ISSN:0975-9646

BiFree: An Efficient Biclustering Technique for Gene Expression Data Using Two Layer Free Weighted Bipartite Graph Crossing Minimization

Suvendu Kanungo[#], Gadadhar Sahoo[*], Manoj Madhava Gore[#]

[#]*Department of Computer Science, BIT Mesra, Ranchi*
Allahabad Campus, India

[*]*Department of Information Technology*
BIT Mesra, Ranchi, India

***Abstract*— Conventional clustering technique for gene expression data provides a global view of the data. In the biological prospective, a local view is essential for better analysis of gene expression data with simultaneous grouping of genes and conditions. Several biclustering techniques have been proposed in the literature based on different problem formulation. Therefore, it is difficult to compare these techniques with respect to their biological significance, effectiveness and accuracy. In this paper, we have proposed a biclustering technique based on two layer free weighted crossing minimization of a bipartite graph. Using this technique, we can mine different types of biclusters amid noise and it works well in practice for real and synthetic gene expression data. The experimental evaluation reveals the accuracy and effectiveness of this technique with respect to noise handling and execution time in comparison to other biclustering approaches.**

***Keywords*— Bipartite Graph, Crossing Minimization, Clustering, Biclustering, Gene Expression Data, Microarray.**

I. INTRODUCTION

The DNA microarray technologies have makes it possible to study the expressions of thousands of genes over hundreds of experimental conditions. It can be used to study fundamental biological phenomena such as development or evolution, to determine the function of new genes, to infer the role of genes in diseases, and to monitor the effect of drugs and other compounds on gene expression. Data arising from microarray experiment, called gene expression data, are usually arranged in the form of a data matrix, where rows represent genes and columns represent experimental conditions. Each element of this matrix represents the expression level of a particular gene under a specific condition and denoted by a real number, which is usually the logarithm of relative abundance of mRNA of the gene under specific condition. The analysis of such high volume data matrix poses challenges to existing data mining tools. One of the most important data analysis tool, called cluster analysis, group data into a set of clusters. The goal of clustering is to separate a finite, unlabeled data set into a finite and discrete set of natural, hidden data structures. This powerful technique can be used to reveal biologically meaningful patterns in microarray data. Conventional one-way clustering methods are based on similarity among genes across all conditions. Recently, biclustering or co-clustering is commonly employed on microarray data, as genes may be co-regulated under limited conditions. An illustrative discussion on many of these methods can be found in [1].

The Minimum Linear Arrangement (MinLA) is commonly employed to solve the module replacement problem in a VLSI circuit layout, where the edges represent the wires and the nodes represent modules [2],[3]. MinLA is basically used here to minimize the total wire length, which is equivalent to crossing minimization in a bipartite graph [4]. The efficiency and effectiveness of biclustering algorithm largely depends on the problem formulation. Our method is based on weighted two layer free crossing minimization of a bipartite graph. The optimal crossing minimization is NP-hard [5] and as a result large numbers of heuristics and approximation algorithms have been devised for these problems with reasonable accuracy. Basically, two layer free crossing minimization problem is solved by iteratively applying the one layer free crossing minimization solution.

In this paper our contributions are:

- We model the gene expression data as a weighted bipartite graph by keeping experimental condition in the upper layer and genes in the lower layer.
- An efficient implementation of the proposed model, called BiFree, is provided. We employ barycenter heuristic [11] for the upper layer, and an approximation algorithm for the lower layer.
- An efficient algorithm for bicluster extraction is proposed. We provide an efficient local search based algorithm for bicluster extraction using conditional entropy, which can extract coherent, constant and overlapped biclusters.

II. RELATED WORK

Biclustering of gene expression data is a new field of research. Researchers have used different similarity criteria and techniques for biclustering.

Cheng and Church [6] identify biclusters with the help of mean squared residue score, which is a measure of the coherence of rows and columns in the bicluster. Here the user has to input a value of mean residue score δ and the number of biclusters to be extracted. This method involve several iterations and each iteration mine only a single bicluster while previously identified biclusters are masked with random values. However they did not address the issue of noisy data, where as in this paper we concentrate on noisy data.

Tanay et al. [7] introduced SAMBA, in which the data are modelled as a bipartite graph with genes corresponding to vertices in one bipartition and samples corresponding to vertices in other bipartition, where edges representing significant changes in expression. Edges and non-edges are weighted by likelihood scores derived from a probabilistic model for the bipartite graph. A bicluster is defined as a heavy subgraph, where the weight of the subgraph is the sum of the weights of the corresponding edges and non-edges. It repeatedly finds the maximal highly-connected subgraph in the bipartite graph and perform local improvement by adding or deleting a single vertex until no further improvement is possible. In order to avoid exponential runtime, they assumed that row vertices have d-bounded degree. However, our technique can handle graphs of arbitrary degrees.

Ben-Dor et al. [8] proposed OSPM, in which a bicluster is defined as a cluster of genes with the same rank profile across the biclustered samples. This method can mine large and statistically significant biclusters with the help of a greedy algorithm for identifying a fixed pattern of rows in a data set, one at a time. The time complexity of this technique is $O(nm^2l)$, where n and m are the number of rows and columns of the data matrix and l is the number of biclusters, which is slower than our approach.

Ahsan and Amir[9] identify biclusters by recursively removing noise with the help of crossing minimization technique. This method is based on binary representation of the bipartite graph corresponding to input data matrix. It is difficult to mine coherent biclusters, as this method use a static discretization of the input data matrix.

Wang et al. [10] proposed RMSBE, which can identify optimal square biclusters with the maximum similarity score. This method performs multiple scans of the data matrix in order to compute similarity score, reference gene identification and bicluster identification. The time complexity of this technique is $O(nm(n+m)^2)$, where n is number of rows and m is number of columns. Due to this cubic nature of complexity, it is not feasible for very high dimensional data. Prelic et al. [6] proposed BiMax, which can identify constant biclusters. This method discretize the input expression matrix into a binary matrix based on a threshold value. Therefore it is difficult to identify coherent biclusters.

Waseem and Asfaq [14] proposed cHawk, to identify biclusters with the help of crossing minimization paradigm. This method employs the barycenter heuristic to arrange vertices in both layers of a bipartite graph. The similarity test is done based on bregman divergence. This approach is similar to our approach as we also employ bipartite graph for representation of gene expression data. The time complexity of this technique is O(dnm), where n and m are the number of rows and columns of the input data matrix and d is the average degree of overlap among biclusters, which is slower than our approach.

Bergmann et al. [24] proposed the iterative signature algorithm (ISA) that uses gene signatures and condition signatures in order to extract biclusters with both up and down-regulated expression values. They identify several transcription modules (biclusters) by executing the algorithm on reference gene sets. The reference gene sets needs to be carefully selected for extraction of good quality biclusters.

III. MODEL FORMULATION

Bipartite Graph: A graph $G(V,E)$ is called Bipartite if its vertex set V can be decomposed into two disjoint subsets V_0 and V_1 (i.e. $V = V_0 \cup V_1$) such that every edge in E joins a vertex in V_0 with a vertex in V_1 (i.e. $V_0 \cap V_1 = \phi$).

Weighted Bipartite Graph: A graph $G(V_0,V_1,E,W)$ is called weighted bipartite graph if $W = (w_{ij})$ where $w_{ij} \geq 0$ denotes the weight of the edge (i, j) between vertices i and j .

Bipartite Drawing: A bipartite drawing or 2-layer drawing of $G(V_0,V_1,E)$ is a graph representation where the nodes of V_0 and V_1 are placed in two parallel lines $y = 0$ and $y = 1$, while the edges are drawn with straight lines between them.

Crossing Number: Let h be a bipartite drawing of bipartite graph $G(V_0,V_1,E)$. Let $bcr_h(e)$ represent the number of crossing of the edge $e \in E$ with other edges of E . Let $bcr(h)$ represent 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 bipartite drawings of G i.e. $bcr(G) = \min_h bcr(h)$.

Weighted Crossing: Let two edges $e_1, e_2 \in E$ of a bipartite drawing cross each other with nonnegative weights $w(e_1)$ and $w(e_2)$ respectively. Then, this crossing amount to $w(e_1) \times w(e_2)$ in the total weighted crossings.

The barycenter [11] and median heuristics [12] are the most popular heuristics for crossing minimization problem. A

survey of various heuristics has been taken up in detail in [13] and shown that the barycenter method produces better results than the median heuristics. The barycenter heuristic orders the vertices in upper layer V_1 by computing the averages of the positions of the adjacent vertices in lower layer V_0. This technique is employed when the ordering of vertices in both layers of a bipartite graph is to be determined. As a result, similar vertices come closer and form biclusters. Each iteration of this heuristic can be implemented in $O(|E|+|V|\log|V|)$ time. In gene expression data, the numbers of experimental conditions are very few in comparison to the number of genes. This motivates us to place experimental conditions in the upper layer (V_1) and genes in the lower layer (V_0) of the bipartite graph. We employ weighted version of barycenter heuristic to reorder vertices of upper layer for a single iteration. This crossing minimization process brings similar vertices in the vicinity of each other and helps us to achieve biclustering in less time. In order to rearrange the vertices of lower layer, we have employed an approximation algorithm that brings similar vertices in the vicinity of each other, with a constant approximation ratio of 3.

Let $a \in V_1$ and A_a denotes the set of neighbours of a. Also let n and m be the size of lower and upper layer of the bipartite graph respectively. We assume that nodes in V_0 and V_1 are labeled from 1 to n and 1 to m respectively. The new label of a denoted as R can be computed as follows:

$$R = \sum_{i=1}^{|A_a|} w(aA_a[i] \times A_a[i]) / \sum_{i=1}^{|A_a|} w(aA_a[i]) \quad (1)$$

where $A_a[i]$ denotes the i^{th} neighbour's label of a. Here we have assumed the weight to be binary. If an edge exit, the weight is considered as 1 otherwise 0.

Let $c, d \in V_0$ such that c is placed to the left of d. Here we have assumed the weight to be the real weight, which is weight of an edge. If M_{cd} denote the sum of the weighted crossings between the edges incident on c and d, it can be computed as follows:

$$M_{cd} = \sum_{k=1}^{m-1} \sum_{j=k+1}^{m} w(cj) \times w(dk) \quad (2)$$

We perform hard partitioning of V_0 into m sets $S_0 \ldots S_{m-1}$, such that:

$$S_0 = \left\{ c \in V_0 \middle| w(c1) \geq \sum_{j=2}^{m} w(cj) \right\} \quad (3)$$

$$S_q = \left\{ c \in V_0 \middle| \sum_{j=1}^{q-1} w(cj) \langle \sum_{j=q+1}^{m} w(cj) and \sum_{j=1}^{q} w(cj) \geq \sum_{j=q+2}^{m} w(cj) \right\} \quad (4)$$

where $0 \leq q \leq m-1$, and $q \geq 1$.

We have ordered the partitions in non decreasing order of their indices from left to right. After placing the vertices in their respective partitions, the correct position of any vertex inside a partition can be determined by the following condition:

$$\sum_{j=1}^{q} w(dj) \times \sum_{j=q+1}^{m} w(cj) \leq \sum_{j=1}^{q} w(cj) \times \sum_{j=q+1}^{m} w(dj) \quad (5)$$

If the equation 5 holds, then c is placed to the left of d, otherwise c is placed to the right of d. The pseudocode for the two layer free weighted crossing minimization is given in Algorithm I.

Algorithm I: Two Layer Free Weighted Crossing Minimization
Input: *Bipartite graph G*
Output: *Reordered bipartite graph G′*

1: *forall* $a \in V_1$ *do*
2: $R \leftarrow \sum_{i=1}^{|A_a|} w(aA_a[i]) \times A_a[i] / \sum_{i=1}^{|A_a|} w(aA_a[i])$
3: *if* $A_a[i] \neq R$ *then*
4: $A_a[i] = R$
5: *endif*
6: *endfor*
7: *sort label of vertices in* V_1
8: *forall* $c \in V_0$ *do*
9: $W_r \leftarrow \sum_{i=1}^{|A_c|} w(cA_c[i])$; $W_l \leftarrow 0$
10: *for* $t: 1 \rightarrow |A_c| - 1$ *do*
11: $r = A_c[t] - 1$
12: $W_r \leftarrow W_r - w(cA_c[t])$
13: *if* $W_l \geq W_r$ *then goto step* 20
14: $W_l \leftarrow W_l + w(cA_c[t])$
15: *if adjacent vertices are consecutive then*
16: *if* $W_l \geq (W_r - w(cA_c[t+1]))$
17: $r \leftarrow r + 1$
18: *goto step* 20
19: *endfor*
20: *if* S_r *is empty then* $S_r \leftarrow S_r \cup \{c\}$
21: *else*
22: *if* $d \in S_r$ *then*
23: *if* $\sum_{j=1}^{r} w(dA_d[j]) \times \sum_{j=r+1}^{m} w(cA_c[j]) \leq \sum_{j=1}^{r} w(cA_c[j]) \times \sum_{j=r+1}^{m} w(dA_d[j])$ *then*
24: *place c to the right of d in* S_r
25: *else*
26: *place c to the left of d in* S_r
27: *endfor*
28: *Reorder vertices in* V_0 *in the order* $S_0, S_1, \cdots S_{m-1}$

IV. IMPLEMENTATION

For the implementation of the proposed model, our technique consists of the following basic steps:

a. Preprocessing of data in D

b. Weighted crossing minimization of G to get G'

c. Extraction of biclusters from G'

We have implemented our proposed algorithm in C++ under windows environment on a computer with configuration of Core 2 Duo 2.2 GHz of CPU and 3 GB RAM. We evaluate its accuracy and performance using synthetically generated dataset and real dataset. For real gene expression dataset, we have considered the model organism Saccharomyces Cerevisiae, provided by Gasch et al. [17], since the yeast GO annotations are more extensive compared to other organisms. This gene expression dataset contains 2,993 genes and 173 different stress conditions.

A. *Preprocessing of Data*

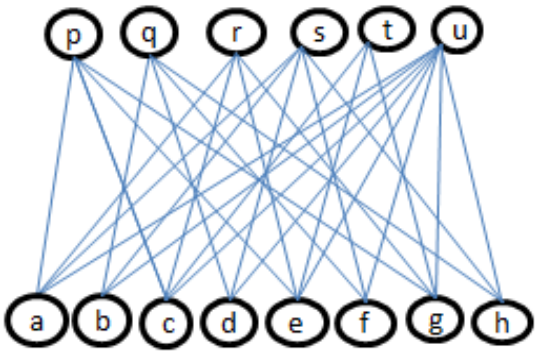
Gene expression data is usually noisy and may contain missing values. In order to deal with these missing values, we have adopted the simple approach used in [15], where all missing values are replaced by zero. Further, preparing data for cluster analysis requires transformation, such as standardization or normalization. Before normalizing data, we have temporarily removed data beyond a threshold value (three standard deviation), to reduce the effect of outliers in the data. Then, the gene expression data is transformed using z-score standardization, where the transformed variables have a mean of 0 and variance of 1. Finally, the temporarily removed outliers that are below the mean value are replaced by the minimum value, where as the outliers above the mean values are replaced by the maximum value of the final normalized data. Further, to handle outliers more efficiently, we have adopted the approach used in [16], where the normalized sample data is partitioned into unequal length intervals based on mean value of the partition. Here, we discertize each gene expression level into unequal length intervals, in order to handle extreme outlier values.

B. *Crossing Minimization*

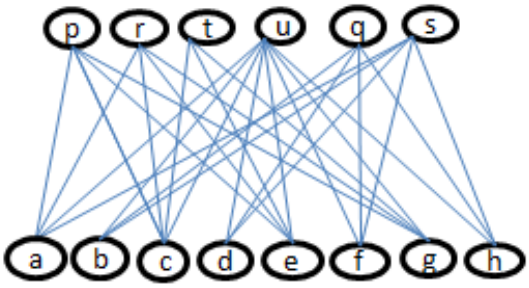
Most of the crossing minimization takes place in the first few iterations and hence we employ the weighted barycenter heuristic on the upper layer for only one iteration to place similar vertices in the vicinity of each other. We decide the correct partition for a node in the lower layer based on its left and right weighted sum value. Starting from the first adjacent node of any node in the lower layer, we increase the left weighted sum each time and decrease the right weighted sum value whenever necessary. When the left weighted sum is greater than or equal to the right weighted sum, then the node is placed in the corresponding partition. Finally, based on the amount of weighted crossing between any two nodes in the lower layer, we decide the correct place of a node inside the partition. Fig. 1 illustrates the application of Algorithm I for crossing minimization that leads to rearrangement of vertices in both top and bottom layer of the bipartite graph. Fig. 1(c) shows the rearrangement of top layer vertices whereas Fig. 1(d) shows the rearrangement of bottom layer vertices.

	p	q	r	s	t	u
a	0.5	0	1	0	1.5	2
b	0	3	0	3	0	3
c	0.5	0	1	0	1.5	2
d	0	3	0	3	0	3
e	0.5	0	1	0	1.5	2
f	0	3	0	3	0	3
g	0.5	0	1	0	1.5	2
h	0	3	0	3	0	3

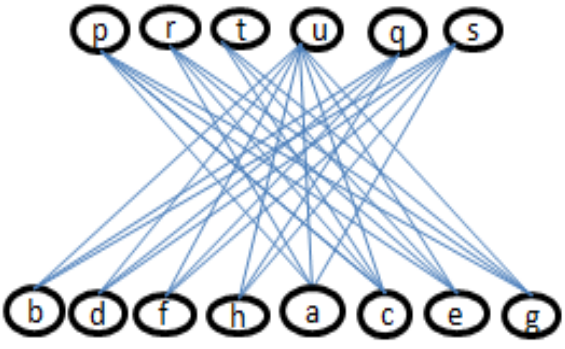
(a)



(b)



(c)



(d)

	p	r	t	u	q	s
b	0	0	0	3	3	3
d	0	0	0	3	3	3
f	0	0	0	3	3	3
h	0	0	0	3	3	3
a	0.5	1	1.5	2	0	0
c	0.5	1	1.5	2	0	0
e	0.5	1	1.5	2	0	0
g	0.5	1	1.5	2	0	0

(e)

Fig. 1 (a) Input data matrix before crossing minimization (b) Bipartite graph representation of input data matrix (c) Bipartite graph after application of barycenter heuristic to the top layer (d) Bipartite graph after application of approximation algorithm to the bottom layer (e) Input data matrix after crossing minimization of the top and bottom layer.

Finally, Fig. 1(e) shows the formation of two biclusters i.e. {b,d,f,h}×{u,q,s} and {a,c,e,g}×{p,r,t,u}, which can be extracted by our proposed bicluster extraction algorithm i.e. Algorithm II. Here, {b,d,f,h}×{u,q,s}is a constant bicluster whereas {a,c,e,g}×{p,r,t,u} is a coherent bicluster.

C. *Bicluster Extraction*

After we employ Algorithm I for crossing minimization, vertices of both layers for the bipartite graph are rearranged so that similar rows and columns come close to each other. This process enables us to apply a local search algorithm for extraction of constant, coherent and overlapping biclusters. In information theory, the entropy plays a vital role as measures of information and uncertainty. It is a measure of the average uncertainty in the random variable X [21] and can be defined as

$$H(X) = \sum_{x} p(x) \log p(x), \qquad (6)$$

where p(x) is the probability mass function of X.

Cheng et al. [22] used entropy to evaluate and prune subspaces for clustering. As similar objects form a cluster, the entropy of a cluster tends to be very low. This motivates us to employ entropy measure for the extraction of biclusters. In order to detect mutual interaction between two genes $(c \text{ and } d)$, we employ conditional entropy measure, which is defined as

$$H(c|d) = \sum_{l=1}^{L} p(c^l) \sum_{v=1}^{L} p(c^v|d^l) \log p(c^v|d^l), \qquad (7)$$

where L is the number of discretization levels, $p(c^l)$ is the probability of values in discretization level l of gene c, and $p(c^v|d^l)$ is the conditional probability of the values in the interval u of gene c given values in interval l of gene d.

Our technique is row major, as we compare two genes simultaneously for similarity. We keep reference set of matching conditions based on threshold value (ρ) of the conditional entropy. Conditions are added to this set, if the conditional entropy is less than ρ . We also keep a gene cluster in order to add similar genes that contain similar conditions. If subsequent genes contain more matching conditions\, then the overlap flag is set. When a sub matrix contains minimum number genes and conditions, and satisfies the threshold value, we declare it as a bicluster and store the same in a bicluster set. Pseudocode for bicluster extraction is given in Algorithm II.

Algorithm – II : Bicluster Extraction
Input : An $n \times m$ discretized and reordered matrix D', minimum
// number of genes ($g_{\min}$), minimum number of conditions ($c_{\min}$),
// entropy threshold ρ
Output : A set of biclusters
1 : *sgene* $\leftarrow 0$; *scon* $\leftarrow 0$; *overlap* $\leftarrow$ *false*
2 : *forall* i : *sgene* $\rightarrow n-2$ *do*
3 : *forall* j : *scon* $\rightarrow m-1$ *do*
4 : *compute conditional entropy between gene i and $i+1$*
using eq. 7 for condition set $0 \rightarrow j$
5 : *if* i = *sgene then*
6 : *if conditional entropy* $< \rho$ *then*
7 : *reference condition set* $(c_r) \leftarrow j$
8 : *if* $j = m-1$ *then gene cluster* $(g_c) \leftarrow i$ *and* $i+1$
9 : *if* $i >$ *sgene then*
10 : *if conditional entropy* $< \rho$ *then*
11 : *current condition cluster* $(c_{cc}) \leftarrow j$
12 : *if* $j = m-1$ *then*
13 : *if* c_{cc} *contain similar conditions then*
14 : $c_r \leftarrow c_{cc}$, $g_c \leftarrow i+1$
15 : *if* c_{cc} *is empty then goto step* 19
16 : *if* c_{cc} *contain more similar conditions then*
17 : *overlap* $\leftarrow$ *true* , *sgene* $\leftarrow i$, *scon* $\leftarrow j$
18 : *if* $i = n-2$ *goto step* 19
19 : *if* $|g_c| \geq g_{\min}$ *AND* $|c_r| \geq c_{\min}$ *then*
20 : $B \leftarrow g_c$, c_r ; *remove* g_c , c_{cc} *and* c_r
21 *if overlap* = *true then*
22 : $i \leftarrow$ *sgene* ; $j \leftarrow$ *scon* ; *overlap* $\leftarrow$ *false*
23 : *endfor*
24 : *endfor*
25 : *Return B*

V. EVALUATION FRAMEWORK

A. *Complexity Analysis*

The time complexity for partitioning the conditions of input data matrix into unequal length intervals is $O(n \log n)$. In the first stage for Algorithm I, we rearrange the layer V_1 nodes of pre-processed gene expression data by weighted barycenter

method. This process would take time $O(|E| + (m+n)\log(m+n))$. In the second stage, we rearrange layer V_0 nodes into m partitions. The correct partitions for a node in V_0, which occur when W_l value is larger than W_r, would take time $O(|E| + m + n)$. Then placing this node in correct position in the corresponding partition would take time $O(n + m\log m)$. Therefore, the total time for discretization into unequal length intervals and reordering the rows of D would take $O(|E| + m + n + m\log m + n\log n)$. For Algorithm II, it involve extraction of biclusters and would take time $O(mn)$. Thus, the overall time complexity of the proposed two layer free crossing minimization based biclustering which is denoted as T_{BiFree} and defined as:

$$T_{BiFree} = O(|E| + m + n + m\log m + n\log n) + O(mn).$$

The second term $O(mn)$ tends to dominate, which shows that the time complexity has linear relationship with size of the given problem.

B. Synthetic Dataset

In case of synthetic gene expression data, we used the technique proposed by Zimmermann et al. [18] for evaluation of implanted constant, coherent and overlap biclusters. For constant bicluster generation, we adopt [23] the following steps:

a. Generate a 100 × 100 matrix A with all elements 0
b. Generate ten biclusters (modules) of size 10 × 10 with all elements 1
c. Replace elements of biclusters with random noise values from uniform distribution $(-\delta, \delta)$
d. Implant the ten biclusters into A without overlap

For all experimentation, we set the noise level range from 0.0 to 0.25. In case of overlapping biclusters, we used 10 degrees of overlap (o_d = 0,1,2,3,4,5,6,7,8,9) , where the size of matrix and bicluster vary from 100 × 100 to 110 × 110 and from 10 × 10 to 20 × 20, respectively. The steps for evaluation of coherent biclusters are same as that of constant bicluster, but rows and columns in a bicluster have a 0.02 increasing trend. In order to validate the accuracies of different algorithms, we apply the gene match score proposed by Zimmermann et al. [18]. Let M_1 and M_2 be two sets of biclusters. The match score of M_1 with respect to M_2 is given by:

$$S_G(M_1, M_2) = \frac{1}{|M_1|} \sum_{(G_1,S_1)\in M_1} \max_{(G_2,S_2)\in M_2} \frac{|G_1 \cap G_2|}{|G_1 \cup G_2|}, \quad (8)$$

where G and S are set of genes and a set of samples in a bicluster respectively. Let M_{opt} represent the set of implanted biclusters and M be the set of output biclusters of an algorithm. The score $S(M, M_{opt})$ represents the degree of similarity between extracted biclusters and the implanted biclusters, where as the score $S(M_{opt}, M)$ represents how well each of the true biclusters extracted by the bicluster algorithm. As per our experimental results, in case of high noise level for extraction of constant biclusters; cHawk, BiFree, ISA and RMSBE shows high accuracies; BiMax and SAMBA perform moderately, and CC perform poorly. For coherent biclusters, BiFree has a comparable accuracy with cHawk and RMSBE. In case of overlapped biclusters, BiFree is marginally affected by the overlap degree of the implanted biclusters.

C. Real Dataset

We have adopted the approach used by Zimmermann et al. [18] to evaluate the performance of BiFree with other algorithms for real gene expression data, provided by Gasch et al. [17]. In order to evaluate extracted biclusters based on Gene Ontology (GO) annotations [19] for their enrichment level, we have also used a web tool called FuncAssociate [20]. The adjusted significance scores (α) were computed using FuncAssociate and is shown in Fig. 2. Based on this score, the results for BiFree is compared with other algorithms like cHawk, BiMax, RMSBE, OSPM, SAMBA and CC.

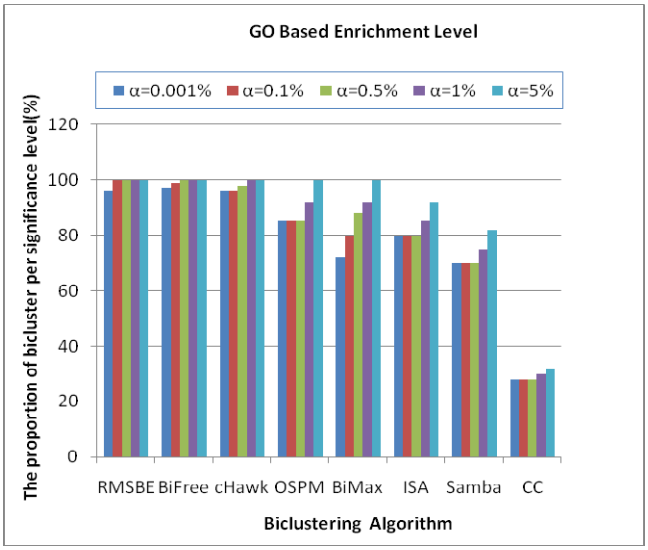


Fig. 2 Proportion of GO based enriched bicluster

D. Performance of BiFree

In this section we analyse the performance of our proposed BiFree algorithm. We have synthetically generated datasets with sizes ranging from 2000 × 100 to 100000 × 500 and implant constant biclusters in this matrix. Fig. 3 illustrates the performance of BiFree, cHawk and RMSBE with respect to execution time for different size of dataset. As per our complexity analysis, the execution time of BiFree and cHawk increase linearly with size of the dataset, while execution time for RMSBE increases at a much higher rate. This confirms the practical applicability of our proposed algorithm.

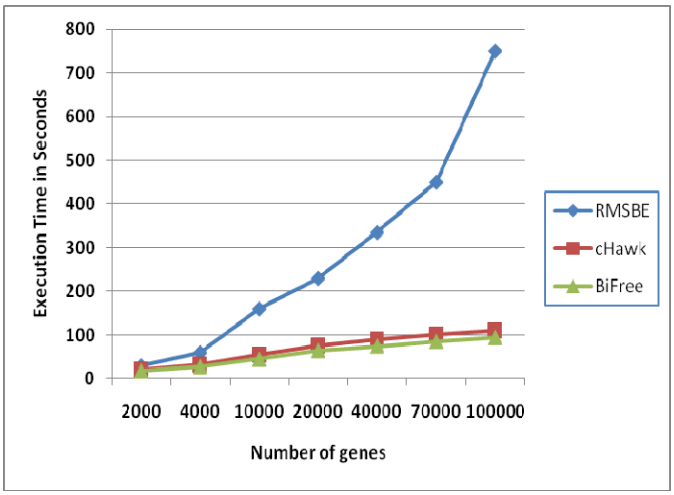


Fig. 3 Performance of BiFree, cHawk and RMSBE

VI. Conclusions

We have proposed and implemented a biclustering technique, called BiFree, in order to extract constant, coherent and overlapping biclusters. The technique employs weighted two layer free crossing minimization on the conditioned gene expression data, so that similar genes and conditions come close to each other that form different types of biclusters. In order to extract these biclusters, we have proposed a local search algorithm based on conditional entropy. We have verified the accuracy and performance of our algorithm for synthetic and real gene expression data sets. The experimental results reveal that our technique outperforms other conventional techniques in terms noise removal and execution time.

Acknowledgment

The authors are grateful to the reviewers for their invaluable suggestions that have helped immensely to improve the quality of this paper.

References

[1] S. C. Madeira and A. L. Oliveira, "Biclustering algorithms for biological data analysis: a survey", IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 1(1), pp. 24-45, 2004.

[2] D. Adolphson and T. C. Hu, "Optimal linear ordering", SIAM Journal on Applied Mathematics, vol. 25(3), pp. 403–423, Nov. 1973.

[3] L. H. Harper, "Chassis layout and isoperimetric problems", Technical Report SPS, pp. 37–66, vol II, Jet Propulsion Laboratory, 1970.

[4] J. Pach, F. Shahrokhi, and M. Szegedy, "Applications of the crossing number", Algorithmica,, vol. 16, pp. 111–117, 1996.

[5] M. R. Garey and D. S. Johnson, "Crossing number is np-complete", SIAM Journal on Algebraic and Discrete Methods, vol. 4, pp. 312-316, 1983.

[6] Y. Cheng and G. M. Church, "Biclustering of expression data", Proceedings of 8th International Conference on Intelligent Systems for Molecular Biology, pp. 93-103, 2000.

[7] A. Tanay, R. Sharan and R. Shamir, Discovering statistically significant biclusters in gene expression data. Bioinformatics, vol. 18, pp. S136-S144, 2002.

[8] A. Ben-Dor, B. Chor, R. Karp and Z. Yakhini,, "Discovering Local Structure in Gene Expression Data: The Order-Preserving Sub-matrix Problem", Proceedings of Sixth International Conference on Computational Molecular Biology (RECOMB '02). pp. 49-57, 2002.

[9] A. Hussain and A. Abdullah, "A new biclustering technique based on crossing minimization", Neurocomputing Journal, vol. 69, pp. 1982-1896, 2006.

[10] L. Wang and X. Liu, "Computing the maximum similarity bi-clusters of gene expression data", Bioinformatics, vol. 23(1), pp. 50-56, 2007.

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

[12] P. Eade and N. Wormald, "The Median heuristic for drawing 2-layers networks", Technical Report 69, Department of Computer Science, University of Queensland, Brisbane, Australia, 1986.

[13] M. Jünger and P. Mutzel, "2-layer straightline crossing minimization, Performance of exact and heuristic algorithms", Journal of Graph Algorithms and Applications, vol. 1(1), pp. 1-25, 1997.

[14] W. Ahmad and A. Khokhar, "cHawk : An efficient biclustering algorithm based on bipartite graph crossing minimization", VLDB, 2007.

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

[16] S. Kanungo, G. Sahoo and M.M. Gore, "BiCross: A Biclustering Technique for Gene Expression Data Using One Layer Fixed Weighted Bipartite Graph Crossing Minimization", International Journal of Computer Applications, vol. 29(4), pp. 28-34, 2011.

[17] A.P. Gasch,. "Genomic expression programs in the response of yeast cells to environmental changes", Molecular Biology Cell, vol. 11, 4241-4257, 2000.

[18] P. Zimmermann, A. Wille, P. Buhlmann, W. Gruissem, L. Hennig, L.Thiele, E. Zitzler, A. Prelic and S. Bleuler, "A systematic comparison and evaluation of biclustering methods for gene expression data. Bioinformatics", vol. 23(1), pp. 50-56, 2007.

[19] Gene Ontology Consortium, "Gene ontology: tool for the unification of biology", *Natural Genetics*, vol. 25, pp. 25-29, 2000.

[20] G. Berriz, O. Bryant, C. Sander and F. Roth, "Charactering gene sets with FuncAssociate", Bioinformatics, vol. 22, pp. 1282-1283, 2003.

[21] T. M. Cover and J. A. Thomas, "Elements of Information Theory", Wiley, 2006.

[22] C. Cheng, A. Fu and Y. Zhang, "Entropy-based subspace clustering for mining numerical data", Proceeding of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999.

[23] A. Olomola and S. Dua, "Bi-clustering of gene expression data using conditional entropy", LNBI, Springer-Verlag, pp. 224-254, 2009.

[24] S. Bergmann, J. Ihmels and N. Barkai,. "Defining transcription modules using large-scale gene expression data", Bioinformatics, vol. 20(13), pp. 1993-2003, 2004.