Analysis of DNA Chips and Gene Networks **Spring Semester, 2009**

Lecture 11: December 31, 2009

Lecturer: Ron Shamir *Scribe: Boris Kostenko*

BIMAX

11.1 Introduction.

In the course we have already seen different Biclustering methods such as Cheng-Church, ISA, SAMBA (*see scribe 5*), OPSM (*see scribe 9*). The method described in this lecture is Bimax - an algorithm due to Prelić *et al.* [2]. It uses a simple data model reflecting the fundamental idea of biclustering, while aiming to determine all optimal biclusters in reasonable time. This method has the benefit of providing a basis to investigate (1) the usefulness of the biclustering concept in general, independently of interfering effects caused by approximate algorithms, and (2) the effectiveness of more complex scoring schemes and biclustering methods in comparison to a plain approach.

11.2 Model.

The model assumes two possible expression levels per gene: no change and change with respect to a control experiment (To this end, a preprocessing step normalizes log expression values and then transforms matrix cells into discrete values, e.g. by using a 2-fold change cutoff.) Accordingly, a set of m microarray experiments for n genes can be represented by a binary matrix $E^{n\times m}$, where a cell e_{ij} is 1 whenever gene i responds in the condition j and otherwise it is 0. A bicluster (G, C) corresponds to a subset of genes $G \subseteq \{1, \ldots, n\}$ that jointly respond across a subset of samples $C \subseteq \{1, ..., m\}$. In other words, the pair (G, C) defines a sub matrix of E for which all elements equal 1. Note that, by definition, every cell e_{ij} having value 1 represents a bicluster by itself. However, such a pattern is not interesting; instead, we would like to find all biclusters that are inclusion maximal, i.e. those that are not properly contained by any other biclusters. In graph theory if we consider representation in adjacency matrix of the graph (each cell represents if node of the row is connected with node of the column), the problem translates to finding all the maximal bicliques. A Biclique is a fully connected bipartite graph where every vertex of the first set is connected to every vertex of the second set. In our case we present genes as matrix rows and conditions as matrix columns and seek maximal bicliques between genes and conditions.

11.3 An Incremental Algorithm.

The incremental procedure, see below, is based on work by Alexe *et al.* [1], who propose a method to find all inclusion-maximal cliques in general graphs. Shortly summarized, each

node in the input graph is visited, and all maximal cliques are found that contain that node. A visit-to-a-node operation comprises an iteration through all other nodes of the graph as well, and each newly found bicluster is globally extended to its maximality.

```
1:  procedure IncrementalAlgorithm(E)
2:      M ← ∅
3:      for i ← 1 to n do
4:          C* ← {j | e_ij = 1 ∧ 1 ≤ j ≤ m}
5:          for each (G, C) ∈ M do
6:              C' ← C ∩ C*
7:              if ∃ (G'', C'') ∈ M with C'' = C' then
8:                  M ← M \ {(G'', C'')} ∪ {(G'' ∪ {i}, C'')}
9:              else
10:                 M ← M ∪ {(G  ∪ {i}, C')}
11:             end if
12:         end for
13:         if ∄ (G'', C'') ∈ M with C'' = C* then
14:             M ← M ∪ {({i}, C*)}
15:         end if
16:     end for
17:     return M
18: end procedure
```

Figure 11.1: An Incremental Algorithm

Algorithm description.

We add the rows one by one to the pool of biclusters (rows with the same condition set fall into the same bicluster). For each added row we run through all the biclusters in the pool and concentrate on the common conditions of bicluster group and the current row - C'. If there is a bicluster in the pool with condition set equals to C', we add the current row to it. Otherwise we create a new bicluster: genes set contains the current row and all the genes of bicluster, with which we calculated C', and C' as conditions set. After algorithm run we have set of all possible maximal biclusters, including the one with an empty condition set.

Theorem 11.1 *The time complexity of the Incremental Algorithm is* $\Theta(nm\beta \log \beta)$*, where* β *is the number of all inclusion-maximal biclusters in* $E^{n\times m}$*. The space complexity is* $O((m+n)\beta)$*.*

Proof: We maintain the set M in a lexicographic order according to the sets C. For each row the algorithm performs the following:

- calculates C^* — in $O(m)$ time
- iterates through $O(\beta)$ biclusters and for each bicluster:

- calculates the intersection with C — in $O(m)$ time
- seeks in M a bicluster with $C'' = C'$ — this takes $O(m \log \beta)$ time by binary search on M
- updates M by removing at most one bicluster and adding to it at most two keeping sorted order. Each of teh three operations takes $O(m \log \beta)$ time

Hence, for each row the algorithm performs $O(m\beta \log \beta)$ operations, and in total the time complexity is $O(nm\beta \log \beta)$. Note that since $\beta \leq 2^{\min(n,m)} - 1$ we have $\log \beta \leq \min(n, m)$. Replacing $\log \beta$ in the complexity we receive $O(nm \min(n, m)\beta)$, and if $m < n$ the total complexity becomes $O(nm^2\beta)$. The Space complexity is $O((m + n)\beta)$, since for each of the $O(\beta)$ biclusters we need to record the sets of size at most m and n. ■

11.4 Bimax algorithm.

A key problem in the above algorithm is the space complexity. We now decribe the Bimax algorithm due to Prelić *et al.* [2] that is more space efficient. The algorithm realizes the divide-and-conquer strategy. Rows are added one by one. When a row is added, the column set is partitioned into C_U — the columns in which the new row has ones, and its complement C_V (compare Figure 11.2[1]). The row set is split into G_U — the rows that have only ones in C_U, G_V — those that have ones in C_V only, and G_W - those that have ones in both. Let U be the submatix $(G_U \cup G_W, C_U)$ and let V be the submatix $(G_W \cup G_V, C_V)$.

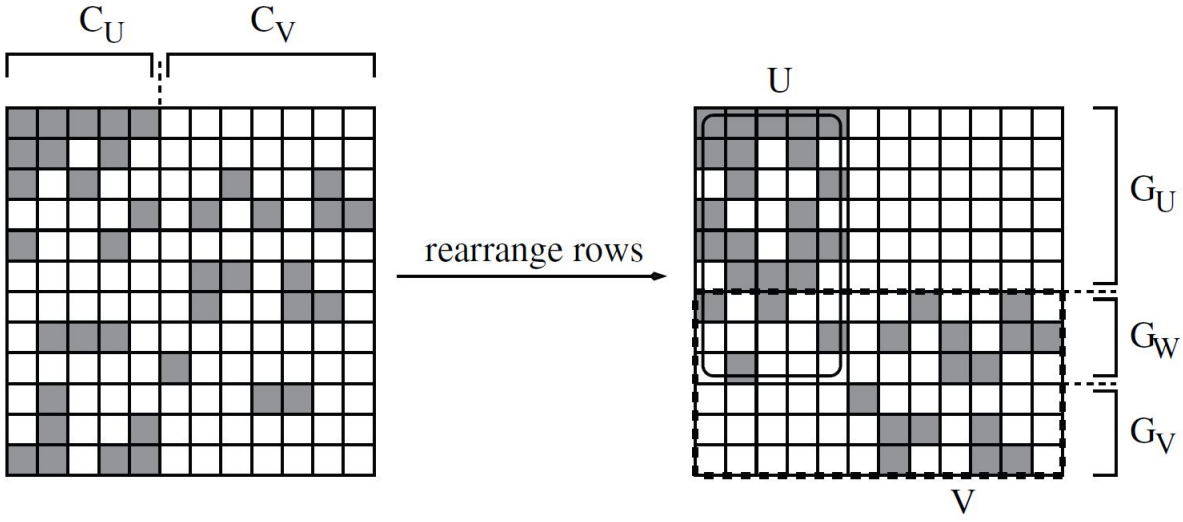

Figure 11.2: Bimax Algorithm run example.

Special operations are required for processing the V submatrix. The algorithm needs to guarantee that only optimal, i.e., inclusion-maximal biclusters are generated. The problem

[1]The source of all figures in this scribe is Prelić *et al.* [2]

arises because V contains parts of the biclusters found in U, and as a consequence we need to ensure that the algorithm only considers those biclusters in V that extend over C_V . The parameter Z serves this goal. It contains sets of columns that restricts the number of admissible biclusters. A bicluster (G, C) is admissible, if (G, C) shares one or more columns with each column set C^+ in Z, i.e., $\forall C^+ \in Z : C \cap C^+ \neq \emptyset$.

The code of the algorithm is presented in Figure 11.3 below.

Theorem 11.2 *Algorithm* Bimax *can be implemented in* $O(nm \min(n, m)$ *time and* $O(nm \log \beta)$ *space.*

Proof: In the procedure *reduce*, one can observe that the number of column sets stored in Z is bounded by the number of rows, n, and each column set contains at most m elements. If Z is implemented as a list and C^* is represented by an array, the `If` statement in line 49 can be executed in $O(nm)$ time. Accordingly, one call to *reduce* takes $O(n^2m)$ steps resp. $O(m^2n)$ steps, if $n > m$ and the transposed matrix is considered. Overall, the running time complexity is of order $O(nm \min(n, m))$. The partitioning of a submatrix is accomplished by the procedure *divide*.We assume that all sets except of C^* are implemented using list structures, while C^* is stored in an array. Thereby, the inclusion tests can be performed in time $O(m)$, and the entire loop takes $O(nm)$ steps. Overall, the running time of the procedure including *reduce* is $O(nm \min(n, m))$.

We now calculate *conquer* time complexity excluding recursive calls. The main procedure *conquer* requires $O(nm)$ steps to check whether (G, C) represents a valid bicluster (lines 7 to 9), and $O(1)$ steps to perform the union operations at lined 18 and 21, again assuming a list implementation. Altogether, one invocation of *conquer* including *divide* takes $O(nm \min(n, m))$ time.

The question now is how many times *conquer* is executed. Taking into account that every invocation of *conquer* returns at least one inclusion-maximal bicluster, there are at maximum β procedure calls that do not perform any further recursive calls. In other words, the corresponding recursion tree, where each node represents one instance of conquer and every directed edge stands for a recursive invocation, has at most β leaves. Each inner node of the recursion tree has an outdegree of 1 or 2, depending on whether G_W and G_V are empty (G_U is always non-empty except of the special case that E contains only 0-cells).

Suppose an instance of *conquer* in the tree that only has one child to which the submatrix U is passed. U has at least one row that contains a 1 in all columns of U; this is the row according to which the partitioning in the parent is performed. Now, either there is another row in U that contains both 0s and 1s (line 25) or all remaining rows only contain 1s. In the former case, the partitioning of U produces a non-empty set G_W and therefore the outdegree of the child is two. In the latter case, the submatrix resulting from the partitioning contains only 1s, which in turn, means that the following invocation of conquer is a leaf in the recursion tree. Therefore, at least one half of all inner nodes have an outdegree greater than 1.

We first give an upper bound for the number of inner nodes with more than one child, and for this purpose disregard all nodes with outdegree 1. Consider a tree where all inner nodes have an outdegree of 2 and the number of leaves equals β. Then the number of inner nodes is less than $2^{(log_2\beta)+1} = 2\beta$. For the recursion tree, this means that there are at maximum $2 \cdot 2\beta$ inner nodes, and as a consequence the overall number of nodes and invocations of *conquer* is $O(\beta)$. By combining the two main results, (i) one *conquer* call needs $O(nm \min(n, m))$ steps and (ii) there are at maximum $O(\beta)$ invocations of *conquer*, we obtain the upper bound for the running-time of the *Bimax* algorithm.

We now show that the space complexity is $O(nm \log \beta)$ or $O(nm \min(n, m))$. As it is shown above, the number of *conquer* invocations is $O(\beta)$. Having two invocation in a *conquer* call, the depth of the recursion tree is not more than $O(\log \beta)$. In each invocation we need to maintain the set Z of size less than $n \times m$, hence total space consumption is $O(nm \log \beta)$. ■

```
1:  procedure Bimax(E)
2:      Z ← ∅
3:      M ← conquer(E, ({1,...,n}, {1,...,m}), Z)
4:      return M
5:  end procedure

6:  procedure conquer(E, (G, C), Z)
7:      if ∀i ∈ G, j ∈ C : e_ij = 1 then
8:          return {(G, C)}
9:      end if
10:     (G_U, G_V, G_W, C_U, C_V) = divide(E, (G, C), Z)
11:     M_U ← ∅, M_V ← ∅
12:     if G_U ≠ ∅ then
13:         M_U ← conquer(E, (G_U ∪ G_W, C_U), Z)
14:     end if
15:     if G_V ≠ ∅ ∧ G_W = ∅ then
16:         M_V ← conquer(E, (G_V, C_V), Z)
17:     else if G_W ≠ ∅ then
18:         Z' ← Z ∪ {C_V}
19:         M_V ← conquer(E, (G_W ∪ G_V, C_U ∪ C_V), Z')
20:     end if
21:     return M_U ∪̇ M_V
22: end procedure

23: procedure divide(E, (G, C), Z)
24:     G' ← reduce(E, (G, C), Z)
25:     choose i ∈ G' with 0 < Σ_{j∈C} e_ij < |C|
26:     if such an i ∈ G' exists then
27:         C_U ← {j | j ∈ C ∧ e_ij = 1}
28:     else
29:         C_U = C
30:     end if
31:     C_V ← C \ C_U
32:     G_U ← ∅, G_V ← ∅, G_W ← ∅
33:     for each i ∈ G' do
34:         C* ← {j | j ∈ C ∧ e_ij = 1}
35:         if C* ⊆ C_U then
36:             G_U ← G_U ∪ {i}
37:         else if C* ⊆ C_V then
38:             G_V ← G_V ∪ {i}
39:         else
40:             G_W ← G_W ∪ {i}
41:         end if
42:     end for
43:     return (G_U, G_V, G_W, C_U, C_V)
44: end procedure

45: procedure reduce(E, (G, C), Z)
46:     G' ← ∅
47:     for each i ∈ G do
48:         C* ← {j | j ∈ C ∧ e_ij = 1}
49:         if C* ≠ ∅ ∧ ∀C+ ∈ Z : C+ ∩ C* ≠ ∅ then
50:             G' = G' ∪ {i}
51:         end if
52:     end for
53:     return G'
54: end procedure
```

Figure 11.3: Bimax Algorithm

11.5 Testing the Bimax algorithm

11.5.1 Validation Approach

Theoretically the number of inclusion-maximal biclusters for $n \times m$ matrices is $2^{\min(n,m)}$, but practically the numbers seem much smaller. The average number for random matrices with 6000 genes and varying number of columns and densities is shown in the table 11.4. Each number gives the average over 100 matrices. The last row comprises the theoretical upper bounds for the number of inclusion-maximal biclusters.

density	number of samples m				
$D^{6000\times\dots}$	50	150	250	350	450
1 %	530.0	3475.5	7594.2	12405.5	17919.9
2 %	1468.7	11829.2	28938.8	53438.2	86657.3
3 %	2490.1	21693.7	62005.3	132435.8	238598.5
4 %	3933.7	44463.7	155929.8	367228.8	694202
5 %	6554.9	100213.8	390835	956255	1838979.7
	1.13e+15	1.43e+45	1.81e+75	2.29e+105	2.91e+135

Figure 11.4: Actual Number of Bics in the Randomized Matrix with given percentage of 1s

Therefore, as one can see the number of biclusters is really modest compared to the theoretical upper bounds, and comparing the solution is not too expensive practically.

To evaluate the performance of *Bimax* compared to other methods, five prominent biclustering methods have been chosen according to three criteria: (1) to what extent the methods have been used or referenced in the community, (2) whether their algorithmic strategies are similar and (3) whether an implementation was available or could be easily reconstructed based on the original publications. The selected algorithms, which all are based on greedy search strategies, are Cheng and Church's algorithm CC [3]; Samba (Tanay *et al.* [4]); Order Preserving Submatrix Algorithm, OPSM (Ben-Dor *et al.* [5]); Iterative Signature Algorithm, ISA (Ihmels *et al.* [6, 7]); xMotif (Murali and Kasif [8]).

All of the selected methods have been re-implemented according to the specifications in the corresponding papers, except of Samba for which a publicly available software tool, Expander (Sharan et al., 2003), has been used. The OPSM algorithm has been slightly extended to return not only a single bicluster but also the q largest biclusters among those that achieve the optimal score; q has been set to 100. Furthermore, the standard hierarchical clustering method (HCL) in MATLAB has been included in the comparison. HCL uses single linkage

in combination with Euclidean distance. For the reference method, Bimax, the discretization threshold has been set to $(\max(ExpressionValues) + \min(ExpressionValues))/2$. The output filtering procedure adopted here follows a greedy approach: in each step, the largest of the remaining biclusters is chosen that has less than 25% of its cells in common with any previously selected bicluster; the algorithm stops if either 100 biclusters have been selected or none of the remaining ones fulfills the selection criterion.

11.5.2 Building Synthetic Data

The artificial model used to generate synthetic gene expression data is similar to an approach proposed by Ihmels *et al.* [6]. In this setting, biclusters represent *transcription modules*; these modules are defined by (i) a set G of genes regulated by a set of common transcription factors, and (ii) a set C of conditions in which these transcription factors are active. More specifically, we consider

- a set of t transcription factors;
- a binary activation matrix $A^{t \times m}$ where $a_{ij} = 1$ iff transcription factor i is active in condition j;
- a binary regulation matrix $R^{t \times n}$ where $r_{ij} = 1$ iff transcription factor i regulates gene j;

The *Prelić et al* studied two synthetic scenarios: disjoint biclusters with varying noise levels and noiseless overlapping biclusters. In the first scenario, 10 non-overlapping transcription modules, each extending over 10 genes and 5 conditions are created. Each gene is regulated by exactly one transcription factor and in each condition only one transcription factor is active. The corresponding data sets contain 10 disjoint implanted biclusters and have been used to study the effects of noise on the performance of the biclustering methods.

For the second scenario, the regulatory complexity has been systematically varied: here, each gene can be regulated by d transcription factors and in each condition up to d transcription factors can be active. As a consequence, the original 10 biclusters overlap where d is an indicator for the overlap degree; overall, nine different levels have been considered with $d = 0, 1, \ldots, 8$.

For each scenario two types of biclusters were concidered: (i) constant biclusters and (ii) additive biclusters. In the first case, the corresponding gene expression matrix E is defined by setting the expression value $e_{ij} = \max_{1 \leq k \leq t} r_{ki} \cdot a_{kj}$; E is a binary matrix where the cells contained in biclusters are set to 1. In the second case, E is constructed as follows: $e_{ij} = m + (j - 1)$ for (i, j) that imply $\max_{1 \leq k \leq t} r_{ki} \cdot a_{kj} \neq 0$; otherwise e_{ij} is a uniformly randomly chosen integer in the interval $[0, m - 1]$.

In order to assess the performance of the selected biclustering approaches, the following gene match score is used:

Definition Let M_1, M_2 be two sets of biclusters. The *gene match score* of M_1 with respect

to M_2 is given by the function

$$S_G^*(M_1, M_2) = \frac{1}{|M_1|} \sum_{(G_1,C_1)\in M_1} \max_{(G_2,C_2)\in M_2} \frac{|G_1 \cap G_2|}{|G_1 \cup G_2|}$$

which reflects the average of the maximum match scores for all biclusters in M_1 with respect to the biclusters in M_2.

Now, let M_{opt} denote the set of implanted biclusters and M the output of a biclustering method. The average *bicluster relevance* is defined as $S_G^*(M, M_{opt})$ and reflects to what extent the recovereded biclusters represent true biclusters in the gene dimension. In contrast, the average *module recovery*, given by $S_G^*(M_{opt}, M)$, quantifies how well each of the true biclusters is recovered by the biclustering algorithm under consideration.

11.5.3 Artificial Data Results

The first artificial scenario, where all biclusters are non-overlapping, serves as a basis to assess the sensitivity of the methods to noise in the data. Noise is imitated by adding random values drawn from a normal distribution to each cell of the original gene expression matrix. The noise level, i.e. the standard deviation σ, is systematically increased, and for each noise value, 10 different data matrices have been generated from the original gene expression matrix E.

In the absence of noise, ISA, Samba and Bimax are able to identify a high percentage ($> 90\%$) of implanted modules; as expected, the same holds for the hierarchical clustering approach, if the number k of clusters to be generated corresponds to the actual number of implanted modules. In contrast, the scores obtained by CC and xMotif are substantially lower. CC tends to find large groups of genes extending over a few columns only, which owes to the used greedy heuristic. Since xMotif is mainly designed to find biclusters with coherent row values, the underlying bicluster problem formulation is not well suited for the second bicluster type. A similar argument applies to OPSM which seeks clear trends of up- or down-regulation and cannot be expected to perform well in the scenarios with constant biclusters.

The only method that fully recovers all hidden modules in the data matrix is by design the reference method, Bimax. Among the remaining methods, Samba provides the best performance: most of the biclusters found ($> 90\%$) represent hidden modules; however, not all implanted modules are recovered. While OPSM is not significantly affected by the overlap degree (only the non-constant bicluster datasets have been considered as OPSM cannot handle identical expression values), ISA appears to be more sensitive to increased regulatory complexity, especially with the second bicluster type. As to CC, the performance increases with larger overlaps degrees, but the gene match scores are still lower than the ones by Bimax, Samba and ISA. xMotif shows the same behavior on the data matrices with constant biclusters. Comparing the biclustering methods with HCL, one can observe that already a minimal overlap causes a large decrease in the performance of HCL, even if the optimal number of clusters is used.

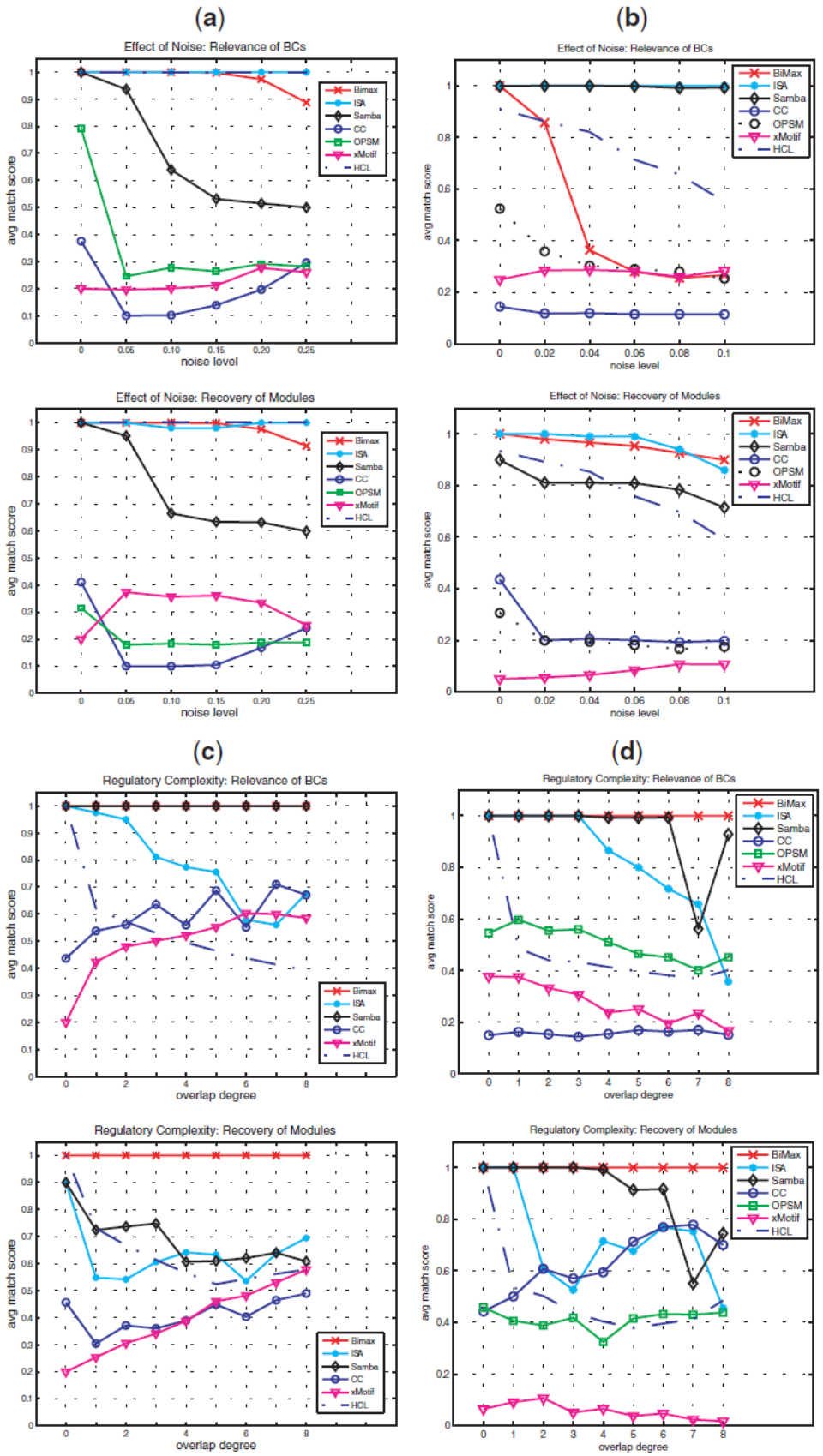

Figure 11.5: Results for the artificial scenarios: non-overlapping modules with increasing noise levels for (**a**) constant and (**b**) additive biclusters, overlapping modules with increasing overlap degree and no noise for (**c**) constant and (**d**) additive biclusters. Y-axis is *average match score*, X-axis of (**a**) and (**b**) shows *noise level*, X-axis of (**c**) and (**d**) shows the *overlap degree*. The first row of charts shows *relevance of biclusters* and the second row shows the *recovery of modules*.

11.5.4 Real Data Results

The biclustering algorithms were also tested on real datasets, normalized using mean centering, and the biological relevance of the obtained biclusters was evaluated with respect to GO annotations, metabolic pathway maps and protein-protein interaction data. Unfortunately, testing data details are not covered in the article, so it is impossible to tell, for example, how big the matrix was.

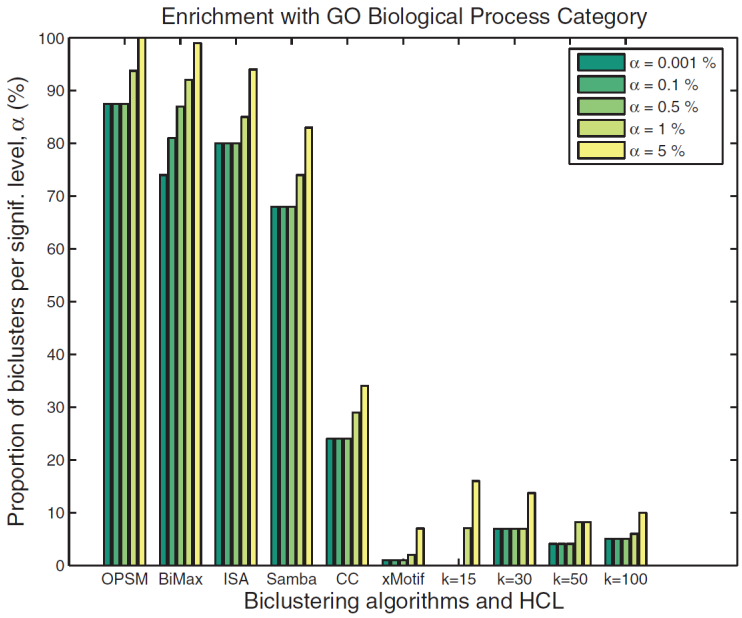

Figure 11.6: Proportion of biclusters significantly enriched by any GO Biological Process category (*S.cerevisiae*) for the six selected biclustering nethods as well as for hierarchical clustering with $k \in \{15, 30, 50, 100\}$. The columns are grouped method-wise, and different bars within a group represent the results obtained for five different significance levels α.

The histogram in Figure 11.6 reflects for each method the proportion of biclusters for which one or several GO categories are overrepresented at different levels of significance. Best results are obtained by OPSM. However, the comparison to other method is not meaningful, since only 12 modules were generated by OPSM, while all other methods produced 100 modules.

Bimax, ISA and Samba also provide a high portion of functionally enriched biclusters, with a slight advantage of Bimax and ISA ($> 90\%$ at a significance level of 5%) over Samba ($> 80\%$ at a significance level of 5%). In contrast, the scores for CC are considerably lower (30%) due its greedy nature. Except for xMotif, though, all biclustering methods achieve higher scores than HCL.

Under the assumption that the structure of a metabolic pathway map, respectively, a protein-protein interaction network is somehow reflected in the gene expression data, the degree of connections number of the genes associated with a bicluster can be used to assess its biological relevance. In particular, one may expect that both the number of disconnected gene pairs and the average shortest distance between connected gene pairs tend to be smaller

for the biclusters found than for random gene groups. Although for most methods, the biclusters are better connected than random gene groups, the differences to the random case are not as striking as for the average gene pair distance.

Method	Proportion of disconnected gene pairs				Average shortest distance in the graph			
	Smaller		Greater		Smaller		Greater	
	MPM	PPI	MPM	PPI	MPM	PPI	MPM	PPI
Bimax	58.9	14.0	19.5	64.0	85.3	58.0	3.4	16.0
CC	70.0	52.0	15.0	26.0	70.0	42.0	15.0	34.0
OPSM	42.8	18.8	28.6	50.0	92.9	56.3	0.0	43.8
Samba	41.6	0.0	37.5	100.0	75.6	25.6	13.1	46.2
xMotif	49.0	2.0	17.0	92.0	84.0	4.0	3.0	72.0
ISA	25.0	58.0	25.0	22.0	50.0	70.0	25.0	22.0

Figure 11.7: Biological relevance of biclusters with respect to a metabolic pathway map (**MPM**) for *A. thaliana* and a protein-protein interaction network (**PPI**) for *S.cerevisiae*.

The results for the corresponding comparison for the protein-protein interaction, though, are ambiguous, Figure 11.7. In this table for each bicluster, a Z-test is carried out to check whether its score is significantly smaller or greater than the expected value for random gene groups; the table gives for each method the proportion of biclusters with statistically significant scores (significance level $\alpha = 10^{-3}$). The results for HCL are omitted as all scores equal 0%. As to the degree of disconnectedness, there is no clear tendency in the data which can be attributed to the fact that not all possible protein pairs have been tested for interaction. Focusing on connected gene pairs only, ISA and Bimax seem to mostly generate gene groups that have a low average distance within the protein network in comparison to random gene sets; for xMotif, the numbers suggest the opposite. *Prelić et al* conclude that overall the differences between the biclustering methods demonstrate that special care is necessary when integrating gene expression and protein interaction data: not only the incompleteness of the data needs to be taken into consideration, but also the confidence in the measurements has to be accounted for.

We note, however, that the method by which the randomized networks were generated is flawed, since it does not take into account the high variability in node degrees. Reevaluation should be done after degree-preserving randomization.

11.6 Conclusions

- Meaningful biological results with ISA, SAMBA, OPSM.

  On the real datasets, ISA, Samba and OPSM provide similarly good results: a large portion of the resulting biclusters is functionally enriched and indicates a strong correspondence with known pathways.

- Similar performance. SAMBA slightly more robust to overlaps, more sensitive to noise. OPSM (still) oriented to find a single bic.

In the context of the synthetic scenarios, Samba is slightly more robust regarding increased regulatory complexity, but also more sensitive regarding noise than ISA. While Samba and ISA can be used to find multiple biclusters with both constant and coherently increasing values, OPSM is mainly tailored to identify a single bicluster of the second type. The scores for CC and xMotif are significantly lower than that for the other biclustering methods under consideration.

- BIMAX gets similar results. How come naïve binarization works?

  The Bimax algorithm achieves similar scores as the best performing biclustering techniques. But from the data we do not see obvious evidence that this algorithm is much better than other five prominent methods. An advantage of Bimax is that it is capable of generating all optimal biclusters, given the underlying binary data model.

- We are making progress, but still far from fully understanding the problem and formulating it well.

  The reference method can be useful as a preprocessing step by which potentially relevant biclusters may be identified; later, the chosen biclusters can be used, e.g. as an input for more accurate biclustering methods in order to speed up the processing time and to increase the bicluster quality.

Bibliography

[1] Alexe,G., Alexe,S., Crama,Y., Foldes,S., Hammer,P.L. and Simeone,B.: *Consensus algorithms for the generation of all maximal bicliques.* Technical Report TF-DIMACS-2002-52. 2002.

[2] Prelić A., Bleuler S., Zimmermann P., Wille A., Buhlmann P., Gruissem W, Hennig L, Thiele L. and Zitzler E.: *A systematic comparison and evaluation of biclustering methods for gene expression data.* Bioinformatics Vol. 22 no. 9 2006.

[3] Cheng,Y. and Church,G. *Biclustering of expression data.* Proc. Int. Conf. Intell. Syst. Mol. Biol. pp. 93-103. 2000

[4] Tanay,A. et al. *Discovering statistically significant biclusters in gene expression data.* Bioinformatics, 18 (Suppl. 1), S136-S144. 2002

[5] Ben-Dor,A., Chor,B., Karp,R. and Yakhini,Z. *Discovering local structure in gene expression data: the order-preserving sub-matrix problem.* In Proceedings of the 6th Annual International Conference on Computational Biology, ACM Press, New York, NY, USA, pp. 49-57. 2002

[6] Ihmels,J. et al. *Revealing modular organization in the yeast transcriptional network.* Nat. Genet., 31, 370-377. 2002

[7] Ihmels,J. et al. *Defining transcription modules using large-scale gene expression data.* Bioinformatics, 20, 1993-2003. 2004

[8] Murali,T.M. and Kasif,S. *Extracting conserved gene expression motifs from gene expression data.* Pac. Symp. Biocomput., 8, 77-88. 2003