SUPPLEMENTARY MATERIAL

A Systematic Comparison and Evaluation of Biclustering Methods for Gene Expression Data

(Bioinformatics, 2006)

Amela Prelić,[a] Stefan Bleuler [a], Philip Zimmermann [b], Anja Wille [c,d], Peter Bühlmann [d], Wilhelm Gruissem [b], Lars Hennig [b], Lothar Thiele [a], and Eckart Zitzler [a]
Reverse Engineering Group: [a]Computer Engineering and Networks Laboratory, [b]Institute for Plant Sciences and Functional Genomics Center Zurich, [c]Colab, [d]Seminar for Statistics, ETH Zurich, 8092 Zurich, Switzerland.

Considered Biclustering Algorithms

Five prominent biclustering methods have been chosen for this comparative study according to three criteria: (i) to what extent the methods have been used or referenced in the community, (ii) whether their algorithmic strategies are similar and therefore better comparable, and (iii) whether an implementation was available or could be easily reconstructed based on the original publications. The selected algorithms are briefly described in the following; they are all based on greedy search strategies.

Cheng and Church's Algorithm (CC) Cheng and Church, 2000 define a bicluster to be a submatrix for which the *mean squared residue score* is below a user-defined threshold δ, where 0 represents the minimum possible value. In order to identify the largest δ-bicluster in the data, they propose a two-phase strategy: first, rows and columns are removed from the orginal expression matrix until the above constraint is fulfilled; later, previously deleted rows and columns are added to the resulting submatrix as along as the bicluster score does not exceed δ. This procedure is iterated several times where previously found biclusters are masked with random values. Recently, Yang *et al.*, 2003 proposed an improved version of this algorithm which avoids the problem of random interference caused by masked biclusters.

Samba Tanay *et al.*, 2002 presented a graph-theoretic approach to biclustering in combination with a statistical data model. In this framework, the expression matrix is modelled as a bipartite graph, a bicluster is defined as a subgraph, and a likelihood score is used in order to assess the significance of observed subgraphs. A corresponding heuristic algorithm called Samba aims at finding highly significant and distinct biclusters. In a recent study (Tanay *et al.*, 2004), this approach has been extended to integrate multiple types of experimental data.

Order Preserving Submatrix Algorithm (OPSM) In (Ben-Dor *et al.*, 2002), a bicluster is defined as a submatrix that preserves the order of the selected columns for all of the selected rows. In other words, the expression values of the genes within a bicluster induce an identical linear ordering across the selected samples. Based on a stochastical model, the authors developed a deterministic algorithm to find large and statistically significant biclusters. This concept has been taken up in a recent study by Liu and Wang, 2003.

Iterative Signature Algorithm (ISA) The authors of (Ihmels *et al.*, 2002, 2004) consider a bicluster to be a transcription module, i.e., a set of co-regulated genes together with the associated set of regulating conditions. Starting with an initial set of genes, all samples are scored with respect to this gene set and those samples are chosen for which the score exceeds a predefined threshold. In the same way, all genes are scored regarding the selected samples and a new set of genes is selected based on another user-defined threshold. The entire procedure is repeated until the set of genes and the set of samples converge, i.e., do not change anymore. Multiple biclusters can be identified by running the iterative signature algorithm on several initial gene sets.

xMotif In the framework proposed by Murali and Kasif, 2003, biclusters are sought for which the included genes are nearly constantly expressed—across the selection of samples. In a first step, the input matrix is preprocessed by assigning each gene a set of statistically significant *states*. These states define the set of valid biclusters: a bicluster is a submatrix where each gene is exactly in the same state for all selected samples. To identify the largest valid biclusters, an iterative search method is proposed that is run on different random seeds, similarly to ISA.

Bimax (Reference Method)

Algorithm The following algorithm realizes the divide-and-conquer strategy as illustrated in Fig. 1. Note that special operations are required for processing the V submatrices. As mentioned in the discussion of the reference model, the algorithm needs to guarantee that only optimal, i.e., inclusion-maximal biclusters are generated. The problem 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., $\forall C^+ \in Z : C \cap C^+ \neq \emptyset$.

```
 1: procedure $Bimax(E)$
 2:     $Z \leftarrow \emptyset$
 3:     $M \leftarrow conquer(E, (\{1, \ldots, n\}, \{1, \ldots, m\}), Z)$
 4:     return $M$
 5: end procedure

 6: procedure $conquer(E, (G, C), Z)$
 7:     if $\forall i \in G, j \in C : e_{ij} = 1$ 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:     $M_U \leftarrow \emptyset, M_V \leftarrow \emptyset$
12:     if $G_U \neq \emptyset$ then
13:         $M_U \leftarrow conquer(E, (G_U \cup G_W, C_U), Z)$
14:     end if
15:     if $G_V \neq \emptyset \wedge G_W = \emptyset$ then
16:         $M_V \leftarrow conquer(E, (G_V, C_V), Z)$
17:     else if $G_W \neq \emptyset$ then
18:         $Z' \leftarrow Z \cup \{C_V\}$
19:         $M_V \leftarrow conquer(E, (G_W \cup G_V, C_U \cup C_V), Z')$
20:     end if
21:     return $M_U \dot\cup M_V$
22: end procedure

23: procedure $divide(E, (G, C), Z)$
24:     $G' \leftarrow reduce(E, (G, C), Z)$
25:     choose $i \in G'$ with $0 < \sum_{j \in C} e_{ij} < |C|$
26:     if such an $i \in G'$ exists then
27:         $C_U \leftarrow \{j \mid j \in C \wedge e_{ij} = 1\}$
28:     else
```

```
29:            $C_U = C$
30:        end if
31:        $C_V \leftarrow C \setminus C_U$
32:        $G_U \leftarrow \emptyset,\ G_V \leftarrow \emptyset,\ G_W \leftarrow \emptyset$
33:        for each $i \in G'$ do
34:            $C^\star \leftarrow \{j \mid j \in C \wedge e_{ij} = 1\}$
35:            if $C^\star \subseteq C_U$ then
36:                $G_U \leftarrow G_U \cup \{i\}$
37:            else if $C^\star \subseteq C_V$ then
38:                $G_V \leftarrow G_V \cup \{i\}$
39:            else
40:                $G_W \leftarrow G_W \cup \{i\}$
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:        $G' \leftarrow \emptyset$
47:        for each $i \in G$ do
48:            $C^\star \leftarrow \{j \mid j \in C \wedge e_{ij} = 1\}$
49:            if $C^\star \neq \emptyset \wedge \forall C^+ \in Z : C^+ \cap C^\star \neq \emptyset$ then
50:                $G' = G' \cup \{i\}$
51:            end if
52:        end for
53:        return $G'$
54:    end procedure
```

Running-Time Analysis

THEOREM 1. *The running-time complexity of the Bimax algorithm is $O(nm\beta \min\{n, m\})$, where β is the number of all inclusion-maximal biclusters in $E^{n\times m}$.*

Proof of Theorem 1. To derive an upper bound for the running-time complexity, we will first calculate the number of steps required to execute the procedure $conquer$ once, disregarding the recursive procedure calls. Afterwards, the maximum number of invokations of $conquer$ will be determined, which then leads to the overall running-time complexity.

As to 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 amounts to $O(nm)$.

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 invokation of $conquer$ takes $O(nm)$ time.

The question now is how many times $conquer$ is executed. Taking into account that every invokation 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 invokation, has at most β leafs. Each inner node of the recursion tree has an outdegree of 1 or 2, 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 invokation 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 leafs 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 invokations of $conquer$ is of order $O(\beta)$.

By combining the two main results, (i) one $conquer$ call needs $O(nm \min n, m)$ steps and (ii) there are at maximum $O(\beta)$ invokations of $conquer$, we obtain the upper bound for the running-time of the Bimax algorithm. □

Limitations Theoretically, the number of inclusion-maximal biclusters can be exponential in n and m and therefore generating the entire set can become infeasible. For real data, though, the actual number lies within reasonable bounds as the number of 1-cells is small. For instance, for a 6000×50-matrix with a density of 5%, around 6500 biclusters are returned by the algorithm, while the theoretical bound is $1.13e^{+15}$, see Table 2. The running time for such a matrix is below 1 second on a 3 GHz Intel Xeon machine, and about 10 minutes for corresponding 6000×450-matrices.

Furthermore, a secondary filtering procedure, similarly to other biclustering approaches such as (Tanay *et al*., 2002; Ihmels *et al*., 2004), can be applied to reduce the number of biclusters to the desired size; this issue will be discussed in the next section. Another possibility is to constrain the size of the biclusters during the search process. The advantage of the Bimax algorithm over the incremental procedure is that such size constraints can be naturally integrated—thereby, further speed-ups are achievable.

Incremental Procedure

Algorithm The incremental procedure, see below, is based on work by Alexe *et al*., 2002, 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. For the special class of bipartite graphs we are dealing with, it is important

Table 2. Average number of inclusion-maximal biclusters for random matrices with 6000 genes and varying number of columns and densities, i.e., proportion of 1 cells to 0-cells. 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				
$D^{6000\times\cdots}$	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

to notice that several steps of the above method are redundant: it suffices to iterate through only one partition of the graph nodes—in matrix terminology this means we will have to iterate either through the set of rows or columns, but not both. Moreover, extending new biclusters can be avoided with a guarantee that no bicluster will be missed this way.

```
1:  procedure $IncrementalAlgorithm(E)$
2:      $M \leftarrow \emptyset$
3:      for $i \leftarrow 1$ to $n$ do
4:          $C^\star \leftarrow \{j \mid e_{ij} = 1 \wedge 1 \le j \le m\}$
5:          for each $(G, C) \in M$ do
6:              $C' \leftarrow C \cap C^\star$
7:              if $\exists\,(G'', C'') \in M$ with $C'' = C'$ then
8:                  $M \leftarrow M \setminus \{(G'', C'')\} \cup \{(G'' \cup \{i\}, C'')\}$
9:              else
10:                 $M \leftarrow M \cup \{(G'' \cup \{i\}, C')\}$
11:             end if
12:         end for
13:         if $\nexists\,(G'', C'') \in M$ with $C'' = C^\star$ then
14:             $M \leftarrow M \cup \{(\{i\}, C^\star)\}$
15:         end if
16:     end for
17:     return $M$
18: end procedure
```

Running-Time Analysis

THEOREM 2. *The running-time complexity of the Incremental Algorithm is $\Theta(nm\beta \log \beta)$, where β is the number of all inclusion-maximal biclusters in $E^{n\times m}$.*

LEMMA 1. *Given the binary matrix $E^{n\times m}$, a duplicate row or column in E does not contribute to the total number of all inclusion-maximal biclusters in E.*

LEMMA 2. *Given the binary matrix $E^{n\times m}$, the upper bound on the number of all inclusion-maximal biclusters in E is $(2^{min(n,m)} - 1)$.*

Proof of Theorem 2. The incremental algorithm proceeds in stages: at stage i, a row/gene i of the matrix is considered and the steps within the outer **for** instruction are performed. The set of instructions within steps 5 to 12 amounts to: *i)* computing an intersection of the sets of samples (having value 1) corresponding to gene i and a currently considered bicluster, which takes $\Theta(m)$, and *ii)* the search through the list M, followed by a set equality comparison operations, which costs further $\Theta(m \log_2 \beta)$, assuming that binary search through the list M is made. This inner cycle (steps 5 - 12) is performed β times, and the outer one n times, where n is the number of rows of the matrix E. We then obtain $\Theta(n\,\beta\,(m + m\,log_2\beta)) = \Theta(n\,m\,\beta\,\log_2 \beta)$. Note that the worst-case running time complexity amounts to $O(n\,m^2\beta)$ in the case that $m \le n$, because the upper bound on β is then exponential in m, hence, $\log_2 \beta \le m$.

In the algorithm proposed by Alexe *et al.*, 2002, the main differences to our incremental approach is an additional step that is performed within the steps 7 to 11 of globally extending newly created biclusters to their maximality, and an additional "absorption check" operation is made which costs $\Theta(n\,m\,\log_2 \beta)$. Hence, the difference in the running-time complexities. $\square$

Validation Using Synthetic Data

Data Sets The artificial model used to generate synthetic gene expression data is similar to an approach proposed by Ihmels *et al.*, 2002. 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;

on the basis of which two scenarios have been created.

In the first scenario, 10 non-overlapping transcription modules, each extending over 10 genes and 5 conditions, emerge. 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 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 $d = 0, 1, \ldots, 8$.In detail, activation and regulation matrices were created as follows:

$$r_{ij} = \begin{cases} 1 & \text{if } (i-1)n'/t + 1 \le j \le in'/t + d \\ 0 & \text{else} \end{cases}$$

for $1 \le i \le t$, $1 \le j \le n' + d$, and

$$a_{ij} = \begin{cases} 1 & \text{if } (i-1)m'/t + 1 \le j \le im'/t + d \\ 0 & \text{else} \end{cases}$$

for $1 \leq i \leq t$, $1 \leq j \leq m' + d$. For scenario 1, the parameters were $n' = 100, m' = 50, t = 10$, and $d = 0$. For scenario 2, the parameter setting was $n' = 100, m' = 100, t = 10$ in combination with different overlap degrees $d \in \{0, \ldots, 8\}$.

Moreover, we have investigated for each scenario two types of biclusters: (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} of gene i at condition j to $e_{ij} = \max_{1 \leq k \leq t} r_{ki} \cdot a_{kj}$; E is a binary matrix where the cells contained in biclusters are set to 1. In the second case, E is constructed as follows

$$e_{ij} = \begin{cases} m + (j-1) & \text{if } \max_{1 \leq k \leq t} r_{ki} \cdot a_{kj} \neq 0 \\ U[0, m-1] & \text{else} \end{cases}$$

where $U[l, u]$ is a uniformly randomly chosen integer in the interval $[l, u]$. In the resulting matrix, all cells belonging to an implanted bicluster have a value greater than or equal to m, while the background contains random numbers in the range of 0 to $m - 1$. Within each bicluster, the values increase column-wise by one.

Match Scores In order to assess the performance of the selected biclustering approaches, we will use a score that describes the degree of similarity between the computed biclusters and the transcription modules implanted in the synthetic data sets.

The following score is designed to compare two biclusters.

DEFINITION 3. *Let $G_1, G_2 \subseteq \{1, \ldots, n\}$ be two sets of genes. The* match score *of G_1 and G_2 is given by the function*

$$S_G(G_1, G_2) = \frac{|G_1 \cap G_2|}{|G_1 \cup G_2|}$$

which characterizes the correspondence between the two gene sets.

This match score, which resembles the *Jaccard coefficient*, cf. (Halkidi *et al.*, 2001), is symmetric, i.e., $S_G(G_1, G_2) = S_G(G_2, G_1)$, and its value ranges from 0 (the two sets are disjoint) to 1 (the two sets are identical). A match score S_C for sample sets can defined by analogy.

On this basis, a score for comparing two sets of biclusters can be introduced as follows.

DEFINITION 4. *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*

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

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

The gene match score is not symmetric and usually yields different values when M_1 and M_2 are exchanged; accordingly, both $S^*_G(M_1, M_2)$ and $S^*_G(M_2, M_1)$ need to be considered. Although, this comparative study takes only the gene dimension into account, an overall match score can be defined as $S^*(M_1, M_2) = \sqrt{S^*_G(M_1, M_2) \cdot S^*_C(M_1, M_2)}$ where S^*_C is the corresponding *condition match score*.

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 $S^*_G(M, M_{\text{opt}})$ and reflects to what extent the generated biclusters represent true biclusters in the gene dimension. In contrast, the *average module recovery*, given by $S^*_G(M_{\text{opt}}, M)$, quantifies how well each of the true biclusters is recovered by the biclustering algorithm under consideration. Both scores take the maximum value of 1, if $M_{\text{opt}} = M$.

Additional Tables and Figures

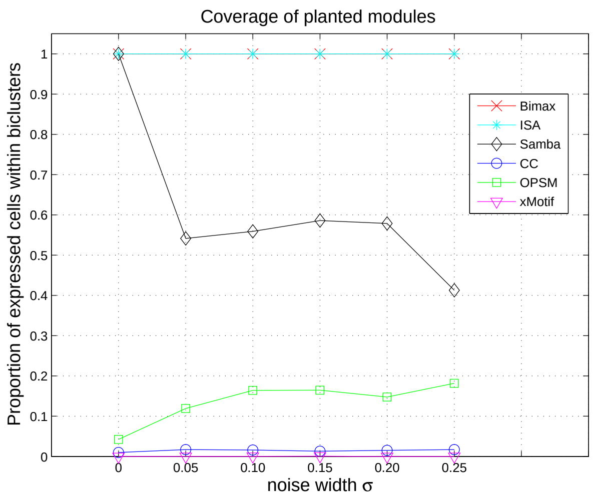


Fig. 6: This figure shows for the first artificial scenario with constant biclusters what proportion of computed biclusters contain overexpressed cells. As argued in the article, the two methods CC and xMotif tend to produce large biclusters covering the background area of the input matrix, i.e., the cells containing 0).

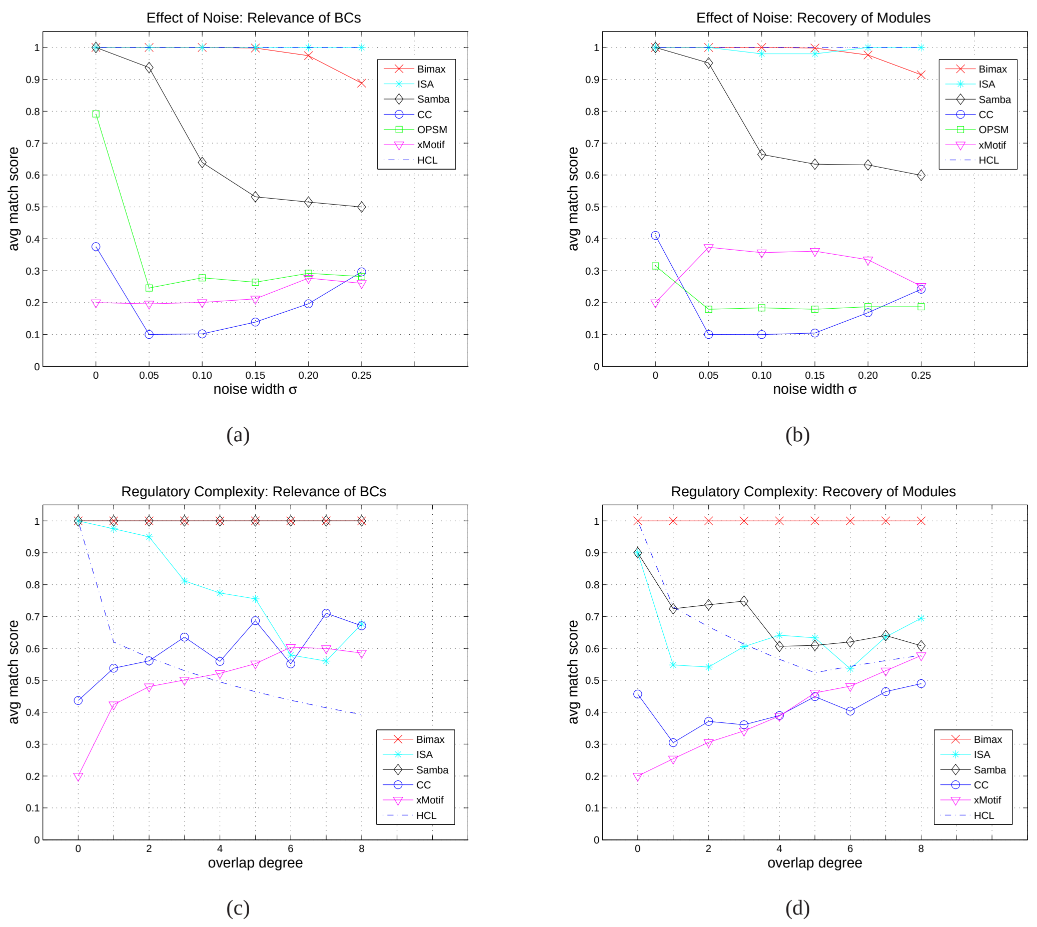
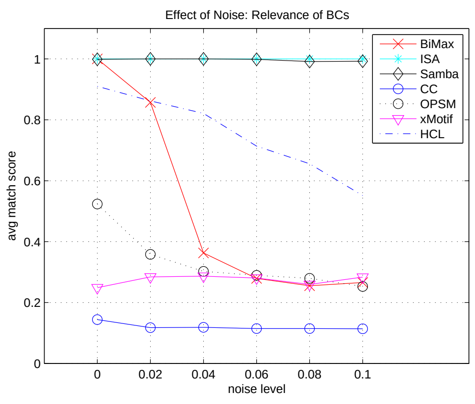


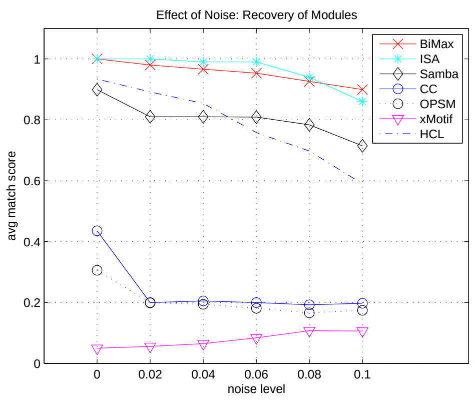
Fig. 4: Results for the artificial scenarios where the implanted biclusters are characterized by constant expression values: (a), (b) non-overlapping modules with increasing noise levels; (c), (d) overlapping modules with increasing overlap degree and no noise. Note that OPSM is excluded in the lower two figures as explained in the results section.

Table 3. Parameter settings used for different biclustering methods. Default settings (i.e., the parameter values recommended/used by the authors of original papers) were occasionally changed in order to force the methods to output at least a single bicluster. The changed values are reported in the third column (an empty third column cell indicates the default values have always been used). For the meaning of different parameters, please refer to the original papers.

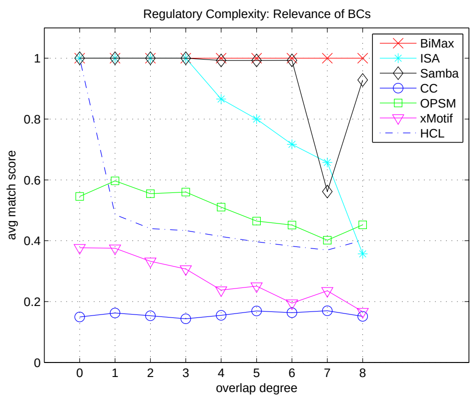
Algorithm	Default Parameter Settings	Changed values
Samba	$D = 40,\ N_1 = 4,\ N_2 = 6,\ k = 20,\ L = 30$	
ISA	$t_g = 1.8 - 4.0$ (step 0.1), $t_c = 2.0$, nr. seeds $= 20000$	$t_g = 2.0$, nr. seeds $=\ 500$
CC	$\alpha = 1.2$, δ lower end of the range of expression values	$\delta \leq 0.5$, for biclusters with increasing values $\delta = 0.1$
OPSM	$l = 100$	
xMotifs	$n_s = 10, n_d = 1000, s_d = 7 - 10, \alpha$ not given, P value 10^{-10}, max_length not given	$s_d = 7, \alpha = 0.1, max\_length = 0.7m$ for increasing noise matrix p-value 10^{-7}



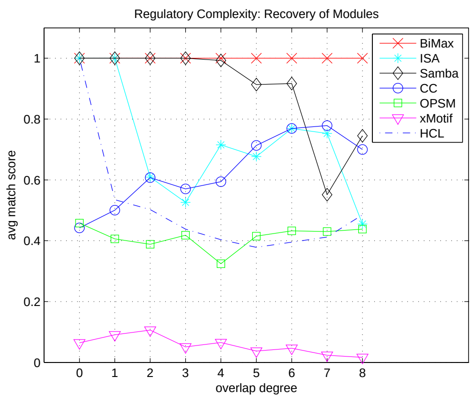
(a)



(b)

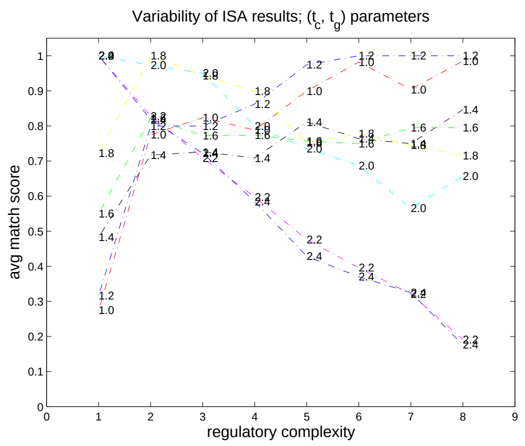


(c)

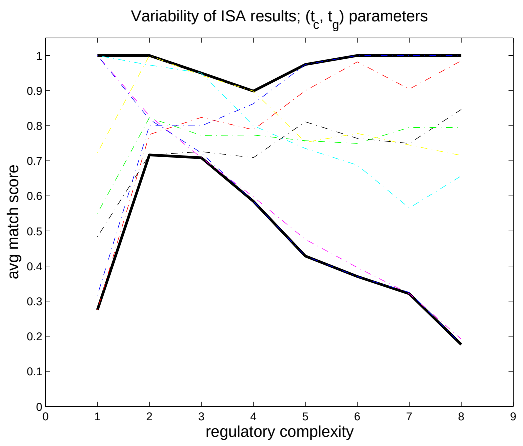


(d)

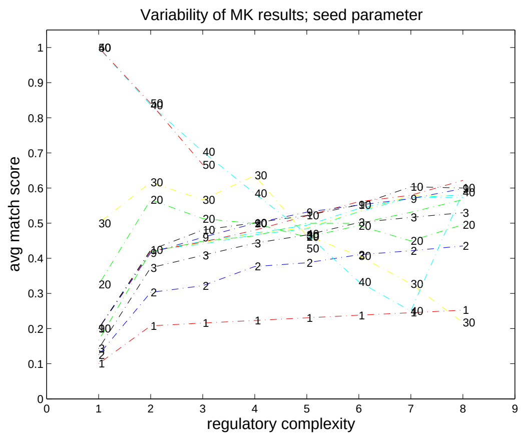
Fig. 5: Results for the artificial scenarios where the biclusters follow an additive model: (a), (b) non-overlapping modules with increasing noise levels; (c), (d) overlapping modules with increasing overlap degree and no noise.



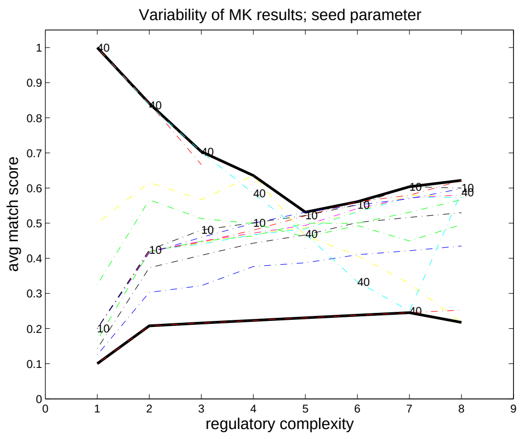
(a)



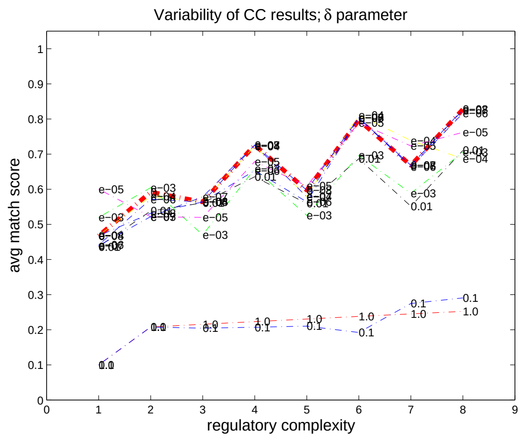
(b)



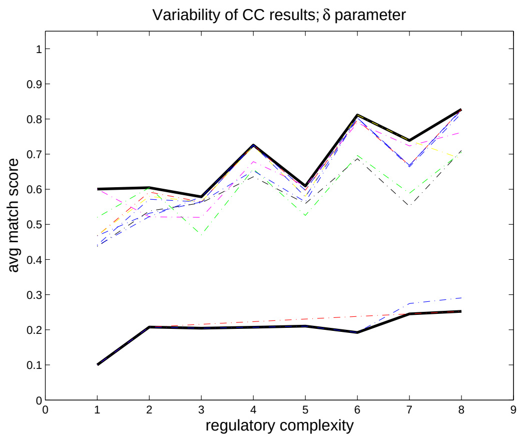
(c)



(d)



(e)



(f)

Fig. 7: Variability of the average bicluster relevance score depending on the parameter settings (constant biclusters). The plotted values represent averages over the biclusters obtained by ISA, xMotif and CC. (a), (b): For ISA, we varied the (t_g, t_c) parameters, in all cases, $t_g = t_c$, with $1.0 \leq t_g \leq 2.4$; the value recommended by authors is $(2.0, 2.0)$. (c), (d): As to xMotif, the size of the random seeds was changed in the range $1 - 50$; values recommended by the authors are in the range $7 - 10$. (e), (f): For CC, the homogeneity threshold, δ, has been systematically varied; the red bold line in (e) shows the results obtained for $\delta = 0$, i.e., when only perfect biclusters are sought.