

## Lecture 5: May 24, 2007

*Lecturer: Ron Shamir**Scribe: Shelly Mahlev and Shaul Karni<sup>1</sup>*

## 5.1 Introduction

As we have seen in previous lectures, the technology of DNA chips allows the measurement of mRNA levels simultaneously for thousands of genes. The abundance of readily available data allows us the option to try to extract information, not easily seen while analyzing a single test, from the results of a multitude of tests. The results of DNA chip experiments are usually organized together in a *gene expression matrix*, with rows corresponding to genes and columns corresponding to conditions. A *bicluster* can be defined as a submatrix spanned by a subset of genes and a subset of conditions in which the genes have joint behavior in the corresponding conditions. Different algorithmic approaches to the biclustering problem use different measures for the quality of a given biclustering solution. Hence, the definition of the goal function of each algorithm is part of its description. There are two important themes regarding biclustering algorithms:

1. Data normalization. A common preprocessing step before applying any clustering algorithm is to normalize the rows or columns of the gene expression matrix, depending on the dimension on which the clustering is applied (rows/columns). In biclustering algorithms, for which the two dimensions are considered, the question of how normalization should be done is not simple. Some of the biclustering algorithms do not refer to the normalization problem and assume that the given data is normalized.
2. Eliminating redundancies when searching for all (or many) biclusters. Usually in clustering solutions, each gene/condition participates in at most one cluster, in biclustering solutions genes and conditions are often part of several biclusters. A bicluster algorithm should eliminate redundancies (“similar” biclusters) from its output.

This lecture can be viewed as a direct continuation of Lecture 4 which already described the notion of biclustering and presented Cheng and Church’s biclustering algorithm. In this lecture, we will first recollect on Cheng and Church’s algorithm (section 5.2) and then describe three more biclustering algorithms. First, we will describe the elegant simple Iterative

---

<sup>1</sup>Based in part on a scribe by Itamar Elem and Seagull Chalamish, March 2005; Michal Ozery-Flato and Israel Steinfeld, May 2004; Irit Gat and Amos Tanay, May 2002.

Signature Algorithm (section 5.3). Then, the hierarchical Coupled Two Way Clustering algorithm (section 5.4). And finally, we will describe the Statistical Algorithmic Method for Biclustering Analysis (section 5.5), a fast, combinatorial, statistically based algorithm.

## 5.2 Cheng and Church's algorithm

First, let's briefly describe Cheng and Church's algorithm [4]. For a full description, the reader should refer to Lecture 4.

Cheng and Church [4] 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  $e_{Ij} = \frac{\sum_{i \in I} e_{ij}}{|I|}$  as column subset average,  $e_{iJ} = \frac{\sum_{j \in J} e_{ij}}{|J|}$  as row subset average and  $e_{IJ} = \frac{\sum_{i \in I, j \in J} e_{ij}}{|I||J|}$  as 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  $H(I, J) = \sum_{i \in I, j \in J} \frac{RS_{IJ}(i, j)^2}{|I||J|}$ . Given the score definition, the *maximum bicluster problem* seeks a bicluster of maximum size among all biclusters with score not exceeding a threshold  $\delta$ . 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 ( $|I| + |J|$ ).

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.

## 5.3 The Iterative Signature Algorithm

In the Iterative Signature Algorithm (ISA), the goal is to find a bicluster in which the conditions induce a correlated expression of the genes in the bicluster. I.e., the expression profiles of the genes in the bicluster are the most similar to each other when compared over its conditions. Conversely, the patterns of gene expression obtained under the conditions of the bicluster are most similar to each other when compared only over its genes. The degree of similarity is determined by a pair of threshold parameters. The gene threshold constrains the gene set, while the condition threshold constrains the condition set. Importantly, distinct biclusters may share common genes and conditions. Therefore, for each condition the average gene expression over all the genes of the bicluster should be exceptionally different (unusually high or low), and for each gene the average gene expression over all biclusters conditions

should be surprising as well. This intuition is formalized using a simple linear model for gene expression assuming normally distributed expression levels for each gene or condition as shown below.

### 5.3.1 Normalization

The algorithm, presented in Figure 5.1, uses two normalized copies,  $E^C$  and  $E^G$ , of the original gene expression matrix  $E$ . The matrix  $E^G$  ( $E^C$ ) results from  $E$  by normalizing each column (row) so that the expression level of a random condition (gene) has mean 0 and variance 1. Note that the normalized matrices  $E^G$  and  $E^C$  in general are not equal.

### 5.3.2 Bicluster definition

Suppose  $G' = \{g\}$  is a set of independent random variables, where each random variable  $g \in G$  has mean 0 and variance 1. In this case, the average of all the random variables in  $G'$  (i.e.  $\frac{1}{|G'|} \sum_{g \in G'} g$ ) has a mean and a variance equal to 0 and  $\frac{1}{|G'|}$  respectively. For  $G' \subseteq G$  and  $C' \subseteq C$  we define:

$$e_{G'c}^G = \frac{1}{|G'|} \sum_{g \in G'} E_{gc}^G \quad (5.1)$$

$$e_{gC'}^C = \frac{1}{|C'|} \sum_{c \in C'} E_{gc}^C \quad (5.2)$$

Note that  $e_{Gc}^G = e_{gC}^C = 0$  due to the normalization made in  $E^G$  and  $E^C$ .

For a fixed  $C'$  ( $G'$ ) let  $\sigma_G$  ( $\sigma_C$ ) be the standard deviation of the means  $\{e_{gC'}^C : g \in G\}$  ( $\{e_{G'c}^G : c \in C'\}$ ). The standard deviation  $\sigma_G$  ( $\sigma_C$ ) can be predicted as  $\frac{1}{\sqrt{|C'|}}$  ( $\frac{1}{\sqrt{|G'|}}$ ) as  $e_{gC'}^C$  ( $e_{G'c}^G$ ) being a linear sum of  $|C'|$  ( $|G'|$ ) independent standard random variables. Alternatively, the standard deviations,  $\sigma_G$  and  $\sigma_C$ , can be estimated directly from the data, correcting for possible biases in the statistics of the specific condition and gene sets used.

Given a set of genes  $G'$  and a threshold parameter  $T_C$  define:

$$\text{ISA}(G') = \{c \in C : |e_{G'c}^G| > T_C \sigma_C\} \quad (5.3)$$

Given a set of conditions  $C'$  and a threshold parameter  $T_G$  define:

$$\text{ISA}(C') = \{g \in G : |e_{gC'}^C| > T_G \sigma_G\} \quad (5.4)$$

A (perfect) bicluster  $B = (G', C')$  is required to have:

$$\text{ISA}(G') = C', \quad \text{ISA}(C') = G' \quad (5.5)$$

The above definition states that for each condition  $c$  in the bicluster the average expression level of the genes in the bicluster,  $|e_{G'c}^G|$ , is above a certain threshold  $T_C$ . Conversely, for each gene  $g$  in the bicluster the average expression level over the conditions of the bicluster,  $|e_{gC'}^C|$ , is also above some threshold  $T_G$ . This reciprocal dependence between the genes and the conditions associated with a bicluster implies that, considering only the genes of the bicluster, its conditions are exactly those for which the coexpression is the most stringent. In other words, in a bicluster, the  $z$ -score of each gene, measured w.r.t. the conditions of the bicluster, and the  $z$ -score of each condition, measured w.r.t. the genes of the bicluster, should exceed a threshold. As we shall see below, ISA will not discover biclusters for which the conditions (5.5) hold strictly, but will use a relaxed version.

### 5.3.3 Searching for biclusters

Define a directed bipartite graph  $B(\mathcal{G}, \mathcal{C}, E)$  as follows.  $\mathcal{G} = \{G' : G' \subseteq G\}$ ,  $\mathcal{C} = \{C' : C' \subseteq C\}$ ,  $E = \{(G', C') : \text{ISA}(G') = C', G' \subseteq G, C' \subseteq C\} \cup \{(G'', C'') : \text{ISA}(C'') = G'', G'' \subseteq G, C'' \subseteq C\}$ , that is, every node in the graph is either a subset of conditions or a subset of genes. The goal is to find a cycle as small as possible following the conditions (5.5), and therefore a (perfect) bicluster is a cycle of size 2. However, as we mentioned before, the algorithm searches for an approximation of a (perfect) bicluster.

The algorithm starts from an arbitrary set of genes  $G_0 = G_{in}$ . The set may be randomly generated or selected based on some prior knowledge. The algorithm then repeatedly applies the update equations:

$$C_i = \text{ISA}(G_i), \quad G_{i+1} = \text{ISA}(C_i) \quad (5.6)$$

Let  $g_n$  be a binary vector of size  $|G|$  for which  $g_n^{(i)} = 1$  iff  $i \in G_n$ . An approximated bicluster is a pair  $(G_n, C_n)$  satisfying:

$$\frac{|g_n - g_{n-i}|}{|g_n + g_{n-i}|} < \epsilon \quad (5.7)$$

for all  $i > 0$  smaller than some  $m$ . The ISA thus searches for a series of gene sets  $\{G_i\}$  which converges to an approximated fixed point that is considered to be a bicluster. The actual fixed point depends on both the initial set  $G_{in}$  and the threshold parameters  $T_C, T_G$ .

The ISA algorithm can be generalized by assigning weights for each gene/condition such that genes/conditions with a significant behavior (higher  $z$ -score) will have larger weights. In this case, there are two changes to the algorithm:

1. the simple means,  $e_{G'c}^G$  and  $e_{gC'}^C$ , are replaced by weighted means.
2. The vector  $g_n$  (see 5.7) has real-valued elements and is defined as follows:  $g_n^{(i)} = \text{weight}(i)$  if  $i \in G_n$ , otherwise  $g_n^{(i)} = 0$ .

```

ISA( $G, C, E, G_{in}, T_G, T_C, m, \epsilon$ ):
\*  $G$  : genes.  $C$  : conditions.  $E$  : Gene expression matrix.
   $G_{in}$  : Initial gene set.
   $T_G, T_C$  : gene and condition z-score thresholds.
   $m, \epsilon$  : stopping criteria. *\
Construct matrix  $E^C$ .
Construct matrix  $E^G$ .
Initialize counter  $n = 0$ .
Initialize the current genes set  $G' = G_{in}$  and compute  $g_0$  based on  $G'$ 
While ( true) do
  Compute  $e_{G'_c}^G$  for  $c \in C$ .
   $C' = \{c \in C : |e_{G'_c}^G| > \frac{T_C}{\sqrt{|G'|}}\}$ 
  Compute  $e_{gC'}^C$  for  $g \in G$ .
   $G' = \{g \in G : |e_{gC'}^C| > \frac{T_G}{\sqrt{|C'|}}\}$ 
   $n = n + 1$ 
  Compute  $g_n$  from  $G'$ .
  If ( $n < m - 1$ ) continue.
  Initialize counter  $i = 1$ 
  while  $i < m$ 
    If ( $\frac{|g_n - g_{n-i}|}{|g_n + g_{n-i}|} \geq \epsilon$ ) break
     $i = i + 1$ 
  If  $i == m$  break
Report  $G', C'$ 

```

Figure 5.1: The ISA algorithm for finding a single bicluster.

### 5.3.4 Applications

There are a few difficulties and redundancies that we have to take into account:

- Starting from different seeds yields different fixed points (biclusters).
- Using different thresholds changes the graph structure and gives a different number of fixed points.
- There is need to filter similar solutions and report a short list of significant biclusters.

The following practical applications take the above into consideration:

- Start from genes with a known functional annotation and refine them into a bicluster.
- Start from genes with known transcription factor binding sites.

- Start from a set of sequence orthologs, to find similar genes from different species.
- Start from a random set (the algorithm might end up reporting an empty bicluster).

Therefore, to generate a representative set of biclusters, do the following:

1. Run ISA with different seeds.
2. Run ISA with variant thresholds (see equations 5.3 and 5.4). This action affects the ISA function and consequently the traversed bipartite graph. Scanning over different values for  $(T_C, T_G)$  reveals the modular structure at different resolutions: Lower thresholds yield larger biclusters whose coregulation is relatively loose, while higher thresholds lead to smaller, tightly coregulated biclusters.

After eliminating redundancies (similar solutions), the set of fixed points can be analyzed as a set of biclusters.

### 5.3.5 Discussion on ISA

ISA is an elegant algorithm which has quite a fast implementation. It offers a satisfying simple solution to the normalization problem. It is known to have good empirical results in several cases. However, there are some difficulties in the use of ISA:

1. The size of a bicluster depends critically on the associated set of two thresholds that determine the similarity between the genes and conditions of the module, respectively, and there is no scheme for choosing the values of the thresholds.
2. Finding good seeds for the algorithm can be problematic.
3. We cannot know if we found all or even most of the existing (approximated) biclusters.
4. Since the algorithm uses  $Z$ -score statistics, it implicitly assumes that the data (genes and conditions) have normal distribution. However, if the data have non-normal behavior, then the algorithm may fail. For instance, assume that the data contains a gene  $g$  and a condition  $c$  for which the expression level is much higher than all the other levels ( $E_{gc}$  is extremely high). In this case, if a set of genes  $G'$  contains  $g$  then  $c \in \text{ISA}(G')$  and vice versa:  $c \in C'$  implies  $g \in \text{ISA}(C')$ . Hence, we can get a bicluster in which there are a condition and a gene which have no relevance to the other genes/conditions in the bicluster.
5. Redundancies are a general problem of biclustering, that most methods do not deal with. ISA, for example, might find two overlapping biclusters as significant biclusters.

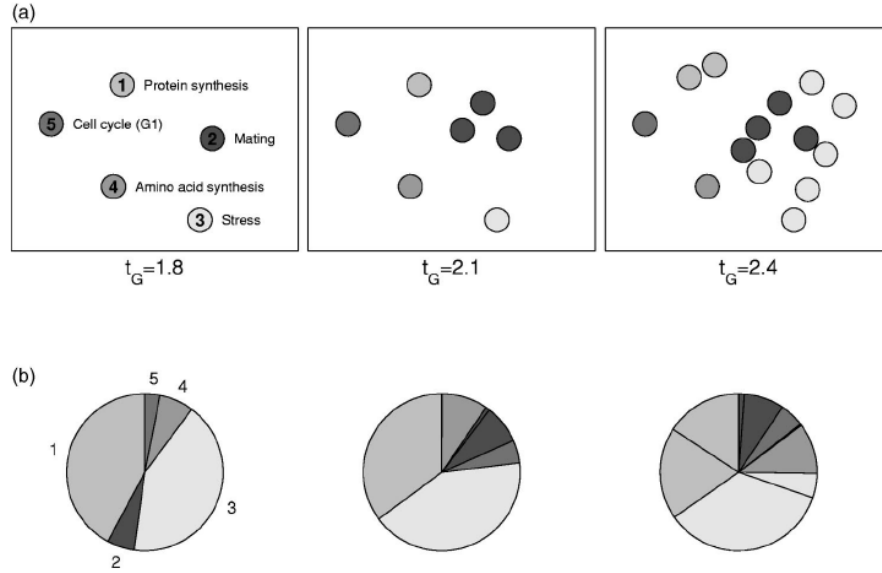


Figure 5.2: Modular organization of yeast expression data. The iterative signature algorithm was applied to genome wide yeast expression data gathered by more than 1000 DNA-chip experiments. (a) The figure shows the identified biclusters at three different gene thresholds  $t_G = 1.8, 2.1, 2.4$ . For each threshold the corresponding biclusters are displayed in a plane, such that their distance reflects their correlation with respect to conditions. Moving to a higher threshold, corresponding biclusters are kept in the same position in each plane, while the new biclusters are placed such that their position reflects best their correlation with the other biclusters. The leftmost plane corresponds to the lowest threshold ( $t_G = 1.8$ ), where only five fixed points exist. The corresponding biclusters can be associated with central functions of the yeast organism: protein synthesis, cell cycle ( $G_1$ , mating, amino-acid biosynthesis, and stress response. Different gray scales were used to indicate which of the fixed points that emerge at higher thresholds are related to these five central biclusters (i.e., they would converge to the respective module at the lowest threshold). (b) The pie charts show the number of random input sets that converged to the respective fixed point. The gray scales are as in (a).

## 5.4 Coupled two-way clustering

Coupled two-way clustering (CTWC), introduced by Getz, Levine and Domany [8] is another algorithm for finding biclusters. The main idea is to identify subsets of the genes and conditions, such that when one of these is used to cluster the other, stable and significant partitions emerge. This algorithm is based on iterative clustering, that performs a search for such subsets. A straightforward approach to finding pairs of subsets,  $(G, C)$ , that lead to meaningful clusters, could be to take all possible submatrices of the original data and apply the standard (uncoupled) two-way clustering procedure to every one of them. By keeping track of all stable clusters that are formed in this process, and storing the identity of both genes and conditions that define the particular submatrix, one is guaranteed to find every possible stable partition in the data. This approach is, of course, impossible to implement, because the number of such submatrices grows exponentially with the size of the problem. CTWC provides an efficient heuristic to generate such pairs of genes and conditions subsets by an iterative process that severely restricts the possible candidates for such subsets; Only those submatrices whose rows (columns) belong to genes (conditions) that were identified (in a previous iteration) as a stable cluster were considered and tested. This algorithm does not define any goal function or other measure for the quality of a given biclustering solution. Instead it defines a generic scheme for transforming a one-dimensional clustering algorithm into a biclustering algorithm. The iterative process is initialized with the full matrix i.e., the sets of all genes ( $g^0$ ) and of all conditions ( $s^0$ ) are used to perform standard two-way clustering. Denote by  $g_i^1$  and  $s_i^1$  stable clusters of genes and conditions found in this first step. Every pair  $(g_i^n, g_j^m)$  (made of clusters obtained so far) defines a submatrix of the expression data; for every such submatrix two-way clustering was performed. The resulting stable gene (or condition) clusters are denoted by  $g_k^2$  (or  $s_l^2$ ). Each cluster is stored. Together with each new cluster we also store pointers that identify the pair of parent clusters  $(g_i^n, s_j^m)$  that were used in the clustering process that generated it. These steps are iterated further, using pairs of all previously found clusters. Note that every pair is treated only once; the process is terminated when no new clusters that satisfy some criteria (such as stability, critical size, etc.) are found.

The problem with this algorithm is that its success depends on the clustering algorithm that was used, and therefore the CTWC procedure might miss significant biclusters.

## 5.5 SAMBA

We next describe SAMBA (Statistical Algorithmic Method for Bicluster Analysis) [16]. The motivation underlying its development is the need for a fast biclustering method that would produce statistically significant results as part of its design. The key point in the understanding of SAMBA is the statistical model used by the bicluster scoring scheme. The



normalization is done by translating the gene expression matrix to a weighted bipartite graph, using a statistical model for the data. The hope is that a model which captures the essential features of the data would guarantee high quality results.

### 5.5.1 Data Modeling

The intuitive notion of a bicluster is a subset of genes that exhibit similar expression patterns over a subset of conditions. Following this intuition, the authors defined a bicluster as a subset 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.

The expression data is modeled as a bipartite graph whose two parts correspond to conditions and genes with edges spanned between the parts where significant expression changes occur. Later, we shall assign weights to the vertex pairs of the bipartite graph according to a statistical model, so that heavy subgraphs would correspond to significant biclusters. We can tag each edge to incorporate the direction of regulation (up or down), as we shall see later. For now assume edges are not tagged.

Formally, given an input gene expression data, we form a bipartite graph  $G = (U, V, E)$  (e.g. Figure 5.3). 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  $u$ , i.e., if the expression level of  $v$  changes significantly in  $u$ . This discretization can be further developed using a discretisation function, adding edges according to the expression level of  $v$ . A bicluster corresponds to a subgraph  $H = (U', V', E')$  of  $G$ , and represents a subset  $V'$  of genes that are coregulated under a subset of conditions  $U'$  (see Figure 5.3). The *weight* of a subgraph (bicluster) is the sum of the weights of gene-condition pairs in it (edges and non-edges).

In order to assign statistical meaning to the weight of a subgraph, the authors developed statistical models for the bipartite graph representation of expression data. Using these models one can derive scoring schemes for assessing the significance of an observed subgraph (corresponding to a bicluster). This is done so that the score can be expressed as a sum of independent contributions from each of the node pairs (condition-gene) in the subgraph. Using this model, we can reduce the biclustering problem to the problem of finding heavy subgraphs in a bipartite graph.

### 5.5.2 A Simple Model

The first statistical model we present is a simplistic one, and is presented as a motivation for the more sophisticated model that will follow. Let  $H = (U', V', E')$  be a subgraph of  $G$ . Denote  $|U'| = m, |V'| = n, |E'| = k$ . The model assumes that edges occur independently with an equal probability  $p$ , where  $p = \frac{|E|}{|U||V|}$  (edge density). We shall now calculate  $P(H)$  the  $p$ -value of  $H$  (i.e. the tail of a binomial distribution). Then, the probability of observing

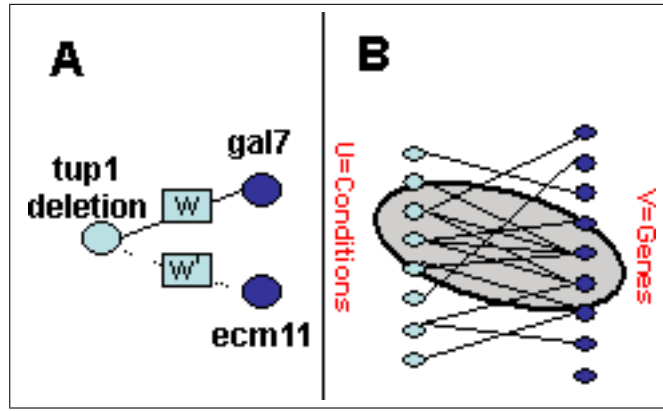


Figure 5.3: Source: [16]. SAMBA model : Gene expression data is modeled using a bipartite graph whose two sides correspond to the set of conditions  $U$  and the set of genes  $V$ . An edge  $(u, v)$  indicates the response of gene  $v$  in condition  $u$ . A statistical model assigns weights to the edges and non-edges of the graph. A: Part of the graph showing the condition “tup1 deletion” and its effect on the genes “gal7” (response) and “ecm11” (no response). B: A heavy subgraph (shaded) representing a significant bicluster.

a graph at least as dense as  $H$  according to this model is:

$$P(H) = \sum_{k'=k}^{nm} \binom{nm}{k'} p^{k'} (1-p)^{nm-k'} \quad (5.8)$$

This can then be bounded, assuming that  $p < 1/2$  by :

$$P(H) = \sum_{k'=k}^{nm} \binom{nm}{k'} p^{k'} (1-p)^{nm-k'} \leq 2^{nm} p^k (1-p)^{nm-k} \quad (5.9)$$

Our goal is to find a subgraph  $H$  with the lowest  $P(H)$ . Seeking a minimal weighted subgraph  $H$ , is equivalent to minimizing the logarithm of the upper bound:

$$\sum_{edges} (1 + \log p) + \sum_{non\ edges} (1 + \log(1-p)) \quad (5.10)$$

Which is the sum of the edges and the sum of non-edges.

### 5.5.3 A Refined Model

The simple model presented above is actually far from reality. In fact, if we look at the degree distribution of real gene expression data, we can identify a very non uniform behavior,

where some of the conditions and genes have very high degrees and others very low degrees. Because, a simple random graph model would result in very high scores for the "bicluster" defined by all high degree nodes in the graph, we next describe a refined null model that takes into account the variability of the degrees in  $G$ . This model incorporates the characteristic behavior of each specific condition and gene. This behavior may reflect some experiment artifact (some genes are more likely to have noisy measurements) or some biological actual meaning (some of the genes participate in many of the cellular pathways).

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  $d_w^G$  denote its degree in  $G$ . The refined 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  $(u, v)$ , or

$$p_{(u,v)} = \frac{|\{G' = (U, V, E') | \forall w, d_w^{G'} = d_w^G, (u, v) \in E'\}|}{|\{G' = (U, V, E') | \forall w, d_w^{G'} = d_w^G\}|} \quad (5.11)$$

We can estimate  $p_{(u,v)}$  using a Monte-Carlo Markov Chain process, starting from the original graph and performing a sequence of random edge swaps that preserve the degrees (A formal proof of a sampling lemma in the space of fixed degree graphs is however still an open problem). The probability of observing  $H$  is thus  $P(H) = (\prod_{(u,v) \in E'} p_{(u,v)}) \cdot (\prod_{(u,v) \in \overline{E'}} (1 - p_{(u,v)}))$ . However, we cannot simply compare subgraphs according to this probability, since it improves (decreases) as the size of  $H$  increases (decreases).

To overcome this problem, we use a likelihood ratio to capture the significance of biclusters. The null model is as stated above. For the alternative model we assume that each edge of a bicluster occurs with constant probability  $p_c > \max_{(u,v) \in U \times V} p_{(u,v)}$ . This model reflects our 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  $\log \frac{1-p_c}{1-p_{(u,v)}} < 0$ , we conclude that the score of  $H$  is simply its weight.

### 5.5.4 Finding heavy subgraphs

Having assigned weights for edges in our model bipartite graph, such that the weight of a subgraph encode some reasonable likelihood ratio, we now turn to the problem of identifying the maximum likelihood bicluster, which is, under our formulation, the heaviest subgraph.

The computational problem of finding the largest node biclique (the one with the largest number of vertices) in a bipartite graph has an elegant polynomial time algorithm (using

matching). Our problem, however, is more closely related to the problem of finding the largest edge biclique (the biclique with the largest number of edges) which is NP-hard for both unweighted [13] and weighted graphs [9]. In fact, the problem of finding the heaviest bipartite subgraph when edges are assigned positive weights and non edges negative weights is hard as well (by reduction from CLIQUE). We thus enforce additional limitation on our graph, namely, restrict the degree of the genes side, so that a gene that responds in more than  $d$  conditions is ignored. This enables us to develop a polynomial algorithm which is later used as the basis for a practical implementation that can avoid the degree restriction. Note that according to the statistical model, genes with high degree contribute less to the significance of a bicluster, so running the algorithm without them may not be a very restricting limitation.

### Maximum Bounded Biclique

We start by describing an  $O(|V|2^d)$ -time algorithm to find a maximum weight biclique in a bipartite graph whose gene vertices have  $d$ -bounded degree. This algorithm will be a key component in the algorithms that follow.

Let  $G = (U, V, E)$  be a bipartite graph. We say that  $G$  has  $d$ -bounded gene side, if every  $v \in V$  has degree at most  $d$ . Let  $w : U \times V \rightarrow \mathcal{R}$  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  $G$ . We denote  $|V| = n$  throughout.

**Problem 5.1 (Maximum Bounded Biclique)** Given a weighted bipartite graph  $G$  with a  $d$ -bounded gene side, find a maximum weight complete subgraph of  $G$ .

**Theorem 5.1** *The maximum bounded biclique problem can be solved in  $O(n2^d)$  time and space.*

**Proof:** Observe that a maximum bounded biclique  $H^* = (U^*, V^*, E^*)$  in  $G$  must have  $|U^*| \leq d$ . If not, it's easy to show that there must be a vertex in  $U^*$  whose removal will give a better biclique, with contradiction to the assumption that  $H^*$  is maximal. Figure 5.5 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. An iteration over subsets of  $N(v)$  can be done in  $O(2^d)$  time. Hence, the algorithm spends  $O(n2^d)$  time on the hashing and finding  $U_{best}$ . Since  $|U_{best}| \leq d$ , computing  $V_{best}$  can be done in  $O(nd)$  time, so the total running time is  $O(n2^d)$ . The space complexity is  $O(n2^d)$  due to the hash-table. ■

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.

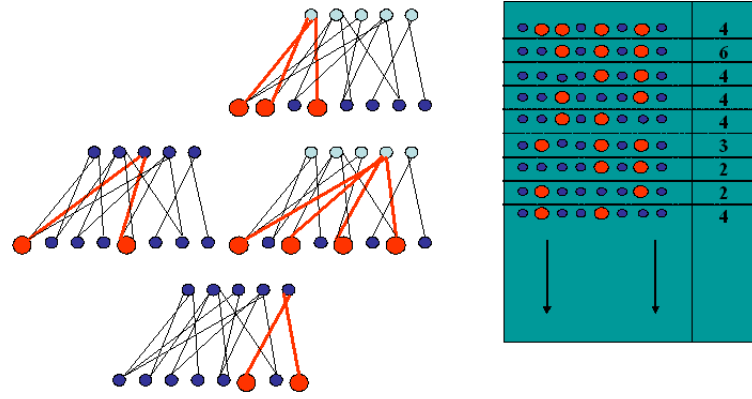


Figure 5.4: Example of the hashing technique, in which a maximal Biclique, is found for a subgraph. Each row stands for a subset of genes, and a score (for the row). The score is initialized as zero, and for every condition that is connected to all genes in the subset, the size of the subset is added to it. Thus, the score of a subset is the number of genes multiplied by the number of conditions connected to the whole subset. The maximal biclique is the genes in the row with the highest score, and the conditions that are connected to the whole subset.

### Finding Heavy Subgraphs Algorithm

We will now show how to extend the latter algorithm to find heavy subgraphs which are not necessarily complete. For simplicity we shall describe the algorithm assuming that each edge has weight  $+1$  and each non-edge has weight  $-1$ . Extension to more general weights can be done in a similar manner. Formally, given a bipartite graph  $G = (U, V, E)$  define a weight function  $w : U \times V \rightarrow \{-1, 1\}$  such that  $w((u, v)) = 1$  for  $(u, v) \in E$ , and  $w((u, v)) = -1$  for  $(u, v) \in (U \times V) \setminus E$ . Consider the following problem:

**Problem 5.2 (Maximum Bounded Bipartite Subgraph)** Given a bipartite graph  $G$  with  $d$ -bounded gene side, find a maximum weight subgraph of  $G$ .

**Lemma 5.2** *Let  $H^* = (U^*, V^*, E^*)$  be a maximum weight subgraph of  $G$ . Then every vertex in  $H^*$  is connected to at least half the vertices on the other side of  $H^*$ .*

**Proof:** Follows from the choice of weights, since if a vertex  $v \in V^*$  has less than  $\lceil |U^*|/2 \rceil$  neighbors, then removing  $v$  from  $H^*$  will result in a heavier subgraph. The proof for  $u \in U^*$  is symmetric. ■

**Lemma 5.3** *Let  $H^* = (U^*, V^*, E^*)$  be a maximum weight subgraph of  $G$ . For each set  $X \subseteq U^*$  there exists a subset  $Y \subseteq X$  with  $|Y| \geq \lceil |X|/2 \rceil$  such that  $Y \subseteq N(v)$  for some  $v \in V^*$ .*

```

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
         $weight[S] \leftarrow weight[S] + w(S, \{v\})$ 
        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 5.5: Source: [16]. An algorithm for the maximum bounded biclique problem.

**Proof:** Assume there exists  $X \subseteq U^*$  such that all subsets  $X \cap N(v), v \in V^*$  are of size smaller than  $\lceil |X|/2 \rceil$ . Then the weight of the subgraph induced on  $(U^* \setminus X, V^*)$  exceeds that of  $H^*$ , a contradiction. ■

**Corollary 5.4** *A maximum weight subgraph of  $G$  has at most  $2d$  vertices from  $U$ .*

**Proof:** let  $X$  be  $U^*$ , since  $Y \subseteq N(v)$  then  $d \geq |Y|$ . From lemma 5.3 we know  $|Y| \geq \lceil |U^*|/2 \rceil$ , so we get  $U^* \leq 2d$ . ■

**Lemma 5.5** *Let  $H^* = (U^*, V^*, E^*)$  be a maximum weight subgraph of  $G$ . Then  $U^*$  can be covered by at most  $\lfloor \log(2d) \rfloor$  sets, each of which is contained in the neighborhood of some vertex in  $V^*$ .*

**Proof:** Denote  $|U^*| = t$ . By Lemma 5.3 there exists a subset  $Y \subseteq U^*$  with  $|Y| \geq \lceil t/2 \rceil$ , such that  $Y \subseteq N(v)$  for some  $v \in V^*$ . The same holds for the set  $U^* \setminus Y$ , and we can continue in this manner until we cover  $U^*$ . By construction we have at most  $\lfloor \log t \rfloor$  sets in the cover. Since  $t \leq 2d$  by Corollary 5.4, the result follows. ■

Lemma 5.5 implies an algorithm to find a maximum weight subgraph. The algorithm tests all collections of at most  $\lfloor \log(2d) \rfloor$  subsets of neighborhoods of vertices in  $V$ . Since there are  $O(n2^d)$  such subsets we have:

**Corollary 5.6** *The maximum bounded bipartite subgraph problem can be solved in  $O((n2^d)^{\log(2d)})$  time.*

### 5.5.5 Incorporating the Direction of Expression Changes

In our discussion so far, the underlying bipartite graph used for modeling the data contained edges for significantly changed genes, but ignored the type of change (increase or decrease in the expression level). We can integrate additional information into the model by associating a sign of "up" or "down" with each edge. We now have three types of binary relations in our bipartite graphs: An "up" edge, a "down" edge or no edge. It is reasonable to look for a bicluster in which the conditions tend to affect genes in a *consistent* way, i.e., two clustered conditions should either have always the same effect or always the opposite effect on each of the genes. This leads to the definition of a consistent biclique:

Given a bipartite graph  $G = (U, V, E)$  with edge sign function  $c : E \rightarrow \{-1, 1\}$ , we say that an induced biclique  $H = (U', V', E')$  is *consistent* if there exists an assignment  $\tau : U' \cup V' \rightarrow \{-1, 1\}$  such that for every  $v \in V', u \in U'$  we have  $c((u, v)) = \tau(u)\tau(v)$ .

The maximum consistent biclique problem in degree-bounded graphs can be solved in polynomial time by reduction to the standard maximum biclique problem:

**Lemma 5.7** *There is an  $O(n2^d)$ -time algorithm for the maximum consistent bounded biclique problem on graphs with  $d$ -bounded gene side.*

**Proof:** Given  $G$  and  $c$ , we construct the graph  $G' = (U \cup \bar{U}, V \cup \bar{V}, E')$ , where  $\bar{U}$  and  $\bar{V}$  are copies of  $U$  and  $V$ , respectively, and  $E' = \{(u, v), (\bar{u}, \bar{v}) | (u, v) \in E, c((u, v)) = 1\} \cup \{(u, \bar{v}), (\bar{u}, v) | (u, v) \in E, c((u, v)) = -1\}$ . Suppose that  $(U', V')$  induce a consistent biclique in  $G$  of size  $k$  with a sign assignment  $\tau$ . Then  $\{v \in U' \cup V' | \tau(v) = 1\} \cup \{\bar{v} | v \in U' \cup V', \tau(v) = -1\}$  induce a biclique of size  $k$  in  $G'$ . Conversely, if  $(U', V')$  induce a biclique in  $G'$ , then no pair  $u, \bar{u}$  is contained in it, so  $\{v \in U \cup V | v \in U' \cup V' \text{ or } \bar{v} \in U' \cup V'\}$  induce a consistent biclique in  $G$  of the same size, where  $\tau(v) = 1$  if  $v \in U' \cup V'$  and  $\tau(v) = -1$  if  $\bar{v} \in U' \cup V'$ . The claim thus follows from Theorem 5.1. ■

We can use similar ideas to handle consistent subgraphs. All we need is to ensure that the sum of weights of edges from a node to two opposite sign neighbors is negative. Given this condition, the algorithms from previous sections can be applied directly and generate consistent subgraphs.

### 5.5.6 The SAMBA Algorithm

The scoring scheme and combinatorial algorithms developed above can be combined to create the practical SAMBA algorithm (Figure 5.6). We cannot apply the theoretical algorithm directly on large data sets since for reasonable degree bounds (e.g. 60),  $2^d$  is too large. The suggested solution avoids hashing all subsets of incoming neighbors per gene and exhaust only subset of small size (typically 4 – 6 conditions). The heaviest condition sets are then used as kernels that are extended by a local search procedure.

SAMBA thus works as follows: We first form the bipartite graph and calculate vertex pair weights using one of the weighting methods described above. We consider a gene to be up (down) regulated in a condition if its standardized level with mean 0 and variance 1 is above 1 (below -1). Depending on the data, we may choose to work with signed or unsigned graphs.

In the second phase of the algorithm we apply the hashing technique of the algorithm previously shown (Figure 5.5) to find the heaviest bicliques in the graph. In fact, SAMBA looks for the  $k$  best bicliques intersecting every given condition or gene. We ignore 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 third phase of the algorithm performs a local improvement procedure on the biclusters derived from the previous phase. The procedure iteratively applies the best modification to the bicluster (addition or deletion of a single vertex) until no score improvement is possible.

To avoid similar biclusters whose vertex sets differ only slightly, a greedy algorithm is applied. We iterate over all generated biclusters, ordered by their score, and filter out biclusters whose intersection with a previous solution (number of shared conditions times number of shared genes) is above  $L\%$ .

An implementation of SAMBA can handle large data sets in a few minutes (15,000 genes, 500 conditions,  $d = 40$ ,  $N_1 = 4$ ,  $N_2 = 6$ ,  $K = 20$ ,  $L = 30$ ).

### 5.5.7 Validating Biclustering Quality

As the reader may have noted, for many applications in computational biology, it is hard to compare different algorithms and methodologies and state clearly which one is "better". It is however, very important for any scientific discipline to have means for evaluating the quality of a given result and to make sure the field is indeed making progress. Described below are two examples of general methodologies for assessing algorithms performance : comparative analysis, which matches algorithmic results with some external knowledge, and intrinsic validation, which uses randomization to evaluate the significance of the signals discovered.

#### Comparative analysis

One way for evaluating bicluster algorithms is by using prior biological knowledge as some form of a gold standard that we can compare our set of biclusters to. A *correspondence plot* depicts the distribution of  $p$ -values of the produced biclusters, using for evaluation a known (putatively correct) classification of conditions (e.g., to various cancer types) or a given gene annotation. We describe the plot when a classification is given. For each value of  $p$  on a logarithmic scale, the plot presents the fraction of biclusters whose  $p$ -value is at most  $p$  out of the  $k$  best biclusters.



```

SAMBA( $U, V, E, w, d, N_1, N_2, k$ ):
 $U$  : conditions.  $V$  : genes.
 $E$  : graph edges.  $w$  : edge/non-edge weights.
 $N_1, N_2$  : 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  $best[v][1 \dots k]$  to the  $k$  heaviest  $S$  such that  $v \in S$ .
For each  $v \in V, i \in \{1..k\}$ 
     $S = best[v][i]$ 
     $V' \leftarrow \cap_{u \in S} N(u)$ .
     $B \leftarrow S \cup V'$ .
    Do {
         $a = \operatorname{argmax}_{x \in V \cup U} (w(B \cup x))$ 
         $b = \operatorname{argmax}_{x \in B} (w(B - x))$ 
        if  $w(B \cup a) > w(B - b)$  then  $B = B \cup a$ 
        else  $B = B - b$ 
    } While improving
    Store  $B$ .
Post process to filter overlapping biclusters.

```

Figure 5.6: The practical biclustering algorithm.

$p$ -values are calculated according to the known classification as follows: Suppose prior knowledge partitions the  $m$  conditions into  $k$  classes,  $C_1, \dots, C_k$ . Let  $B$  be a bicluster with  $b$  conditions, out of which  $b_j$  belong to class  $C_j$ . The  $p$ -value of  $B$ , assuming its most abundant class is  $C_i$ , is calculated as  $p(B) = \sum_{k=b_i}^b \binom{|C_i|}{k} \binom{m-|C_i|}{b-k} / \binom{m}{b}$ . Hence, the  $p$ -value measures the probability of obtaining at least  $b_i$  elements from the class in a random set of size  $b$ . One should note, that high quality biclusters can also identify phenomena that are not covered by the given classification. Nevertheless, we expect a large fraction of the biclusters to conform to the known classification.

As an example for the usage of correspondence plots, we present the analysis of outputs from SAMBA algorithm compared to Cheng and Church's algorithm. Running on the same data set (the lymphoma data of [2]), a collection of biclusters from both algorithm were analyzed using the known classification of conditions to different clinical types (DLBCL, CLL, FL and more). As a reference, a correspondence plot calculated on a random annotation of the 96 samples (preserving class sizes) was added. The results (Figure 5.7(A)) clearly indicate that SAMBA's biclusters are much more concordant with the biological information.

## Intrinsic validation

A second method for validating the quality of biclusters is by analyzing the results on random data sets. We should make sure that the results we consider as statistically significant are not obtained from random data. The details of randomization may be critical to the integrity of such test. Assume we are using a uniformly random graph model (the simple model described above) and we randomize the data according to it. Then the artifact we have mentioned before, causing the identification of high degree nodes as biclusters would not be discovered since the random graph model will follow our originally restricting assumption.

Figure 5.7(B) describes the results of a randomization test done on SAMBA using a degree preserving random graph model. The analysis was done on two data sets, first the real data was biclustered and the significance of each bicluster was calculated. Then the same procedure was done with a random graph preserving all vertex degrees. It can be seen that significance values on the random data are well separated from those computed on the original data and, furthermore, only two random biclusters have significance values below zero. The scatter plot not only demonstrates that heavy biclusters are non random but also provides empirical evidence to the relation of the SAMBA likelihood score and the more formal significance measure.

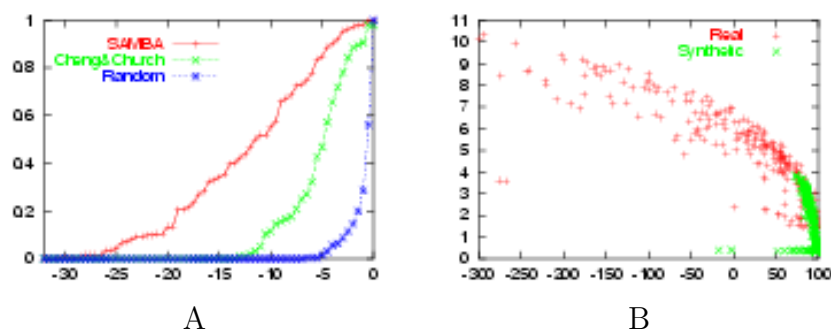


Figure 5.7: Source: [16]. A: Performance of different weighting schemes and algorithms. Correspondence plots for SAMBA, the algorithm of Cheng and Church [4], and random biclusters. Likelihood weights use  $p_c = 0.9$ . For each value of  $p$  on a logarithmic scale, the plot presents the fraction of biclusters whose  $p$ -value is at most  $p$  out of the (say) 100 best biclusters. B: Scatter plots of significance values of biclusters (x-axis) vs. their log likelihood (weight) on synthetic and real data. x-axis: significance value, y-axis: bicluster weight.

### 5.5.8 Source Heterogeneity

SAMBA can be used to evaluate several data sources, separately or combined. The more information we can add to the graph, the more precise our results will be, this is because

true results will be amplified by the new properties added. Thus, different types of datasets are used in conjunction to produce more accurate results. Examples of other data that can be added include transcription factor binding profiles, protein-protein interaction data and phenotype profiles.

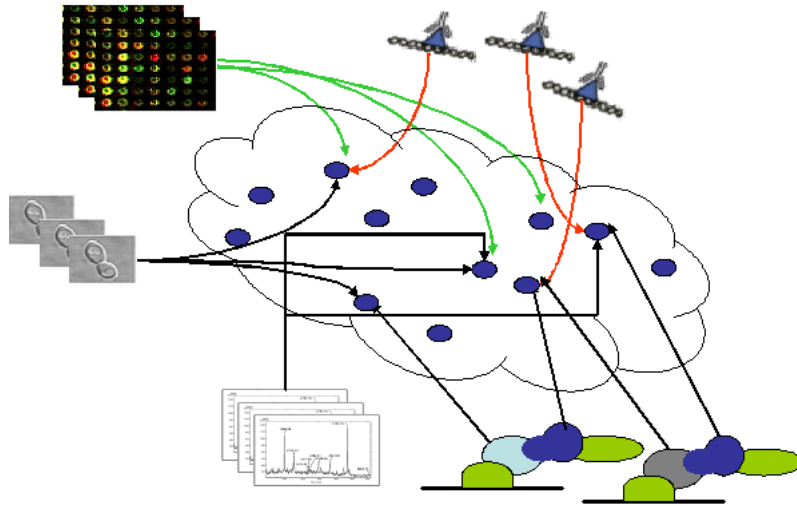


Figure 5.8: Example of a variety of sources combined to produce a better result using the SAMBA algorithm.

### 5.5.9 Functional Annotation using biclusters

The output of a bicluster algorithm is a collection of significant local signals in the data. Such signals may be used in diverse applications. One example for an application of biclusters is for automatic annotation of genes. The budding yeast, which is one of the most characterized eukaryotes up to date, contains about 6,200 genes, only half of which have a known function. We can use a large database of gene expression and a set of derived biclusters to try and associate unknown genes with a function. The idea is simple, whenever a significant fraction of the characterized genes in a bicluster share a common functional class, it is likely that the other genes in the bicluster are also related to this class.

A compiled data set of yeast gene expression, including 515 conditions for the 6,200 yeast ORFs was used to test this idea [10, 6, 7, 14, 11]. The data was collected from five different experiments. Analysis by SAMBA generated 2,406 biclusters ranging over 4,961 genes and 515 conditions. The source for the known functional annotation was the SGD database [1], which includes 3,000 annotations from the Gene Ontology [5] vocabulary.

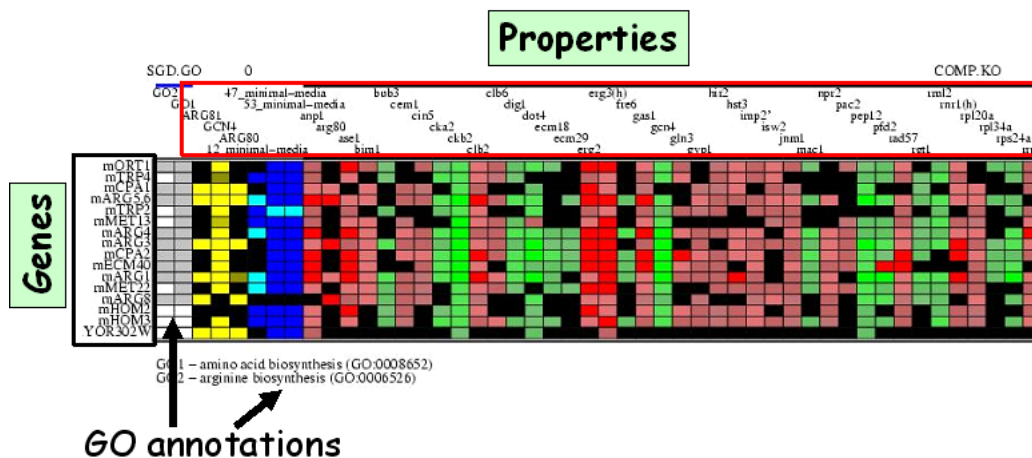


Figure 5.9: Example of a SAMBA bicluster based on different properties. Other than the gene expression profile, transcription factor binding profiles and different cell growth conditions are combined in this bicluster. On the left, two GO annotations are added to show the quality of the results.

The bicluster set was filtered to include only those biclusters in which more than 60% of their annotated members had the same class. Out of those, only biclusters that were functionally enriched ( $p$ -value below  $10^{-4}$ ) were used. The unannotated genes in those biclusters were now assigned to the most abundant class. Note that each gene may be annotated more than once, as is the case for the curated GO annotations. For cross validation, 100 runs were performed and in each one 30% of the annotations were hidden. The overall average success rate in annotating the hidden genes was calculated.

The results of these runs are summarized in Figure 5.10(A,B). Overall, 81.5% of the test set annotations matched those known from SGD, demonstrating that we can extrapolate functional annotation using biclusters.

### 5.5.10 Global Transcriptional Network

Another application that can be applied on the biclusters is identification of Transcription Factor (TF) binding sites. Genes in the same biological modules, that is genes that are expressed corporately under some conditions related to a biological process, might be derived by one or more common TFs. TFs are important for understanding the biological regulation mechanism. In several cases, TF binding property can even explain the apparent biological coregulation. In Figure 5.11 we can see a process-TF map derived from samba biclustering results. The biological modules are represented by ovals and the TFs by circles. An edge

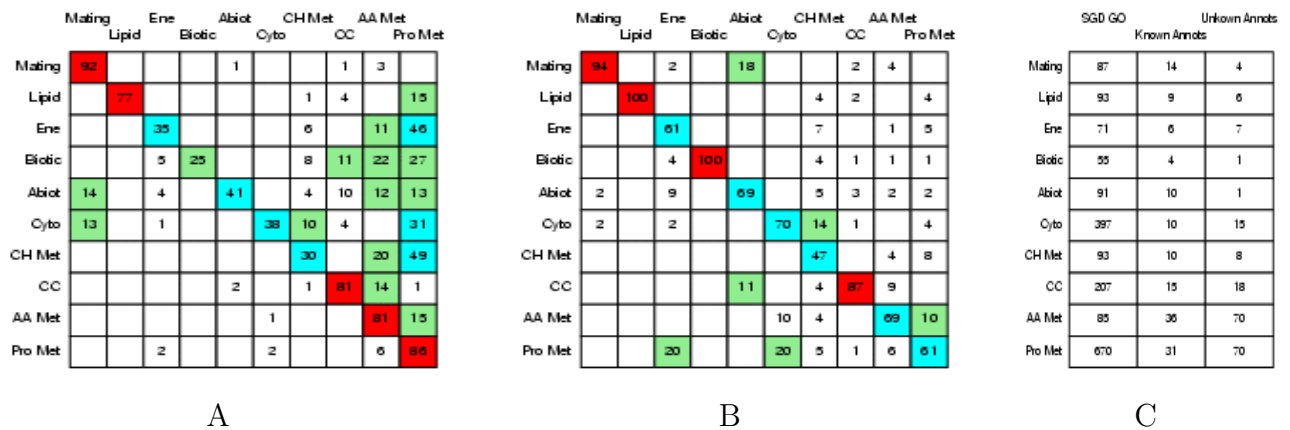


Figure 5.10: Source: [16]. Yeast functional annotation. A: Annotation specificity. The table depicts the annotation accuracy measured using 70:30 cross-validation. Rows represent classes assigned using our method and columns represent SGD GO classes. Cell  $(x, y)$  contains the percentage of genes annotated  $x$  that belong to GO class  $y$ . Higher percentages are darker. B: Annotation sensitivity calculated w.r.t. annotated genes only. Cell  $(x, y)$  contains the percentage of SAMBA annotated genes that belong to GO class  $y$  and were annotated  $x$ . C: Annotation of unknown genes. The table shows for each functional class its size in the SGD GO annotation, the number of genes that belong to this class and were annotated by SAMBA, and the number of unknown genes assigned to this class by SAMBA.

between a process and a TF represents an enrichment of the TF in the module's genes. This map enables us to generally view the Transcriptional regulation mechanism. As can be seen, some TFs can coregulate several processes. For example MBP1 that has considerable representation in the *DNA metabolism* and *Cell Cycle* processes.

### 5.5.11 SAMBA - Discussion

The SAMBA algorithm is fast and allows simultaneous normalization of genes and conditions and integration of heterogenous data. It is well suited for query based usage. Disadvantages of SAMBA: discretisation and redundancies.

## 5.6 Summary

In this and the previous lectures, we have reviewed some methods for biclustering. In each of these methods we focused on the discovery of one, presumably best, bicluster. To obtain a set of biclusters we had to apply our algorithm many times, possibly in parallel. A different approach may be to try and discover a multiple biclustering model for the entire data. This

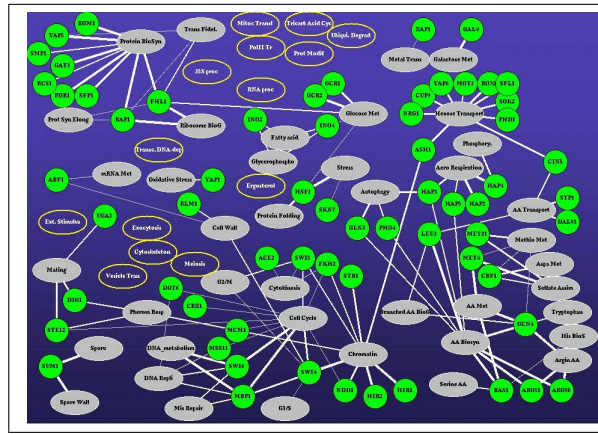


Figure 5.11: Source: [15] Functional modules and their transcription factors in *S. cerevisiae*. Modules with significant functional enrichment for a particular process (with  $p - value \leq 0.01$ ) are grouped and plotted as ovals with the process name. TFs with binding profiles associated with any of these modules are marked as green circles and connected to the associated process. Modules enriched in more than one process may appear in several places in the figure. The thickness of the connecting lines is inversely proportional to the p-value of the functional enrichment in the associated module. The map was automatically generated by SAMBA using no prior biological knowledge. Key abbreviations: Met: Metabolism, AA: Amino Acid, Tran: Transport.

approach has the advantage of enabling a global view of the data and reducing overfitting and redundancies. On the other hand, as biclusters are by definition local patterns, a global approach to biclustering is not a trivial task. A family of methods using a basically global approach is based on probabilistic models. For example, we can model each sub column and row as having a typical value distribution, which may be different from the background. We can then model the entire data set by tiling it with a small number of biclusters, each having a characteristic distribution. To discover such tiling, it is possible to use variants of the EM algorithm.

As we have seen, the many possible definitions of the biclustering problem have a central importance in the design and implementation of algorithms for solving it. As for now, there is no single solution that is accepted as a standard. As gene expression data accumulates rapidly, the need for a robust and efficient biclustering algorithms is nevertheless increasing.

# Bibliography

- [1] <http://genome-www.stanford.edu/saccharomyces/>.
- [2] A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. C. Boldrick, H. Sabet, T. Tran, X. Yu, J. I. Powell, L. Yang, G. E. Marti, T. Moore, J. Hudson, L. Lu, D. B. Lewis, R. Tibshirani, G. Sherlock, W. C. Chan, T. C. Greiner, D. D. Weisenburger, J. O. Armitage, R. Warnke, and L. M. Staudt. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.
- [3] 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.
- [4] Y. Cheng and G.M. Church. Biclustering of expression data. In *Proc. ISMB’00*, pages 93–103. AAAI Press, 2000.
- [5] The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
- [6] A. P. Gasch et al. Genomic expression programs in the response of yeast cells to environmental changes. *Mol Biol Cell*, 11:4241–57, 2000.
- [7] 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.
- [8] 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.
- [9] Dorit S. Hochbaum. Approximating clique and biclique problems. *Journal of Algorithms*, 29(1):174–200, 1998.
- [10] TR. Hughes et al. Functional discovery via a compendium of expression profiles. *Cell*, 102:109–26, 2000.

- [11] T. Ideker, J.A. Thorsson, V. Ranish, R. Christmas, J. Buhler, J.K. Eng, R. Bumgarner, D.R. Goodlett, R. Aebersold, and Hood L. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science*, 291:929–34, 2001.
- [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] R. Peeters. The maximum edge biclique problem is NP-complete. [cite-seer.nj.nec.com/peeters00maximum.html](http://cite-seer.nj.nec.com/peeters00maximum.html).
- [14] 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.
- [15] 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.
- [16] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(Suppl 1):S136–44, 2002.