cHawk: An Efficient Biclustering Algorithm based on Bipartite Graph Crossing Minimization

Waseem Ahmad
Department of Electrical and Computer Engineering
University of Illinois
Chicago, IL, 60607
wahmad@acm.org

Ashfaq Khokhar
Department of Electrical and Computer Engineering
University of Illinois
Chicago, IL, 60607
ashfaq@uic.edu

1. ABSTRACT

Biclustering is a very useful data mining technique for gene expression analysis and profiling. It helps identify patterns where different genes are co-related based on a *subset* of conditions. Bipartite Spectral partitioning is a powerful technique to achieve biclustering but its computation complexity is prohibitive for applications dealing with large input data. We provide a connection between spectral partitioning and crossing minimization which is amenable to efficient implementations. Theoretical construction of Biclustering model based on crossing minimization is provided. Based on this model, an efficient biclustering algorithm, which is termed as cHawk, is developed. We have evaluated cHawk on both synthetic and real data sets. We show that cHawk is able to identify, with good accuracy, constant, coherent and overlapped biclusters amid noise. Moreover, its execution time grows linearly with input data size.

2. INTRODUCTION

Biclustering (Subspace Clustering) is a very useful technique for gene expression analysis and profiling. It has gained increasing popularity in the analysis of gene expression data. Biclustering is significantly useful and considerably harder problem than traditional clustering. Whereas a *cluster* is a set of objects with *similar* values over the *entire* set of attributes, a *bicluster* can be composed of objects with similarity over only a *subset* of attributes. Euclidean Distance is a popular similarity function for clustering. However, it is known to suffer from *curse of dimensionality*. When the dimensions increase, the similarity diffuses in all attributes. Therefore, the distance between two records in such cases becomes meaningless. For example, in text mining, the size of keywords set describing different documents is huge and yields sparse data matrix. In this case, clusters based on entire keywords set may have no meaning for end users.

Biclustering is an approach that finds *local patterns* where a *subset* of objects might be similar to each other based on only a *subset* of attributes. Note that biclusters can cover just part of rows or columns and may overlap with each other. Biclustering process can generate a wide variety of object groups that capture all the significant correlation information present in a data set. For DNA microarray experiments rows of the input matrix correspond to the genes and columns correspond to the conditions. Since a bicluster can identify patterns among a subset of genes based on a subset of conditions, it therefore models condition-specific patterns of co-expression. Moreover, biclustering can identify overlapping patterns thus catering to the possibility that a gene may be a member of multiple pathways.

Biclusters can be categorized into following categories based on the patterns exhibited by the underlying submatrices.

- **Constant Biclusters:** Constant Biclusters refer to those biclusters which (1) exhibit same values over the entire submatrix, (2) same values over only the rows or (3) same values over the columns.
- **Coherent Biclusters:** Values over Rows and Columns of Coherent Biclusters are either (1) offset by a constant (additive coherent biclusters) or (2) offset by some model generated values (model based coherent biclusters).

Formally, the Biclustering problem can be defined as follows: *"Given a data matrix A with set of rows R and set of columns C, identify a set of biclusters $B_k = (R_k, C_k)$ where $R_k \in R$ and $C_k \in C$ such that each bicluster B_k satisfies some specific characteristics of homogeneity."*. Although biclustering has been primarily used in biological data analysis [9, 16, 5, 4, 24, 18, 26], its use in other domains such as the word-document clustering in text mining [10] and for collaborative filtering in online recommendation systems [30, 8] has also gained momentum.

Spectral Partitioning is a popular technique for clustering high dimensional data in different domains [29, 11]. It requires computation of the second eigenvalue and corresponding eigenvector, termed as Fiedler Vector, for the Laplacian matrix of the input data. Computation of eigenvectors using techniques such as Singular Value Decomposition (SVD) incurs a quadratic computation cost in terms of input data size. Moreover it also requires random access to complete data matrix. Such restrictions limit its use in applications involving large input data sets.

In this paper, we provide a connection between bipartite spectral partitioning and bipartite graph crossing minimiza-

VLDB '07, September 23-28, 2007, Vienna, Austria.

tion. We note that computation of Fiedler vector can yield a solution for the Linear Arrangement problem [19]. Linear arrangement problem has been shown to provide bipartite graph crossing minimization solutions which are within reasonable bounds of the optimal solution [28]. This observation leads to the conclusion that crossing minimization can be employed for bipartite spectral partitioning (clustering). Crossing minimization has been extensively used in Graph Drawing and in reducing the wiring congestion in VLSI circuit placement. Although it has been shown that optimal crossing minimization is an NP-hard problem [13], extensive research in this area has yielded efficient heuristics to solve the crossing minimization problem with reasonable accuracy. Computation complexity of most of these heuristics is linear (or log-linear in some cases) with size of the input matrix [20].

Based on the proposed model of biclustering based on crossing minimization, we formulate the optimal biclustering problem as *"maximal crossing minimization of the weighted bipartite graph representing the input matrix"*. This formulation leads to a very efficient and accurate biclustering algorithm which is termed as cHawk.

Our contributions, in this paper, are listed as follows.

- We provide a theoretical connection between spectral partitioning and crossing minimization of a bipartite graph. Using this connection, a Biclustering Model based on Crossing minimization is proposed.

- An efficient implementation of the proposed model, termed as cHawk, is provided. We use barycenter heuristic [20] for solving the crossing minimization problem efficiently. Convergence of this heuristic was theoretically and experimentally proved in [20] [22]. We note that crossing minimization reorders the vertices on both layers of the bipartite graph such that vertices belonging to the same bicluster are brought into the vicinity of each other. This essentially reduces the bicluster identification problem from a global search to local search. Asymptotic complexity of barycenter heuristic is only $O(|E| + |V|log|V|)$ where E is the set of edges and V is the set of vertices in the input graph.

- An efficient algorithm for bicluster identification is proposed. This algorithm employs local search and is capable of finding constant, coherent and overlapped biclusters amid noise. The underlying similarity test is done based on Bregman Divergence [21].

Our results show that the proposed crossing minimization based approach is computationally very efficient. Moreover the proposed bicluster identification algorithm is robust against noise and as our evaluation framework reveals, it outperforms most biclustering algorithms in terms of accuracy and computational efficiency.

cHawk was implemented as part of Biclustering Analysis Toolbox (BicAT) [27]. We have evaluated cHawk over S. cerevisiae data set [14] against known results [1]. We also evaluate the performance and accuracy of cHawk on synthetically generated data sets. The evaluation results are detailed in Section 7.

Organization: Rest of the paper is organized as follows. We give an overview of related research work in Section 3. Section 4 provides theoretical details on the construction of the proposed biclustering model. Section 5 provides a reference implementation, termed as cHawk, of the proposed model. Section 6 discusses the complexity of cHawk. Experimental framework is given in Section 7, conclusions are drawn in Section 8.

3. RELATED WORK

Several approaches have been proposed for solving the biclustering problem. A good discussion on many of these algorithms can be found in [23]. For comparative analysis, we have limited our focus to a selected set of algorithms which are (1) popular in terms of citations and usage and (2) are available in implemented form or can be easily implemented for actual comparative analysis. Following is a description of these selected algorithms.

Cheng and Church [9] define a bicluster to be a submatrix for which the mean squared residue score is below a user-defined threshold δ, where 0 represents the minimum possible value. They proposed a two-phase strategy: first, rows and columns are removed from the original expression matrix until the above constraint is fulfilled; later, previously deleted rows and columns are added to the resulting submatrix as long as the bicluster score does not exceed δ. This procedure is composed of several iterations and each iteration is restricted to identification of only one bicluster while previously defined biclusters are masked with random values. Recently, Wang et al., [32] proposed an improved version of this algorithm which avoids the problem of random interference caused by masked biclusters. Problem with mean squared residue criterion is that it is aimed at identification of constant biclusters and is unable to cater for coherent biclusters. Moreover, it is not very robust against noise [5,33]. On the other hand, our approach employs bregman distance as a similarity criterion which effectively handles noisy data as shown in Section 7. We also use Manhattan Distance for effective identification of coherent patterns.

Tanay et al., [6] presented SAMBA, a graph-theoretic approach to biclustering in combination with a statistical data model. In Samba framework, expression matrix is modeled as a bipartite graph, a bicluster is defined as a subgraph, and a likelihood score is used in order to assess the significance of observed subgraphs. A corresponding heuristic algorithm is aimed at finding highly significant and distinct biclusters. This approach is similar to ours as we also employ a bigraph representation of the input data. It should, however, be noted that SAMBA's time complexity is $O(n2^d)$ where n is the number of vertices and d is the upper bound on the degree of each vertex. This complexity which grows exponentially with d would become non-polynomial for graphs with arbitrarily large degree. As we show in Section 6, our approach has complexity which is just proportional to the size of the input data.

In Order Preserving Sub-matrix Algorithm (OPSM), by Ben-Dor et al. [4], a bicluster is defined as a submatrix that preserves the order of the selected columns for all of the selected rows. In other words, the expression values of the genes within a bicluster induce an identical linear ordering across the selected samples. Based on a stochastic model, the authors developed a deterministic algorithm to find large and statistically significant biclusters. The time complexity of this technique is $O(nm^3l)$ where n and m are the number of rows and columns of the input data matrix respectively and l is the number of biclusters. Since the time complexity

for this technique is cubic with regards to the number of columns (dimensions) of the input matrix, it does not scale well for high dimensional data sets.

Iterative Signature Algorithm (ISA) by Ihmels et al. [18] considers a bicluster to be a transcription module, i.e., a set of co-regulated genes together with the associated set of regulating conditions. Starting with an initial set of genes, all samples are scored with respect to this gene set and those samples are chosen for which the score exceeds a predefined threshold. In the same way, all genes are scored regarding the selected samples and a new set of genes is selected based on another user-defined threshold. The entire procedure is repeated until the set of genes and the set of samples converge, i.e., do not change anymore. Multiple biclusters can be identified by running the iterative signature algorithm on several initial gene sets. This approach requires identification of a reference gene set which needs to be carefully selected for good quality results. In the absence of prespecified reference gene set, random set of genes is selected at the cost of quality of overall biclustering solution.

In xMotif biclustering framework proposed by Murali and Kasif [24] biclusters are sought for which the included genes are nearly constantly expressed across the selection of samples. In a first step, the input matrix is preprocessed by assigning each gene a set of statistically significant states. These states define the set of valid biclusters: a bicluster is a submatrix where each gene is exactly in the same state for all selected samples. To identify the largest valid biclusters, an iterative search method is proposed that is run on different random seeds, similar to ISA. This technique identifies the largest bicluster with probability greater than 0.5 in time $mn^{\frac{\log(\frac{1}{\alpha})}{\log(\frac{1}{\beta})}}$ where $\alpha > 0$ and $\beta < 1$. This complexity is at least $\frac{\log(\frac{1}{\alpha})}{\log(\frac{1}{\beta})}$ order more than the complexity of our approach. Moreover, xMotif framework requires preidentification of the classes of biclusters present in the data which does not suit many kinds of real life data sets.

Liu et al [33] recently proposed an algorithm, referred to as RMSBE, which is claimed to find optimal bi-clusters with the maximum similarity score. The algorithm works for a special case, where the bi-clusters are approximately squares. The algorithm requires multiple scans of the data matrix for similarity score calculation, for reference gene identification and eventually for bicluster identification. The computational cost of the algorithm is $O(nm(n+m)^2)$ [33] which is cubic in both n(rows) and m(columns). This is around $O(n+m)^2$ times more than the complexity of our proposed approach.

The BiMax algorithm proposed by Prelic et al [5] focuses on finding constant biclusters. They discretize the input expression matrix into a binary matrix. This discretization makes it harder to determine coherent biclusters.

It was proposed in [2] that crossing minimization can be used as a recursive noise removal process leading to biclustering. They provided no theoretical justification for this hypothesis. Moreover, their approach employs a static discretization of the input data matrix which makes it hard to identify coherent biclusters. Also, the proposed approach in [2] lacks a quantitative approach for bicluster identification.

Spectral partitioning based co-clustering(biclustering) algorithm was proposed in [11]. The algorithm employs Singular Value Decomposition (SVD) for Fiedler vector computation. A generalized approach for K-way graph clustering was presented in [15]. The computation cost of these algorithms is at least quadratic with respect to the input data size.

Dhillon et al in [17] gave an information theoretic biclustering (Co-Clustering) algorithm. They treat the (normalized) non-negative contingency table as a joint probability distribution between two discrete random variables that take values over the rows and columns. Their subsequent work [3] provided a Bregman Divergence based loss function which was applicable to all density functions belonging to the exponential family. Bregman Divergence is defined as follows [21].

If f is a strictly convex real-valued function, the f-entropy of a discrete measure $p(x) \geq 0$ is defined by

$$H_f(p) = -\sum_x (f(p(x))$$

and the Bregman divergence $B_f(p;q)$ is given as

$$B_f(p;q) = -\sum_x f(p(x)) - f(q(x)) - \nabla f(q(x))(p(x) - q(x)) \tag{1}$$

When $f(x) = x\log x$, H_f is the Shannon entropy and $B_f(p;q)$ is the I-divergence, when $f(x) = -\log(x)$ we get the Burg entropy and discrete Itakura-Saito distortion $B_f(p;q) = \sum_x (\log\frac{q(x)}{f(x)} + \frac{p(x)}{q(x)} - 1)$

4. CONSTRUCTION OF THE PROPOSED BICLUSTERING MODEL

4.1 Preliminaries:

Throughout the paper, we denote a matrix by capital boldface letters such as **G,I** etc. Vectors are denoted by small boldface letters such as **p** and matrix elements are represented by small letters such as w_{ij}. Also, we denote a graph by $G(V,E)$ where V is the vertex set and E is the edge set of the graph. Moreover each edge, denoted by $\{i,j\}$, has a weight w_{ij}. The adjacency matrix of G, denoted as **G**, is defined as

$$\mathbf{G} = \left\{ \begin{array}{ll} w_{ij} & \text{if there is an edge } \{i,j\} \\ 0 & \text{otherwise} \end{array} \right\}$$

In graph theory, a cut is a partition of the vertices of a graph into two sets. Formally, for a partition of vertex set V into two subsets S and T, a *cut* can be defined as follows

$$cut(S,T) = \sum_{i\in S, j\in T} w_{ij}$$

DEFINITION 4.1. *Bipartite Graph: A graph $G(V,E)$ is termed as Bipartite if $V = V_0 \cup V_1$ where V_0 and V_1 are the disjoint sets of vertices (i.e. $V_0 \cap V_1 = \phi$) and each edge in E has one end point in V_0 and the other end point in V_1.*

We consider weighted bipartite graph $G(V_0, V_1, E, W)$ with $W = (w_{ij})$ where $w_{ij} > 0$ denotes the weight of the edge $\{i,j\}$ between vertices i and j. Moreover, $w_{ij} = 0$ if there is no edge between i and j.

For microarray experiment data, genes and conditions are represented by V_0 and V_1 vertex sets respectively. The edge

weight w_{ij} represents the response of i'th gene to j'th condition. We transform the data so that each edge weight is positive.

4.2 Spectral Clustering, Normalized Cut and Fiedler Vector

It was shown in [11] that partitioning (clustering) of vertices on one layer of the bipartite graph will induce a specific clustering of vertices on the other layer which then itself induces a new clustering on the first layer. This recursive process would yield the "best" clustering of vertices on both layers when it corresponds to a partitioning of the graph such that the crossing edges between partitions have minimum weight. This essentially implies

$$cut(V_{01} \cup V_{11}, V_{02} \cup V_{12}, \ldots, V_{0k} \cup V_{1k}) = min_{V_1,V_2,\ldots,V_k} cut(V_1, V_2, \ldots, V_k)$$

where $V_1, V_2, \ldots, V_k$ is any partitioning of the overall Vertex set $V = V_0 \cup V_1$ into k vertex subsets. And V_{0k} and V_{1k} denote the k'th subsets of the two groups of vertices in a bipartite graph.

The minimum cut only considers external cluster connections and ignores the intra-cluster similarity which leads to poor clustering. The poor results are because of its tendency to trap in local minima. The solution is to employ normalized cut instead. Minimum Normalized cut for two partitions S and T can be defined as

$$Ncut(S,T)_{min} = \frac{cut(S,T)}{vol(S)} + \frac{cut(S,T)}{vol(T)}$$

Here $vol(S)$ refers to the total weight of all edges originating from group S. Normalized cut yields more balanced partitioning.

It is well known that graph partitioning problem is NP-complete [12]. Many heuristic methods exist that can find the local minimum of the problem. Spectral graph partitioning heuristics are known to perform well [11]. Following is a brief introduction to the spectral partitioning heuristic.

DEFINITION 4.2. *Laplacian Matrix: The Laplacian matrix* $\mathbf{L_G}$ *of a Graph* G *is the matrix* $\mathbf{I(G)} - \mathbf{A(G)}$ *where* $\mathbf{A(G)}$ *is the adjacency matrix of the graph* G *and* $\mathbf{I(G)}$ *is the diagonal matrix with vertex degrees on the diagonal, i.e.* $i_{vv} = d_v$ *and* $i_{uv} = 0$ *if* $u \neq v$

Given a bipartitioning of V into V_1 and V_2, a partition vector, denoted by $\mathbf{p}$ is defined as follows

$$p_i = \left\{ \begin{array}{ll} +1 & i \in V_1 \\ -1 & i \in V_2 \end{array} \right\}$$

We can minimize the cut of the partition by finding a non-trivial vector $\mathbf{p}$ that minimizes the function

$$f(p) = \sum_{i,j \in V} w_{ij}(p_i - p_j)^2 = \mathbf{p^T L_G p}$$

OBSERVATION 4.3. *The Rayleigh Theorem shows that the minimum value for* $f(p)$ *is given by the second smallest eigenvalue of the Laplacian* $\mathbf{L_G}$*. The optimal solution for* $\boldsymbol{p}$ *is given by the corresponding eigenvector* λ_2*, referred to as the Fiedler vector. Thus spectral partitioning problem requires calculation of the Fiedler vector.*

4.3 Crossing Number and Linear Arrangement Problem

DEFINITION 4.4. *Bipartite Drawing: A bipartite drawing of* G *is the embedding of its vertex sets* V_0 *and* V_1 *onto distinct points on two horizontal lines* y_0, y_1 *respectively while edges are drawn by straight line segments.*

We will assume that y_0 is the line $y = 0$ and y_1 is the line $y = 1$. Any bipartite drawing of G is identified by a coordinate function $h : V_0 \cup V_1 \rightarrow \mathbb{R}$ where for any $v \in V_0 \cup V_1$, $h(v)$ is the x-coordinate of the vertex v in the drawing. If h is a coordinate function of a bipartite drawing, then the permutations π_0 and π_1 are injective functions and determine the order in which vertices are placed on the lines y_0 and y_1 respectively.

DEFINITION 4.5. *Crossing Number: Let* h *be a bipartite drawing of Bipartite graph* $G = (V_0, V_1, E)$*. Let* $bcr(e)$ *denote the number of crossings for an edge* $e = i, j$ *with other edges. Let* $bcr(h)$ *denote the total number of crossings in* h *i.e* $bcr(h) = \frac{1}{2}\sum_e bcr_h(e)$*. The bipartite crossing number of* G *denoted by* $bcr(G)$ *is the minimum number of crossings over all possible drawings of* G *i.e* $bcr(G) = min_h bcr(h)$

DEFINITION 4.6. *Linear Arrangement Problem: The linear arrangement problem is to find a bijective function* $f : V \rightarrow \{1, 2, 3, \ldots, \mid V \mid\}$ *of minimum length. This minimum value is denoted by* $L(G)$ *and its calculation is known to be NP-hard [28]. If* h *is the bipartite drawing of* $G = (V_0, V_1, E)$*, then the length of* h*, denoted by* L_h *is defined as*

$$\sum_{uv \in E} \mid h(u) - h(v) \mid$$

CLAIM 4.7. *Linear arrangement and Spectral Partitioning problems are dual of each other and can be solved by finding the Fiedler Vector of corresponding Laplacian matrix.*

PROOF. In [19], a heuristic was proposed to solve the linear arrangement problem in polynomial time. This heuristic solves the linear arrangement problem by computing the corresponding eigenvector, termed as Fiedler vector, to the second Laplacian eigenvalue λ_2 of the Graph G. Results from [19] combined with Observation 4.3 prove the claim. □

CLAIM 4.8. *Crossing minimization can be used to solve the Linear Arrangement problem with accuracy guarantee of* $O(\log n)$ *from the optimal value.*

PROOF. Shahrokhi et al. [28] provided a connection between crossing minimization and linear arrangement problem. They provided lower and upper bounds for bipartite crossing number $bcr(G)$ in terms of optimal arrangement value. If d_v denotes the degree of v, δ_G is the minimum degree of G and Δ_G is the maximum degree of G, then it was shown that

$$\frac{1}{36}\delta_G L(G) - \frac{1}{12}\sum_{v \in V} d_v^2 \leq bcr(G) \leq 5\Delta_G L(G)$$

In [28] a heuristic was proposed using the above result to solve the crossing minimization in polynomial time with accuracy guarantee of $O(\log n)$ from the optimal value based on the condition that $\Delta_G = O(\delta_G)$ □

Proofs of claims 4.7 and 4.8 provide a clear connection between crossing minimization and spectral clustering problems. Spectral clustering solutions are based on finding the Fiedler Vector (an eigenvector) of the Laplacian of the given graph. Eigenvalues and their corresponding eigenvectors are computed using techniques such as Singular Value decomposition (SVD). These techniques have typically quadratic complexity and require random access to complete data sets. This complexity limits their effectiveness for processing very large matrices.

To the best of our knowledge, proposed model is the first to show that spectral clustering problem can be solved through crossing minimization heuristics. Crossing minimization heuristics have been a subject of research for many years and a number of efficient solutions have been proposed to date [20], [22]. The complexity of most of these heuristics is linear (or log-linear in some cases) with the matrix size. This clearly indicates that the proposed problem formulation can lead to more efficient solutions for clustering problem.

In this paper, we make use of the proposed model to develop an efficient biclustering solution, termed as cHawk. Implementation details of cHawk are provided in next section.

5. cHAWK IMPLEMENTATION

Gene microarray experiment data is usually very noisy and contains missing values too. Therefore, accurate biclustering requires conditioning of the experimental data to reduce the effect of noise and cater for the missing values.

Missing value prediction has been taken up in great detail in [25] and [31]. Since missing value prediction is out of scope for this paper, we will rely on a simple approach whereby each missing value is taken as zero. Similar approach was used in [7].

Several techniques have been developed to cater for the noise in data including those based on binning and regression. The approach used in cHawk is to employ the binning method. According to this method, attribute values are partitioned into equal depth bins. The values within bins are then smoothed by bin means. This is a simple yet effective way of catering for the noise. More elaborate schemes for noise removal are beyond the scope of this paper. Moreover a joint probability distribution table is built during this process. This table will be used for computing similarity based on Bregman divergence during bicluster identification process.

After the input data is properly conditioned, we then build bipartite graph model of the data. Crossing minimization heuristics are performed on this model to obtain biclusters. After the crossing minimization, vertices on both layers of the bipartite graph are reordered so that "similar" vertices are in the vicinity of each other. Bregman Divergence is then employed as a similarity criterion in a local search based bicluster identification algorithm to identify all significant biclusters.

cHawk's crossing minimization based biclustering approach is composed of following steps.

- Construction of the Bigraph
- Bigraph Crossing Minimization
- Bicluster Identification

5.1 Construction of Bigraph

The cHawk algorithm takes the input matrix as adjacency matrix. It then builds a bipartite graph (bigraph) representation of this matrix where rows are represented by $Layer0$ nodes and columns are represented by $Layer1$ nodes. There is an edge between a node i at $Layer0$ and a node j at $Layer1$ if corresponding matrix element i.e. w_{ij} is greater than 0. If $w_{ij} = 0$ then there is no edge between node i and node j. This essentially means that bigraph representation will ignore zero values which lends itself to efficient implementations for sparse matrices. Figure. 1 illustrates an example input matrix and its bipartite graph representation where $Layer0$ is represented by the bottom layer and $Layer1$ is represented by top layer. Figure. 1(a) shows the input matrix where each element is shown inside a box in a grid. The color of this box is dependent on the value of the matrix element. Figure. 1(b) shows the bipartite graph representation of the input matrix. For tidiness, edges are not labeled with their weights in Figure 1(b)

5.2 Bigraph Crossing Minimization Heuristics

After construction of the bigraph, crossing minimization heuristic is performed. Several heuristics for bigraph crossing minimization have been proposed in the literature. There are two phases of each heuristic, first is initial ordering and the second is iterative improvement. Initial Ordering involves some global computation on the graph to sequence the nodes in each layer. The iterative improvement phase solves a fixed layer problem on the alternating layers. The nodes on one layer(say $Layer0$) are kept static and ordinal values (x-coordinates on the plane) of the nodes on the other layer ($Layer1$) are calculated as a function of the ordinal values of its neighbor nodes on $Layer0$. This function varies from heuristic to heuristic. The nodes on $Layer1$ are reordered based on the new ordinal values. In the next step, nodes on $Layer1$ are kept static and new ordinal values of nodes on $Layer0$ are calculated. This process continues till there is no further change in the ordinal values of any node on a layer. By the end of crossing minimization step, we have essentially reordered the input matrix such that all nodes with similar neighbors and edge weights are brought together. Figure 1(c) shows the reordered matrix after crossing minimization and Figure 1(d) presents the bipartite graph representation of the reordered matrix. The rectangles with dashed lines represent the biclusters.

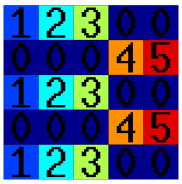
The details of each phase involved in crossing minimization are given below.

5.2.1 *Initial Ordering*

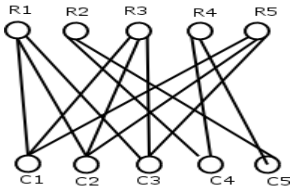
In their experimental evaluation, Stallman et al in [22] use three types of initial orderings. These are random, breadth first search (BFS) and guided breadth first search (GBFS). Interested reader is referred to [22] for detailed discussion of these techniques. For simplicity, cHawk considers random initial ordering which has been shown to be very effective while incurring minimal computation cost [22].

5.2.2 *Iterative Improvement*

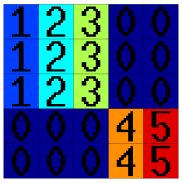
As its name suggests, second phase – the iterative improvement – is used to iteratively minimize the crossings by repeatedly applying a fixed layer heuristic alternating from layer to layer. Median and Barycenter heuristics are the two



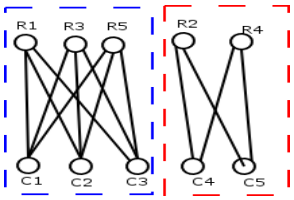
(a) The input matrix.



(b) Bipartite Graph representation of the input matrix.



(c) Reordered matrix after crossing minimization.



(d) Bipartite Graph representation after crossing minimization

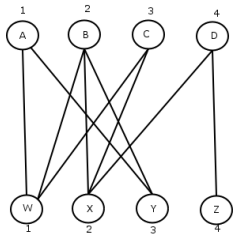
Figure 1: Illustration of Crossing Minimization Process for Biclustering

most popular heuristics applied towards iterative improvement. A detailed description of these heuristics follows.

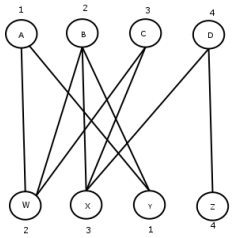
Median Heuristic: Median heuristic treats the neighbors of each node as a set of integers representing their ordinal numbers on the opposite layer. Nodes are sorted using medians of these sets as keys. Traditionally, the median is defined as the mean of the two middle elements. The median heuristic which was evaluated in [22] always uses the smaller of the two candidates, but with the added condition that, in case of ties, nodes with odd degree always precede those of even degree. Assuming random initial ordering and a stable sort, the probability that the median heuristic embeds a simple path optimally is $O(\frac{1}{n})$. It embeds a simple cycle optimally every time.

Figure 2 illustrates the crossing minimization process based upon median heuristic. For simplicity, we are considering a binary matrix. Figure 2(a) shows the original bigraph. Initial ordering is shown on top/bottom of the nodes in top/bottom layer. For first iteration, upper layer is kept static and positions of nodes in the bottom layer are changed according to median Heuristic. According to this heuristic, node W will get the new order 2 which is a median of rank values $(1, 2, 3)$ of its neighbors on the upper layer. Similarly new ordering information will be assigned to remaining nodes. This ordering information is used to calculate the rank for the node among all nodes in the same layer. This rank is used in subsequent iterations. Figure 2(b) shows the new positions for nodes in the lower layer based upon the median heuristic and Figure 2(c) shows the final bigraph with reduced crossings. Note that the crossing number of the graph based on new ordering has been reduced from 9 to 2 in just one iteration.

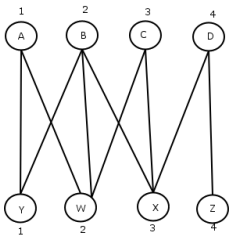
Note that Figure 2 illustrates only one iteration of an actual crossing minimization step. Next step would be to fix the lower layer and update the positions of the nodes in the upper layer. This iterative process goes on till there is no change in the node positions.



(a) Original Graph G. Crossing Number($C(G)$)= 9



(b) Updated Positions of the lower layer nodes with respect to ranks of upper layer neighbors



(c) Graph G' after one iteration of Median Heuristic. $C(G') = 2$

Figure 2: Illustration of Median Heuristic For Crossing Minimization

Barycenter Heuristic : Barycenter heuristic assigns a new rank to each node based on the mean of ranks of its neighbor nodes. Median heuristic, on the other hand, employs median instead of mean. On graphs that are more random and have several nodes of high degree, barycenter does better than median in terms of crossing minimization. Each iteration of the median heuristic can be implemented in linear time, while one iteration of barycenter requires $O(|E| + |V|log|V|)$ time. For graphs with constant degree bound, the barycenter heuristic can also be implemented in linear time per iteration. Convergence of barycenter heuristic was theoretically and experimentally proved in [20] [22].

Figure 3 shows the crossing minimization process based on Barycenter Heuristic for the graph of Figure 3(a). Figure 3(b) shows the updated positions of the nodes in the bottom layer based on Barycenter Heuristic. For example, Node W is assigned a position 2 because the mean of its neighbors' rank values $(1, 2, 3)$ is $\frac{1+2+3}{3} = 2$. Similarly the ranks of other nodes are also calculated. Figure 3(c) shows the Graph after one iteration of Barycenter Heuristic. The number of crossings after one iteration have been reduced to 2 from 9.

Our solution employs a weighted version of the Barycenter heuristic in order to cater for the practical non-binary matrices. Pseudocode for barycenter heuristic based crossing minimization is given in Algorithm 1.

Let v_i represent the i'th node in the non-static(dynamic) layer and set N_i represent the set of neighbors of v_i. Also let r_j represent the rank of j'th member of the set N_i. Then new rank of v_i, denoted as r_i can be calculated as follows.

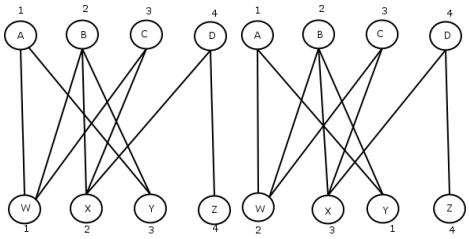
1. We first calculate the Weighted Mean, denoted as μ, of the ranks of neighbor nodes using Equation 2.

2. These weighted means represent the new ordering of the nodes. Since these means are not necessarily unique, we adjust them so that each node is assigned a unique rank based on its weighted mean.

$$\tilde{s} = \sum_{j \in N_i} w_{i,j} \times r_j \tag{2a}$$

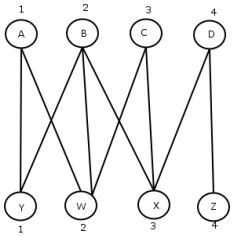
$$s = \sum_{j \in N_i} w_{i,j} \tag{2b}$$

$$\mu_i = \frac{\tilde{s}}{s} \tag{2c}$$



(a) Original Graph G. Crossing Number($C(G)$)= 9

(b) Updated Positions of Lower Layer Nodes using Barycenter Heuristic.



(c) Graph G' after one iteration of Barycenter Heuristic. $C(G') = 2$

Figure 3: Illustration of Barycenter Heuristic for Crossing Minimization

5.3 Bicluster Identification

After crossings are minimized, we perform the bicluster identification process. This process employs the joint probability distribution table built during the data pre-processing stage. Algorithm for Bicluster Identification is given in Algorithm 2.

Bicluster identification process starts from the first element of the reordered matrix and keeps on adding new columns and rows while calculating and updating a score representing the coherence of the values in the current block of the reordered matrix. This coherence is determined by virtue of two kinds of distance functions namely Bregman Divergence and Manhattan Distance. Bregman Divergence

Algorithm 1 Crossing Minimization in a Bipartite Graph

Require: Bigraph BG
Ensure: An embedding of BG (new ordering of the nodes in the two layers) which results in maximal crossing minimization for BG.

```
positionChanged ⇐ 1
DynamicLayer ⇐ 1
while PositionChanged ≠ 0 do
    PositionChanged = 0
    for all i such that v_i belongs to the vertices(nodes) in the Dynamic Layer do
        Compute Weighted mean for v_i using Equation 2
        if Node_i.Rank ≠ WeightedMean then
            Node_i.Rank = WeightedMean
            PositionChanged = PositionChanged + 1
        end if
    end for
    Sort all the nodes in Dynamic Layer
    Now adjust node ranks such that each node has a unique rank
    DynamicLayer = (1 − DynamicLayer)
end while
```

is used to compare two rows over same set of columns. As described earlier, we can use different functions in Bregman Divergence Equation (Equation 1) to emulate Euclidean Distance, KL-Divergence and Itakura-Saito distortion [21]. Manhattan Distance is used to compare the values of adjacent columns in the same row. Manhattan distance for two points $P_1(x1, y1)$ and $P_2(x2, y2)$ in XY plane is given in Equation 3.

$$D(P_1, P_2) = \mid x_1 - x_2 \mid + \mid y_1 - y_2 \mid \tag{3}$$

Manhattan distance is computed over edge weights w_{ij} and bregman divergence is computed using joint probability distribution table as mentioned above. Manhattan distance is simple to compute and works well in our case as we are using it to compare distance among columns of the same row which are assumed to be identically distributed.

The bicluster identification procedure is a local search procedure. We keep pointers $StartRow$ and $StartColumn$ which are initialized to first row and column of the reordered matrix respectively. Coherence score for adjacent rows is calculated using Bregman Distance. Similarly the evolution pattern over adjacent columns is determined using Manhattan Distance. Our iterative procedure is row-major. We compare two rows at one time to see if they have matching columns. Starting with the $StartRow$ and $Startrow + 1$, we keep a reference set of matching columns. Columns are added to this set if either of following two conditions is satisfied.

1. Bregman Distance between two rows over same set of columns is less than the given threshold δ

2. The manhattan distance between adjacent columns on both rows is the same.

The first condition makes sure that we always identify the constant value biclusters constrained by the noise. Second condition, on the other hand, guarantees that we would be able to determine the biclusters which exhibit coherent evolution over current set of columns. For each subsequent row, we find out that if it has the same coherent columns. If it has, we add it to our row set. If any row has more coherent columns than the ones in reference set, we set the

OverlapFlag and store the current value of row iterator i in *StartRow* and that of column iterator j to *StartColumn*.

If both of the above mentioned conditions fail, we call the current submatrix a bicluster and continue with the Bicluster identification process. The next iteration of the identification process would start from *StartRow* and *Startcolumn* if *OverlapFlag* was set otherwise it could continue with the next value of row iterator till it reaches the last row. Moreover, when we reach the end of columns, we add the current row to our current Row Cluster. If we reach the end of rows, and current column and row clusters contain more than required minimum number of columns($Column_{min}$) and rows (Row_{min}) respectively, then we declare these row and column clusters to be a bicluster and add it to the Bicluster set S.

Algorithm 2 Bicluster Identification

Require: Reordered Matrix M, A convex function f (required for computing Bregman Distance), Dissimilarity threshold δ, Minimum Rows in a Bicluster row_{min}, Minimum Number of Columns in a Bicluster $Column_{min}$
Ensure: A set S of all interesting biclusters is returned.
1: $StartRow \Leftarrow 1$
2: $StartColumn \Leftarrow 1$
3: **for all** $i : 0 \rightarrow Rows - 1$ **do**
4: **for all** $j : 0 \rightarrow Columns - 1$ **do**
5: Calculate the Bregman Distance between Rows i and $i+1$ for the Columns Set $0 \rightarrow j$. Use Equation 1 and the given function f to compute the value of Bregman Distance.
6: Calculate Manhattan Distance, using Equation 3, between adjacent columns (j and $j-1$) for Rows i and $i+1$
7: **if** Distance Calculated in Step 5 is less than δ OR Manhattan Distance calculated in Step 6 is the same for both rows. **then**
8: **if** $i = StartRow$ **then**
9: Add column j to current column cluster if its not already contained in it.
10: **end if**
11: **if** $(i = Rows - 2)$ and $|RowCluster| > Row_{min}$ and $|ColumnCluster| > Column_{min}$ **then**
12: Declare current Row and Column Clusters as a Bicluster and add it to S.
13: **end if**
14: If the Column is not contained in current Column Cluster and $i > StartRow$, then we have potentially overlapped biclusters.
15: $OverlapFlag = True$.
16: $StartRow = i$; $StartColumn = j$ {We will start finding biclusters from this row and column in the next iteration of the identification process.}
17: **if** $j \Leftarrow Columns - 1$ **then**
18: Add Row i to current Row Cluster
19: **end if**
20: **else if** $|RowCluster| > row_{min}$ and $|ColumnCluster| > Column_{min}$ **then**
21: Declare a bicluster Bix composed of current row and column clusters. Add Bix to set of Biclusters S.
22: Empty the Row and Column Clusters.
23: **if** $OverlapFlag = True$ **then**
24: $i \Leftarrow StartRow$
25: $j \Leftarrow StartColumn$
26: $OverLapFlag \Leftarrow False$
27: **end if**
28: **end if**
29: **end for**
30: **end for**
31: **return** S

6. COMPLEXITY ANALYSIS OF cHAWK

Lets assume that $n = \mid V_0 \mid$= total number of rows of the input data matrix, $m = \mid V_1 \mid$= total number of columns of the input matrix and R_c = average number of rows per bicluster. Also C_c = average number of columns per bicluster and C = total number of biclusters and k = average number of iterations of crossing minimization process. Now if $O(n)$ = time to compute weighted means and $O(nlogn)$ = time to perform sorting based on means and $O(n)$ = time taken in adjusting node positions. Also $O(CR_cC_c)$ = time to identify biclusters (when there is no overlap). In case of overlapped biclusters, the bicluster identification process would take $O(dCR_cC_c)$ where d is the average degree of overlap among biclusters. Combining all of these components would yield time complexity which is given as $T_{cHawk} = O(k(n + nlog(n) + n)) + O(dCR_cC_c)$. Here, without loss of generality, we can assume that $O(dCR_cC_c) = O(dnm)$. This implies that

$$T_{cHawk} = O(k(n + nlog(n) + n)) + O(dnm)$$

We have noted that k is a fairly small number for most cases. In fact, during all of our experiments the maximum value that k took was 5. This implies that in the expression for T_{cHawk}, the term $O(dnm)$ tends to dominate which essentially means that proposed crossing minimization based biclustering process has time complexity which has linear relationship with the problem size given that the degree of overlap remains constant.

7. EXPERIMENTAL FRAMEWORK

7.1 Experimental Setup

The proposed biclustering algorithm is implemented in C++ and has been integrated with Java based BicAT [27] using Java Native Interface (JNI)[1].

The algorithm's accuracy and performance was determined using the gene expression data of S.cerevisiae provided by Gasch et al. [14]. The dataset contains 2993 genes and 173 conditions. Evaluation of cHawk was also carried out on synthetically generated data sets of larger sizes. The experiments were performed on a Linux machine which is composed of a 2.4 GHz Pentium4 processor with 512 MB RAM. The operating system is Ubuntu Linux 6.10 and compiler is the GNU gcc 3.3.6. In all the experiments the execution time is reported in seconds. The execution time takes into account the time spent in reading data from Files in the process of initializing the structure.

7.2 Accuracy Evaluation

To evaluate the accuracies of different methods, we use the measure (match score) similar to the score proposed by Prelic et al. [5]and Liu et al [33]. Let $M1, M2$ be two sets of bi-clusters. The match score of $M1$ with respect to $M2$ is given by

$$S(M_1, M_2) = \frac{1}{\mid M_1 \mid} \sum_{A(I_1,J_1) \in M_1} max_{A(I_2,J_2) \in M_2} \frac{\mid I_1 \cap I_2 \mid\mid J_1 \cap J_2 \mid}{\mid I_1 \cup I_2 \mid\mid J_1 \cup J_2 \mid}$$

Let M_{opt} denote the set of implanted bi-clusters and M the set of the output bi-clusters of a bi-clustering algorithm. $S(M_{opt}, M)$ represents how well each of the true bi-clusters

[1]cHawk is available under GPL at
http://www2.uic.edu/~wahmad1/software/cHawk/

Method	Parameter Settings
cHawk	$\delta = 0.5, I = 5$,Function=KL-Divergence
Samba	$D = 40, N_1 = 4, N_2 = 4, k = 20, L = 10$
BiMax	Minimum no. of genes and chips =4
ISA	$t_g = 2, t_c = 2, seeds = 100$
CC	$\delta = \{0.5, 0.002\}, \alpha = 1.2$
RMSBE	$\alpha = 0.4, \beta = 0.5, \gamma = \gamma_e = 1.2$
OPSM	l = 100

Table 1: Parameter Settings for Different Biclustering Methods

is discovered by the bi-cluster algorithm. Our evaluation methodology is to compare our algorithm with other algorithms for two kinds of bi-clusters i.e. Constant Biclusters and Coherent Biclusters. Other algorithms include CC [9], Samba [6], ISA [18], RMSBE [33] and Bimax [5]. We used the Biclustering Analysis Toolbox (BicAT) developed by Barkow et al. [27] and EXPANDER developed by Shamir et al. [26] for evaluation and bicluster visualization purposes. CC, ISA and Bimax are implemented in BicAT. Samba is implemented as part of EXPANDER and RSMBE implementation was downloaded from [34].

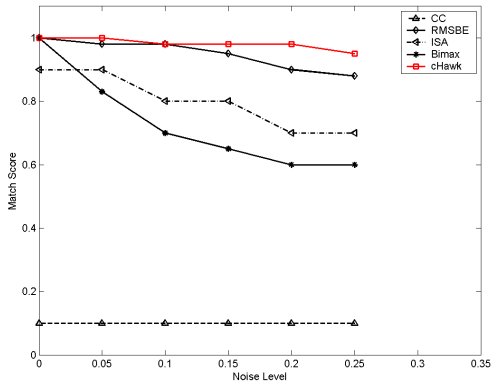
7.2.1 *Evaluation for Constant Biclusters*

In order to compare our algorithm with other programs for constant bi-cluster data, we follow the approach used by Liu et al. [33]. To cater for the missing values in real life data, we add noise by replacing some elements in the matrix with random values. There are three variables b, c and γ in the generation of the bi-clusters. b and c are used to control the size of the implanted bi-cluster. γ is the noise level of the bi-cluster. The matrix with implanted constant bi-clusters is generated with four steps: (1) generate a 100×100 matrix A such that all elements of A are 0s, (2) generate ten 10×10 bi-clusters such that all elements of the bicluster are 1s, (3) implant the ten bi-clusters into A without overlap, (4) replace $\gamma(100 \times 100)$ elements of the matrix with random noise values (0 or 1) . For each test on constant bi-clusters, we generate 10 matrices and consider the average matching scores of different biclustering methods on these matrices.

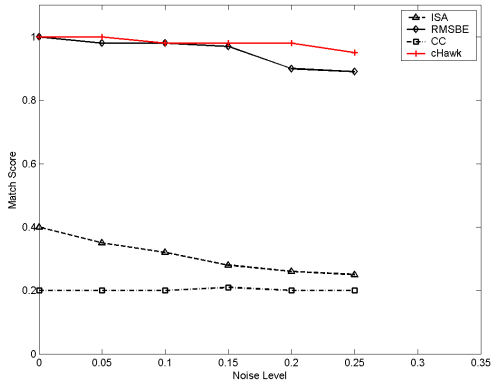
In the experiment, the noise level ranges from 0 to 0.25. The parameter settings used for different bi-clustering methods are the default settings and are listed in Table1. The results are shown in Figure. 4(a). In the absence of noise, cHawk, ISA, Samba, Bimax and RMSBE can always find the implanted bi-clusters correctly. As mentioned in [5], CC uses the similarity of the selected elements as the bi-clustering criterion. The criterion does not work for the constant bi-cluster with only up-regulated values as is the case with current scenario. When the noise level is high, cHawk, ISA and RMSBE have the best accuracies. The normalization of the data (in case of ISA) and data cleaning through binning (in case of cHawk) is the main reason that these algorithms perform well on noisy data. The inclusion-maximal bi-cluster model of Bimax is limited in finding error-free bi-clusters. This model limits its performance on noisy data. The performance of Samba is sensitive to the statistical significance of the bi-clusters. When the noise level is high, the significance of the bi-clusters decreases rapidly. Therefore, the performance of Samba is not good for noisy data.

7.2.2 *Evaluation for Coherent Biclusters*

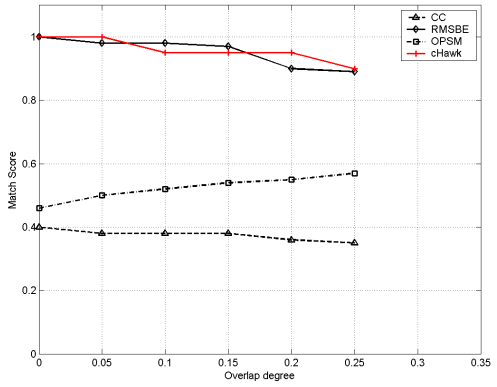
For accuracy evaluation of Biclustering methods in find-



(a) Accuracy Evaluation for Constant Biclusters.



(b) Accuracy Evaluation for Coherent Biclusters.



(c) Accuracy Evaluation for Overlapped Biclusters

Figure 4: Results for Synthetic data sets

ing coherent biclusters, we randomly generate the values in the 100×100 (background) matrix A such that the data fits the standard normal distribution with the mean of 0 and the σ of 1.2. This is the same approach for synthetic data generation as taken by Liu et al in [33]. To generate a coherent $b \times c$ bi-cluster, we first randomly generate the matrix values in a reference row $(a_1, a_2, ..., a_c)$ according to the standard normal distribution. To get a row $(a_{i1}, a_{i2}, ..., a_{ic})$ in the coherent bi-cluster, we randomly generate a distance d_i (based on the standard normal distribution) and set $a_{ij} = a_j + d_i$ for $j = 1, 2, ..., c$. After we get the $b \times c$ coherent bi-cluster, we can add some noise by randomly selecting $\delta.(b \times c)$ elements in the bi-cluster and changing the values to a random number (according to the standard normal distribution). Finally, we insert the noisy coherent bi-cluster into the 100×100 background matrix A. For each test on coherent biclusters, 50 matrices are generated.

Parameters selection. Based on the simulation results, we find that cHawk works well for a wide range of parameters settings and that Euclidean distance based coherence score outperforms KL-divergence in case of overlapped biclusters while KL-Divergence is better in case of Constant and Coherent Biclusters. We recommend to use the following parameter settings: Function= Euclidean Distance (Overlapped Biclusters) and KL-Divergence (Coherent Biclusters). Now, we test different methods for coherent biclusters. The discretization methods used by Samba and Bimax cannot identify the elements in the coherent bi-clusters. Without reasonable discretized data, the two methods cannot find the implanted coherent biclusters. Thus, the two methods are not included in the comparison on coherent biclusters. In the test on coherent biclusters, we compared cHawk with RMSBE, ISA and CC. We generated the coherent bi-clusters of size 15×15 with different noise level $\delta \in [0, 0.25]$. The parameter settings of different methods are listed in 1.

Figure 4(b) shows that cHawk has better accuracy than CC and ISA on different noise levels and has a comparable accuracy with RMBSE. ISA uses only up-regulated and down-regulated expression values in its bi-clustering method. When coherent bi-clusters contain elements of normal expression levels, ISA may miss some rows and columns of the implanted bi-clusters. When the signal of the implanted bi-cluster is weak comparing with the background noise of the whole matrix, the greedy method of CC may delete some rows and columns of the implanted bi-cluster in the beginning of the algorithms and miss the deleted rows and columns in the output biclusters. RMBSE requires selection of a reference gene for its accuracy. Their algorithm tries to work with all genes as reference genes which make it computationally very expensive but more accurate. On the other hand, cHawk is much more computationally efficient and yet very accurate.

Finding Overlapped Biclusters. To test methods with overlapped biclusters, we first generate two $b \times b$ coherent bi-clusters with o overlapped rows and columns. o is called the overlap degree. Also, we replace δ fraction of the two biclusters with random noise values and implant them into a 100×100 randomly generated matrix. The elements also fit the standard normal distribution. To find overlapped biclusters in a given matrix, some methods, e.g. CC, need to mask the discovered bi-clusters with random values. Advantage of the cHawk is that it does not need to mask discovered bi-clusters. We test the accuracy of cHawk, RMSBE, CC and OPSM on overlapped biclusters by using 20×20 coherent bi-clusters with noise level $\delta = 0.1$ and overlap degree o ranging from 0 to 10. The results in Figure. 4(c) show that cHawk is only marginally affected by the overlap degree of the implanted bi-clusters.

7.2.3 *Accuracy Evaluation for Non-Contiguous Distributed Biclusters*

So far, we have evaluated all methods for biclusters which were all contiguously allocated in the input data matrix. This does not capture the scenario in real data where biclusters are non-contiguous and distributed all across the matrix. For this set of experiments, we generated a 100×50 matrix and 10 biclusters of size 10×5. But instead of placing these bicluster contiguously in the data matrix, we place them in interleave fashion such that if first row belonged to Bicluster 1, then the second one would belong to Bicluster2 and so on. Similarly, if first Column belonged to first Bicluster then second one would belong to second bicluster and so on. The purpose of implanting rectangular biclusters was to determine if algorithms under investigation are capable of handling non-square biclusters too. In practical scenarios, it is quite common to have rectangular biclusters. Moreover, rectangular biclusters are harder to find and provide a good benchmark for determining the effectiveness of underlying bicluster identification procedure.

cHawk was compared against RMSBE on this synthetic data amid noise levels ranging from 0 to 2.5. The results are shown in Figure 5. It is clear that cHawk maintains its accuracy even in this case while RMSBE's accuracy deteriorates significantly on this data set. The reason for this deterioration of accuracy in case of RMSBE is that it is unable to find rectangular biclusters accurately. It was also pointed out in [33]. On the other hand, crossing minimization based reordering approach employed by cHawk not only finds the distributed patterns but is able to do so with high computational efficiency. Results on the computational performance are presented at the end of this section.

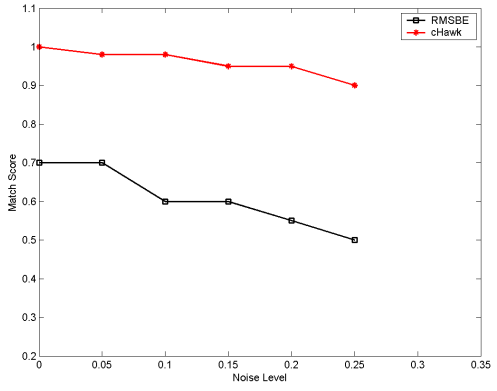


Figure 5: Accuracy Comparison of cHawk and RMSBE for Distributed Interleaved Biclusters.

7.2.4 *Accuracy Evaluation for Real Data*

In order to evaluate the proposed Biclustering algorithm

cHawk against other Biclustering methods on real data, we follow the approach used by Prelic et al [5]. Their method evaluated the discovered biclusters using Gene Ontology (GO) annotations and protein-protein interaction networks for S. cerevisiae and A. thaliana data sets. Here we present results for evaluation based on GO annotations. The idea is to determine the enrichment level of discovered biclusters by each method with respect to a specific GO provided by Gene Ontology Consortium. Similar to Liu et al's approach, we also use FuncAssociate to determine these enrichment levels. The analysis is performed on the gene expression data of S.cerevisiae provided by Gasch et al [14]. The data set contains 2993 genes and 173 conditions. The adjusted significance scores were calculated using FuncAssociate. These scores are represented in the graph of Figure.6. The results for cHawk are compared against reported scores of RMSBE, BiMax, OPSM, ISA, Samba and CC from [5] and [33]. RMSBE and cHawk seem to outperform other methods. RMSBE achieves this accuracy at a very high computational cost. Apart from CC, other algorithms have reasonably good performance. CC under-performs because of its inability to find coherent biclusters and its lack of robustness against noise.

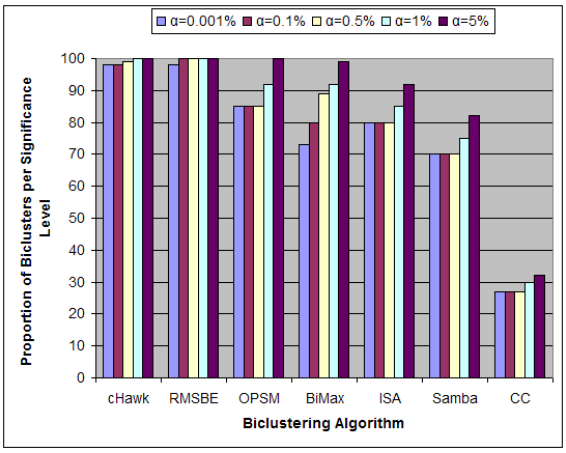


Figure 6: Proportions of Biclusters significantly enriched by Gene Ontology biological process category(S.cerevisiae). α is the adjusted significance scores of the biclusters.

7.2.5 *Performance Evaluation of cHawk*

As mentioned above, cHawk is designed to be computational efficient from the ground up. Our second set of experiments were aimed at analyzing the performance of cHawk. For this purpose, synthetic data sets with sizes ranging from 1000 rows to 200000 rows were generated. Implanted biclusters were all of constant nature. RMSBE was originally implemented in Java, for accurate performance comparison we compared Java version of cHawk with RMSBE.

Figure 7 shows the performance of both cHawk and RMSBE in terms of execution time for varying set of problem sizes. In case of cHawk, we can see that execution time only increases linearly with the problem size. This confirms our complexity analysis done in Section 6. On the other hand, the execution time for RMSBE increase at a much higher rate. cHawk clearly outperforms RMSBE for all problem sizes. In fact, cHawk runs approximately 10 times faster than RMSBE. The reason for this performance difference is that RMSBE is an exact algorithm. It first of all creates a similarity matrix based on input matrix, then attempts to find best suitable set of reference rows which can take time in the order of number of rows in worst case. Given that cHawk is comparable to RMSBE in accuracy, the performance difference solidifies the practical useful nature of cHawk.

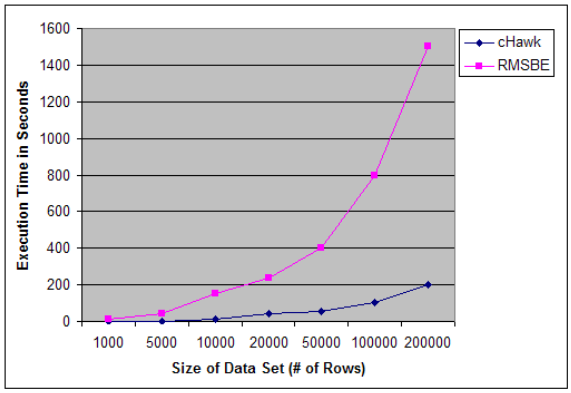


Figure 7: Performance of cHawk and RMSBE with increasing data sizes.

8. CONCLUSIONS

We have proposed to reformulate the optimal biclustering problem as "optimal crossing minimization in the representative bipartite graph". We have provided a theoretical connection between crossing minimization and spectral partitioning problems. Based on the new problem formulation, an efficient and scalable biclustering algorithm is proposed. Both accuracy and performance of the algorithm are tested and verified for synthetic as well as practical data sets. The experiments reveal that the algorithm not only outperforms traditional biclustering approaches in terms of execution time but also maintains good accuracy throughout.

9. REFERENCES

[1] Gene Ontology Consortium (2000). Gene ontology: tool for the unification of biology. *Nat. Genet.*, 25:25–29.

[2] A. Hussain A. Abdullah. A new biclustering technique based on crossing minimization. *Neurocomputing Journal*, 69:1882–1896, 2006.

[3] I. S. Dhillon A. Banerjee, S. Merugu and J. Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

[4] R. Karp A. Ben-Dor, B. Chor and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *Proceedings of the Sixth International Conference on Computational Molecular Biology (RECOMB)*, pages 49–57, 2002.

[5] P. Zimmermann A. Wille P. Buhlmann W. Gruissem L. Hennig L. Thiele E. Zitzler A. Prelic, S. Bleuler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 2006.

[6] M. Kupiec R. Shamir A. Tanay, R. Sharan. Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *PNAS*, 101(9):2981–2986, 2004.

[7] A. B. Tchagang A.H. Tewfik. Robust biclustering algorithm (roba) for dna microarray data analysis. In *Proceedings of IEEE Workshop on Statistical Signal Processing*, 2005.

[8] Waseem Ahmad and Ashfaq Khokhar. An architecture for privacy preserving collaborative filtering on web portals. In *The Third International Symposium on Information Assurance and Security (IAS)*, 2007.

[9] Y. Cheng and G.M. Church. Biclustering of expression data. In *Proceedings of Intelligent Systems for Molecular Biology*, 2000.

[10] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD)*, 2001.

[11] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Knowledge Discovery and Data Mining*, pages 269–274, 2001.

[12] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[13] M.R. Garey and D. S. Johnson. Crossing number is np-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4:312–316, 1983.

[14] A.P. Gasch. Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Biol. Cell*, 11:4241–4257, 2000.

[15] M. Gu, H. Zha, C. Ding, X. He, and H. Simon. Spectral relaxation models and structure analysis for k-way graph clustering and bi-clustering, 2001.

[16] Y. Guan H. Cho, I. S. Dhillon and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the fourth SIAM International Conference on Data Mining*, pages 114–125, 2004.

[17] S. Mallela I. S. Dhillon and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Miing (KDD)*, pages 89–98, 2003.

[18] S. Bergmann J. Ihmels and N. Barkai. Defining transcription modules using large-scale gene expression data. *Bioinformatics*, 20(13):1993–2003, 2004.

[19] Martin Juvan and Bojan Mohar. Optimal linear labelings and eigenvalues of graphs. *Discrete Appl. Math.*, 36(2):153–168, 1992.

[20] S. Tagawa K. Sugiyama and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transaction on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

[21] J. Lafferty, S. Pietra, and V. Pietra. Statistical learning algorithms based on bregman distances. In *Proceedings of the Canadian Workshop on Information Theory, 1997.*, 1997.

[22] F. Brglez M. Stallmann and D. Ghosh. Heuristics and experimental design for bigraph crossing number minimization. *Goodrich and McGeoch, editors, Algorithm Engineering and Experimentation*, LNCS 1619:74–93, 1999.

[23] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1, 2004.

[24] T.M. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. In *Proceedings of the 8th Pacific Symposium on Biocomputing*.

[25] Patrick O. Brown Orly Alter and David Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *PNAS*, 97:10101 – 10106, Aug 2000.

[26] A. Maron-Katz R. Sharan and R. Shamir. Click and expander: A system for clustering and visualizing gene expression data. *Bioinformatics*, 19(14):1787–1799, 2003.

[27] A. Prelic P. Zimmermann S. Barkow, S. Bleuler and E. Zitzler. Bicat: a biclustering analysis toolbox. *Bioinformatics*, 22(10):1282–1283, 2006.

[28] Farhad Shahrokhi, Ondrej Sýkora, László A. Székely, and Imrich Vrto. On bipartite drawings and the linear arrangement problem. *SIAM Journal on Computing*, 30(6):1773–1789, 2001.

[29] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[30] S. Merugu T. George. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, 2005.

[31] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 2001.

[32] Haixun Wang and Philip Yu. δ-clusters: Capturing subspace correlation in a large data set. In *Proceedings of the 18th IEEE International Conference on Data Engineering*, pages 517–528, 2002.

[33] L. Wang X. Liu. Computing the maximum similarity bi-clusters of gene expression data. *Bioinformatics*, 23(1):50–56, 2007.

[34] L. Wang X. Liu. Msbe software download. http://www.cs.cityu.edu.hk/ liuxw/msbe/help.html, 2007.