Council of Turkey). They supported me and paid monthly stipends via BIDEB. With their payments, I could be able to buy a new computer for my project.

Moreover, I am very grateful to my university. During my two year graduate period with assitantship, I have taken full support from my university. My family and I are thankful again to the Işık University.

Finally, I am very lucky to have a family that has full confidence to my success. Also, without their support, I could probably have a lack of concentration.

All in all, my perspective of view to the acadamic life has changed a lot when I compare before Işık University and after Işık University. I appreciate for all of these changes.

My best Regards

Preface

This master thesis is about the work that I have done during two years study period. Also, thesis includes study period when I was senior undergraduate student. There are two main topics that I have studied. The first topic is *Biclustering* and the second topic is *Graph Union*. In both field we are using graph theoretical approaches.

For thesis research, I have looked at many related databases and found 100 published papers. I have given more than 50 references at bibliography and that means countless hours in front of computers and sacrificing from daily life in order to study.

The motivating force my research is purely satisfaction after each work, paper and poster. A study in the last day of deadline and struggling to submit new work is different feeling than others. The relaxation after conference deadline or maybe academic presentation is perfect comfort. The passed deadline is also big frustration, if you are having ideas in mind. These are the feelings that I feel during my thesis study.

Contributing to computer science, feeling of future contributions, and the feeling of usefulness of your study are another motivation.

Indeed, working over these topics gives me pleasure and for phd studies I am planning to go over these topics.

Table of Contents

List of Figures

List of Tables

List of Symbols

G	:	Graph
E	:	The Set of All Edges in the Graph
V	:	The Set of All Vertices in the Graph
$cross(e)$	:	Total Number of Crossings Between Two Layers of Bipartite Graph
$w(u, v)$	:	Weigth value of the Edge Between two vertices u and v
L_0	:	First Layer of Bipartite Graph
L_1	:	Second Layer of Bipartite Graph
B_1, B_2	:	Bicliques
RS	:	Residue Score
H	:	H Value
$\|L_0\|$	:	Number of Vertices in L_0
$\|L_1\|$	:	Number of Vertices in L_1
$\|E\|$	:	Number of Edges for a Given Graph G
$\|e\|$	:	The Length of Edge e
$\|V\|$	:	Number of Vertices for a Given Graph G
$density$	:	The Reqired Minimum Density for Each Bag to Extract Constant Biclusters
α	:	The Length of the Bags for LEB Algorithm at x Direction
β	:	The Length of the Bags for LEB Algorithm at y Direction
F	:	Feedback Arcset Edges
C	:	The edges in Cycle

List of Abbreviations

ANH	:	Adaptive Noise Hiding
BICAT	:	Biclustering Analysis Toolbox
BIMAX	:	Fast Divide-and-Conquer Algorithm of Prelic *er al*
CC	:	Cheng and Church Algorithm
CMH	:	Crossing Minimization Heuristics
FAS	:	Feedback Arcset Problem
GRE	:	Greedy Algorithm
ISA	:	Iterative Signature Algoritm
LEB	:	Localize-and-Extract Biclusters
LEDA	:	Library of Efficient Data types and Algorithms
OLF	:	One Layer Free Problem
OPSM	:	Order Preserving Sub Matrix Algorithm
PM	:	Penalty Minimization
SAMBA	:	Statistical Algorithmic Method for Bicluster Analysis
W-BARY	:	Weighted Barycenter Heuristics
W-MED	:	Weighted Median Heuristics
WOLF	:	Weighted One Layer Free Algorithm
xMOTIF	:	Conserved Gene Expression Motifs
VLSI	:	Very-large-scale integration

Chapter 1

Introduction

In the beginning, fear was the dominant motivating force.
Robert Vaughn

In this master thesis, we worked on two different topics that are related with the Bioinformatics and the Graph Visualization. Previously, we proposed an specific algorithm for the Graph Theory. In this thesis, we give two implementations for each main topic. First of all, we proposed a Biclustering algorithm related with our main algorithm on Graph Theory. We give detailed theory and experiments starting from Chapter 2. In addition, we also used our technique in order to draw graphs in layered fashion. In that case, we suggest a new layout for drawing graphs and it could be used to union the graphs. The detail of that part starts with Chapter 7.

Chapter 2

Biclustering

2.1 Motivation

Clustering refers to the process of organizing a set of input vectors into clusters based on similarity defined according to a predefined distance measure. In many cases it is more desirable to simultaneously cluster the dimensions as well as the vectors themselves. This special instance of clustering referred to as *biclustering*, was introduced by Hartigan [1]. It has many applications in areas including data mining, pattern recognition, and computational biology.

Biclustering is a new developing paradigm in the literature. Recently, many people proposed different approaches to this paradigm. Biclustering is different from the existing clustering approach. By using usual clustering methods, we can only make one way clustering. This gives us a global perspective of view to the data in consideration. On the other hand, by using biclustering, we can predict locally important information, since it works on both way of the data.

Biclusters are identified among four major classes. These are:

1. Biclusters with constant values
2. Biclusters with constant values on rows or columns
3. Biclusters with coherent values
4. Biclusters with coherent evolutions

Constant valued biclusters represent the property of having the same value inside the biclusters. If it happens in row or column of bicluster, it is called constant values on rows or columns.

Coherent valued biclusters represent the pattern that each row or column is obtained by adding or multiplying a constant to another row or column.

It is easier to find constant valued clusters. Coherent valued clusters are more complicated than constant ones. Because they have assigned by using a formulation such as additive bicluster or multiplicative bicluster.

Biclusters with constant rows or columns are assesed with some formulas. According to additive bicluster model, we can find perfect constant row biclusters by

$$a_{i,j} = \delta + \alpha_j \tag{2.1}$$

where $a_{i,j}$ is each value in input matrix, δ typical value for the row and α_i is the adjustment value for the row i. Above formula is changed to

$$a_{i,j} = \delta + \gamma_j \tag{2.2}$$

when we are considering constant column perfect biclusters, and γ_i adjustment ratio for the column i.

According to the multiplicative model Equation 1.1 changes to

$$a_{i,j} = \delta * \alpha_i \tag{2.3}$$

and Equation 1.2 changes to

$$a_{i,j} = \delta * \gamma_i \tag{2.4}$$

On the other hand, if we are considering coherent values, they have adjustment values for both column and row. As a result, bicluster could be more complicated.

Furthermore, we can divide biclusters in categories according to its structure. In survey of Madeira and Oliveira [2], you can see detailed structures.

Our motivation is to find both constant valued biclusters and coherent valued biclusters. We can find constant valued biclusters because our algorithm

of extraction could do that and also we can find coherent valued ones because we have robust noise removal and thereshold of obtaining sustainable density.

Biclustering mainly deals with bioinformatics' applications such as microarray analysis, drug activity analysis, motif detection [3, 4, 5, 6, 7, 8, 9, 10]. Adding that, biclustering can also deal the problem of information retrieval and textmining, database research and data mining and analysis of electoral data. In this thesis, we are dealing with microarray analysis.

2.2 Previous Work

2.2.1 Biclustering

One of the early approaches for biclustering expression data is that of Cheng and Church [11]. They provide a greedy, iterative algorithm with running time $O(mn)$, where m and n are the dimensions of the data matrix. The *mean squared residue* score is defined and the algorithm greedily inserts/removes rows and columns to arrive at a certain number of biclusters achieving some predefined score value. When they discover a bicluster, they put random values inside and iterates over it again. These random values may be interfered as a candidate for another bicluster, and they could be not good for postcoming bicluster results. They need also predefined value for number of biclusters. According to this value, algorithm iterates itself. Finally, they focus on determining coherent values due to scoring function. Order-Preserving Sub Matrix(OPSM) [3] is another greedy, iterative algorithm, that finds a statistically significant bicluster at each iteration. It greedily runs over the data matrix and enlarges the best bicluster and continues until there is no reported bicluster. They defined a bicluster as a set of selected rows that preserves the ordering in columns. This means that algorithm focuses on columns. The time complexity of OPSM is $O(nm^3l)$ where n and m are dimensions of input matrix and l is the number of output biclusters. Because of cubic runtime in second dimension, algorithm does not scale for high dimensional input sizes. Also, in essence, OPSM does a method like catch a possible bicluster, and try it if it is good or not. This greedy strategy

called partial models. OPSM technique is to expand these partial models at each step. OPSM also focuses on coherent valued biclusters because of catch and try strategy. Conserved gene expression motifs or xMOTIFs [5] is another greedy algorithm which finds all biclusters at a single run. Basically, they are looking largest xMOTIF and XMOTIF is identified according to up-regulated and down-regulated patterns when there are two state in concept. It is a kind of discretization and like working on binary data matrix. After that step, they collect these stated values in order to maximize their predefined fraction for rows and columns and xMOTIF could do that simultaneously. However the algorithm does not work for more than 64 conditions.

ISA [7] introduces a statistical approach to the biclustering problem. It requires normalized data with mean 0, variance 1 and assigns weights for each input that represents significant behaviors with similar weights. These significant weights assigned according to z-scores and each higher score results with larger weights. Thus the resulting biclusters are expected to be the ones with similarly assigned weights. It has been applied to determine cis-regulatory modules in yeast dataset.

Prelic et al. modified the approach of Alexe et al. [12] in a way that works fast and requires less space. The resulting divide-and-conquer algorithm, Bimax [10], is simple and fast. The algorithm runs on discretized binary data. Algorithm mainly focuses on finding constant biclusters. Discretized binary data makes it harder to find coherent values. Since they rely on discretization strategy, the results of BIMAX could change according to used strategy. Coupled two way clustering is proposed by Getz et al. [13]. First stable forms on submatrices that are used to divide the original dataset are found. Then one-way clustering is applied recursively on a single dimension of the submatrices until there is no newly found stable submatrix. They guarantee that each submatrix pairs is not encountered more than once. Their success depends on the performance of one dimensional clustering algorithm such as K-means, Hierarchical, SOM. They used their algorithm in order to determine diseases on clinical data sets.

Several graph-theoretical approaches have been suggested. In SAMBA [4] the data matrix is viewed as a bipartite graph where the genes/conditions constitute the layers of the bipartite graph and edges in the graph correspond to the expression changes. The goal is to find out heavy bicliques inside the graph. Running time of SAMBA is exponential on the size of the conditions set in a maximum bounded biclique because of the employed exhaustive bicluster enumeration. A similar model is constructed in [8] where crossing minimization in unit-weight bipartite graphs is used as a means to extract bicliques corresponding to biclusters in the data matrix. They used barycenter [14] like method in order to crossing minimization. The proposed approach in [8] lacks of quantativeness to extract biclusters. Also, they are not given any clue about median [15], GRE [16] and PM [17] that could also solve unit weight crossing minimization.

2.2.2 Bipartite Crossing Minimization

Jünger and Mutzel survey various heuristics and experimentally compare their performances of bipartite crossing minimization heuristics [18]. They conclude that the barycenter method yields slightly better results than the median heuristic in practice. On the other hand from a theoretical point of view median heuristic is better. Specifically, they both run in linear time and the median heuristic is a 3-approximation, whereas the approximation ratio of the barycenter method is $\Theta\left(\sqrt{|L_0|}\right)$ [15]. Yamaguchi and Sugimoto [16] provide a greedy algorithm GRE that has the same approximation ratio of 3 in the worst case and that works well in practice. However the running time of GRE is quadratic. Recently, Nagamochi devised a 1.47-approximation algorithm for OLF [19]. Another promising technique for OLF is the penalty graph approach introduced by Sugiyama *et al.* [14]. The performance of this method depends on an effective solution to the minimum feedback arc set (FAS) problem which is also NP-complete [20]. Demetrescu and Finocchi experimentally compare the performance of the penalty graph method based on their algorithm for FAS to that of the barycenter, median, and the GRE heuristics [21]. In addition to the

mentioned heuristics, approaches based on integer programming formulations that solve the problem exactly have been suggested. Jünger and Mutzel showed that OLF can be formulated as a linear ordering problem which is then solved optimally by employing a branch-and-cut technique [18]. Although directly applicable to WOLF, this exact approach works well mostly for sparse graphs of relatively small size.

In [22], we also proposed weighted version of Barycenter as W-BARY, GRE as W-GRE, PM as W-PM, and Median as W-MEDBARY. W-MEDBARY does barycenter in its second phase. Since the crossing minimization performance of each algorithm depends on the distribution of weights, there is no prediction that the algorithm 'x' could be better.

In this paper, we are interested in with two problems, *Biclustering problem* and *Bipartite Crossing Minimization*

Biclustering problem is NP-Complete problem, reduced from maximum edge biclique problem [23]. The problem could be approximated and be given heuristics.

Bipartite crossing minimization is also NP-Complete problem. The former is usually referred to as the both layers free bipartite crossing minimization whereas the latter as the one layer free bipartite crossing minimization. Both problems have been extensively studied in literature. Unfortunately they are both NP-hard [15, 24].

One drawback of approaches that is using bipartite graph based approach is the assumption that the corresponding bipartite graph is unweighted. For example, Abdullah et.al.'s algorithm does discretization step before the algortihm. Also, in BIMAX, they need discretization in order to run the algorithm. Since, in both BIMAX and Abdullah et.al.'s approach, they lost original input matrix with discretization, they are actually running their algorithm over different input rather than original one. In our algorithm, we are following a direct graph-drawing approach is [8]. A crossing minimization procedure is applied on the unweighted bipartite graph resulting

from preprocessing the original input data matrix. Our approach is similar in essence. However we do not have a discretization/normalization step to convert the weighted bipartite graph into an unweighted one as this would cause some data loss and produce erroneous output. Instead we apply crossing minimization directly on the original weighted graph. Various efficient crossing minimization heuristics have been shown to work well on weighted bipartite graphs as mentioned before and a 3-approximation algorithm has been suggested [22].

2.3 Summary of Main Results

We make a general outline of our work

1. We proposed a biclustering algorithm using weighted bipartite crossing minimization.

2. We proposed generic bicluster extraction algorithm in order to finalize biclustering.

3. We have given various experiments over the biclustering topic and tested on several data sets and compared the results of several biclustering algorithms.

Chapter 3

Preliminaries

Bicluster:A bicluster N is subset of vertices and edges over the graph. Graph G is considered as $G = (L_0, L_1, E)$ where G is bipartite graph, L_0 and L_1 are layers of graph G and $L_0 \cup L_1 = V$, E is set of edges, bicluster N is a subgraph $G' = (L'_0, L'_1, E')$ of G. Also, a bicluster could be represented as matrix. If the input is given as adjacency matrix M, bicluster will be submatrix M' of given M. For each element at the input matrix could be represented as $a_{i,j}$. That value also represents the relation between gene and condition pair in bioinformatics.

Weighted Bipartite Graph:A graph G with set of edges E and set of vertices V with layers L_0 and L_1. There is no edge between same layer node. Weighted means that for every edges e at E, there is a real number representing the weight value of edge as $w(u, v)$, where u from layer L_0 and v from layer L_1. In our problem, we represent the matrix input of microarray data as a weighted bipartite graph.

Weighted Bipartite Graph Crossing Minimization: Weighted bipartite graph crossing minimization is an optimization problem that is to be solved by heuristics. The cost function is

$$\min \sum_{e}^{E} cross(e) \tag{3.1}$$

where *cross(e)* is total crossings between edge e and all other edges in set E.

In order to solve *Weighted Bipartite Graph Crossing Minimization*, Weighted One Layer Free(WOLF) and two sided crossing minimization are two options. WOLF is the new form of OLF. In order to solve OLF problem, barycenter[14], median[15], GRE[16] and PM [17] are proposed. In WOLF

setting, one layer is set as free layer, meaning that positions of the nodes should be identified. In addition, second layer is fixed while first layer is free. On the other hand, two sided crossing minimization is done while there is no fixed layer. In this approach, problem is more difficult to solve.

Chapter 4

Crossing Minimization and Biclustering

No writer should minimize the factor that affects everyone, but is beyond control: luck.
John Jakes

We mention that *Bipartite Crossing Minimization* and *Biclustering* are both NP-Hard problems.

Biclustering problem is related with bipartite crossing minimization problem; in essence, two problems are related. Biclustering is aiming a structure which is the largest possible biclique. Bipartite crossing minimization is aiming to make an arrangement of free layer of graph in order to minimize crossings. Let, there are two complete biclique, B_1 and B_2 in a graph. If we run crossing minimization heuristic (*CMH*), *CMH* will avoid crossings of edges over B_1 and B_2. Meanwhile, *CMH* decides the most possible place for each node on a free layer. If we run biclustering algorithm, we may find B_1 and B_2 as perfect biclusters. But, assumption is that we are using unit weight graph.

However, we are taking an input as weighted bipartite graph, and running weighted bipartite crossing minimization. There is no ideal proof that weighted biclique is also be a bicluster. Since we don't know the correlation and heuristics for weighted bipartite crossing minimization are not considering correlation as biclustering did. On the other hand, we believe that weighted biciliques should be grouped in the graph. In order to explain this claim, we designed a pre-experiment. According to this experiment, we implant two weighted biclique inside the bipartite graph. These cliques are the half of the graph at each experiment. We made 10,000 times by creating bipartite graph starting with

10x10. At each 2,000 experiments, we multiply the size of the graph with 2. Also, at each experiment, we implant random edges with random sized in to the graph. Then, we permute the graph and run CMH such as WOLF. As a result of that, at 9,919 try out of 10,000 experiments, there is no change in clique positions and node positions as well. This experiment shows us that CMH results with weighted bicliques which may be biclusters if they are correlated.

Theorem 4.0.1 ([22]). *Fixing the columns of M, Phase-1.1 orders the rows in such a way that the weighted crossings in the resulting bipartite graph is at most three times the optimum.*

According to these theorem, we are expecting that by applying CMH over the matrix form of the Graph G, we obtain 3 times optimum result.

Chapter 5

Localize-and-Extract Biclusters

An algorithm must be seen to be believed.
Donald Knuth

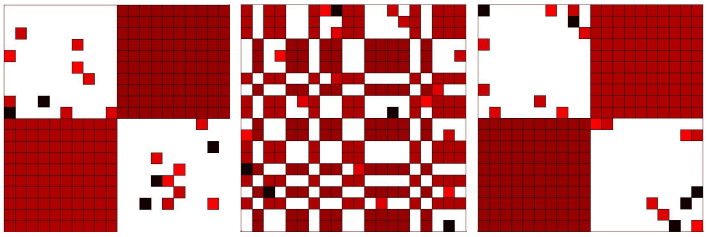

Figure 5.1 Assumed noise is 0.05.(a)Initial artificial design with 2 biclusters of $K_{10,10}$;(b)Without noise removal;(c)Our complete algorithm

Our algorithm, Localize-and-Extract Biclusters(LEB), consists mainly of three steps. *Initial placement* phase applies a two-sided crossing minimization on the weighted graph until there is no change on the node orders. To do this we employ algorithm `3-WOLF` of [22] (one-sided crossing minimization procedure) repeatedly, each time alternating the fixed layer. The pseudocode is given at Algorithm 1.

Algorithm 1 Initial Placement: Input is bipartite graph G with layers L_0, L_1.

Require: Graph G, Layer L_0, layer L_1
  while no change in node positions **do**
    `/*run Weighted One Layer Free approach */`
    free layer A and fix layer B
    run Weighted CMH
    free layer B and fix layer A
    run Weighted CMH.
  end while

We have verified that if the input data is noise-free then this initial placement is usually enough to identify bicliques and extract the biclusters. However, in real data, finding biclusters is not easy because of high level of noise. In order to deal with noise, we proposed *Adaptive Noise Hiding* phase. *Adaptive Noise Hiding* phase removes the weighted edges in the graph that correspond to noise in the original input data. Sliding a window around the perimeter of each node pair, where $(i \pm 1, j), (i, j \pm 1), (i \pm 1, j \pm 1)$ constitutes the perimeter of a pair (i, j), we check whether the window satisfies a threshold density in terms of the number of nonzero weight edges. If it does not, the pairs on the perimeter are considered *suspicious*. The pseudocode of ANH procedure is given at Algorithm 2.

Algorithm 2 Adaptive Noise Hiding: Input is bipartite graph G with layers L_0, L_1.

Require: k, the maximum threshold density
  while $n \leq k$ **do**
    for all pairs (i, j) where $i \in L_0, j \in L_1$ **do**
      neighboring pairs $= (i \pm 1, j), (i, j \pm 1), (i \pm 1, j \pm 1)$
      count = number of pairs with nonzero weight;
      if *count* $\leq n$ store those pairs as *suspicious*;
    end for
    Find the *most suspicious* weight
    Hide the pairs (i, j) with weight equal to the most suspicious
    run Two-sided-crossing-minimization
    if no *suspicious* nodes then $n++$;
  end while

Once sliding is finished we find the *most suspicious* weight and remove all the suspicious pairs with that weight. We adaptively apply our two-sided crossing minimization on the new graph and continue noise hiding after incrementing the threshold density. The removal of the *suspicious* edges and the crossings couple each other in terms of noise removal. Each time the partitions of the graph are reordered to reduce crossings, new suspicious pairs are created. The simple run showcase is in Figure 5.1. Once the noise removal phase is over, we finally gather the biclusters by applying a procedure similar to the one described in [8] and weighted version is called as *Bicluster Extraction Method*. We note that different from previous approaches we directly apply weighted

crossing minimization on the original input data, not to lose possibly important data that can not be considered noise. Secondly our application of the crossing minimization is two-folds. Besides providing a good initial placement, crossing minimization is also used to handle noise removal.

5.1 Bicluster Extraction Method

Algorithm 3 Extracting Biclusters: Input: bipartite graph G with layers L_0, L_1.

Require: M ,matrix representation of graph G
Require: α as size of bag as square submatrix
  Divide M into the bags of size $\alpha x \alpha$, store in $BagList$
  for all bags in BagList **do**
    GenericEvaluation()
    Remove bag according to evaluation
  end for
  while no unmarked *bag* in $BagList$ **do**
    Choose one evaluated *bag* from $BagList$
    Mark *bag* as bicluster
    for all bags b in x coord of marked *bag* **do**
      if GenericEvaluation(b, *bag*) == true **then**
        Expand bicluster with the bag
      end if
    end for
    for all bags b in y coord of marked *bag* **do**
      if GenericEvaluation(b, *bag*) == true **then**
        Expand bicluster with the bag
      end if
    end for
    Define Cluster Boundaries and mark all bags inside the bicluster
  end while

Once the localization phase is over the next step in LEB is to extract biclusters form the reorganized data matrix by considering local neighborhoods. For this method, we defined two parameters as γ and α. In order to get stable biclusters, We divide the adjacency matrix form of data into bags according to given bag size as αxα. For example, consider the input with 100x100 that has 100 nodes in both layer A and B. If α is 10, we divide 10x10 bags of the input. Total bag number is 100 in that case.

We do a traversal over the bag set and try to enlarge the bags. All bags are unmarked initially. At each iteration during the traversal we first mark a bag

b_i. We compute the evaluation score of b_i with the next unmarked bag b_j in the x-direction and construct their union if the score satisfies the γ.

We mark b_j and continue with the union as the current bag. Once we check over all bags in the x-direction, we continue with the y-direction and follow the same procedure. Thus at the end of one such iteration we have the boundaries of a bicluster determined. We continue the traversal starting from an unmarked bag and follow the same enlargement procedure. Once all bags are marked we are left with nonoverlapping, different sized bags corresponding to biclusters all of which satisfy the γ score.

5.1.1 Evaluation Score

For constant biclusters the evaluation score of b_i, b_j is the ratio of the number of most frequent weight in the union to the size of the union. A score satisfies the constraint if it is larger than γ. We note that we have an additional initial traversal step for constant biclusters where we remove bag b_i that scores lower than γ when evaluated with the empty set.

On the other hand for coherent biclusters we first define the H-value of a submatrix [11]. Assume the submatrix consists of I rows and J columns. The residue R of an entry (i, j) is

$$RS_{I,J}(ij) = a_{ij} - a_{Ij} - a_{iJ} + aIJ \tag{5.1}$$

$$a_{Ij} = \frac{1}{J}\sum_{j=0}(a_{i,j}) \tag{5.2}$$

$$a_{iJ} = \frac{1}{I}\sum_{i=0}(a_{i,j}) \tag{5.3}$$

$$a_{IJ} = \frac{1}{|I||J|}\sum_{i=0,j=0}(a_{i,j}) \tag{5.4}$$

where a_{iJ} is the mean of row i, a_{Ij} is the mean of column j and a_{IJ} is the mean of the submatrix. H-value is defined as

$$H_{I,J}(i,j) = \frac{1}{|I||J|} \sum_{i=0,j=0} (RS_{I,J}(i,j)^2) \tag{5.5}$$

The evaluation score of two bags is the difference between their H-values. A score satisfies the γ constraint if it is less than γ in this case.

5.2 Running Time

In order to calculate total running time, we have to know the running time of weighted bipartite crossing minimization heuristics. Given a bipartite graph $G = (L_0, L_1, E)$, first of all algorithm `3-WOLF` 3-approximates WOLF in time $O(|E| + |L_0| + |L_1| \log |L_1|)$ [22]. In addition, the approximation ratio of the barycenter method applied in OLF settings is $\Theta\left(\sqrt{|L_0|}\right)$ [15]. Furthermore, both `W-BARY` and `W-MEDBARY` run in linear time. The running time of `W-GRE` is $O(|E|^2 + |L_1|^2)$ and that of `W-PM` is $O(|E|^2 + |L_1|^4)$ [22]. Both `W-GRE` and `W-PM` require the computation of a cross table. All c_{uv} values are retrieved from this table which is computed beforehand. A straightforward implementation of this computation requires time $O(|E|^2)$.

As a result, running time will change according to selected heuristics. If chosen heuristic is WOLF, running time of *Initial Placement* is the same of running time above.

Running time of *Extracting Bicluster Procedure* is $O(|N|)$, where N is the number of bags and equals to $\frac{|m|*|n|}{\alpha}$. Since each bag is marked in order not to pass again, each bag is passed once so running time depends on number of bags.

Theorem 5.2.1. *The running time of the algorithm LEB is* $O(|N| * |\pi| + |m| + |n| \log |n| + \frac{|m|*|n|}{\alpha})$.

Proof. Running time of WOLF is $O(|\pi| + |m| + |n| \log |n|)$ [22] where $|\pi|$ is total number of entries in data matrix, m and n are the sizes of dimensions. Running time of noise hiding procedure is $O(|\pi| * |N|)$ where $|\pi|$ is total number

of entries and N is number of iteration. The experimental upper bound of N is no more than 25. So the total running time is $O(|\pi|)$. Running time of extraction procedure is $O(\frac{|m|*|n|}{\alpha})$ where α is bag size, and $|m|$ and $|n|$ are the size of dimensions of the data matrix. Since each bag is marked once, the running time is total number of bags.

Chapter 6

Experiments

To consult the statistician after an experiment is finished is often merely to ask him to conduct a post mortem examination. He can perhaps say what the experiment died of.
Ronald Fisher

During the experiment part we have used, Pentium 4 with 3.2 GHZ computer with 1 GB of RAM. Before go into the topic, we can divide experiments into two part, *Experiments on Artificial Data* and *Experiments on Real Data*. Furthermore, we are using LEDA C++ Library in order to implement the details of our algorithm [25]. LEDA is suitable for implementation with structures and implementation support in C and C++. In order to run other algorithms, we have use Bicustering Analysis Toolbox (BicAT) [9].

6.1 Setting for Artificial Experiment and Evaluations

Settings:In order to maintain experimental results in artificial data testing part, we defined *the noise ratio* α. *The noise ratio* α is the parameter of input that determines the noise number inside the data. For example, for $m * n$ input, if the β is 0.01, then the total number of noise should be, $\frac{m*n}{100}$. Additionally, we create input matrix with constant n biclusters with constant values and size of $m * m$ matrices. Since our difference is weighted input, we give weight values for each implanted biclusters. Then, we add the noise according to given β value. Finally, we permute the input matrix. As a result, we get almost random input with weighted values and we can run algorithms over that input. We have chosen

5 different algorithms for comparison. These are BIMAX [10], CC [11], ISA [7], OPSM [3].

Evaluation Scores:Cover ratio is the ratio that for each weighted value in input matrix, we are looking that is there any biclusters that covers the values inside the matrix. It is the ratio of covered ones over all logical entries in input matrix. Also, one of the vital part of biclustering is the quality of found biclusters. To do evaluation we used the formula named *H-score* is given at Equation 5.5. In this equation, $H_{I,J}$ is H-value, I and J are bicluster dimensions, $RS_{I,J}$ is residual score. Lower H-scores means that there is a correlation inside the bicluster.

6.2 Experiment on Artifical Data

In this section we artificially implant 10 constant valued biclusters into the input matrix. In this experiment, we have 100x100 input data matrices with 10 implanted biclusters with the size of $K_{10,10}$.

In addition, noise ratio represents the number of different independent valued gene-condition pairs. We tested with noise ratios,β,*0.001,0.005,0.01,0.1,0.15*. For this section, each setting parameters of the used algorithms are for LEB $density = 0.6, \alpha = 2$, for BIMAX $Dscrzt = 0.2, \alpha = 2, \beta = 2$, for CC $\rho = 13, \delta = 0.5, \alpha = 1.2, outputBiclusters = 10$, for ISA $\rho = 13, t_g = 2.0, t_c = 2.0, n = 100$, for OPSM $\gamma = 10$.

6.2.1 100x100 Experiment

We have done the experiment for the artificial data with 100x100 input data and Our algorithm LEB did well at 100x100 data matrix with artificially implanted 10 $K_{10,10}$ constant biclusters. With this experiment we are questioning the performance of the algorithms over constant bicluster extraction. Since perfect constant bicluster should have H-value with 0, during the experiment, we are expecting the low *H-value* results from the biclusterings of each algorithm. In Figure 6.1-a, we see that CC and OPSM are not performing well. That is the reason of their evaluation to the biclusters. They are performing well in

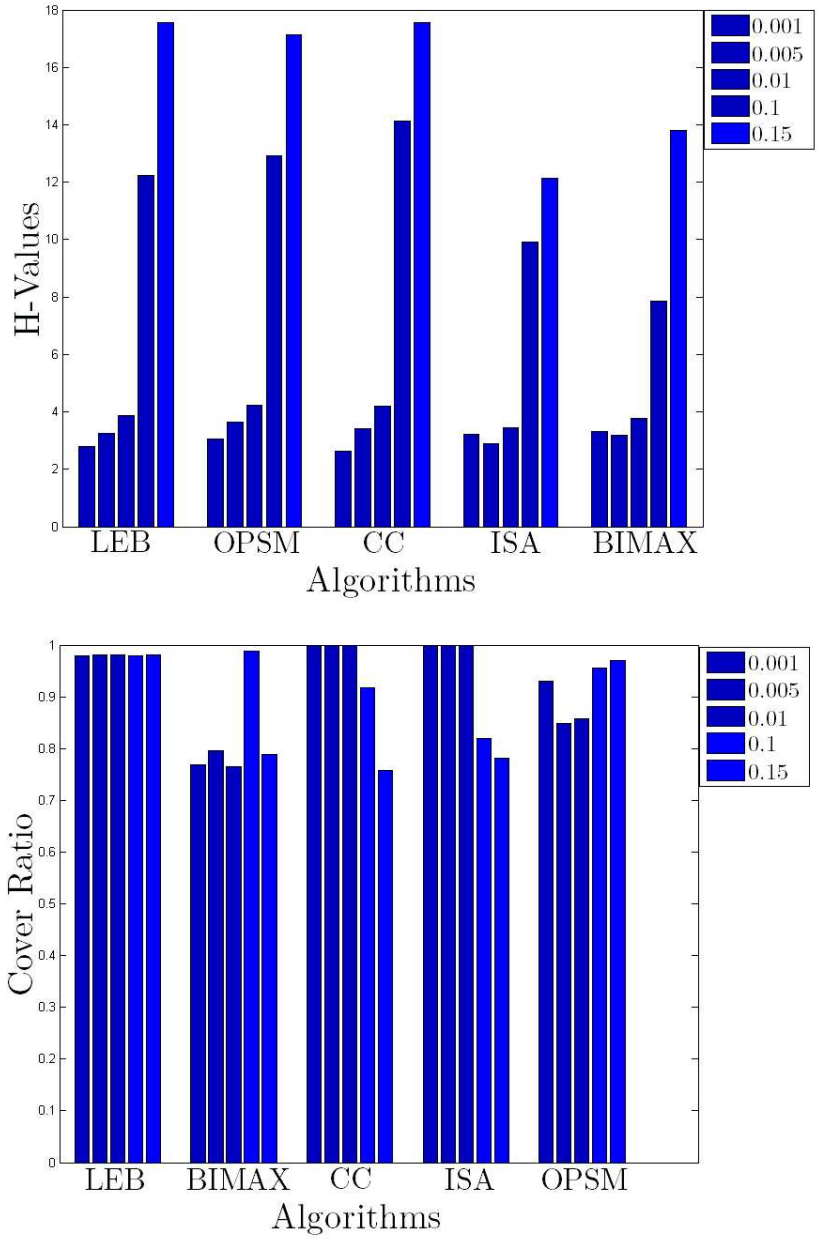

Figure 6.1 (a) H-values of each algorithm, on artificial test data with noise ratios, *0.001, 0.005, 0.01, 0.1, 0.15* . Data is 100x100 matrix with 10 constant $K_{10,10}$ biclusters at the beginning; (b)Covered gene and condition ratio of each algorithm, on artificial test data with noise levels, *0.001, 0.005, 0.01, 0.1, 0.15* . Data is 100x100 matrix with 10 constant $K_{10,10}$ biclusters at the beginning.

coherent type of biclusters. LEB has also fair performance with low H-values. This implies that we can run on constant valued biclusters. ISA and BIMAX have similar performance and when noise level increases, they perform better. This may imply, that they are noise robust. However, their problem is that they are finding many biclusters and since we don't restrict the bicluster sizes, small sized ones may change the H-value positively.

In Figure 6.1-b, we notify that, our algorithm covers almost all gene-condition pairs. Also, cover ratio for ISA and CC decreases with increasing noise ratio because they could not find all biclusters. Since we have significant implanted biclusters because of their sizes, the gene-condition covering should be high in lower noise levels. High cover ratio for LEB at low noise levels implies that LEB successfully determines the constant biclusters.

Indeed, our algorithm has the property of increasing H-values from sparse input to dense. Since we apply noise hiding procedure, we believe that in low noise ratios, we are working well. Also, at high noise ratios we could compete with ISA and CC. Finally, LEB almost covers all gene-condition pairs and does it with stationary cover ratio.

6.2.2 200x200 Experiment

For this experiment we try to run LEB on bigger size such as 200 in both dimension 1 and 2 and there are 20 $K_{10,10}$ implanted biclusters. Again, we have noise ratios and they are *0.01, 0.05, 0.10, 0.50* . We experimented two different run of LEB in Table 6.1. We tested and compared with BIMAX, CC, ISA and OPSM. We could not use XMOTIF since chip size is larger than 64. In Table 6.1, there are two main rows representing two different α values. According to these results, H-values of each algorithm are similar. For the α values of 2 we see that LEB gives fine low H-values than others in low noise ratios. Nevertheless, LEB has a problem with high noise ratios. The reason of that is LEB *density* parameter. Since it is too low (0.5) for the bag size $alpha = 2$, bags should differ at high noise ratios. In addition, at the second main row of Table 6.1, α is 3 and for this value LEB gives better H-values. The

Table 6.1 Artificial Dataset 200x200

Whole run of the algorithms on 200x200 dataset					
Treatment 1		LEB *density* = 0.5			
		0.01	0.05	0.10	0.50
LEB $\alpha = 2$	LEB	1,55	9,30	12,68	15,46
	OPSM	5,01	7,82	11,36	22,15
	CC	5,72	2,96	3,77	11,04
	ISA	0,07	2,84	6,59	18,64
	BIMAX	0,00	0,00	3,64	3,78
Treatment 2		LEB *density* = 0.8		LEB *density* = 0.9	
		0.01	0.05	0.10	0.50
LEB $\alpha = 3$	LEB	5,22	7,34	1,61	2,01
	OPSM	4,74	6,52	10,77	21,07
	CC	3,44	3,36	3,32	9,79
	ISA	0,66	2,26	6,44	17,54
	BIMAX	5,87	4,46	3,18	3,80

reason of that is because of *density* parameter. We tried two different densities. For noise ratios, *0.01 and 0.05*, *density* is 0.8 and for *0.10 and 0.50* is 0.9. Since we are extracting constant ones, 0.8 and 0.9 densities support LEB results to be constant. The problem of high *density* parameter is under extraction. For instance, at the noise ratio *0.50*, algorithm could not extract biclusters because of high *density* parameter. But we are not expecting to deal with such noise in real environment. Furthermore in Figure 6.2, you could see the resulting run of LEB. In these figures, you can see the input, input permutation and output. Note that, the resulting figures don't include the extraction part. The settings for this figure are "Assumed noise is 0.01(a)Initial artificial design with 20 biclusters of $K_{10,10}$;(b)Without noise removal;(c)Our complete algorithm", "Assumed noise is 0.05(c)Initial artificial design with 20 biclusters of $K_{10,10}$;(d)Without noise removal;(e)Our complete algorithm", "Assumed noise is 0.10(f)Initial artificial design with 20 biclusters of $K_{10,10}$;(g)Without noise removal;(h)Our complete algorithm", "Assumed noise is 0.50(i)Initial artificial design with 20 biclusters of $K_{10,10}$;(j)Without noise removal;(k)Our complete algorithm".

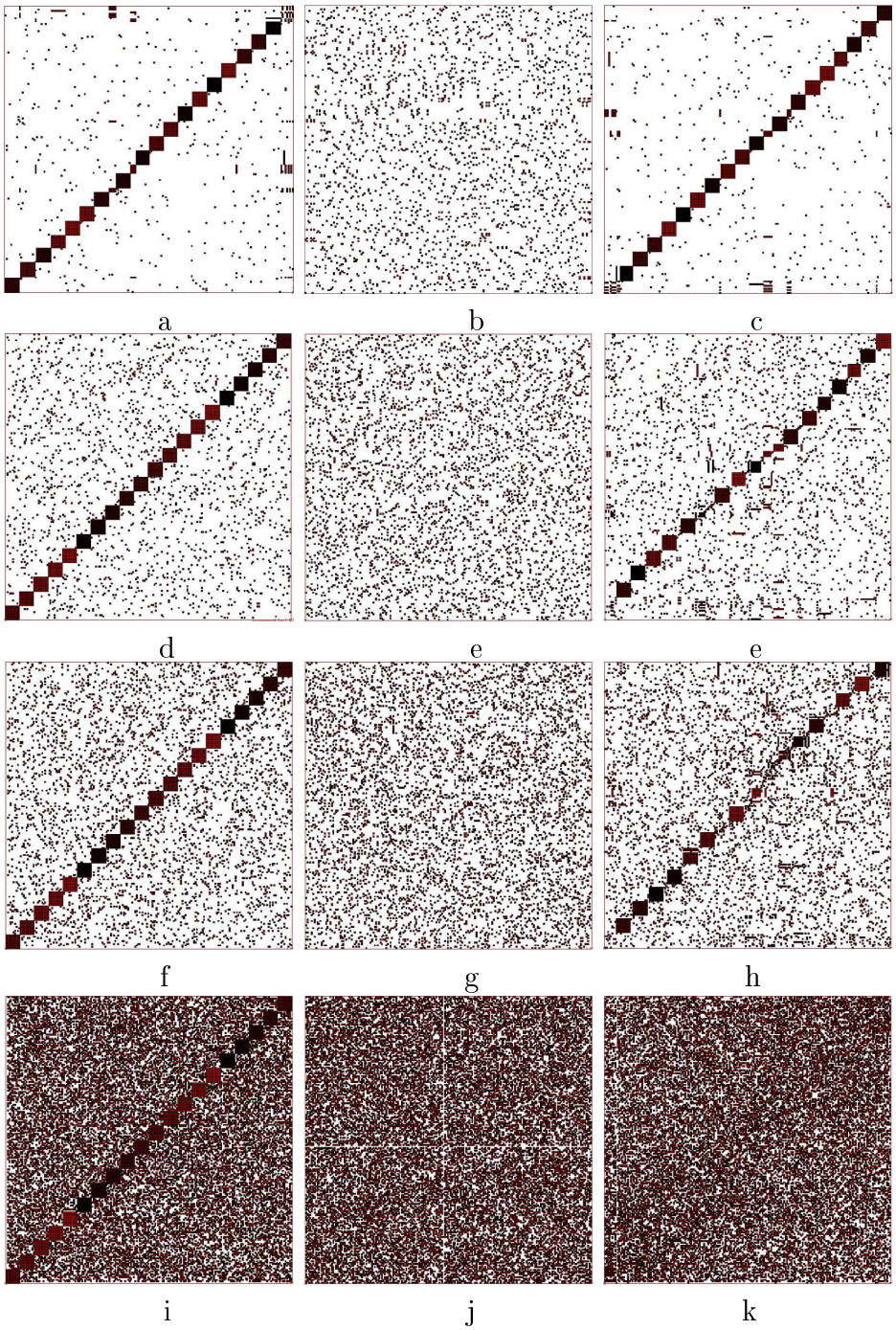

Figure 6.2 Figures for 200x200 Experiments

6.3 Experiments on Real Data

We have tested our algorithm on two dataset,*Arabidopsis thaliana*[26] and *Saccharomyces cerevisiae*(Yeast Cell Cycle)[11]. For thaliana dataset, we tested the cover ratio, bicluster sizes, h-values and for yeast dataset, we tested the enrichment ratio of each selected algorithms.

6.3.1 Arabidopsis Thaliana

We have data set of *Arabidopsis thaliana* from [26]. *Arabidopsis thaliana* is a widely used plant for bioinformatics applications. It was the first plant that is sequenced with its whole genome. The database we maintained is with 734 genes and 69 conditions. Over this database we have done several tests.

Table 6.2 Arabidopsis thaliana dataset experiment 1

Whole run of the algorithms on *Arabidopsis thaliana* dataset						
Alg.	Max Bic.	Min Bic.	Dim1 Avrg.	Dim2 Avrg.	Bic. Number	H-values
LEB	630x66	2x4	237,25000	17,5000	15	402,6922
BIMAX	10x4	4x4	4.428301	4.065273	50189	963,3856
CC	690x69	242x69	531.4200	69.00000	100	3660,31
ISA	234x5	198x2	220.3250	2.537500	80	2349,431
OPSM	2x57	10x5	10.33333	15.41667	12	909,4981

Experiment 1: We have taken the default parameters from BicAT tool at the previous experiment. We checked all the the parameters in a way that they have been used in their original paper. Settings are for LEB $\gamma = 10, \alpha = 2$, for BIMAX $Dscrzt = 0.6, \alpha = 4, \beta = 4$, for CC $\rho = 13, \delta = 100, \alpha = 1.2, outputBiclusters = 100$, for ISA $\rho = 13, t_g = 2.0, t_c = 1.0, n = 500$, for OPSM $\gamma = 100$. As a consequence of *Arabidopsis thaliana* experiment, CC and ISA performed well in that setting in terms of number of output biclusters. But, one problem is that the size of biclusters. For example, for CC, max bicluster is almost the size of data, and for all biclusters, algorithm gives us

the whole dimension 2 that is 69. In addition, ISA has a problem in dimension 2. Dimension 2 of ISAs biclusters is not sized as others. Finally, for this experiment we change BIMAX parameters. We increase discreatization value to 0.6, and also we increase $\alpha and \beta$ values. In that setting, BIMAX finds many biclusters, but there is no biclusters more than size of *10x4*. On the other hand, LEB, ISA and CC are finalized with fine dimension sizes.

In addition, H-values are important criteria to determine biclusters. LEB has the lowest H-value scores. This means that we have the biclusters with best correlation. You can also see the resulting figures of two significant biclusters at Figure 6.3. According to looking at that figures, LEB catches highly correlated biclusters. There are some peaks inside the plots. These may be noised values. We applied noise hiding procedure, but data is so dense. Because of that, noise hiding may not be effective. Also, in some part of the plot, they are symmetric and could not be a noise. Because there are similar patterns around. These values should be correlated with itselves.

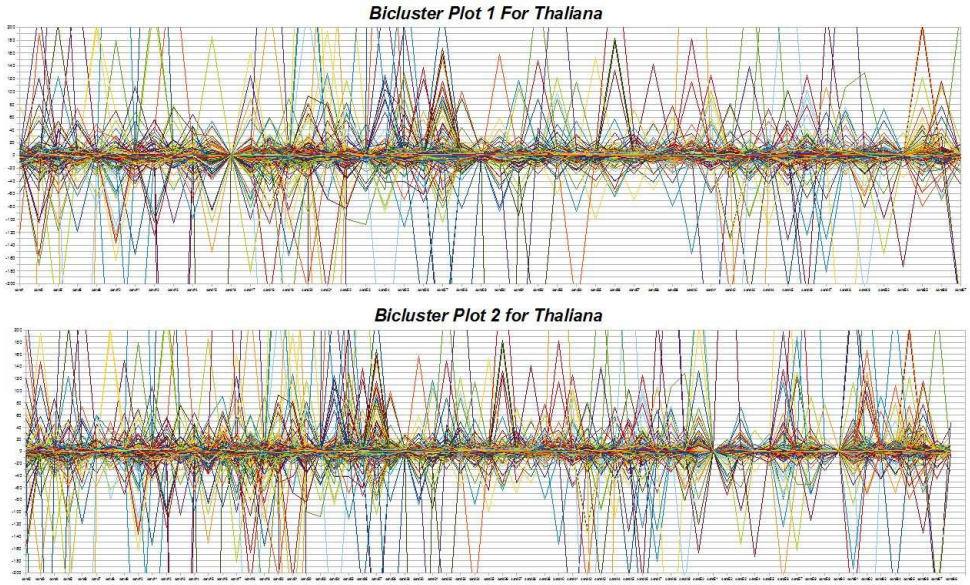

Figure 6.3 Thaliana Bicluster plots for LEB($\gamma = 10, \alpha = 2$),conditions at X-axis

Discussion on Parameters of Arabidopsis thaliana experiments: First of all, during experiments, we have experienced that BIMAX has a problem when the discretization parameter when it is low. For instance, with 0.2 value of that

parameter, it finds millions of biclusters. It is not easy to analyze millions of biclusters. It is obvious that biclusters are permuted. It has also running time problem. It takes more than several days, so we have to interrupt the experiment. Moreover, OPSM does not give fine results when we change its parameters. It is the worst one in *Arabidopsis thaliana* experiment.

In addition, our parameters, α and *gamma* could be used to determine correlation level. Choosing α smaller such as 3 should be better in order to catch small biclusters. For large datasets with shorter range of weights, α parameter can be larger than 3. Also, by changing α values you could catch different bicluster results. One who runs LEB by changing α values, could catch different biclusters. Furthermore, γ value is necessary in order to identify coherence. Smaller *gamma* values should give better correlated biclusters.

6.3.2 Yeast

As a second real data, we selected *Yeast Cell Cycle (S.cerevisiae)* dataset [11] that is widely used in biclustering applications (http://arep.med.harvard.edu/ biclustering/yeast.matrix). It has 2884 genes and 17 conditions. In most of previous papers, they used that dataset to test their algorithms. Since *Yeast* dataset was categorized in terms of functionality of each genes at MIPS(http://mips.gsf.de/ genre/proj/mpact/yeast), we are able to test enrichment ratio of each bicluster by looking at genes. We designed a experiment as Bryan et.al. [27] did. We identified categories of genes. There are 13 pre-identified categories. During the experiment, we find enrichment ratio as a number of genes at the most specific category in bicusters over total number of genes in bicluster. This is a ratio between 0 and 1. In Table 6.4, you can see the functional enrichment values. During this experiment we extracted small biclusters such as smaller than 40 genes. Indeed, according to that table, OPSM fails to enrich biclusters. LEB has 7 wins. CC has 3 wins. OSPM has 1 win. Wins are determined according to best functional enrichment values in each category. Finally, CC and LEB has a one draw among them and all of them has one draw at category B. Settings of CC and OPSM are default parameters given at their papers and for LEB, setting is $\gamma = 100, \alpha = 4$. In this test, OPSM failed because it has high dimension sizes. Adding that, CC competes with our algorithm as it has 100 biclusters and we have also 42 biclusters. In Figure 6.4, you can also see the graph of enrichment values for each category. In that figure, we append 3 different run of LEB with different parameters. Indeed, we have still have advantage over CC and OPSM for different settings.

Secondly, we do not run BIMAX and ISA for that experiment. The reason is that, ISA does not give any biclusters although we tried different parameters. In addition, BIMAX has a problem of number of biclusters and duplicate results. There are so many biclusters but many of them are duplicate of another. That is surely overlapping problem. Furthermore, parameter selection is a issue. Defining dimension sizes and giving discretezation parameters give advantage

Table 6.3 Yeast Dataset Experiment 1

Warfield for OPSM, CC, LEB on *Yeast 2884x17* dataset for each category						
	OPSM		CC		LEB	
Functional Catogory	ORF in Bicluster	Func. Enrich.	ORF in Bicluster	Func. Enrich.	ORF in Bicluster	Func. Enrich.
E - Energy Production	543	0,03	55	**0,04**	100	**0,04**
G - Amino Acid Metabolism	1282	0,03	51	0,04	186	**0,05**
M - Other Metabolism	62	0,11	59	0,14	79	**0,22**
P - Translation	2342	0,03	57	**0,19**	79	0,09
T - Transcription	1282	0,06	55	**0,19**	143	0,08
B - Transcriptional control	124	**0,08**	42	**0,08**	152	**0,08**
F - Protein Fate	2342	0,06	51	**0,14**	83	0,08
O - Cellular Org.	2342	0,04	51	0,10	46	**0,11**
A - Transport and Sensing	124	**0,13**	59	0,07	143	0,10
R - Stress and Defense	62	0,06	42	0,05	105	**0,06**
D - Genome Maintenance	62	0,11	51	0,10	293	**0,14**
C - Cellular Fate / Org.	196	0,47	51	0,47	111	**0,48**
U - Uncharacterized	124	0,06	51	0,08	79	**0,13**

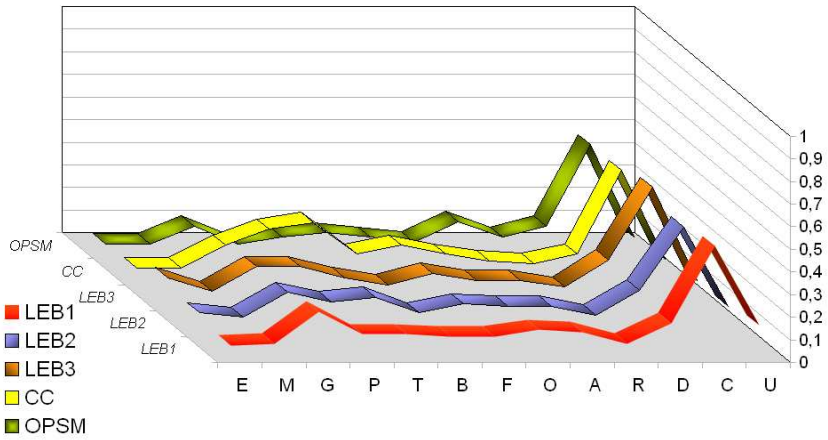

Figure 6.4 Yeast Results for
OPSM,CC,LEB1($\gamma = 100, \alpha = 4$),LEB2($\gamma = 50, \alpha = 3$),LEB3($\gamma = 25, \alpha = 3$)

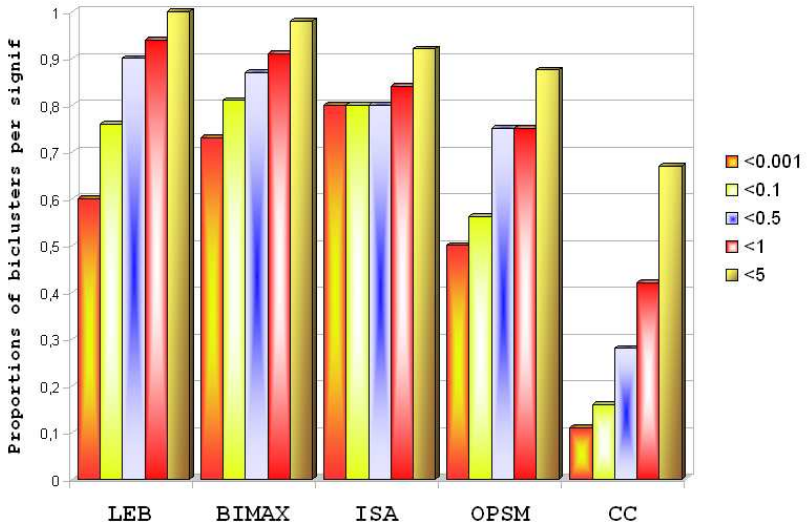

Figure 6.5 Proportion of biclusters significantly enriched by any GO biolological category of Yeast(*S.cerevisiae*) for (LEB $\gamma = 10, \alpha = 2$),BIMAX,ISA,OPSM,CC

to BIMAX itselves to permute biclusters. Indeed, we decided not to run BIMAX and ISA for the Yeast Cell Cycle data set but we are giving the results of these from [27, 10, 28] for better comparison on hand. Furthermore, we used FuncAssociate tool [29] (http://llama.med.harvard.edu/ cgi/func/funcassociate) in order to measure with GO accepted categories. FuncAssociate computes the hypergeometric functional score by using "Fisher's Exact Test". According to results, we are good at covering genes with large amounts with high hit ratio inside the bicluster. In addition we have given the Figure 6.5. According to this figure, comparing with [27, 10, 28], on yeast (*Saccharomyces cerevisiae* dataset, we are performing similar and better bar graphs as seen on these paper and their algorithms. For each significance level, we give a enrichment ratio of our biclusters. The value in consideration as α is adjusted p-values gathered from FuncAssociate tool [29]. According to adjusted p-values, we have increasing enrichment ratio when significance level increases as expected. Also, we have the best results among three algorithms. Our week point is lower ratio at the first enrichment category. The problem of CC is that it can not have p-values smaller. As a result it can not find enriched categories. Also, OPSM has the problem of number of output biclusters. One not enriched bicluster may decrease the enrichment ratio. Finally, LEB does fair results with 50 biclusters and all of them is enriched with ratio lower than 0.5.

Indeed, at *Yeast* data set, our results are fairly good. We are better than OPSM and CC at two experiment environment.

Table 6.4 Yeast dataset experiment 2: FuncAssociate results for LEB

FunAssociate Result for LEB						
Rank	N	X	LOD	P	P-adj	GO Attribute
1	704	3585	0.413	4.1e-37	< 0.001	0005737: cytoplasm
2	423	1876	0.349	3.2e-28	< 0.001	0003824: catalytic activity/enzyme activity
3	92	239	0.592	2.7e-20	< 0.001	0016491: oxidoreductase activity/redox activity
4	122	372	0.491	9.1e-20	< 0.001	0005783: endoplasmic reticulum/ER
5	880	5389	0.429	1.2e-18	< 0.001	0005623: cell
6	33	46	1.172	3.2e-18	< 0.001	0000502: proteasome complex (sensu Eukaryota)/26S proteasome
7	95	294	0.472	4.7e-15	< 0.001	0019752: carboxylic acid metabolism
8	95	294	0.472	4.7e-15	< 0.001	0006082: organic acid metabolism
9	57	157	0.539	9.8e-12	< 0.001	0006066: alcohol metabolism
10	106	393	0.359	5.2e-11	< 0.001	0009056: catabolism
11	61	180	0.493	5.2e-11	< 0.001	0006520: amino acid metabolism
12	31	64	0.748	1.4e-10	< 0.001	0004175: endopeptidase activity/endoprotease/proteinase
13	63	195	0.462	2.4e-10	< 0.001	0006519: amino acid and derivative metabolism
14	40	99	0.609	3.2e-10	< 0.001	0008652: amino acid biosynthesis
15	42	108	0.582	4.8e-10	< 0.001	0044271: nitrogen compound biosynthesis
16	42	108	0.582	4.8e-10	< 0.001	0009309: amine biosynthesis
17	13	15	1.501	1.1e-09	< 0.001	0005839: proteasome core complex (sensu Eukaryota)/20S core complex/macropain
18	98	373	0.341	1.4e-09	< 0.001	0044248: cellular catabolism
19	85	310	0.364	2.1e-09	< 0.001	0009057: macromolecule catabolism
20	65	216	0.417	3.4e-09	< 0.001	0009308: amine metabolism

Chapter 7

Graph Unions

Union gives strength.
Aesop

7.1 Motivation

There have been numerous applications, heuristics and tools over graph visualization. Several techniques depend on these graph based knowledge. Each technique is resulting with layout. Sugiyama layout is one of them. Sugiyama layout is a popular one used in layered graph drawing. Layered graph drawing is designed in steps. These are

Removing Cycles Layer Assignment Adding Dummy Vertices Crossing Minimization Horizontal Coordinate Assignment

Among these steps, many of the problems such as Feedback-arc Set [30], Precedence Constrained Multiprocessor Scheduling [31], 2-layer Crossing Minimization [32], Optimal Linear Arrangement [33] are NP-hard. So we need to use heuristics in order to complete layout.

After choosing the layout, we identify the application areas. Applications such as process scheduling, social network visualization, protein precedence graph and related graphs [34], VLSI (circuit schematics), data flow diagrams, subroutinecall graphs suit to the layered graph drawing approach. Up to now, the previous applications of layered graph drawing using Sugiyama layout runs over unweighted directed graphs. This means that problems at each step are related with the unweighted directed graphs.

Weighted strategy is new area for this topic. If we consider weights of the graph, we need to modify each step considering weights. For some steps, weighted versions of algorithms are in literature. Weighted feedback arcset problem and its heuristics, and weighted 2-layer crossing minimization and its heuristics are usable.

In our case study we are considering the company social network graph in such a way that we have pre-identified relations among workers in a company. Since weighted relations are possible, we can represent the importance of relation by weight. Also, assuming that each worker belongs to a department or project, a natural clustering occurs.

7.2 Related Work

Graph visualization is important topic in computer science. Since there are many related work with the whole topic, we are giving related work with social network visualization.

Recent years, there have been several approaches over social network visualization. Approach are analyzing email communication [35], online social networks [36], and co-authorship networks for scientific publications [37]. There are also several tools for generating social network visualizations and performing analysis and research. These are UCINet [38], GUESS [39], JUNG [40], Vizster [41], Visone [42].

Social network visualization is divided into parts in itself. One approach is aiming to online social networks such as email networks, Facebook,other networking utilities networks. Another approach is to represent terrorist groups and their networks. Approaches are application dependent. They rely on specific properties. For terrorist group network, they rely on real connections

There are also approaches in order to mine communities from signed social network with algorithm FEC [43, 44] uses one of the simplest maximum flow algorithms to cover community. Also Flake et.al [45] tried to extract the data from online network. According to their method, despite the Web's decentralized,

unorganized, and heterogeneous nature, Web self-organizes and its link structure allows them to determine communities efficiently. In [46], they retrieved data from search engine. In [47], they performed on a special dataset and they extracted the information based on web application.

Finally, IBM is proposed a tool on Lotus called ATLAS(http://ibm.com/software /lotus/services) in order to visualize the organization. Their tool is related with our approach. But they are not using layered graph drawing approach. We believe that best suited approach for drawing social network hierarchy could be done with layered graph drawing.

Moreover, in many graph related visualizations, they disregard the edge weights. With our approach we are trying to handle, layered drawing in weighted directed graphs.

Layered graph drawing is one of the main approaches of graph drawing. Its each step there are NP-hard problems. Removing cycles is one of these problems. Feedback arcset problem is approximated with heuristics [48, 49]. In [50, 51], they have performance ratio $O(lognlog(logn))$. Also, [52] proposed weighted feedback arcset problem with worst case $O(mn)$ where m is number of arcs and n is number of vertices. Result after feedback arcset problem complements the problem of Maximum Acyclic Subgraph. There are also older approaches, Fast Heuristic and an Enhanced Greedy Heuristics. However, these are not applicable to weighted graphs.

The second step is assigning layers for each node inside the graph. There are two main approaches: Coffman Graham Layering and Longest Path Layering. Coffman's algorithm is aiming to minimize width of layering. Longest Path Layering is aiming to minimize height of layering. In literature, there is no corresponding version for weighted graphs.

Two-layer Crossing minimization is also another issue after obtained acyclic layered graph. However, there is no applicable algorithm over k-layer crossing minimization, we have to add dummy vertices before crossing minimization. Crossing minimization can be run over bipartite graph. There are techniques

in order to minimize crossings of whole graph. One of these techniques is layer by layer sweep method. With this method one layer is free and neighbor layer is constant at one time, and next, constant layer is freed and its neighbor is constant by sweeping. The process continues until there is no layer remained. Another method is to have two layer free sweeping. Since the problem is NP-hard, it is harder to design heuristics for two layer free problem.

There are numerous algorithms designed for one layer free problem. In [22], we also proposed weighted version of one layer free algorithms, Barycenter as W-BARY, GRE as W-GRE, PM as W-PM, Median as W-MEDBARY and our algorithm WOLF.

Finally, horizontal coordinate assignment is not necessarily vital for layered drawing, it is helpful for neat design and to arrange edge bends.

Chapter 8

Our Method

To accomplish great things we must first dream, then visualize, then plan... believe... act!
Jack Youngblood

Our method is based weighted layered drawing. We are considering the company social graph such that each worker belongs to specific department or departments. That means that natural clustering occurs before we implement our algorithms. If another application area is chosen, we may apply clustering before execution of our method. Since it is possible that there are many departments inside the company, we designed upper level visualizations based on departments and their relations. These relations are represented as in circular layout.

Circular layout is also high level graph. Users could gain information about department relations. There is an edge between two departments, if they have worker(s) shared.

At the low level graphs, to represent the hierarchy of the specific department. Layered layout best suits. For readability aim layered layout help users to understand key points. The problem occurs when there is a relation between two departments. This could be solved by adding a duplicate edge for both of two departments. Furthermore, these edges could be represented in high level graph, representing that there is a relation between two departments.

Each relation in low level graphs is represented with weighted edge that represents the weight information. Since unweighted method of layered drawing do not consider weight, unweighted algorithms work in a way that loosing the weight information. By using weighted versions of these algorithms we take

advantage over to represent weight information. In low level graphs, weight information gives us the importance of the relation. Higher weights mean that there is a important relation between employees.

8.1 Methods in Design

We use Demetrescu et al's algorithm to remove cycles, Coffman Graham Algorithm for layering, WOLF for crossing minimization. Since Coffman's algorithm considers the graph as weighted, we need to change the algorithm in a way that consider weights. Since weights have special meaning, we need to represent weight with high values better. So, each algorithm should execute and give result with better understanding for high weighted values. In Figure 8.1 you can see our tool sample visualization.

8.1.1 Modified Coffman Graham Algorithm

Over the original algorithm of Coffman, we have added some weight modifications. In original algorithm, there are two main phases. In the first phase, we are making an initial ordering. At first, this initial ordering merges duplicate edges by adding their edges. In unweighted case we need to delete duplicate ones. Then for each leaf node, that means no outgoing edges, it gives a numbering that represents a type of lexicographical ordering. After these, second phase begins. At the second phase, actual layering does, according to given constraints that no such layer has a size of nodes larger than predefined W value, and no such node that there is an outgoing edge to the upper layer with k values larger than its layer number. For the second phase, we can also add greedy strategy to collect and assign layer to the most weighted node that means having maximized edge weights. You will also see pseudocode at Algorithm 4.

8.1.2 Demetrescu's Weighted Feedback Arc Set Algorithm

In [52], they proposed an algorithm FAS, in order to deal with cycle removal in weighted directed graphs. It is simple two phase and can be seen at Algorithm 5. At its first phase it catches cycle C until there is no remaining

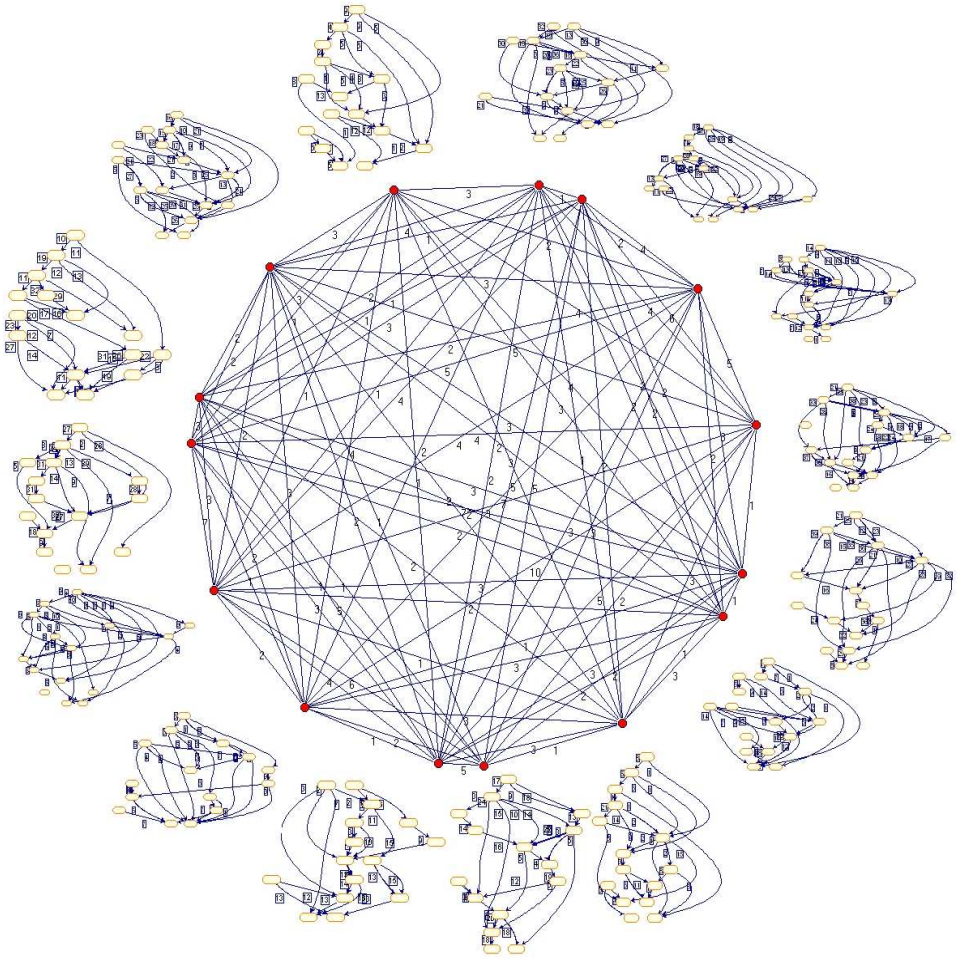

Figure 8.1 Graph Union Showcase

Algorithm 4 Coffman Graham Algorithm with weight modification

Require: Acyclic Directed Weighted Graph G and Width W

```
/*Phase 1:  Initial Ordering*/
for all pairs of nodes u,v of G with edge e do
  if ∃ a different path p then
    merge e with p by adding each weight values
  end if
end for
U set of all unlabeled nodes of G
for all v ∈ U do
  L(v) = ∞
end ford = 1
for i = 1 to n do
  if outdegree(v_i)=0 then
    L(v_i)=d
    d = d + 1
  end if
end for
/*Phase 2:  Assigning layers buttom-up*/
L_1 = ∅
while U ≠ ∅ do
  Choose unlabeled vertex u, such that every vertex in v : (u, v) ∈ E is in U,
  and L(u) is maximized and sum of all edge weights is maximized too.
  if |L_k| < W and for every edge (u, w), w ∈ L_1 ∪ L_2 ∪ ...L_{k−1} then
    add u to L_k
  else
    k = k + 1, L_k = u
  end if
end while
```

one. Among that cycle C, they hide suspicious edge with minimum weight and then they subtract its weight from all weights of the cycle C. They repeat until there is no newer cycle. Moreover, at the second phase, we sort suspicious edges according to their values. From the edge list F, we check that each suspicious edge creates a new cycle in graph if it is unhidden. If it does, it remains hided else it is unhidden. Indeed, all edges in list F reversed in terms of their directions, in order to continue with the next step.

Algorithm 5 Weighted Feedback Arcset Heuristic

Require: Directed Weighted Graph G=(V,E)
```
/*Phase 1:  Suspicion*/
```
$F \Rightarrow \emptyset$ *F is feedback arcset*
while $\exists$ no remaining cycle C **do**
  Let C is a simple cylce in G
  ϵ is minimum wighted arc (x, y) in C
  for all $(u, v) \in C$ **do**
    $w(u, v) = w(u, v) - \epsilon$
    if $w(u, v) = 0$ **then**
      $F \Rightarrow F \cup (u, v)$
    end if
  end for
end while
```
/*Phase 2:  Appeal*/
```
Sort F
```
/*Make sure weight values are descending in list*/
```
for all (u, v) in F **do**
  if $V - (F - (u, v))$ is acyclic **then**
    $F \Rightarrow F - (u, v)$
  end if
end for

8.1.3 Weighted Crossing Minimization

There are alternatives for Weighted Crossing Minimization. In [22] we proposed WOLF, and weighted versions of PM,GRE,Barycenter and Median. Since all of these algorithms don't need weighted modifications, we directly apply them to our method.

8.1.4 Method Review

Indeed for our design we used, Demetrescu et al's algorithm FAS for *Cycle Removal*, *Coffman Graham Algorithm* for layering, WOLF for *Weighted Crossing Minimization*. There are some other steps such as *Adding Dummy Nodes* and *Horizontal Coordinate Assignment*. We add dummy nodes if there is an edge between layers L_i and L_j such that $i - j > 0$. Also, *Horizontal Coordinate Assignment* is sub-method that we don't need to use it since it doesn't effect the visualization of weight information.

Chapter 9

Experiments And Results

Don't be too timid and squeamish about your actions. All life is an experiment. The more experiments you make the better.
Ralph Waldo Emerson

For the testing issue of our design, we randomly create a company working artificially.

As a predefined parameter, we have number of departments as d, number of relations for each department as r, number of workers for each department as n and for the coffman's algorithm we need *width* parameter as w.

We also randomly choose names among name database. In Figures 9.1 and 9.2 there are one sample input graph for layered drawing that is also representing departments. There are also resulting graphs that are the output our drawing method. These are at Figures 9.3 and 9.4. For better understanding of hierarchy and weight information it is apparent that layered drawing best suits for the company social network drawing. In Figures 9.5 and 9.6 you can also see the department related graph with real employee names.

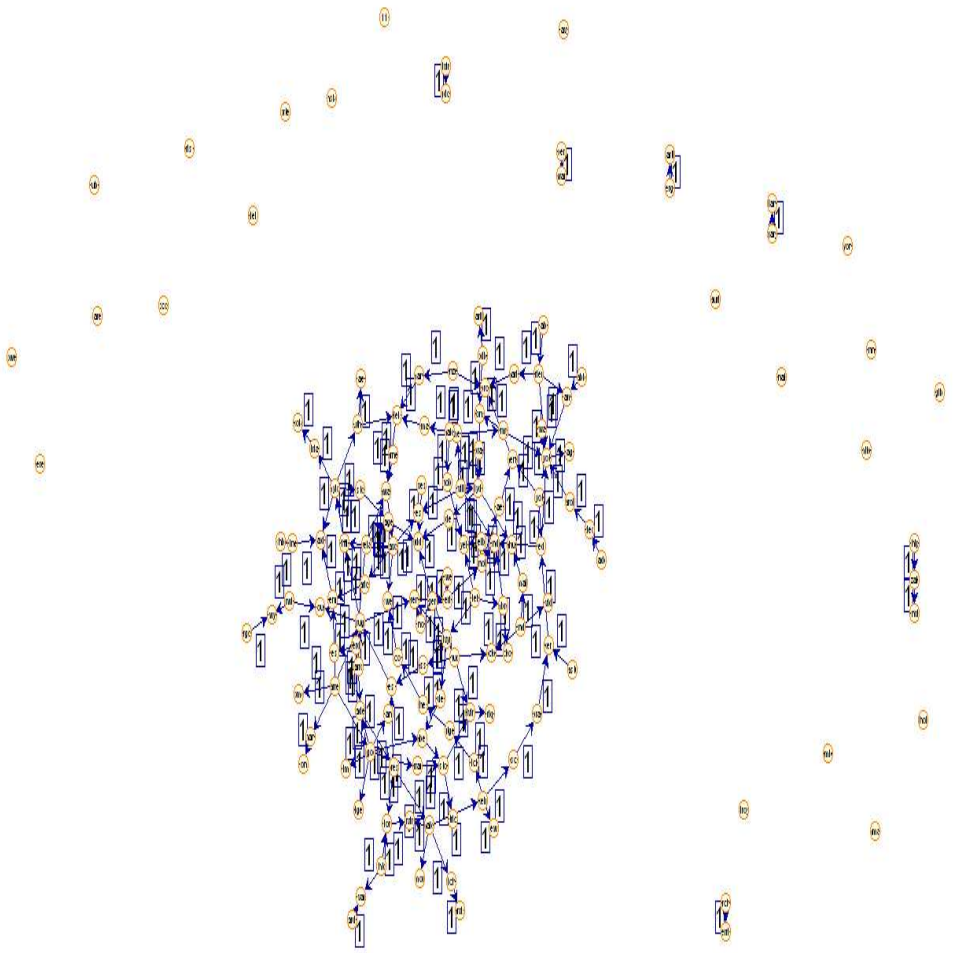

Figure 9.1 Spring 2D embeder for Graph 1 $r = 151, n = 150$

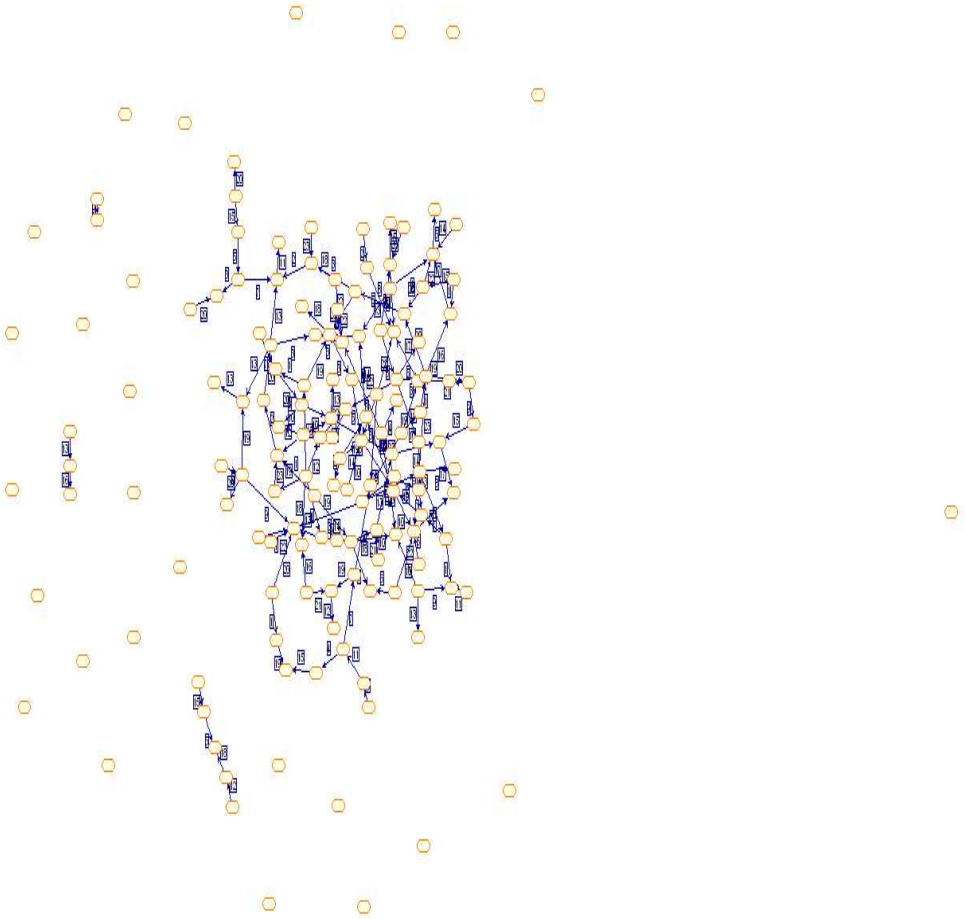

Figure 9.2 Spring 2D embeder for Graph 2 $r = 151, n = 150$

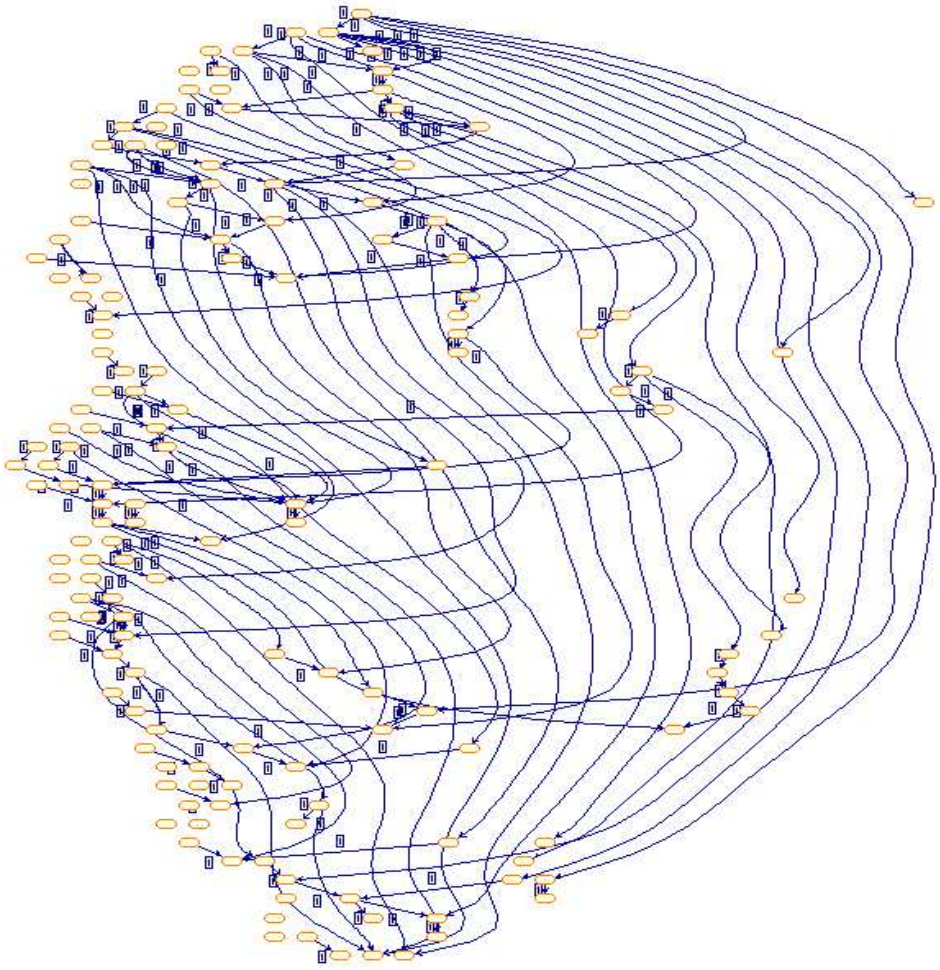

Figure 9.3 Layered Drawing for Graph 1 $r = 151, n = 150, w = 3$

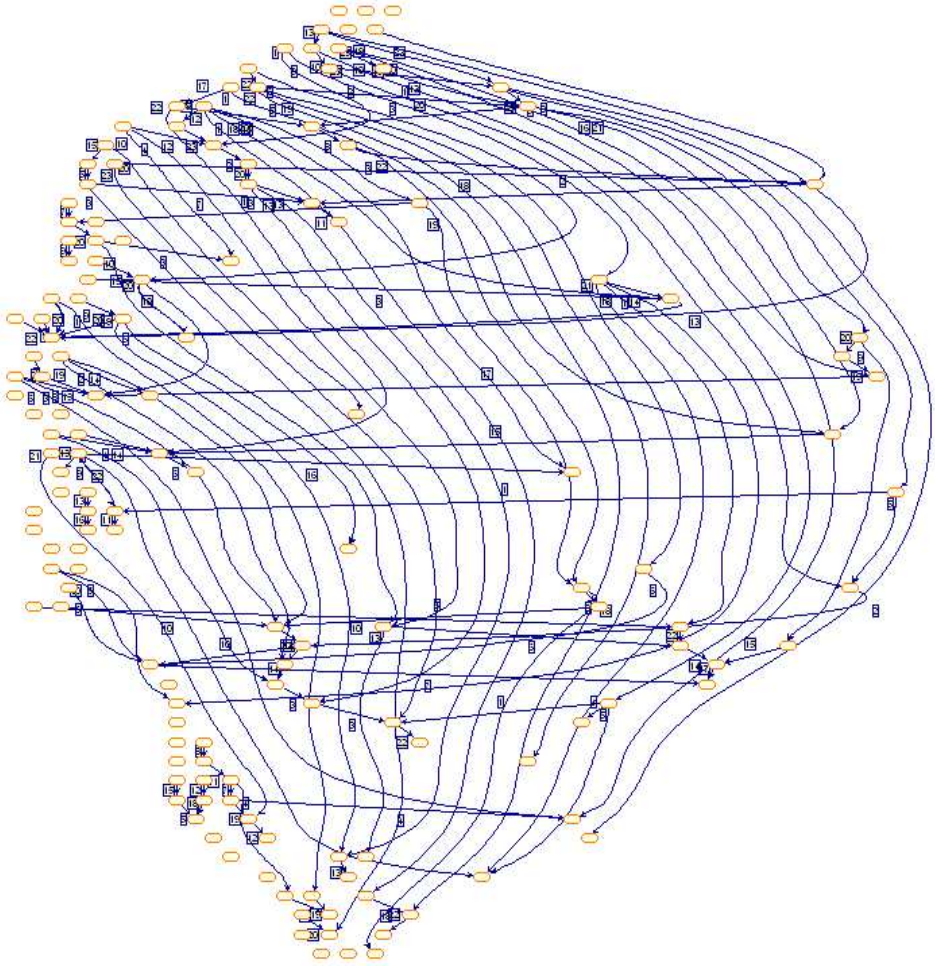

Figure 9.4 Layered Drawing for Graph 2, $r = 151, n = 150, w = 3$

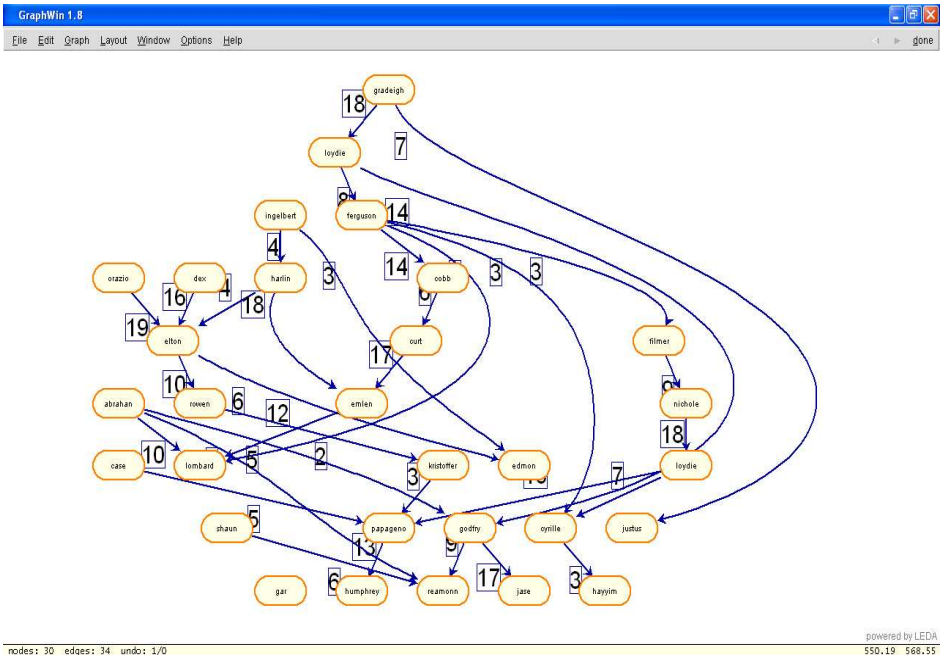

Figure 9.5 Layered Drawing for Graph 1 with employee names
$r = 34, n = 30, w = 5$

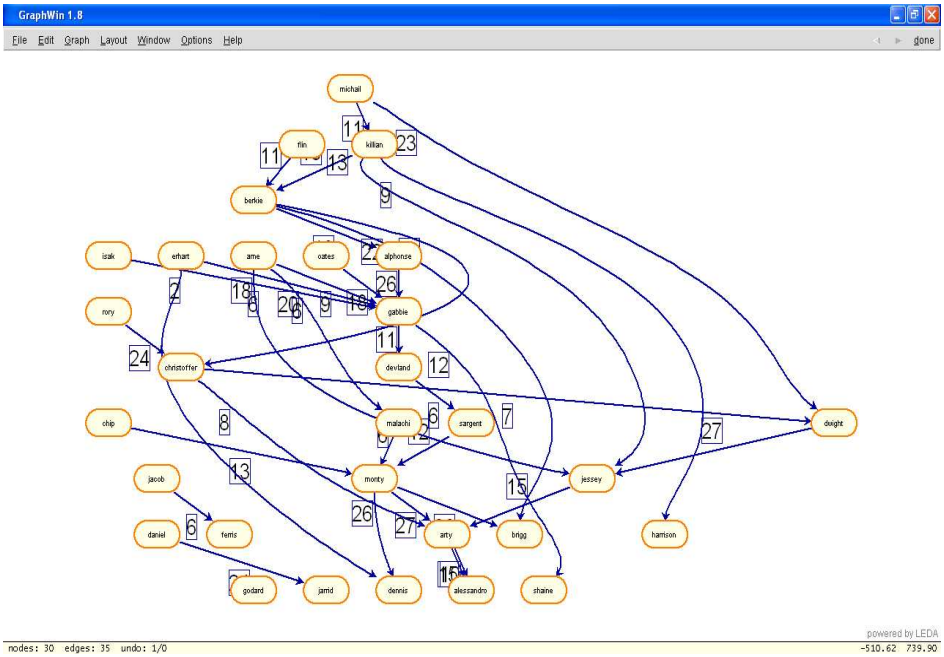

Figure 9.6 Layered Drawing for Graph 2 with employee names
$r = 35, n = 30, w = 5$

9.1 Properties of Designed Tool

1. The tool is simple, and there are available click and drop operations.
2. Tool has high level graph for general perspective and low level graphs for detailed relations. High level graph shows company profile and low level graphs show relations and person interactions.
3. Low level graph is drawn in layered drawing layout that enables user to see the real hierarchy in the departments.
4. Relations are weighted, weight could mean everything. It could represent importance, numbering or counting or even pointer to another file.
5. Shows weight information better.
6. Better Readablity comparing with other layouts.
7. High level graph also represents the information about general relations between department.

9.2 Statistics

9.2.1 Crossing Stats

We also give statistical values for our layout. You can see whole average and averages for each node numbe values in Table 9.1. There are 100 consecutive tests for each *Node* entries. All other values in the table are averages. That makes 2000 trials for the experiment. It takes only a hour to run all of these trials. In Table 9.1, *Node* is node number in tests. *Edge* is the average number of edges at this experiment. W is average width, H is average height found by Coffman's algorithm, $Max\ W$ is maximum weight averages in the graphs, *EdgesD* and *NodesD* are average number of edges and nodes after adding dummy nodes. *Cross* is averages of initial crossings, *WOLF* is average crossings after WOLF algorithm and *Median* is average crossings after Median algorithm.

At the final row, you see the averages of all. It is apparent that crossing minimization is vital part of the layered drawing. Coffman's algorithm gives results with many crossings. After WOLF and Median crossing number is decreased. Also, average width of layered drawing is approximately 10 and for this reason, when there is a increase in both node number and edge number, Average height is increased as well.

9.2.2 Edge Length Stats

Edge Length is also important parameter of Layered Drawing. Since we have predefined parameters for *width* accoding to Coffman's algortihm, we do not have restriction on *height* parameter. Actually, the *width* parameter effects *height* parameter. The relation is given at Equation 9.1. In Table 9.2, we give results for the "Average Height" and the "Averege Edge Length" for given "Node Number". For each "Node Number", we repeat the experiment 100 times for different graphs. "Averege Edge Length" is the average of the averages for each graph's edge length. Edge length is calculated with the below formula,

$$|e| = |Layer_i| - |Layer_j| \tag{9.1}$$

where $|e|$ is the length of edge, i is the Layer Number of the source node of e and j is the Layer Number of the target node of e. The Layer Number is calculated by Coffman's Algorithm.

During this experiment, we neglect the edge weights. Furthermore, from the table, we interfere that the Average Edge Length and Average Height have a direct relationship. Interestingly, the approximate ratio for $\frac{AverageHeight}{AverageEdgeLength}$ is 5.91 according to our experiment. Since we have tested 1100 graphs for this experiment, the ratio may be relaible. But, it could be a future work for us to determine theoric relation.

Table 9.1 Layered Drawing Crossing Experiment

Run of Layered Drawing with Stats for Each Node Values									
Node	Edge	W	H	Max W	EdgesD	NodesD	Cross	WOLF	Median
55	62,76	11,81	15,74	18,87	193,94	186,18	158644	11927	11693
60	67,43	10,75	16,78	22,29	211,62	204,19	227772	16644	16010
65	72,66	9,84	18,03	21,38	246,32	238,66	265316	18818	18666
70	77,56	11,72	17,47	20,61	257,21	249,65	277128	20390	19800
75	82,29	10,05	20,91	21,33	306,37	299,08	351824	24745	24604
80	87,47	10,66	18,71	19,4	292,2	284,73	348314	20839	20883
85	92,53	10,15	22,98	20,87	355,21	347,68	417160	26967	26749
90	97,67	9,76	24,26	20,05	419,74	412,07	466792	28019	27990
95	102,56	10,86	22,88	18,98	405,16	397,6	488073	30143	29848
100	107,37	10,71	24,29	20,35	429,18	421,81	661416	34753	34297
105	103,07	11,81	24,06	18,87	398,52	400,45	410411	24056	23721
110	107,87	10,75	25,08	22,29	414,91	417,04	582947	33932	33498
115	112,94	9,84	27,29	21,38	479,8	481,86	726673	36648	36227
120	118,05	11,72	23,87	20,61	443,3	445,25	632961	37002	36450
125	122,88	10,05	29,93	21,33	560,16	562,28	807475	42309	42021
130	128,06	10,66	25,73	19,4	518,85	520,79	891206	40244	40183
135	133,06	10,15	31,47	20,87	637,33	639,27	966688	50878	50129
140	137,87	9,76	33,02	20,05	694,31	696,44	951456	44943	43930
145	143,24	10,86	31,77	18,98	689,87	691,63	1065368	46966	45910
150	148,16	10,71	32,59	20,35	737,33	739,17	1496093	57409	56315
102,5	105,28	10,631	24,343	20,413	434,57	431,79	609686	32382	31946

Table 9.2 Layered Drawing Edge Length Experiment

Run of Layered Drawing with Avrg Edge Lengths for Each Node Number Values		
Node Number	Average Height	Average Edge Length
100	21.73	3.5132958
95	22.63	3.7497752
90	22.62	3.6906813
85	20.07	3.3361297
80	18.36	3.0569359
75	19.43	3.2867682
70	16.26	2.8371761
65	14.86	2.5370780
60	16.74	2.8463142
55	13.08	2.4678986
50	14.48	2.5647818
75	18.20	3.0806213

Chapter 10

Conclusion

This is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.
Winston Churchill

In a conclusion, we provide two methods.

The first method that we proposed is an algorithm for biclustering. Mainly we trust on the power of LEB heuristics and ANH procedure. They worked fine with impressive amount of noise in artificial data set. Even, sometimes crossing minimization is sufficient to get fine biclusters. Our algorithm seems to be better than counter algorithms. It is also simple to implement and adopt. There is no need to struggle with setting parameters. Additionally, in both real data sets and artificial data sets, our results are fine. We extracted correlated biclusters and we get covered almost each gene-condition pair. We perform better than OPSM and CC in many categories of *Yeast* data and our FuncAssociate results are better than all the algorithms used in experiments.

The second method that we provide is a algorithm for layered graph drawing. It is a new method since there is no previous work that considers edge weights in layered drawing. For an application area, we provide a tool that considers a social network of a company as the union of graphs. These union graphs are in layered drawing and the whole graph is represented with the high level graph in a circular layout. We showed the demonstration of our sample graphs, and we have given statistics with our experiments

References

[1] Hartigan,J.A. "Direct Clustering of a Data Matrix". *Journal of the American Statistical Association*, 67(337):123–129, 1972.

[2] Madeira,S.C. and Oliveira,A.L. "Biclustering Algorithms for Biological Data Analysis: A Survey". *IEEE/ACM Trans. on Comp. Biol. and Bioinformatics (TCBB)*, 1(1):24–45, January-March 2004.

[3] Ben-Dor,A., Chor,B., Karp,R. and Yakhini,Z.. "Discovering local structure in gene expression data: The order-preserving submatrix problem". In *Annual International Conference on (Research in) Computational (Molecular) Biology*, volume 6, 2002.

[4] Tanay,A. , Sharan,R. and Shamir,R. "Discovering Statistically Significant Biclusters in Gene Expression Data". *Bioinformatics*, 18 Suppl 1, 2002.

[5] Murali,T.M. and Kasif,S. "Extracting Conserved Gene Expression Motifs from Gene Expression Data". In *Pacific Symposium on Biocomputing*, pages 77–88, 2003.

[6] Kluger,Y. , Basri,R. , Chang,J.T. and Gerstein,M. "Spectral Biclustering of Microarray Data: Coclustering Genes and Conditions". *Journal Genome Res PMID 12671006*, 13:703–16, 2003.

[7] Bergmann,S. , Ihmels,J. and Barkai,N. "Iterative Signature Algorithm for the Analysis of Large-scale Gene Expression Data". *Physical review. E, Statistical, nonlinear, and soft matter physics*, 67(3 Pt 1), March 2003.

[8] Abdullah,A. and Hussain,A. "A New Biclustering Technique Based on Crossing Minimization". *Neurocomputing*, 69(16-18):1882–1896, 2006.

[9] Barkow,S. , Bleuler,S. , Prelic,A. , Zimmermann,P. and Zitzler,E. "Bicat: a Biclustering Analysis Toolbox". *Bioinformatics (Oxford, England)*, 22(10):1282–1283, May 2006.

[10] Prelic, A. , Bleuler, S. , Zimmermann, P., Wille, A., Buhlmann, P., Gruissem, W., Hennig, L., Thiele, L., and Zitzler, E. "A Systematic Comparison and Evaluation of Biclustering Methods for Gene Expression Data". April 18 2006.

[11] Cheng,Y. and Church,G.M. "Biclustering of Expression Data". In Russ Altman, L. Bailey, Timothy, Philip Bourne, Michael Gribskov, Thomas Lengauer, and Ilya N. Shindyalov, editors, *Proceedings of the 8th International Conference on Intelligent Systems for Molecular (ISMB-00)*, pages 93–103, Menlo Park, CA, August 16–23 2000. AAAI Press.

[12] Alexe,G. , Alexe,S. , Crama,Y. , Foldes,S. , Hammer,P.L. , and Simeone,B. "Consensus Algorithms for the Generation of All Maximal Bicliques". *Discrete Appl. Math.*, 145(1):11–21, 2004.

[13] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. In *Proc. Natl. Acad. Sci. USA*, pages 12079–12084, 2000.

[14] Sugiyama, K. , Tagawa, S. , and Toda, M. "Methods for Visual Understanding of Hierarchical System Structures". *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(2):109–125, February 1981.

[15] Eades, P. and Wormald, N.C. "Edge Crossings in Drawings of Bipartite Graphs". *Algorithmica*, 11:379–423, 1994.

[16] Atsuko, Y. and Akihiro, S. "An Approximation Algorithm for the Two-layered Graph Drawing Problem". In *COCOON*, pages 81–91, 1999.

[17] Demetrescu, C. and Finocchi, I. "Removing Cycles for Minimizing Crossings". *ACM Journal of Experimental Algorithms*, 6:2, 2001.

[18] Jünger, M. and Mutzel, P. "2-layer Straightline Crossing Minimization: Performance of Exact and Heuristic Algorithms". *Journal of Graph Algorithms and Applications*, 1(1):1–25, 1997.

[19] Nagamochi, H. "An Improved Bound on the One-sided Minimum Crossing Number in Two-layered Drawings". *Discrete Comput. Geom.*, 33(4):569–591, 2005.

[20] Garey, M.R. and Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[21] Demetrescu, C. and Finocchi, I. "Breaking Cycles for Minimizing Crossings". *J. Exp. Algorithmics*, 6:2, 2001.

[22] Çakiroglu, O.A., Erten,C. , Karatas, Ö. , and Sözdinler, M. "Crossing Minimization in Weighted Bipartite Graphs". In Camil Demetrescu, editor, *WEA*, volume 4525 of *Lecture Notes in Computer Science*, pages 122–135. Springer, 2007.

[23] Peeters, R. "The Maximum Edge Biclique Problem is NP-complete". *Discrete Applied Mathematics*, 131(3):651–654, 2003.

[24] Garey, M.R. and Johnson, D.S. "Crossing Number is Np-complete". *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.

[25] Leda c++ algorithm library. http://www.algorithmic-solutions.com/.

[26] Arabidopsis thaliana. http://arabidopsis.info/.

[27] Bryan, K. and Cunningham, P. "Bottom-up Biclustering of Expression Data". *Proceedings of the 2006 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB'06*, (4133177):232–239, 2006.

[28] Chen, K. and Hu, Y. "Bicluster Analysis of Genome-wide Gene Expression". *Computational Intelligence and Bioinformatics and Computational Biology, 2006. CIBCB '06. 2006 IEEE Symposium on*, pages 1–7, Sept. 2006.

[29] Berriz, G.F., King O.D., Bryant, B., Sander, C., and Roth, F.P. "Characterizing Gene Sets With Funcassociate". *Bioinformatics*, 19(18):2502–2504, 2003.

[30] Karp, R.M. "Reducibility Among Combinatorial Problems". In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[31] Coffman Jr., E.G. and Graham, R.L. "Optimal Scheduling for Two-processor Systems". *Acta Inf.*, 1:200–213, 1972.

[32] Eades, P. and Wormald, N.C. "Edge Crossings in Drawings of Bipartite Graphs". *Algorithmica*, 11(4):379–403, 1994.

[33] Garey, M.R., Johnson, D.S., and Stockmeyer, L. "Some Simplified Np-complete Problems". In *STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63, New York, NY, USA, 1974. ACM Press.

[34] Adai, A.T. , Date, S.V. , Wieland, S., and Marcotte, E.M. Lgl: "Creating a Map of Protein Function With an Algorithm for Visualizing Very Large Biological Networks". *Journal of Molecular Biology*, 340(1):179 – 190, 2004.

[35] Fisher, D. and Dourish, P. "Social and Temporal Structures in Everyday Collaboration". In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 551–558, New York, NY, USA, 2004. ACM Press.

[36] Adamic, L.A. , Buyukkokten, O., and Adar, E. A social network caught in the web. *First Monday*, 8(6), 2003.

[37] Newman, M. "Coauthorship Networks and Patterns of Scientific Collaboration", 2004.

[38] Borgatti, S.P. , Everett, M.G. , and Freeman, L.C. "Ucinet 6 for Windows: Software for Social Network Analysis", 2002.

[39] Adar, E. "Guess: A Language and Interface for Graph Exploration". In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 791–800, New York, NY, USA, 2006. ACM.

[40] Madahain, J.O. ,Fisher, D. , Smyth, P., White, S. , and Boey, Y. "Analysis and Visualization of Network Data Using Jung". *Journal of Statistical Software*, VV(II).

[41] Heer, J. and Boyd, D. "Vizster: Visualizing Online Social Networks". In *INFOVIS '05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, page 5, Washington, DC, USA, 2005. IEEE Computer Society.

[42] Ulrik Br and Dorothea Wagner. visone - analysis and visualization of social networks. In *In Michael Jnger and Petra Mutzel (Eds.): Graph Drawing Software*, pages 321–340. Springer, 2003.

[43] Yang, B., Cheung, W.K., and Liu, J. "Community Mining From Signed Social Networks". *Knowledge and Data Engineering, IEEE Transactions on*, 19(10):1333–1348, Oct. 2007.

[44] Flake, G.W. , Lawrence, S. , and Giles, C.L. "Efficient Identification of Web Communities". In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–160, New York, NY, USA, 2000. ACM.

[45] Flake, G.W., Lawrence, S., Giles, C.L., and Coetzee, F.M. "Self-organization and Identification of Web Communities". *Computer*, 35(3):66–70, Mar 2002.

[46] Murata, T. "Visualizing the Structure of Web Communities Based on Data Acquired From a Search Engine". *Industrial Electronics, IEEE Transactions on*, 50(5):860–866, Oct. 2003.

[47] Adamic, L.A. , Buyukkokten, O., and Adar, E. "A social network caught in the web". *First Monday*, 8(6), 2003.

[48] Eades,P. , Lin,X. and Smyth,W. F.. "A Fast and Effective Heuristic for the Feedback Arc Set Problem". *IPL: Information Processing Letters*, 47, 1993.

[49] Flood, M.M. "Exact and Heuristic Algorithms for the Weighted Feedback Arc Set Problem: A Special Case of the Skew-summetric Quadratic Assignment Problem". *Networks*, 20(1):1–23, 1990.

[50] Even, G., Naor, J., Rao, S., and Schieber, B. "Divide-and-conquer Approximation Algorithms Via Spreading Metrics". In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, page 62, Washington, DC, USA, 1995. IEEE Computer Society.

[51] Seymour, P.D. "Packing Directed Circuits Fractionally". *Combinatorica*, 15(2):281–288, 1995.

[52] Demetrescu, C. and Finocchi, I. "Combinatorial Algorithms for Feedback Problems in Directed Graphs". *Inf. Process. Lett.*, 86(3):129–136, 2003.

Curriculum Vitae

Minimization in Weighted Bipartite Graphs", *Journal of Discrete Algorithms (JDA)*,doi: 10.1016/j.jda.2008.08.003, (2008)

[2] Erten,C. , Sozdinler,M. , "A Robust Biclustering Method Based on Crossing Minimization in Bipartite", *Proc. of Graph Drawing (GD 08)*, LNCS 5417, pp.439-440, (2009)

[3] Cakiroglu,A.O. , Erten,C. , Karatas,O. , Sozdinler,M. , "Crossing Minimization in Weighted Bipartite Graphs", *Proc. 6th Workshop on Experimental Algorithms (WEA 07)*, Lecture Notes in Computer Science 4525, pp. 122-135, (2007)