Biclustering Algorithms: A Survey

Amos Tanay* Roded Sharan† Ron Shamir*

May 2004

Abstract

Analysis of large scale geonomics data, notably gene expression, has initially focused on clustering methods. Recently, biclustering techniques were proposed for revealing submatrices showing unique patterns. We review some of the algorithmic approaches to biclustering and discuss their properties.

1 Introduction

Gene expression profiling has been established over the last decade as a standard technique for obtaining a molecular fingerprint of tissues or cells in different biological conditions [18, 7]. Based on the availability of whole genome sequences, the technology of DNA chips (or microarrays) allows the measurement of mRNA levels simultaneously for thousands of genes. The set (or vector) of measured gene expression levels under one condition (or sample) are called the *profile* of that condition. Gene expression profiles are powerful sources of information and have revolutionized the way we study and understand function in biological systems [1].

Given a set of gene expression profiles, organized together as a *gene expression matrix* with rows corresponding to genes and columns corresponding to conditions, a common analysis goal is to group conditions and genes into subsets that convey biological significance. In its most common form, this task translates to the computational problem known as *clustering*. Formally, given a set of elements with a vector of attributes for each element, clustering aims to partition the elements into (possibly hierarchically ordered) disjoint sets, called clusters, so that within each set the attribute vectors are similar, while vectors of disjoint clusters are dissimilar. For example, when analyzing a gene expression matrix we may apply clustering to the genes (as elements) given the matrix rows (as attributes) or cluster the conditions (as elements) given the matrix columns (as attributes). For reviews on clustering see an earlier chapter in this book. Analysis via clustering makes several a-priori assumptions that may not be perfectly adequate in all circumstances. First, clustering can be applied to either genes or samples, implicitly directing the analysis to a particular

*School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. {amos,rshamir}@post.tau.ac.il.

†International Computer Science Institute, 1947 Center St., Berkeley CA 94704, USA. roded@icsi.berkeley.edu

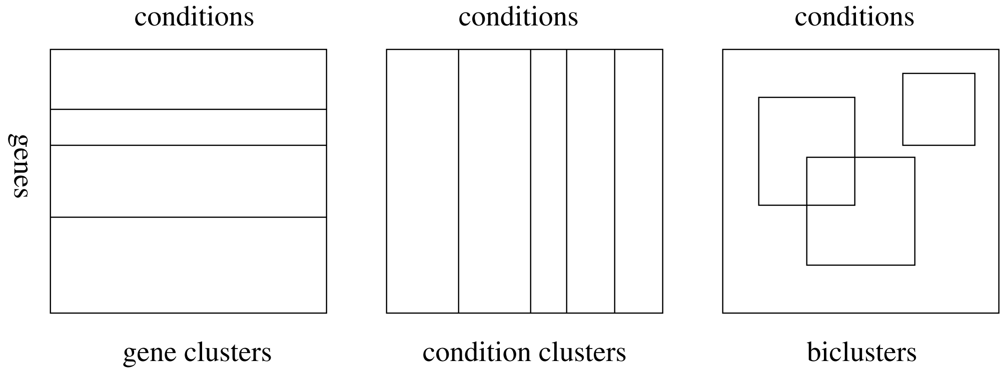

Figure 1: Clustering and biclustering of a gene expression matrix. Clusters correspond to disjoint strips in the matrix. A gene cluster must contain all columns, and a condition cluster must contain all rows. Biclusters correspond to arbitrary subsets of rows and columns, shown here as rectangles. Note that since gene (condition) clusters are disjoint, the rows (columns) of the matrix can be reordered so that each cluster is a contiguous strip. Similar reordering of rows and columns that shows all the biclusters as rectangles is usually impossible.

aspect of the system under study (e.g., groups of patients or groups of co-regulated genes). Second, clustering algorithms usually seek a disjoint cover of the set of elements, requiring that no gene or sample belongs to more than one cluster.

The notion of a bicluster gives rise to a more flexible computational framework. A *bicluster* is defined as a submatrix spanned by a set of genes and a set of samples (compare Figure 1). Alternatively, a bicluster may be defined as the corresponding gene and sample subsets. Given a gene expression matrix, we can characterize the biological phenomena it embodies by a collection of biclusters, each representing a different type of joint behavior of a set of genes in a corresponding set of samples. Note that there are no a-priori constraints on the organization of biclusters and in particular, genes or samples can be part of more than one bicluster or of no bicluster. The lack of structural constrains on biclustering solutions allows greater freedom but is consequently more vulnerable to overfitting. Hence, biclustering algorithms must guarantee that the output biclusters are meaningful. This is usually done by an accompanying statistical model or a heuristic scoring method that define which of the many possible submatrices represent a significant biological behavior. The *biclustering problem* is to find a set of significant biclusters in a matrix.

In clinical applications, gene expression analysis is done on tissues taken from patients with a medical condition. Using such assays, biologists have identified molecular fingerprints that can help in the classification and diagnosis of the patient status and guide treatment protocols [2, 16]. In these studies, the focus is primarily on identifying profiles of expression over a subset of the genes that can be associated with clinical conditions and treatment outcomes, where ideally, the set of samples is equal in all but the subtype or the stage of the disease. However, a patient may be a part of more than one clinical group, e.g., may suffer from syndrome A, have a genetic background B and be exposed to environment C. Biclustering analysis is thus highly appropriate for identifying and distinguishing the biological factors affecting the patients along with the corresponding gene subsets.

In functional genomics applications, the goal is to understand the functions of each of the genes operating in a biological system. The rationale is that genes with similar expression patterns are likely to be regulated by the same factors and therefore may share function. By collecting expression profiles from many different biological conditions and identifying joint patterns of gene expression among them, researchers have characterized transcriptional programs and assigned putative function to thousands of genes [23, 11, 8]. Since genes have multiple functions, and since transcriptional programs are often based on combinatorial regulation, biclustering is highly appropriate for these applications as well.

An important aspect of gene expression data is their high noise levels. DNA chips provide only rough approximation of expression levels, and are subject to errors of up to two-fold the measured value [1]. Any analysis method, and biclustering algorithms in particular, should therefore be robust enough to cope with significant levels of noise.

Below we survey some of the biclustering models and algorithms that were developed for gene expression analysis. Our coverage is not exhaustive, and is biased toward what we believe are the more practical methods. We attempt to cover at least one method from each class of algorithms under development. We do not review methods that are based on extended biological models (e.g., inferring regulation or integrating data types [19, 24]), but focus on algorithms for biclustering per-se. Throughout, we assume that we are given a set of genes V a set of conditions U, and a gene expression matrix $E = (e_{vu})$ where e_{vu} is the expression level of gene v in sample u. We assume that the matrix is normalized, though some of the algorithms below perform additional normalization. A *bicluster* $B = (U', V')$ is defined by a subset of genes $V' \subset V$ and a subset of conditions (or samples) $U' \subset U$. Different algorithmic approaches to the biclustering problem use different measures for the quality of a given biclustering solution. We therefore define the goal function of each algorithm as part of its description.

2 Cheng and Church's Algorithm

Cheng and Church were the first to introduce biclustering to gene expression analysis [6]. Their algorithmic framework represents the biclustering problem as an optimization problem, defining a score for each candidate bicluster and developing heuristics to solve the constrained optimization problem defined by this score function. In short, the constraints force the uniformity of the matrix, the procedure gives preference to larger submatrices and the heuristic is a relaxed greedy algorithm.

Cheng and Church implicitly assume that (gene, condition) pairs in a "good" bicluster have a constant expression level, plus possibly additive row and column specific effects. After removing row, column and submatrix averages, the residual level should be as small as possible. More formally, given the gene expression matrix E, a subset of genes I and a subset of conditions J, we define $e_{Ij} = \frac{\sum_{i \in I} e_{ij}}{|I|}$ (row subset average) $e_{iJ} = \frac{\sum_{j \in J} e_{ij}}{|J|}$ (column subset average) and $e_{IJ} = \frac{\sum_{i \in I, j \in J} e_{ij}}{|I||J|}$ (submatrix average). We define the *residue score* of an element e_{ij} in a submatrix E_{IJ}

as $RS_{IJ}(i,j) = e_{ij} - e_{Ij} - e_{iJ} + e_{IJ}$ and the *mean square residue score* of the entire submatrix as $H(I,J) = \sum_{i \in I, j \in J} \frac{RS_{ij}^2}{|I||J|}$. The intuition behind this definition can be understood via two examples: a completely uniform matrix will have score zero. More generally, any submatrix in which all entries have the form $e_{ij} = b_i + c_j$ would also have score zero. Given the score definition, the *maximum bicluster problem* seeks a bicluster of maximum size among all biclusters with score not exceeding a threshold δ. The size can be defined in several ways, for example as the number of cells in the matrix ($|I||J|$) or the number of rows plus number of columns ($|I| + |J|$).

The maximum bicluster problem is NP-hard if we force all solutions to be square matrices ($|I| = |J|$) or if we use the total number of submatrix cells as our optimization goal (Reductions are from Maximum Balanced Biclique or Maximum Edge Biclique). Cheng and Church suggested a greedy heuristic to rapidly converge to a locally maximal submatrix with score smaller than the threshold. The algorithm (presented in Figure 2) can be viewed as a local search algorithm starting from the full matrix. Given the threshold parameter δ, the algorithm runs in two phases. In the first phase, the algorithm removes rows and columns from the full matrix. At each step, where the current submatrix has row set I and column set J, the algorithm examines the set of possible moves. For rows it calculates $d(i) = \frac{1}{|J|}\sum_{j \in J} RS_{I,J}(i,j)$ and for columns it calculates $e(j) = \frac{1}{|I|}\sum_{i \in I} RS_{I,J}(i,j)$. It then selects the highest scoring row or column and removes it from the current submatrix, as long as $H(I,J) > \delta$. The idea is that rows/columns with large contribution to the score can be removed with guaranteed improvement (decrease) in the total mean square residue score. A possible variation of this heuristic removes at each step all rows/columns with a contribution to the residue score that is higher than some threshold.

In the second phase of the algorithm, rows and columns are being added, using the same scoring scheme, but this time looking for the lowest square residues $d(i), e(j)$ at each move, and terminating where none of the possible moves increases the matrix size without crossing the threshold δ. Upon convergence, the algorithm outputs a submatrix with low mean residue and locally maximal size.

To discover more than one bicluster, Cheng and Church suggested repeated application of the biclustering algorithm on modified matrices. The modification includes randomization of the values in the cells of the previously discovered biclusters, preventing the correlative signal in them to be beneficial for any other bicluster in the matrix. This has the obvious effect of precluding the identification of biclusters with significant overlaps.

An application of the algorithm to yeast and human data is described in [6]. The software is available at http://arep.med.harvard.edu/biclustering.

Cheng-Church(U, V, E, δ):
U : conditions. V : genes.
E : Gene expression matrix.
δ: maximal mean square residue score.
Define $e_{Ij} = \frac{\sum_{i \in I} e_{ij}}{|I|}$
Define $e_{iJ} = \frac{\sum_{j \in J} e_{ij}}{|J|}$
Define $e_{IJ} = \frac{\sum_{i \in I, j \in J} e_{ij}}{|I||J|}$
Define $RS_{IJ}(i, j) = e_{ij} - e_{Ij} - e_{iJ} + e_{IJ}$
Define $H(I, J) = \sum_{i \in I, j \in J} \frac{RS_{ij}^2}{|I||J|}$.
Initialize a bicluster (I, J) with $I = U, J = V$.
Deletion phase:
    While $(H(I, J) > \delta)$ do
        Compute for $i \in I$, $d(i) = \frac{1}{|J|} \sum_{j \in J} RS_{I,J}(i, j)$.
        Compute for $j \in J$, $e(j) = \frac{1}{|I|} \sum_{i \in I} RS_{I,J}(i, j)$.
        If $max_{i \in I} d(i) > max_{j \in J} e(j)$ assign $I = I \setminus \{argmax_i(d(i))\}$.
        Else $J = J \setminus \{argmax_j(e(j))\}$
Addition phase:
    assign $I' = I, J' = J$
    While $(H(I', J') < \delta)$ do
        Assign $I = I', J = J'$
        Compute for $i \in U \setminus I$, $d(i) = \frac{1}{|J|} \sum_{j \in J} RS_{I,J}(i, j)$.
        Compute for $j \in V \setminus J$, $e(j) = \frac{1}{|I|} \sum_{i \in I} RS_{I,J}(i, j)$.
        If $max_{i \in I} d(i) < max_{j \in J} e(j)$ assign $I' = I \cup \{argmax_i(d(i))\}$.
        Else $J' = J \cup \{argmax_j(e(j))\}$
Report I, J

Figure 2: The Cheng-Church algorithm for finding a single bicluster.

3 Coupled Two-way Clustering

Coupled two-way clustering (CTWC), introduced by Getz, Levine and Domany [9], defines a generic scheme for transforming a one-dimensional clustering algorithm into a biclustering algorithm. The algorithm relies on having a one-dimensional (standard) clustering algorithm that can discover significant (termed *stable* in [9]) clusters. Given such an algorithm, the coupled two-way clustering procedure will recursively apply the one-dimensional algorithm to submatrices, aiming to find subsets of genes giving rise to significant clusters of conditions and subsets of conditions giving rise to significant gene clusters. The submatrices defined by such pairings are called *stable submatrices* and correspond to biclusters. The algorithm, which is shown in Figure 3, operates on a set of gene subsets $\mathcal{V}$ and a set of condition subsets $\mathcal{U}$. Initially $\mathcal{V} = \{V\}$ and $\mathcal{U} = \{U\}$. The algorithm then iteratively selects a gene subset $V' \in \mathcal{V}$ and a condition subset $U' \in \mathcal{U}$ and applies the one dimensional clustering algorithm twice, to cluster V' and U' on the submatrix $U' \times V'$. If stable clusters are detected, their gene/condition subsets are added to the respective sets $\mathcal{V}, \mathcal{U}$. The process is repeated until no new stable clusters can be found. The implementation makes sure that each pair of subsets is not encountered more than once.

Note that the procedure avoids the consideration of all rows and column subsets, by starting from an established row subset when forming subclusters of established column subsets, and vice versa. The success of the coupled two-way clustering strategy depends on the performance of the given one-dimensional clustering algorithm. We note that many popular clustering algorithms (e.g. K-means, Hierarchical, SOM) cannot be plugged "as is" into the coupled two-way machinery, as they do not readily distinguish significant clusters from non-significant clusters or make a-priori assumption on the number of clusters. Getz et al. have reported good results using the SPC hierarchical clustering algorithm [10]. The results of the algorithm can be viewed in a hierarchical form: each stable gene (condition) cluster is generated given a condition (resp. gene) subset. This hierarchical relation is important when trying to understand the context of joint genes or conditions behavior. For example, when analyzing clinical data, Getz et al. have focused on gene subsets giving rise to stable tissue clusters that are correlative to known clinical attributes. Such gene sets may have an important biological role in the disease under study.

The CTWC algorithm has been applied to a variety of clinical data sets (see, e.g., [17]), the software can be downloaded via the site http://ctwc.weizmann.ac.il.

TWOWAY(U, V, E, ALG):
U : conditions. V : genes.
E : Gene expression matrix.
ALG : one-dimensional clustering algorithm. Inputs a matrix and outputs significant (stable) clusters of columns or rows
Initialize a hash table $weight$
Initialize $\mathcal{U}_1 = \{U\}$, $\mathcal{V}_1 = \{V\}$
Initialize $\mathcal{U} = \emptyset$, $\mathcal{V} = \emptyset$
Initialize the sets hierarchy table H_V storing for gene clusters the condition subsets used to generate them.
Initialize the sets hierarchy table H_U storing for condition clusters the gene subsets used to generate them.
While ($\mathcal{U}_1 \neq \emptyset$ or $\mathcal{V}_1 \neq \emptyset$) do
    Initialize empty sets $\mathcal{U}_2, \mathcal{V}_2$.
    For all $(U', V') \in (\mathcal{U}_1 \times \mathcal{V}_1) \cup (\mathcal{U}_1 \times \mathcal{V}) \cup (\mathcal{U} \times \mathcal{V}_1)$ do
        Run $ALG(E_{U'V'})$ to cluster the genes in V':
            Add the stable gene sets to $\mathcal{V}_2$
            Set $H_V[V''] = U'$ for all new clusters V''.
        Run $ALG(E_{U'V'})$ to cluster the conditions in U':
            Add the stable condition sets to $\mathcal{U}_2$
            Set $H_U[U''] = V'$ for all new clusters U''.
    Assign $\mathcal{U} = \mathcal{U} \cup \mathcal{U}_1$, $\mathcal{V} = \mathcal{V} \cup \mathcal{V}_1$
    Assign $\mathcal{U}_1 = \mathcal{U}_2$, $\mathcal{V}_1 = \mathcal{V}_2$
Report $\mathcal{U}, \mathcal{V}$ and their hierarchies H_U, H_V.

Figure 3: Coupled two-way clustering.

4 The Iterative Signature Algorithm

In the Iterative Signature Algorithm (ISA) [12, 5] the notion of a significant bicluster is defined intrinsically on the bicluster genes and samples – the samples of a bicluster uniquely define the genes and vice versa. The intuition is that the genes in a bicluster are co-regulated and, thus, for each sample the average gene expression over all the biclsuter's genes should be surprising (unusually high or low) and for each gene the average gene expression over all biclusters samples should be surprising. This intuition is formalized using a simple linear model for gene expression assuming normally distributed expression levels for each gene or sample as shown below.

The algorithm, presented in Figure 4, uses two normalized copies of the original gene expression matrix. The matrix E^G has rows normalized to mean 0 and variance 1 and the matrix E^C has columns normalized similarly. We denote by $e^G_{uV'}$ the mean expression of genes from V' in the sample u and by $e^C_{U'v}$ the mean expression of the gene v in samples from U'. A bicluster $B = (U', V')$ is required to have:

$$U' = \{u \in U : |e^C_{uV'}| > T_C \sigma_C\}, V' = \{v \in V : |e^G_{U'v}| > T_G \sigma_G\} \tag{1}$$

Here T_G is the threshold parameter and σ_G is the standard deviation of the means $e^G_{U'v}$ where v ranges over all possible genes and U' is fixed. Similarly, T_C, σ_C are the corresponding parameters for the column set V'. The idea is that if the genes in V' are up- or down-regulated in the conditions U' then their average expression should be significantly far (i.e., T_G standard deviations) from its expected value on random matrices (which is 0 since the matrix is standardized). A similar argument holds for the conditions in U'. The standard deviations can be predicted as $\frac{1}{\sqrt{|U'|}}, \frac{1}{\sqrt{|V'|}}$ being a linear sum of $|U'|$ (or $|V'|$) independent standard random variables. Alternatively, the standard deviations can be estimated directly from the data, correcting for possible biases in the statistics of the specific condition and gene sets used. In other words, in a bicluster, the z-score of each gene, measured w.r.t. the bicluster's samples, and the z-score of each sample, measured w.r.t. the bicluster's samples, should exceed a threshold. As we shall see below, ISA will not discover biclusters for which the conditions (1) hold strictly, but will use a relaxed version.

The algorithm starts from an arbitrary set of genes $V_0 = V_{in}$. The set may be randomly generated or selected based on some prior knowledge. The algorithm then repeatedly applies the update equations:

$$U_i = \{u \in U : \; |e^C_{uV_i}| > T_C\sigma_C\}, V_{i+1} = \{v \in V : \; |e^G_{U_iv}| > T_G\sigma_G\} \tag{2}$$

The iterations are terminated at step n satisfying:

$$\frac{|V_{n-i} \setminus V_{n-i-1}|}{|V_{n-i} \cup V_{n-i-1}|} < \epsilon \tag{3}$$

for all i smaller than some m. The ISA thus converges to an approximated fixed point that is considered to be a bicluster. The actual fixed point depends on both the initial set V_{in} and the threshold parameters T_C, T_G. To generate a representative set of biclusters, it is possible to run ISA with many different initial conditions, including known sets of associated genes or random sets, and to vary the thresholds. After eliminating redundancies (fixed points that were encountered several times), the set of fixed points can be analyzed as a set of biclusters.

The ISA algorithm can be generalized by assigning weights for each gene/sample such that genes/samples with a significant behavior (higher z-score) will have larger weights. In this case, the simple means used in (1) and (2) are replaced by weighted means.

The signature algorithm has been applied for finding cis-regulatory modules in yeast ([12]) and for detecting conserved transcriptional modules across several species ([4]). For software see http://barkai-serv.weizmann.ac.il/GroupPage/.

ISA($U, V, E, V_{in}, T_G, T_C, m, \epsilon$):
U : conditions. V : genes.
E : Gene expression matrix.
V_{in} : Initial gene set.
T_G, T_C: gene and condition z-score thresholds.
m, ϵ: stopping criteria.
Construct a column standardized matrix E^C.
Construct a row standardized matrix E^G.
Initialize counters $n = 0, n' = 0$.
Initialize the current genes set $V' = V_{in}$
Initialize an empty condition set U'.
While $(n - n' < m)$ do
    Compute $e^C_{uV'} = \frac{1}{|V'|} \sum_{v \in V'} e^C_{uv}$ for $u \in U$.
    $U' = \{u \in U : |e^C_{uV'}| > \frac{T_C}{\sqrt{|V'|}}\}$
    Compute $e^G_{U'v} = \frac{1}{|U'|} \sum_{u \in U'} e^G_{uv}$ for $v \in V$.
    $V'' = V'$
    $V' = \{v \in V : |e^G_{U'v}| > \frac{T_G}{\sqrt{U'}}\}$
    if $(\frac{|V' \setminus V''|}{|V' \cup V''|} < \epsilon)$ then $n' = n$
    $n = n + 1$
Report U', V'

Figure 4: The ISA algorithm for finding a single bicluster.

5 The SAMBA Algorithm

The SAMBA algorithm (Statistical-Algorithmic Method for Bicluster Analysis) [24, 20] uses probabilistic modeling of the data and graph theoretic techniques to identify subsets of genes that *jointly respond* across a subset of conditions, where a gene is termed *responding* in some condition if its expression level changes significantly at that condition w.r.t. its normal level. Within the SAMBA framework, the expression data are modeled as a bipartite graph whose two parts correspond to conditions and genes, respectively, with edges for significant expression changes. The vertex pairs in the graph are assigned weights according to a probabilistic model, so that heavy subgraphs correspond to biclusters with high likelihood. Discovering the most significant biclusters in the data reduces under this weighting scheme to finding the heaviest subgraphs in the model bipartite graph. SAMBA employs a practical heuristic to search for heavy subgraphs. The search algorithm is motivated by a combinatorial algorithm for finding heavy bicliques that is exponential in the maximum gene degree in the graph.

In the following we describe the probabilistic model used by SAMBA and the theoretical algorithm on which the search method is based. Finally, the full SAMBA algorithm is presented.

Applications of SAMBA for gene expression data are described in [25]. SAMBA was also applied to highly heterogeneous data, including expression, phenotype growth sensitivity, protein-protein interaction and ChIP-chip data [24]. The software is available as part of the Expander package [20, 21].

5.1 Statistical Data Modeling

The SAMBA algorithm is based on representing the input expression data as a bipartite graph $G = (U, V, E)$. In this graph, U is the set of conditions, V is the set of genes, and $(u, v) \in E$ iff v responds in condition u, that is, if the expression level of v changes significantly in u. A bicluster corresponds to a subgraph $H = (U', V', E')$ of G, and represents a subset V' of genes that are co-regulated under a subset of conditions U'. The *weight* of a subgraph (or bicluster) is the sum of the weights of gene-condition pairs in it, including edges and non-edges.

Coupled with the graph representation is a likelihood ratio model for the data. Let $H = (U', V', E')$ be a subgraph of G and denote $\overline{E'} = (U' \times V') \setminus E'$. For a vertex $w \in U' \cup V'$ let d_w denote its degree in G. The null model assumes that the occurrence of each edge (u, v) is an independent Bernoulli variable with parameter $p_{u,v}$. The probability $p_{u,v}$ is the fraction of bipartite graphs with degree sequence identical to G that contain the edge (u, v). In practice, one estimates $p_{u,v}$ using a Monte-Carlo process. This model tries to capture the characteristics of the different genes and conditions in the data.

The alternative model assumes that each edge of a bicluster occurs with constant, high probability p_c. This model reflects the belief that biclusters represent approximately uniform relations between their elements. The log likelihood ratio for H is therefore:

$$\log L(H) = \sum_{(u,v)\in E'} \log \frac{p_c}{p_{u,v}} + \sum_{(u,v)\in \overline{E'}} \log \frac{1 - p_c}{1 - p_{u,v}}$$

Setting the weight of each edge (u, v) to $\log \frac{p_c}{p_{u,v}} > 0$ and the weight of each non-edge (u, v) to $\log \frac{1-p_c}{1-p_{u,v}} < 0$, one concludes that the score of H is simply its weight.

5.2 Finding Heavy Subgraphs

Under the above additive scoring scheme, discovering the most significant biclusters in the data reduces under this scoring scheme to finding the heaviest subgraphs in the bipartite graph. Since the latter problem is NP-hard, SAMBA employs a heuristic search for such subgraphs. The search uses as seeds heavy bicliques and we now present the underlying algorithm to find good seeds. In the rest of the section it will be convenient to assume that the degree of every gene is bounded by d.

Let $G = (U, V, E)$ be a bipartite graph with $n = |V|$ genes. Let $w : U \times V \to \mathcal{R}$ be a weight function. For a pair of subsets $U' \subseteq U, V' \subseteq V$ we denote by $w(U', V')$ the weight of the subgraph induced on $U' \cup V'$, i.e., $w(U', V') = \sum_{u\in U',v\in V'} w((u, v))$. The *neighborhood* of a vertex v, denoted $N(v)$, is the set of vertices adjacent to v in G.

The *Maximum Bounded Biclique* problem calls for identifying a maximum weight complete subgraph of a given weighted bipartite graph G, such that the vertices on one side of G have degrees bounded by d. This problem can be solved in $O(n2^d)$ time (and space) as we show next.

Observe that a maximum bounded biclique $H^* = (U^*, V^*, E^*)$ in G must have $|U^*| \leq d$. Figure 5 describes a hash-table based algorithm that for each vertex $v \in V$ scans all $O(2^d)$ subsets of its neighbors, thereby identifying the heaviest biclique. Each hash entry corresponds to a subset of conditions and records the total weight of edges from adjacent gene vertices. The algorithm can be shown to spend $O(n2^d)$ time on the hashing and finding U_{best}. Computing V_{best} can be done in $O(nd)$ time, so the total running time is $O(n2^d)$.

MaxBoundBiClique(U, V, E, d):
Initialize a hash table $weight$; $weight_{best} \leftarrow 0$
For all $v \in V$ do
    For all $S \subseteq N(v)$ do
        $weight[S] \leftarrow weight[S] +$
            $\max\{0, w(S, \{v\})\}$
        If ($weight[S] > weight_{best}$)
            $U_{best} \leftarrow S$
            $weight_{best} \leftarrow weight[S]$
Compute $V_{best} = \cap_{u \in U_{best}} N(u)$
Output (U_{best}, V_{best})

Figure 5: An algorithm for the maximum bounded biclique problem.

Note that the algorithm can be adapted to give the k condition subsets that induce solutions of highest weight in $O(n2^d \log k)$ time using a priority queue data structure.

5.3 The Full Algorithm

Having described the two main components of SAMBA, we are now ready to present the full algorithm, which is given in Figure 6. SAMBA proceeds in two phases. First, the model bipartite graph is formed and the weights of vertex pairs are computed. Second, several heavy subgraphs are sought around each vertex of the graph. This is done by starting with good seeds around the vertex and expanding them using local search. The seeds are found using the hashing technique of the algorithm in Figure 5. To save on time and space the algorithm ignores genes with degree exceeding some threshold D, and hash for each gene only subsets of its neighbors of size ranging from N_1 to N_2. The local improvement procedure iteratively applies the best modification to the current bicluster (addition or deletion of a single vertex) until no score improvement is possible. The greedy process is restricted to search around the biclique without performing changes that would eliminate vertices in it or make vertices in it redundant (having a total negative contribution to the bicluster score). To avoid similar biclusters whose vertex sets differ only slightly, a final step greedily filters similar biclusters with more than $L\%$ overlap.

SAMBA(U, V, E, w, d, N_1, N_2, k):
U : conditions. V : genes.
E : graph edges. w : edge/non-edge weights.
N_1, N_2 : condition set hashed set size limits. k : max biclusters per gene/condition.
Initialize a hash table $weight$
For all $v \in V$ with $|N(v)| \leq d$ do
    For all $S \subseteq N(v)$ with $N_1 \leq |S| \leq N_2$ do
        $weight[S] \leftarrow weight[S] + w(S, \{v\})$
For each $v \in V$ set $best[v][1 \ldots k]$ to the k heaviest sets S such that $v \in S$
For each $v \in V$ and each of the k sets $S = best[v][i]$
    $V' \leftarrow \cap_{u \in S} N(u)$.
    $B \leftarrow S \cup V'$.
    Do $\{$
        $a = argmax_{x \in V \cup U}(w(B \cup x))$
        $b = argmax_{x \in B}(w(B \setminus x))$
        If $w(B \cup a) > w(B \setminus b)$ then $B = B \cup a$ else $B = B \setminus b$
    $\}$ **while** improving
    Store B.
Post process to filter overlapping biclusters.

Figure 6: The SAMBA biclustering algorithm.

6 Spectral Biclustering

Spectral biclustering approaches use techniques from linear algebra to identify bicluster structures in the input data. Here we review the biclustering technique presented in Kluger et al. [13]. In this model, it is assumed that the expression matrix has a hidden checkerboard-like structure that we try to identify using eigenvector computations. The structure assumption is argued to hold for clinical data, where tissues cluster to cancer types and genes cluster to groups, each distinguishing a particular tissue type from the other types.

To describe the algorithm, suppose at first that the matrix E has a checkerboard-like structure (see Figure 7). Obviously we could discover it directly, but we could also infer it using a technique from linear algebra that will be useful in case the structure is hidden due to row and column shufflings. The technique is based on a relation between the block structure of E and the block structure of pairs of eigenvectors for EE^T and E^TE, which we describe next. First, observe that the eigenvalues of EE^T and E^TE are the same. Now, consider a vector x that is *stepwise*, i.e., piecewise constant, and whose block structure matches that of the rows of E. Applying E to x we get a stepwise vector y. If we now apply E^T to y we get a vector with the same block structure as x. The same relation is observed when applying first E^T and then E (see Figure 7). Hence, vectors of the stepwise pattern of x form a subspace that is closed under E^TE. This subspace is spanned by eigenvectors of this matrix. Similarly, eigenvectors of EE^T span the subspace formed by vectors of the form of y. More importantly, taking now x to be an eigenvector of E^TE with an eigenvalue λ, we observe that $y = Ex$ is an eigenvector of EE^T with the same eigenvalue.

$$Ex = \begin{bmatrix} 8 & 8 & 7 & 7 & 3 & 3 \\ 8 & 8 & 7 & 7 & 3 & 3 \\ 6 & 6 & 4 & 4 & 5 & 5 \\ 6 & 6 & 4 & 4 & 5 & 5 \end{bmatrix} \begin{bmatrix} a \\ a \\ b \\ b \\ c \\ c \end{bmatrix} = \begin{bmatrix} d \\ d \\ e \\ e \end{bmatrix} = y, E^Ty = \begin{bmatrix} 8 & 8 & 6 & 6 \\ 8 & 8 & 6 & 6 \\ 7 & 7 & 4 & 4 \\ 7 & 7 & 4 & 4 \\ 3 & 3 & 5 & 5 \\ 3 & 3 & 5 & 5 \end{bmatrix} \begin{bmatrix} d \\ d \\ e \\ e \end{bmatrix} = \begin{bmatrix} a' \\ a' \\ b' \\ b' \\ c' \\ c' \end{bmatrix} = x'$$

Figure 7: An example of a checkerboard-like matrix E and the eigenvectors of EE^T and E^TE. The vector x satisfies the relation $E^TEx = E^Ty = x' = \lambda x$. Similarly, y satisfies the equation $EE^Ty = E\lambda x = \lambda y$.

In conclusion, the checkerboard-like structure of E is reflected in the stepwise structures of pairs of EE^T and E^TE eigenvectors that correspond to the same eigenvalue. One can find these eigenvector pairs by computing a singular value decomposition of E. Singular value decomposition is a standard algebraic technique (cf. [15]) that expresses a real matrix E as a product $E = A\Delta B^T$, where Δ is a diagonal matrix and A and B are orthonormal matrices. The columns of A and B are the eigenvectors of EE^T and E^TE, respectively. The entries of Δ are square roots of the corresponding eigenvalues, sorted in a non-increasing order. Hence the eigenvector pairs are obtained by taking for each i the ith columns of A and B, and the corresponding eigenvalue is the Δ_{ii}^2.

For any eigenvector pair, one can check whether each of the vectors can be approximated using a piecewise constant vector. Kluger et al. use a one-dimensional k-means algorithm to test this fit. The block structures of the eigenvectors indicate the block structures of the rows and columns of E.

In the general case, the rows and columns of E are ordered arbitrarily, and the checkerboard-like structure, if E has one, is hidden. To reveal such structure one computes the singular value decomposition of E and analyzes the eigenvectors of EE^T and E^TE. A hidden checkboard structure will manifest itself by the existence of a pair of eigenvectors (one for each matrix) with the same eigenvalue, that are approximately piecewise constant. One can determine if this is the case by sorting the vectors or by clustering their values, as done in [13].

Kluger et al. further discuss the problem of normalizing the gene expression matrix to reveal checkerboard structures that are obscured, e.g., due to differences in the mean expression levels of genes or conditions. The assumed model for the data is a multiplicative model, in which the expression level of a gene i in a condition j is its base level times a gene term, which corresponds to the gene's tendency of expression under different conditions, times a condition term, that represents the tendency of genes to be expressed under condition j. The normalization is done using two normalizing matrices: R, a diagonal matrix with the mean of row i at the ith position; and C, a diagonal matrix with the mean of column j at the jth position. The block structure of E is now reflected in the stepwise structure of pairs of eigenvectors with the same eigenvalue of the normalized matrices $M = R^{-1}EC^{-1}E^T$ and M^T. These eigenvector pairs can be deduced by computing a singular value decomposition of $R^{-1/2}EC^{-1/2}$. Due to the normalization, the first eigenvector pair (corresponding to an eigenvalue of 1) is constant and can be discarded. A summary of the biclustering algorithm is given in Figure 8.

The spectral algorithm was applied to human cancer data and its results were used for classification of tumor type and identification of marker genes [13].

Spectral(U, V, E):
U : conditions. V : genes.
$E_{n\times m}$: Gene expression matrix.
Compute $R = diag(E \cdot 1_m)$ and $C = diag(1_n^T \cdot E)$.
Compute a singular value decomposition of $R^{-1/2}EC^{-1/2}$.
Discard the pair of eigenvectors corresponding to the largest eigenvalue.
For each pair of eigenvectors u, v of $R^{-1}EC^{-1}E^T$ and $C^{-1}E^TR^{-1}E$ with the same eigenvalue do:
    Apply k-means to check the fit of u and v to stepwise vectors.
Report the block structure of the p u, v with the best stepwise fit.

Figure 8: The spectral biclustering algorithm.

7 Plaid Models

The Plaid model [14] is a statistically inspired modeling approach developed by Lazzeroni and Owen for the analysis of gene expression data. The basic idea is to represent the genes-conditions matrix as a superposition of *layers*, corresponding to biclusters in our terminology, where each layer is a subset of rows and columns on which a particular set of values takes place. Different values in the expression matrix are thought of as different colors, as in (false colored) "heat maps" of chips. This metaphor also leads to referring to "color intensity" in lieu of "expression level". The horizontal and vertical color lines in the matrix corresponding to a layer give the method its name.

The model assumes that the level of matrix entries is the sum of a uniform background ("grey") and of k biclusters each coloring a particular submatrix in a certain way. More precisely, the expression matrix is represented as

$$A_{ij} = \mu_0 + \sum_{k=1}^{K} \theta_{ijk}\rho_{ik}\kappa_{jk}$$

where μ_0 is a general matrix background color, and $\theta_{ijk} = \mu_k + \alpha_{ik} + \beta_{jk}$ where μ_k describes the added background color in bicluster k, α and β are row and column specific additive constants in bicluster k. $\rho_{ik} \in \{0, 1\}$ is a gene-bicluster membership indicator variable, i.e., $\rho_{ik} = 1$ iff gene i belongs to the gene set of the k-th bicluster. Similarly, $\kappa_{jk} \in \{0, 1\}$ is a sample-bicluster membership indicator variable. Hence, similar to Cheng and Church [6], a bicluster is assumed to be the sum of bicluster background level plus row-specific and column-specific constants.

When the biclusters form a k-partition of the genes and a corresponding k-partition of the samples, the *disjointness constraints* that biclusters cannot overlap can be formulated as $\sum_k \kappa_{jk} \leq 1$ for all j, $\sum_k \rho_{ik} \leq 1$ for all i. Replacing $\leq$ by $=$ would require assignment of each row or column to *exactly* one bicluster. Generalizing to allow bicluster overlap simply means removing the disjointness constraints.

The general biclustering problem is now formulated as finding parameter values so that the resulting matrix would fit the original data as much as possible. Formally, the problem is minimizing

$$\sum_{ij}[A_{ij} - \sum_{k=0}^{K} \theta_{ijk}\rho_{ik}\kappa_{jk}]^2 \tag{4}$$

where $\mu_0 = \theta_{ij0}$. If α_{ik} or β_{jk} are used, then the constraints $\sum_i \rho_{ik}\alpha_{ik} = 0$ or $\sum_j \kappa_{jk}\beta_{jk} = 0$ are added to reduce the number of parameters. Note that the number of parameters is at most $k + 1 + kn + km$ for the θ variables, and $kn + km$ for the κ and ρ variables. This is substantially smaller than the nm variables in the original data, if $k << max(n, m)$.

7.1 Estimating Parameters

Lazzeroni and Owen propose to solve problem (4) using an iterative heuristic. New layers are added to the model one at a time. Suppose we have fixed the first $K - 1$ layers and we are seeking for the K-th layer to minimize the sum of squared errors. Let

$$Z_{ij}^{(K-1)} = A_{ij} - \sum_{k=0}^{K-1} \theta_{ijk}\rho_{ij}\kappa_{jk} \tag{5}$$

be the *residual matrix* after removing the effect of the first $K - 1$ layers. In iteration K we wish to solve the following quadratic integer program.

$$\begin{array}{ll} min & Q^{(K)} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{p}(Z_{ij}^{(K-1)} - \theta_{ijK}\rho_{iK}\kappa_{jK})^2 \\ s.t. & \sum_i \rho_{iK}^2\alpha_{iK} = 0, \;\; \sum_j \kappa_{jK}^2\beta_{jK} = 0 \\ & \rho_{iK} \in \{0, 1\}, \;\; \kappa_{jK} \in \{0, 1\} \end{array} \tag{6}$$

The proposed heuristic method to solve (6) is again iterative. To avoid confusion we call the iterations for fixed K *cycles*, and indicate the cycle number by a superscript in parentheses, e.g. $\theta^{(i)}$. The integrality constraints are ignored throughout, and the goal is to solve corresponding relaxation of it. A cycle is done as follows: Compute the best values of the θ parameters given fixed ρ and κ values; Compute the best values of the ρ parameters given new θ and the old κ values;

Compute the best values of the κ parameters given the new θ and the old ρ values. In order to avoid "locking in" of the membership variables to 0 or 1, their values are changed only modestly on the first cycle, and they are allowed to become integral only at the final cycle.

The following optimal parameter values in the relaxed version of (6) are obtained by using Lagrange multipliers:

$$\mu_K = \frac{\sum_i \sum_j \rho_{iK}\kappa_{jK} Z_{ij}^{K-1}}{(\sum_i \rho_{iK}^2)(\sum_j \kappa_{jK}^2)} \tag{7}$$

$$\alpha_{iK} = \frac{\sum_j (Z_{ij}^{(K-1)} - \mu_K \rho_{iK}\kappa_{jK})\kappa_{jK}}{\rho_{iK}\sum_{jK}\kappa_{jK}^2} \tag{8}$$

$$\beta_{jK} = \frac{\sum_i (Z_{ij}^{(K-1)} - \mu_K \rho_{iK}\kappa_{jK})\rho_{iK}}{\kappa_{jK}\sum_{iK}\rho_{iK}^2} \tag{9}$$

So, in cycle s, we use these equations to update $\theta^{(s)}$ using the old values $\rho^{(s-1)}$ and $\kappa^{(s-1)}$. The values for ρ_{iK} and κ_{jK} that minimize Q are:

$$\rho_{iK} = \frac{\sum_j \theta_{ijK}\kappa_{jK} Z_{ij}^{K-1}}{\sum_j \theta_{ijK}^2 \kappa_{jK}^2} \tag{10}$$

$$\kappa_{jK} = \frac{\sum_i \theta_{ijK}\rho_{iK} Z_{ij}^{K-1}}{\sum_i \theta_{ijK}^2 \rho_{iK}^2} \tag{11}$$

At cycle s, we use these equations to update $\rho^{(s)}$ from $\theta^{(s)}$ and $\kappa^{(s-1)}$, and update $\kappa^{(s)}$ from $\theta^{(s)}$ and $\rho^{(s-1)}$. The complete updating process is repeated a prescribed number of cycles.

7.2 Initialization and Stopping Rule

The search for a new layer K in the residual matrix $Z_{ij} = Z_{ij}^{(K)}$ requires initial values of ρ and κ. These values are obtained by finding vectors u and v and a real value λ so that λuv^T is the best rank one approximation of Z. We refer the readers to the original paper for details.

Intuitively, each iteration "peels off" another signal layer, and one should stop after $K-1$ iterations if the residual matrix $Z_{ij} = Z_{ij}^{(K)}$ contains almost only noise. Lazzeroni and Owen define the *importance* of layer k by $\sigma_k^2 = \sum_{i=1}^n \sum_{j=1}^p \rho_{ik}\kappa_{jk}\theta_{ijk}^2$. The algorithm accepts a layer if it has significantly larger importance than in noise. To evaluate σ_k^2 on noise, repeat the following process T times: Randomly permute each row in Z independently, and then randomly permute each column in the resulting matrix independently. Apply the layer-finding algorithm on the resulting matrix, and compute the importance of that layer. If σ_k^2 exceeds the importance obtained for all the T randomized matrices, add the new layer K to the model.

The complete algorithm is outlined in figure 9.

Plaid models have been applied to yeast gene expression data [14]. The software is available at http://www-stat.stanford.edu/∼owen/plaid.

Plaid(U, V, E, S):
U : conditions. V : genes.
E : Gene expression matrix.
S: maximum cycles per iteration.
Set $K = 0$
adding a new layer:
  K=K+1
  Compute initial values of $\kappa_{jK}^{(0)}, \rho_{iK}^{(0)}$. Set $s = 1$
  While $(s \leq S)$ do:
    Compute $\mu_K^{(s)}$, $\alpha_{iK}^{(s)}$, $\beta_{jK}^{(s)}$ using equations (7)- (9).
    Compute $\kappa_K^{(s)}$ using equations (11)
    Compute $\rho_K^{(s)}$ using equations (10)
    If $\rho_K^{(s)} > 0.5$ set $\rho_K^{(s)} = 0.5 + s/2S$, else set $\rho_K^{(s)} = 0.5 - s/2S$
    If $\kappa_K^{(s)} > 0.5$ set $\kappa_K^{(s)} = 0.5 + s/2S$, else set $\kappa_K^{(s)} = 0.5 - s/2S$
  If the importance of layer K is non random then record the layer and repeat
  Else exit.
Report layers $1, \ldots, K - 1$.

Figure 9: The Plaid model algorithm.

8 Discussion

The algorithms presented above demonstrate some of the approaches developed for the identification of bicluster patterns in large matrices, and in gene expression matrices in particular. One can roughly classify the different methods a) by their model and scoring schemes and b) by the type of algorithm used for detecting biclusters. Here we briefly review how different methods tackle these issues.

8.1 Model and score

To ensure that the biclusters are statistically significant, each of the biclustering methods defines a scoring scheme to assess the quality of candidate biclusters, or a constraint that determines which submatrices represent significant bicluster behavior. Constraint based method include the iterative signature algorithm, the coupled two-way clustering method and the spectral algorithm of Kluger et al. In the first two, we search for gene (property) sets that define "stable" subsets of properties (genes). In the last, the requirement is for compatibility of certain eigenvectors to a hidden checkboard-like matrix structure.

Scoring based methods typically rely on a background model for the data. The basic model assumes that biclusters are essentially uniform submatrices and scores them according to their deviation from such uniform behavior. More elaborate models allow different distributions for each condition and gene, usually in a linear way. Such are, for example, the Cheng-Church algorithm and the Plaid model and the alternative formulation in [22]. A more formal statistical model for an extended formulation of the biclustering problem was used in [19, 3]. In this family of algorithms a complete generative model including a set of biclusters and their regulation model is optimized for maximum likelihood given the data. Another approach for the modeling of the data is used in SAMBA, where a degree-preserving random graph model and likelihood ratio score are used to ensure biclusters significance.

8.2 Algorithmic approaches

The algorithmic approaches for detecting biclusters given the data are greatly affected by the type of score/constraint model in use. Several of the algorithms alternate between phases of gene sets and condition sets optimization. Such are, for example, the iterative signature algorithm and the coupled two-way clustering algorithm. Other methods use standard linear algebra or optimization algorithms to solve key subproblems. Such is the case for the Plaid model and the Spectral algorithm. A heuristic hill climbing algorithm is used in the Cheng-Church algorithm and is combined with a graph hashing algorithm in SAMBA. Finally, EM or sampling methods are used for formulations introducing a generative statistical model for biclusters [19, 3, 22]. The overall picture seems to support a view stressing the importance of statistical models and scoring scheme and restricting the role of the search/optimization algorithm to discovering relatively bold structures. A current important goal for the research community is to improve our understanding of the pros and cons of the various modeling approaches described here, and to enable more focused algorithmic efforts on the models that prove most effective.

8.3 Quo vadis biclustering?

Biclustering is a relatively young area, in contrast to its parent discipline, clustering, that has a very long history going back all the way to Aristo. It has great potential to make significant contributions to biology and to other fields. Still, some of the difficulties that haunt clustering are present and are even exacerbated in biclustering: Multiple formulations and objective functions, lack of theoretical and complexity analysis for many algorithms, and few criteria for comparing the quality of candidate solutions. Still, the great potential of the paradigm of biclustering, as demonstrated in studies over the last five years, guarantees that the challenge will continue to be addressed. In time, the concrete advantages and disadvantages of each formulation and algorithm will be made clearer. We anticipate an exciting and fruitful next decade in biclustering research.

9 Acknowledgment

R. Shamir was supported in part by the Israel Science Foundation (grant 309/02). R. Sharan was supported in part by NSF ITR Grant CCR-0121555. A. Tanay was supported in part by a scholarship in Complexity Science from the Yeshaia Horvitz Association.

References

[1] The chipping forecast II. Special supplement to Nature Genetics Vol 32, 2002.

[2] A.A. Alizadeh et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.

[3] A. Battle, E. Segal, and D Koller. Probabilistic discovery of overlapping cellular processes and their regulation. In *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology (RECOMB 2002)*, 2004.

[4] S. Bergman, J. Ihmels, and N. Barkai. Similarities and differences in genome-wide expression data of six organisms. *PLoS*, 2(1):E9, 2004.

[5] S. Bergmann, J. Ihmels, and N. Barkai. Iterative signature algorithm for the analysis of large-scale gene expression data. *Phys Rev E Stat Nonlin Soft Matter Phys*, 67(3 Pt 1):03190201–18, 2003.

[6] Y. Cheng and G.M. Church. Biclustering of expression data. In *Proc. ISMB'00*, pages 93–103. AAAI Press, 2000.

[7] J DeRisi, L Penland, PO Brown, et al. Use of a cDNA microarray to analyse gene expression patterns in human cancer. *Nat Genet*, 14:457–460, 1996.

[8] A.P. Gasch et al. Genomic expression responses to DNA-damaging agents and the regulatory role of the yeast ATR homolog mec1p. *Mol. Biol. Cell*, 12(10):2987–3003, 2001.

[9] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proc. Natl. Acad. Sci. USA*, 97(22):12079–84, 2000.

[10] G. Getz, E. Levine, E. Domany, and M.Q. Zhang. Super-paramagnetic clustering of yeast gene expression profiles. *Physica*, A279:457, 2000.

[11] J.D. Hughes, P.E. Estep, S. Tavazoie, and G.M. Church. Computational identification of cis-regulatory elements associated with groups of functionally related genes in *Saccharomyces Cerevisiae*. *J. Mol. Biol.*, 296:1205–1214, 2000. `http://atlas.med.harvard.edu/`.

[12] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nature Genetics*, 31(4):370–7, 2002.

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

[14] L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12:61–86, 2002.

[15] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge (UK) and New York, 2nd edition, 1992.

[16] S. Ramaswamy et al. Multiclass cancer diagnosis using tumor gene expression signature. *Proc. Natl. Acad. Sci. USA*, 98(26):15149–15154, 2001.

[17] T. Rozovskaia, O. Ravid-Amir, S. Tillib, G. Getz, E. Feinstein, H. Agrawal, A. Nagler, E.F. Rappaport, I. Issaeva, Y. Matsuo, U.R. Kees, T. Lapidot, F. Lo Coco, R. Foa, A. Mazo, T. Nakamura, CM. Croce, G. Cimino, E. Domany, and E. Canaani. Expression profiles of acute lymphoblastic and myeloblastic leukemias with all-1 rearrangements. *Proc Natl Acad Sci U S A*, 100(13):7853–8, 2003.

[18] M Schena, D Sharon, RW Davis, et al. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270:467–470, 1995.

[19] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman. Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat Genet.*, 34(2):166–76, 2003.

[20] R. Sharan, A. Maron-Katz, N. Arbili, and R. Shamir. EXPANDER: EXPression ANalyzer and DisplayER, 2002. Software package, Tel-Aviv University, `http://www.cs.tau.ac.il/~rshamir/expander/expander.html`.

[21] R. Sharan, A. Maron-Katz, and R. Shamir. CLICK and EXPANDER: a system for clustering and visualizing gene expression data. *Bioinformatics*, 2003.

[22] Q. Sheng, Y. Moreau, and B. De Moor. Biclustering gene expression data by Gibbs sampling. *Bioinformatics*, 19(Supp 2):i196–i205, 2003.

[23] P. T. Spellman, G. Sherlock, et al. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell*, 9:3273–3297, 1998.

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

[25] A. Tanay, R. Sharan, and R. Shamir. Biclustering gene expresion data. Submitted for publication, 2002.