**JOURNAL OF COMPUTATIONAL BIOLOGY**
**Volume 15, Number 10, 2008**
**© Mary Ann Liebert, Inc.**
**Pp. 1329–1345**
**DOI: 10.1089/cmb.2008.0066**

# Biclustering Sparse Binary Genomic Data

MIRANDA VAN UITERT,[1] WOUTER MEULEMAN,[1,2] and LODEWYK WESSELS[1,2]

## ABSTRACT

**Genomic datasets often consist of large, binary, sparse data matrices. In such a dataset, one is often interested in finding contiguous blocks that (mostly) contain ones. This is a biclustering problem, and while many algorithms have been proposed to deal with gene expression data, only two algorithms have been proposed that specifically deal with binary matrices. None of the gene expression biclustering algorithms can handle the large number of zeros in sparse binary matrices. The two proposed binary algorithms failed to produce meaningful results. In this article, we present a new algorithm that *is* able to extract biclusters from sparse, binary datasets. A powerful feature is that biclusters with different numbers of rows and columns can be detected, varying from many rows to few columns and few rows to many columns. It allows the user to guide the search towards biclusters of specific dimensions. When applying our algorithm to an input matrix derived from TRANSFAC, we find transcription factors with distinctly dissimilar binding motifs, but a clear set of common targets that are significantly enriched for GO categories.**

**Key words:** biclustering, binary data, transcription factor binding.

## 1. INTRODUCTION

In this paper, we propose *BicBin* (Biclustering Binary data), a new algorithm that is able to find a set of biclusters in sparse binary datasets. The biological motivation for developing such an algorithm comes from problems in several application domains, including transcription factor binding (the main focus of this article), insertional mutagenesis (Lund et al., 2002; Mikkers et al., 2002), and gene expression.

Many algorithms have been proposed in recent years for biclustering gene expression data, such as Plaid (Turner et al., 2005), EXPANDER (Shamir et al., 2005), a method based on Gibbs sampling (Dhollander et al., 2007; Sheng et al., 2003), and several other methods that are compared in a survey (Madeira and Oliveira, 2004). A number of the algorithms reviewed in Madeira and Oliveira (2004) were further evaluated in Prelić et al. (2006). Unfortunately, these algorithms do not perform well on sparse, binary

[1]Bioinformatics and Statistics, Division of Molecular Biology, The Netherlands Cancer Institute, Amsterdam, The Netherlands.
[2]Information and Communication Theory Group, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands.

datasets. The main reason being that they all employ a distance measure designed for continuous vectors, and these measures are ill-suited for sparse binary vectors.

### *1.1. Biclustering binary data*

To the best of our knowledge only two algorithms exist that are especially designed for binary data: the algorithm described in Koyutürk et al. (2004), which will be referred to as *Cmnk* in the remainder of this article, and *Bimax* which is used as a reference in Prelić et al. (2006). Like the algorithms designed for gene expression data, both *Cmnk* and *Bimax* failed on sparse binary datasets. In artificial experiments where a single bicluster was embedded in a noisy background, *Cmnk* was not able to extract the bicluster. *Bimax* is optimal in the sense that it returns all possible biclusters in a dataset. However, if one uses it on a typical genomic dataset (with thousands of columns) it results in a list of hundreds of million *unsorted* biclusters. Moreover, *Bimax* returns biclusters with ones only, whereas a bicluster with a small proportion of zeros could be interesting as well, since it reflects some tolerance to, for example, measurement noise. Hence, *Cmnk* and *Bimax* turned out to be not useful.

The algorithm that we present in this article (*BicBin*) finds biclusters in a noisy background, produces a manageable ranked list of biclusters, and allows some predefined proportion of zeros in the biclusters. In more detail, *BicBin* evaluates each submatrix with a cost function that takes the bicluster size into account. It has a powerful feature, the so-called $\alpha$-$\beta$ space, that allows the user to bias the search towards biclusters of specific dimensions. More specifically, $\alpha$ and $\beta$ can be set such that the dimensions of the biclusters vary from many rows and few columns to few rows and many columns.

### *1.2. Transcription factor binding data: novel regulatory modules*

Over the last years, a substantial amount of literature has appeared on locating transcription factor binding sites and finding their binding motifs (Elnitski et al., 2006; Wasserman and Sandelin, 2004). Lately the focus has changed to finding transcription factor modules responsible for gene regulation. *BicBin* can therefore be employed to decipher transcriptional regulation by searching for transcription factor modules in a purely data-driven way: grouping genes on the basis of their transcription factor motif binding pattern and at the same time grouping transcription factors based on the target genes they putatively bind, might reveal novel regulatory networks. To accomplish this, we applied *BicBin* to a dataset with transcription factor motifs and the genes whose upstream regions they putatively bind. The transcription factor motifs can be either the outcome of a biological assay or taken from an existing database such as JASPAR (Sandelin et al., 2004) or TRANSFAC (Wingender et al., 2000). In this article we chose to use a preprocessed set of motifs from the latter database, and developed a statistical approach to find the genes to which they putatively bind. The resulting matrix with along the rows the transcription factor motifs and along the columns the genes is clearly binary. A one represents a transcription factor motif that putatively binds the upstream region of the gene. Hence, even though transcription factor binding data is not inherently binary, it is natural to represent it in a binary way, since in the end a transcription factor regulates a gene or not. We refer to Harbison et al. (2004), where a similar binary matrix is obtained for yeast. The dataset we obtained is sparse since only 20% of the elements in the matrix are non-zero. As expected, we find biclusters with transcription factors that have very similar motifs (the similarity is based on the Tomtom algorithm as described in Gupta et al. [2007]). This serves as a partial validation of the approach, since similar transcription factor motifs are clustered together. More interestingly, we also obtain biclusters with transcription factors that have very dissimilar motifs and of which the genes are enriched for GO categories. This could point to potentially novel putative regulatory interactions.

## 2. THE *BicBin* ALGORITHM

In this section, we introduce and describe *BicBin*. It consists of three main ingredients: the score function to evaluate a submatrix, the search algorithm to restrict the search space of all possible submatrices and an algorithm to extract all biclusters in an ordered way from a dataset.

### *2.1. Score function*

We will first derive the score function that we use to evaluate a submatrix. Let $G$ be a binary matrix with $M$ rows, $N$ columns and a proportion of ones, $p$. Similar to Koyutürk et al. (2004), we assume each element in this matrix to be the outcome of a Bernoulli trial with succes probability $p$. Defining $X_{m,n}$ to be a random variable denoting the number of ones in a submatrix of size $m \times n$, $X_{m,n}$ follows a Binomial distribution with parameters $mn$ and $p$. Because we are interested in submatrices that are dense with ones, we are actually interested in submatrices with small $\mathbb{P}(X_{m,n} \geq k)$. Due to the sparseness of the dataset, i.e., small $p$, these probabilities become very small and impossible to work with for the submatrices we wish to evaluate, quickly reaching the limits of machine precision. Therefore, instead of using the actual probability, we use an exponentially decaying upper bound for the probability, the Chernoff bound. Such an upper bound is well known to be very accurate in the regime of small probabilities. We choose the multiplicative version of the Chernoff bound, which, when applied to the Binomial distribution (see Proposition 2.4 in Angluin and Valiant [1979], and Section 3 in Erdös and Spencer [1974]), is defined as

$$\mathbb{P}(X_{m,n} > k) \leq \begin{cases} e^{-\frac{(k-mnp)^2}{3mnp}}, & k \in [mnp, 2mnp]; \\ e^{-\frac{(k-mnp)^2}{k+mnp}}, & k > 2mnp. \end{cases}$$

Note that this bound is only valid for $k \geq mnp$, i.e., for $k$ larger than the expected number of ones in a submatrix of size $m \times n$. Instead of finding submatrices with a small probability of occurring, we can search for submatrices with a large value for the exponent

$$\tilde{C}(m,n,k) = \begin{cases} \dfrac{(k-mnp)^2}{3mnp}, & k \in [mnp, 2mnp]; \\ \dfrac{(k-mnp)^2}{k+mnp}, & k > 2mnp. \end{cases}$$

Using the binomial distribution this way, the row and column sizes of the submatrix are not taken into account separately, only the product $mn$ is required for $\tilde{C}(m,n,k)$. To distinguish between the number of rows and the number of columns in the submatrix, we need to normalize $\tilde{C}(m,n,k)$ with some functions of $m$ and $n$. At the same time, minimizing the binomial probability, or equivalently maximizing the exponent, $\tilde{C}(m,n,k)$, introduces a natural bias for larger biclusters. We will illustrate this phenomenon with an example. Consider a submatrix of size $10 \times 10$ with $k = 100$ and $p = 0.016$.

This results in a score of $\tilde{C}(10,10,100) = 95.30$. Now consider a submatrix of size $10 \times 11$, containing $k = 101$ ones. The score for this submatrix is $\tilde{C}(10,11,101) = 95.84$, i.e., $\tilde{C}(10,11,101) > \tilde{C}(10,10,100)$. Intuitively, we would prefer it the other way around: the most dense bicluster should achieve the largest score. In order to take the size of the bicluster into account, we could normalize $\tilde{C}(m,n,k)$ with the size of the bicluster, i.e., with $mn$. However, then a bicluster of $1 \times 1$ with a single one is assigned the same value of $\tilde{C}(m,n,k)$ as a bicluster of size $m \times n$ with $mn$ ones for any $m$ and $n$. This is clearly undesirable, as a bicluster of size $10 \times 10$ containing 99 ones will be scored lower than a $1 \times 1$ bicluster containing a single one, even though the former is clearly the preferred option. Therefore, given the fact that we have to strike a balance between a too stringent normalization ($m^1n^1$) and no normalization ($m^0n^0$, which is simply $\tilde{C}(m,n,k)$) we propose to normalize $\tilde{C}(m,n,k)$ with the factor $m^\alpha n^\beta$, with $\alpha$ and $\beta$ in [0.5, 1]. This results in the following function to score a submatrix of size $m \times n$ with $k$ ones

$$C_{\alpha,\beta} = \frac{\tilde{C}(m,n,k)}{m^\alpha n^\beta} = \begin{cases} \dfrac{(k-mnp)^2}{m^\alpha n^\beta 3mnp}, & k \in [mnp, 2mnp]; \\ \dfrac{(k-mnp)^2}{m^\alpha n^\beta (k+mnp)}, & k > 2mnp. \end{cases} \tag{1}$$

Note again that the function $C_{\alpha,\beta}$ is only defined for $k \geq mnp$. Moreover, there is a subtle difference for $k \in [mnp, 2mnp]$ and $k > 2mnp$. Now we can define a bicluster to be a submatrix with maximal $C_{\alpha,\beta}$.

## 2.2. Search algorithm

Evaluating $C_{\alpha,\beta}$ for all submatrices in order to find the maximum is clearly not feasible. We restrict the search space as in Koyutürk et al. (2004) and employ their efficient search strategy, where the rows and columns are iteratively searched. Before describing the algorithm we introduce some notation. We define $x$ to be an $M$-dimensional binary column-vector, where a one indicates that the row is part of the bicluster, and a 0 indicates that it is not. Similarly we define the column-vector $y \in \{0, 1\}^N$ for the columns.

*Algorithm $A_1$: Iterative algorithm to find the maximal-$C_{\alpha,\beta}$ bicluster*

(1) The algorithm is initiated by selecting a subset of either the columns or the rows. Let us assume that we start with a random subset of the columns (the case where we start with a subset of the rows is similar), and we set $y$ accordingly, i.e., $y_i = 1$ if column $i$ is in the selected set. In principle, one could start with any subset of the columns.
(2) Given the subset of columns, represented by $y$, we compute the rowsums $s$ over the selected columns, $s = Gy$.
(3) We sort the rows in decreasing order of rowsums $s$, and relabel them $\pi_1, \ldots, \pi_M$. Thus $\pi_1$ is the row with the largest rowsum and $\pi_M$ is the row with the smallest rowsum.
(4) Consider the subset of rows $R_m := \{\pi_1, \ldots, \pi_m\}$. For each $m = \{1, \ldots, M\}$ we have a bicluster with columns $C := \{i \,|\, y_i = 1\}$ and the subset of rows $R_m$. Clearly, the $m$th bicluster has $m$ rows, $n = \sum_{i=1}^{N} y_i$ columns and $k = \sum_{i=1}^{m} s(\pi_i)$ ones. We calculate for each of these $M$ biclusters the value of $C_{\alpha,\beta}$, which is for bicluster $m = 1, \ldots, M$ given by:

$$C_{\alpha,\beta}(m) = \begin{cases} \dfrac{\left(\sum_{i=1}^{m} s(\pi_i) - mnp\right)^2}{3m^{1+\alpha} n^{1+\beta} p}, & \sum_{i=1}^{m} s(\pi_i) \in [mnp, 2mnp]; \\[2ex] \dfrac{\left(\sum_{i=1}^{m} s(\pi_i) - mnp\right)^2}{m^{\alpha} n^{\beta} \left(\sum_{i=1}^{m} s(\pi_i) + mnp\right)}, & \sum_{i=1}^{m} s(\pi_i) > 2mnp. \end{cases}$$

(5) Determine

$$m^* = \arg \max_{m=\{1,\ldots,M\}} \{C_{\alpha,\beta}(m)\}.$$

This gives a subset of rows $R_{m^*} = \{\pi_1, \ldots, \pi_{m^*}\}$.
(6) Construct the binary column-vector $x$ such that $x_j = 1$ if $j$ is part of $R_{m^*}$, and $x_j = 0$ otherwise.
(7) Go to Step 2 and do the same for the columns with $s = G^T x$. Stop if $C_{\alpha,\beta}(m^*)$ does not increase. The rows for which $x$ is equal to 1 and the columns for which $y$ is equal to 1 are a bicluster with value $C_{\alpha,\beta}$, which is equal to the last obtained maximum $C_{\alpha,\beta}(m^*)$.
(8) Since each run (Steps 1 to 7) ends up in a locally maximal value of $C_{\alpha,\beta}$, repeat Steps 1 to 7 a number of times $I$, every time generating a new random initialization of the rows or the columns. Return the bicluster with the maximal $C_{\alpha,\beta}$ over all $I$ runs.

Note that the maximal-$C_{\alpha,\beta}$ bicluster is not always the submatrix for which the score function has a global maximum, because we cannot evaluate all possible submatrices. However, the search algorithm is designed such that given $x$ a global optimum is reached for $y$ and vice versa, see (Koyutürk et al., 2004) for more details. We would like to point out here that the $A_1$ algorithm, because of our score function, does very well in finding the biclusters in artificial experiments, as opposed to *Cmnk*.

### 2.3. *Finding all biclusters*

The maximal-$C_{\alpha,\beta}$ bicluster is maximally dense with ones, but the number of ones can in principle vary from $mnp$ to $mn$. Moreover, for each input matrix $G$ and given values of $(\alpha, \beta)$ the number of ones in the maximal-$C_{\alpha,\beta}$ bicluster will be different. The user would probably prefer to indicate in advance the proportion of ones that should be in a submatrix before it is called a bicluster, even though the submatrix has maximum $C_{\alpha,\beta}$. In order to find biclusters that the user would consider dense enough, we introduce a parameter $p_1$ and define a $p_1$-bicluster as a bicluster that has a proportion of ones that is larger than or equal to $p_1$. Suppose the $A_1$ algorithm outputs a maximal-$C_{\alpha,\beta}$ bicluster which has a proportion of zeros that is below $p_1$. Because it *is* the maximal-$C_{\alpha,\beta}$ bicluster it contains all the interesting information. To further enrich this bicluster for ones, i.e., to find a more dense bicluster that meets the $p_1$ constraint, we run the $A_1$ algorithm again on the maximal-$C_{\alpha,\beta}$ bicluster. More formally, this results in the following algorithm, $A_2$, that returns the maximal-$C_{\alpha,\beta}$ $p_1$-bicluster with a proportion of ones larger than or equal to $p_1$.

*Algorithm $A_2$: Algorithm to find the maximal-$C_{\alpha,\beta}$ $p_1$-bicluster*

```
(1) Run $A_1$ on $G$ to obtain a maximal-$C_{\alpha,\beta}$ bicluster $B$.
(2) Calculate $p_B$, the proportion of ones in $B$.
(3) Stop if $p_B \geq p_1$, $B$ is then the maximal-$C_{\alpha,\beta}$ $p_1$-bicluster. If $p_B < p_1$ run $A_1$ on $B$ and denote the
    obtained new maximal-$C_{\alpha,\beta}$ bicluster by $B$. Go to Step 2.
```

In order to find *all* the biclusters in the dataset, we have to run the $A_2$ algorithm to find the maximal-$C_{\alpha,\beta}$ $p_1$-bicluster a number of times. This results in the main algorithm *BicBin*, which finds all biclusters in a dataset for a given $(\alpha, \beta)$ combination, and for a given value of $p_1$.

*Algorithm BicBin: Algorithm to find all maximal-$C_{\alpha,\beta}$ $p_1$-biclusters*

```
(1) Run $A_2$ on $G$ to find the maximal-$C_{\alpha,\beta}$ $p_1$-bicluster.
(2) Put the bicluster in a list of biclusters. Set its elements in $G$ to 0.
(3) Check whether $G$ still contains non-zero elements. If not, stop, else go to Step 1.
```

We decided to set all the elements of a bicluster that has been found to zero. We prefer this over setting the elements of the bicluster to one with probability $p$, since this will certainly lead to biclusters that contain ones that were artificially introduced.

### 2.4. *$\alpha$-$\beta$-space*

What values of $\alpha$ and $\beta$ should be chosen in order to obtain the "true" biclusters in a real dataset? The formula for $C_{\alpha,\beta}$ reveals how the parameters should be interpreted. When $\alpha$ equals $\beta$, rows and columns are weighed equally in $C_{\alpha,\beta}$. This means that when we have to choose between adding a row or a column—both being of equal size with an equal number of ones—to an existing bicluster, it does not matter for the value of $C_{\alpha,\beta}$ which of the two we add. The situation is different when $\alpha$ is larger than $\beta$. Then the columns are weighed more heavily than the rows, so the value of $C_{\alpha,\beta}$ increases more when adding a column than a row, provided they contain both an equal number of ones. Of course, when it is the other way around and $\beta$ is larger than $\alpha$ then $C_{\alpha,\beta}$ increases most rapidly by adding rows, rather than columns. So, depending on the dimensions (number of rows and columns) of the putative bicluster, one may decide that rows are preferred over columns or the other way around, and choose values for $(\alpha, \beta)$ accordingly.

This is illustrated by an artificial example (see Fig. 1) where the notion of the "true" bicluster is ambiguous, just as in real-life datasets. The artificial binary matrix contains 100 rows and 100 columns, with $p$ equal to 0.17. One could argue that Figure 1A contains one big bicluster, as indicated by the gray square. At the same time, one could argue that one of the biclusters, represented by the gray rectangles, in Figure 1B, 1C, or 1D is the "true" bicluster. For instance, choosing $\alpha = 0.5$ and $\beta = 0.5$, we can calculate the value of the score function $C_{0.5,0.5}$ for all four possible biclusters that are shown (in gray) in Figure 1A, 1B, 1C, and 1D: $C_{0.5,0.5} = 13.39$ (Fig. 1A), $C_{0.5,0.5} = 15.84$ (Fig. 1B), $C_{0.5,0.5} = 16.05$ (Fig. 1C) and $C_{0.5,0.5} = 16.01$ (Fig. 1D). So for $\alpha = 0.5$ and $\beta = 0.5$ the bicluster in Figure 1C is the maximal-$C_{\alpha,\beta}$ bicluster. When $\alpha$ is chosen larger than $\beta$, such as for instance when $(\alpha, \beta) = (0.8, 0.6)$, then the bicluster in Figure 1B is maximal-$C_{\alpha,\beta}$, and when $\beta$ is larger than $\alpha$, for instance $(\alpha, \beta) = (0.6, 0.8)$, *BicBin* would pick the bicluster in Figure 1D as maximal-$C_{\alpha,\beta}$. Note that in all these examples $p_1$ is set to 1 for simplicity. This example clearly demonstrates how the $\alpha$-$\beta$-space can be employed to explore the dataset and retrieve biclusters of varying dimensionalities—a feature that is particularly useful in real-world datasets, where the precise definition of a bicluster is often ambiguous.

For a quick first search for unusually dense biclusters in a dataset, we propose to use $\alpha = 0.5$ and $\beta = 0.5$. Applying this procedure to the dataset in Figure 1, we find three biclusters (Fig. 1C): the first is the gray bicluster, the second is the white bicluster to the left and the third is the white bicluster on top of the gray bicluster. Using *BicBin* with $\alpha = 0.5$ and $\beta = 0.5$ on our artificial experiments, we find most of the true artificial biclusters, except for some extreme cases. For a more thorough search through the dataset, and depending on the user preferences with regard to the ratio of the number of rows relative to the number of columns in the retrieved biclusters, *BicBin* can be run for several combinations of parameter values. Obviously, this results in several lists of biclusters, one for each $(\alpha, \beta)$ combination.

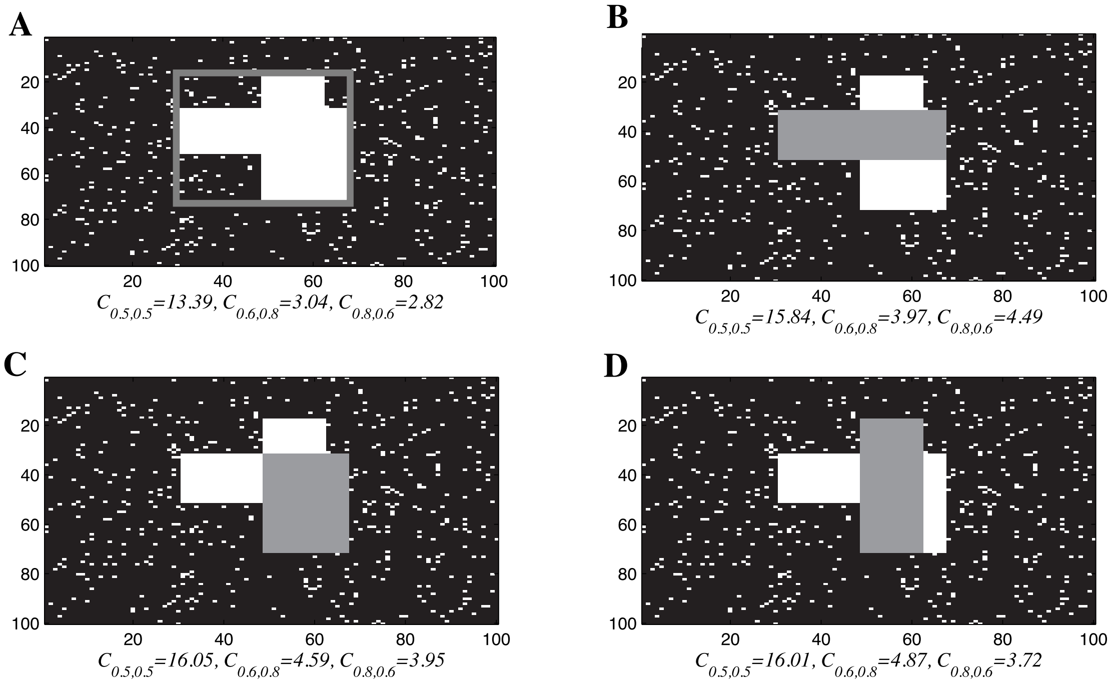


**FIG. 1.** Illustration of $\alpha$-$\beta$-space. The four possible biclusters are depicted in gray in each of the panels A, B, C, and D. Below the panels, the values of $C_{\alpha,\beta}$ for the bicluster in gray are given for the different choices of $(\alpha, \beta)$.

## 3. COMPARING *BicBin* TO *Cmnk* AND *Bimax*

In this section, we compare *BicBin* to the currently available approaches for biclustering binary data. First we perform a comparison of *BicBin* with *Cmnk*, the algorithm proposed in Koyutürk et al. (2004), on a series of artificial datasets containing single biclusters of varying size embedded within larger data matrices which also vary in size. Then we compare *BicBin* with *Bimax* using the artificial dataset proposed in Prelić et al. (2006).

### *3.1. Comparison to Cmnk*

We create binary matrices $G$ of size $M \times N$ with an artificial bicluster as follows. First we set each element in $G$ to one with probability $p$, and to zero with probability $1-p$. This represents the "background noise." Then we insert a bicluster of size $m \times n$, containing $mn$ ones into $G$. We considered the values $M = 10$, 100, 1000, and $N = 50$, 500, 5000. We chose $p$ to range from 0.001 to 0.5, in order to evaluate the performance of the algorithms with low and high levels of background noise. For the artificial biclusters we took size $m \times n$, where $m$ ($n$) was varied from 2 to $M$ ($N$) in 50 (10) equal sized steps.

A natural performance measure for these experiments is the false positive rate (FPrate), which is defined as the number of false positives (FP) divided by the sum of the number of true positives (TP) and the number of false positives, i.e, FPrate = FP/(TP+FP). The false positives are the elements in the bicluster found by the algorithm that are not part of the true bicluster. The true positives are the true elements in the bicluster found by the algorithm. The number of predicted postives, which is TP + FP, is the total number of elements in the bicluster returned by the algorithm. The FPrate measures the relative amount of background noise that is returned by the algorithm as belonging to the true bicluster. The FPrate ranges from 0, when the algorithm only returns true bicluster elements and hardly any background noise, to 1, when the algorithm performs worst and only returns background noise.

To account for locally maximal biclusters when comparing *BicBin* with *Cmnk*, we proceeded as follows. For every combination of $m$, $n$, $M$, $N$, and $p$, we construct 5 instances of a binary matrix of size $M \times N$ with background noise $p$, containing an artificial bicluster of size $m \times n$. For each of the 5 binary matrices, we run both *BicBin* and *Cmnk* 50 times, and retain the result with the highest value of the cost functions: $C(m, n, k)$ for *Cmnk* and $C_{\alpha,\beta}$ for *BicBin*, respectively. For *BicBin* we choose $\alpha = 0.5$ and $\beta = 0.5$, and $p_1 = 1$, and for *Cmnk* we choose $P^* = 10^{-9}$, since that gave the best overall performance for *Cmnk*. For both algorithms we chose $I = 50$ and a random initialization of the columns.

It is important to note that both algorithms performed equally well with respect to the false negative rate, which is defined as the fraction of false negatives (true bicluster elements missed by the algorithm) to the predicted negatives (the elements that are not part of the bicluster returned by the algorithm). This means that both algorithms are capable of detecting the bicluster. However, *BicBin* returns only the true bicluster and *Cmnk* returns a large number of false positives in addition to the bicluster. For this reason it is only necessary to consider the FPrate when comparing the two algorithms.

We averaged the five best FPrates to obtain one FPrate per $m$, $n$, $M$, $N$, $p$-combination. The results for $p = 0.001$ and $p = 0.5$ are given in Figures 2–5 for both *BicBin* and *Cmnk*. Each of the figures displays the results achieved on the nine data matrices associated with the different combinations of $M$ and $N$. Within each matrix, the value at position $(m, n)$ represents the FPrate for a bicluster of size $m \times n$ in a big matrix of size $M \times N$. From the figures it follows immediately that *Cmnk* performs poorly, since the FPrate is very high for most of the $m$, $n$, $M$, $N$, $p$-combinations, while *BicBin* does a very good job of finding the bicluster for almost all parameter settings. It is only for $p = 0.5$ and very small biclusters (top left corners in Fig. 5) where *BicBin* does not achieve a low FPrate. Clearly the noise level has a far greater impact on the perfomance of *Cmnk*. For example, for $p = 0.001$ (Fig. 2) the FPrate for $N = 10$, $M = 50$, 500, 5000 is reasonably good, while the FPrate for the corresponding matrices for $p = 0.5$ (Fig. 4) is dramatically larger.

What is also apparent is that the size of the bicluster relative to the data matrix it is contained in, strongly influences the FPrate of *Cmnk*. For example, in the top-left corners of all the panels in Figure 4, the results are shown for relatively small biclusters contained in a large data matrix. At all these positions the FPrate approaches 1. This indicates that *Cmnk* is not capable of finding relatively small biclusters in large data matrices. To conclude, *Cmnk*, as proposed in Koyutürk et al. (2004) clearly fails to detect biclusters, and

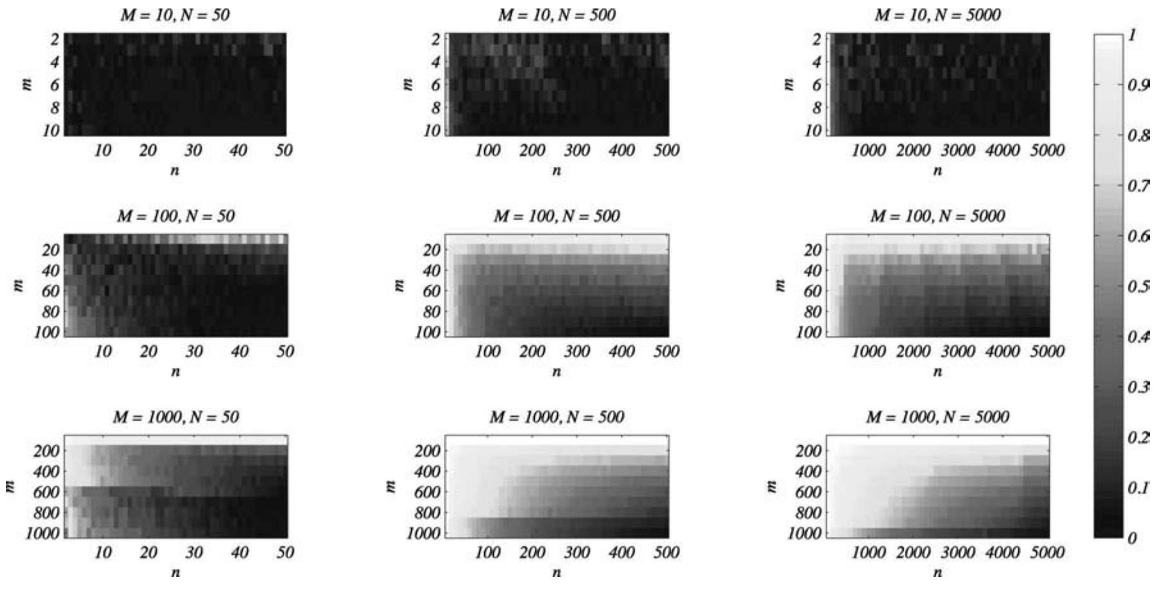


**FIG. 2.** False positive rates for *Cmnk* when $p = 0.001$.

therefore we do not proceed with this algorithm on the artificial dataset in the next subsection, nor on the transcription factor binding data.

## 3.2. Comparison to Bimax

*Bimax* finds all possible biclusters for a given minimal number of rows and columns. Unfortunately *Bimax* only returns biclusters that have all elements equal to one. This is clearly a very restrictive constraint, as we are often interested in "maximally dense" biclusters, for which the vast majority of elements are set to one. This allows for the possibility to cope with noise (some elements being zeros while they should have been one). Especially when the binary matrix is obtained by thresholding, such as for gene expression data, it is important that an algorithm can find biclusters with some zeros. For the transcription factor binding data it

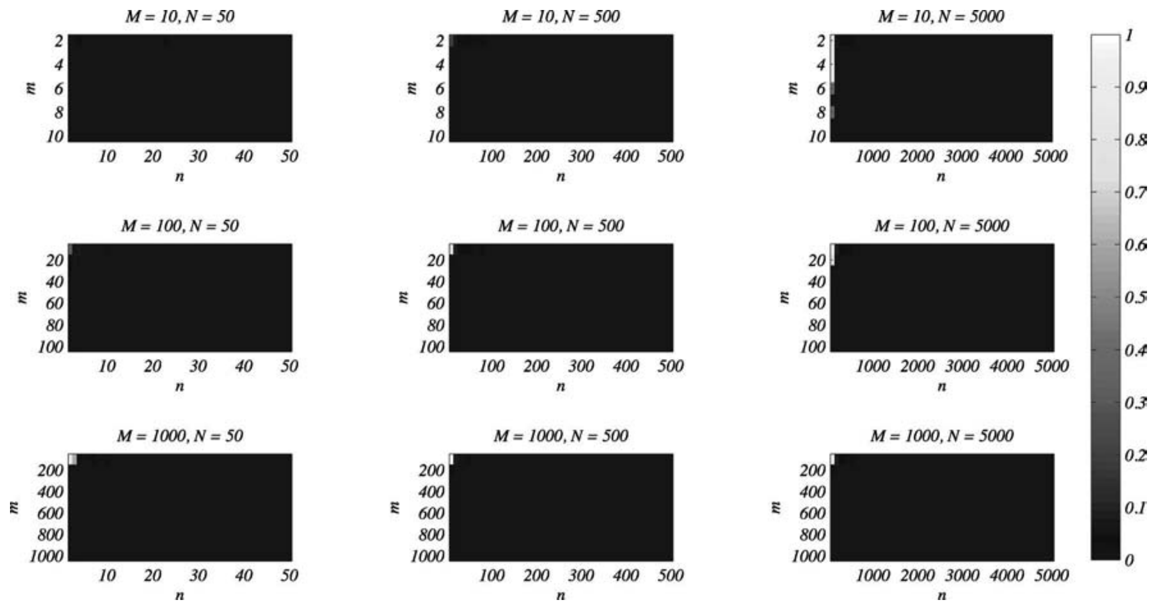


**FIG. 3.** False positive rates for *BicBin* when $p = 0.001$.

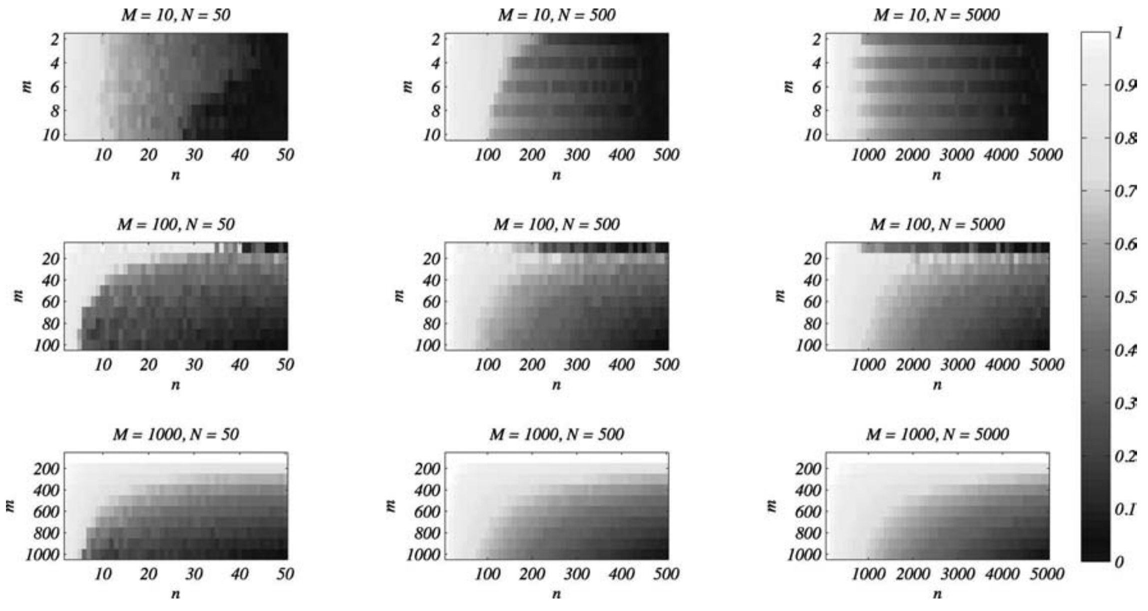


**FIG. 4.** False positive rates for *Cmnk* when $p = 0.5$.

is important to allow for some zeros in the biclusters, since the binary matrix for transcription factor motifs is obtained by thresholding probabilities. We illustrate how *Bimax* works when some elements change from one to zero, using the artificial example that is proposed in Prelić et al. (2006). We first construct the binary matrix $E$ (see the Appendix for details), and apply both *Bimax* and *BicBin*. The output of *Bimax* is a list of 28 unprioritized biclusters. The output of *BicBin* is a ranked list of biclusters (Fig. 11 in Appendix), where the ranks are based on the value of $C_{\alpha,\beta}$. We note that matrix $E$ is very ambiguous, as it contains, in principle, ten equally large overlapping biclusters. While each of these could be the top scoring bicluster for *BicBin*, depending on the columns or rows that are selected in Step 1 of algorithm $A_1$, four are output at the top of the list. Then we randomly change ten of the ones in $E$ into zeros, and obtain binary matrix $E'$ (Fig. 10B in Appendix). The output of *BicBin* applied to $E'$ is the list of ranked biclusters as visualized

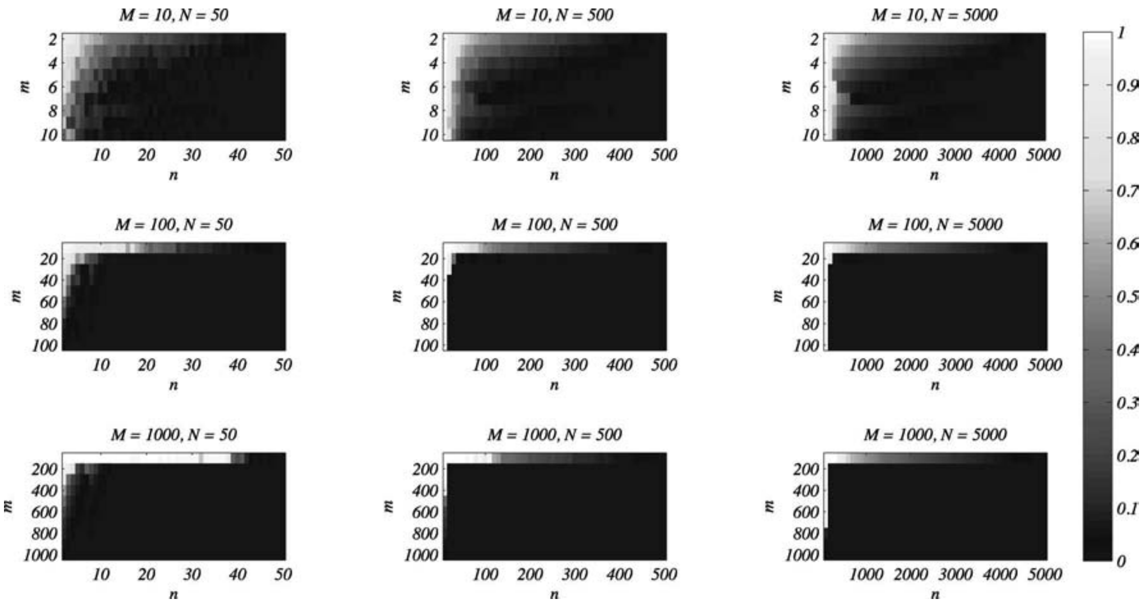


**FIG. 5.** False positive rates for *BicBin* when $p = 0.5$.

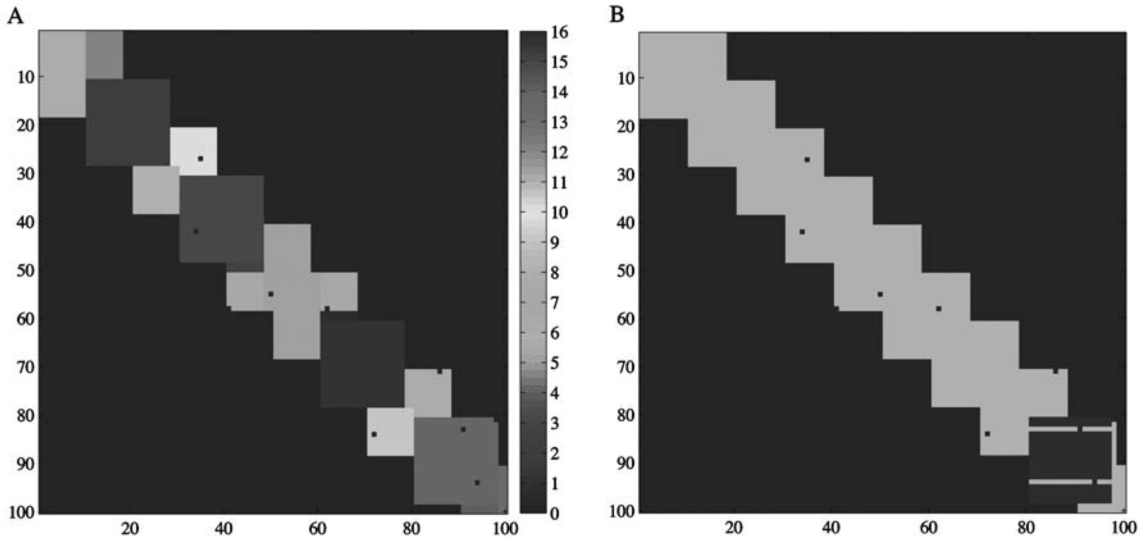


**FIG. 6.** **(A)** Resulting biclusters when *BicBin* is applied to $E'$. **(B)** The 34th bicluster found by Bimax (bottom right, dark) in $E'$.

in Figure 6A. It again returns four of the ten possible top scoring biclusters, even though two of these contain a few zeros. However, as a result of the minor modification to $E$, the output of *Bimax* changes dramatically. It now outputs an unprioritized list of 73 biclusters (compared to 28 for $E$). As an example we show in Figure 6B the bicluster that is output as 34th on the list. This should ideally have been one of the ten big biclusters, but *Bimax* left out two rows and one column because of the additional zeros. To conclude, *Bimax* is clearly very sensitive to noise being added to the binary input matrix, since it requires every bicluster to be completely pure, i.e., free of zeros. This is undesirable, since genomic data matrices typically contain various kinds of noise, and a bicluster should not be divided in subclusters if it contains a few zeros. In addition, Bimax does not provide a prioritization of the obtained biclusters, which makes it extremely difficult to browse the data, or perform some post-processing steps.

## 4. TRANSCRIPTION FACTOR BINDING DATA

In this section, we illustrate the performance of *BicBin* on genomic datasets, by applying *BicBin* to a dataset of putative transcription factor binding sites (TFBS). We have chosen to use the TRANSFAC database (Wingender et al., 2000). TRANSFAC is the database on eukaryotic transcription factors, their genomic binding sites and DNA-binding profiles, i.e., *motifs*. Our objective is to find a bicluster consisting of a group of transcription factors that all putatively bind to the same group of genes. Such a bicluster could reveal potentially novel regulatory complexes.

### *4.1. Biclustering TFBS with* BicBin

As described in the Appendix, we transform the data from TRANSFAC into a binary matrix. In this procedure we corrected for false positives and used the UCSCs most conserved regions to obtain only the high confidence putative relationships. We noticed that the database is quite redundant in the sense that the sequences of some of the motifs are so alike that their binding profiles are almost identical. Hence, applying *BicBin* to a matrix with transcription factor motifs along the rows and target genes along the columns, the most prevalent biclusters will contain groups of motifs that are very similar. These biclusters are clearly uninteresting since they could have been found by one-way hierarchical clustering of the sequences. Therefore, we decided to merge similar motifs together in a so-called *motif group*, and create a binary matrix with transcription factor motif groups along the rows and genes along the columns, where an element is one if the transcription factor associated with the motif group putatively binds the upstream region of the gene. This results in a matrix of 337 motif groups and 10,801 genes with at least one hit.

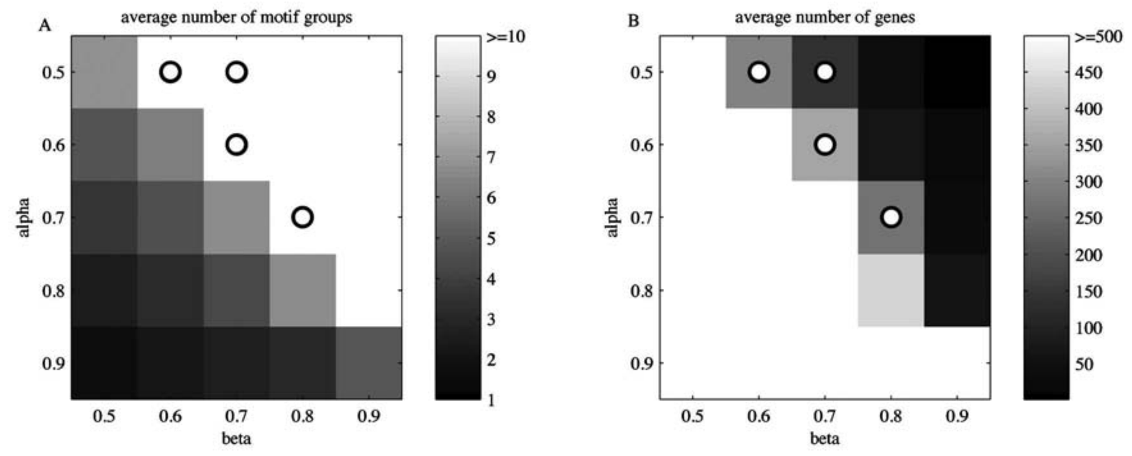


**FIG. 7.** Average number of motif groups **(A)** and genes **(B)** in the first 100 biclusters for combinations of $\alpha$ and $\beta$. The white dots indicate the values of $\alpha$ and $\beta$ that we have chosen for further analysis.

In order to investigate what kind of biclusters are hidden in our binary matrix, we first apply *BicBin* to get the top 100 biclusters for all combinations of $\alpha$ and $\beta$. Note that for all the experiments in this section we have chosen $I = 600$, where we iteratively use a random initiation of $x$ and $y$ (see the $A_1$ algorithm). We have found that the global maximum in most of the applications was reached after $I = 300$ with alternating random initiations of $x$ and $y$. An attractive feature of *BicBin* is the fact that the user can choose a value for $p_1$, since $1 - p_1$ predefines the largest proportion of zeros allowed in a bicluster. The results presented here are for $p_1$ equal to 0.9, because we want to allow some zeros, but not too many. Figure 7A shows the average number of motif groups per bicluster, and Figure 7B shows the average number of genes per bicluster for these combinations. Figure 7A, for $\alpha > \beta$ (lower triangular part of the matrix), most of the biclusters contain only a single motif group, whereas for $\alpha < \beta$ (upper triangular part) the number of motif groups increases. The opposite effect is present for the average number of genes in a bicluster, as is shown in Figure 7B. When $\alpha > \beta$ a large number of genes is found in a bicluster, whereas for $\alpha < \beta$ the number of genes decreases.

We have chosen the $\alpha$ and $\beta$ combinations as indicated with the white dots in Figure 7. For these values of $\alpha$ and $\beta$, we obtain biclusters with a reasonable number of transcription factor motif groups as well as genes, i.e., not too few transcription factor motif groups and not too many genes. We also applied *Bimax* to this matrix, which resulted in an unsorted list of over 200 million biclusters, which is obviously an undesirable result.

## 4.2. Evaluation of the results

*STRING interactions.* A first validation is based on known protein-protein interactions such as those listed in the STRING database (von Mering et al., 2007). These interactions are derived from four sources: genomic context, high-throughput experiments, (conserved) coexpression and previous knowledge. We expect to find biclusters that contain transcription factors of which some are known to interact, as well as novel interactions. The latter reveal potentially novel regulatory modules. In Figure 8, we show the STRING result for the first bicluster that we obtained with $\alpha = 0.5$ and $\beta = 0.6$ (more STRING results can be found on our website: *http://bioinformatics.nki.nl/software.php*). We took the genes that code for the transcription factors in the first bicluster and searched for existing associations between them or their proteins, using the STRING database. We note that most of these genes, coding for the transcription factors in the first bicluster, are indeed present in the STRING database. We observe that roughly half of the transcription factors are known to interact and for a relatively large fraction multiple independent sources of evidence exist for the interactions; for a description of the sources of evidence, see Figure 8. This suggests that for the connected transcription factors that we group together in a bicluster, convincing evidence exists that they are, in fact, functionally related. Based on our output we might also hypothesize that some of the unconnected transcription factors could putatively interact with the other transcription

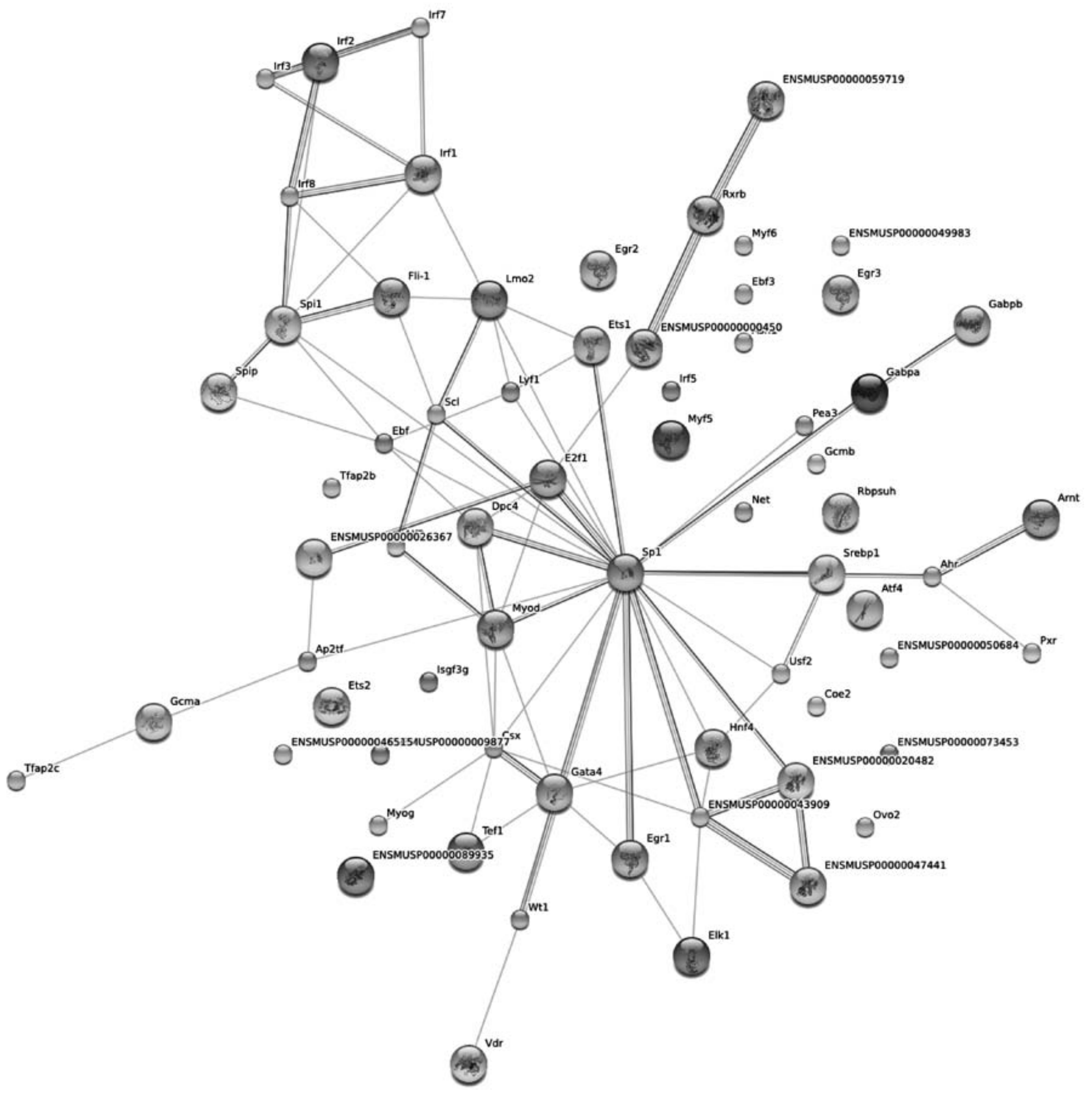


**FIG. 8.** STRING output for the first bicluster ($\alpha = 0.5$, $\beta = 0.6$). Transcription factors are represented by nodes, and a variety of associations between the nodes are represented by the edges. The following associations between nodes are shown in the graph: (1) "neighborhood" indicating frequent occurrence of genes in the same genomic neighborhood; (2) "co-occurrence" of linked orthologous groups across species; (3) "gene fusion" indicating whether fusion occurred; (4) "co-expression" representing evidence of mRNA co-expression of the associated genes; (5) "experiments" representing experimental evidence for an interaction; (6) "databases" representing evidence from pathway databases for association; and (7) "text mining" representing co-publication information. Please refer to von Mering et al. (2007) and http://bioinformatics.nki.nl/software.php for more information on this.

factors in the bicluster, and might be involved in similar regulatory processes as those that are known to interact.

*GO enrichment.* We expect the genes that are bound by the same transcription factors to be involved in similar processes. Hence, a second validation of the biclusters that we obtained with *BicBin* is provided by the number of biclusters whose genes are enriched for GO categories. Following Prelić et al. (2006), we calculate for each bicluster whether the genes contained in the bicluster have a higher degree of overlap with any of the GO categories than could have been expected by chance. We use the hypergeometric distribution and apply Benjamini and Hochberg multiple testing correction to correct

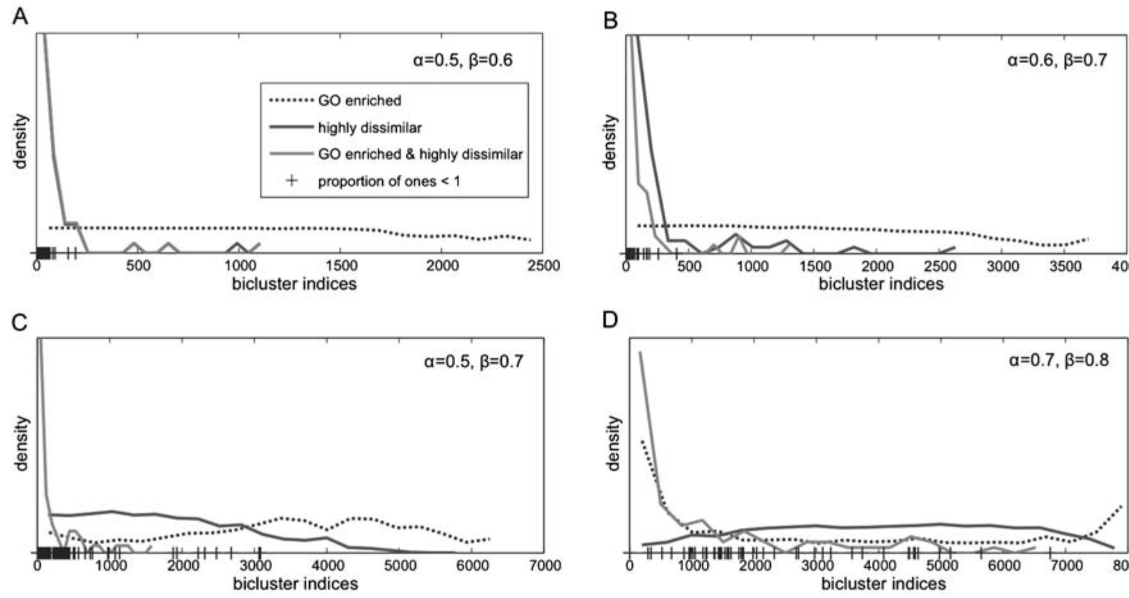


**FIG. 9.** Density plots of the number of GO enriched (dotted line) and highly dissimilar biclusters. We also plot the number of biclusters that are both GO enriched and highly dissimilar. We marked on the horizontal axis the biclusters that have a proportion of ones smaller than 1. Each panel represents an $(\alpha, \beta)$ combination, and the legend in A holds for all panels.

for *both* the number of biclusters and GO categories tested. In Figure 9, we provide an overview of the obtained biclusters using *BicBin* with $(\alpha, \beta) = (0.5, 0.6)$ (Fig. 9A), $(\alpha, \beta) = (0.6, 0.7)$ (Fig. 9B), $(\alpha, \beta) = (0.7, 0.8)$ (Fig. 9C), and $(\alpha, \beta) = (0.5, 0.7)$ (Fig. 9D) (settings corresponding to the white dots in Fig. 7). We plotted (dotted line) the densities of GO enriched biclusters, and we see that GO enrichment is more concentrated towards the first biclusters extracted by *BicBin*. Among the first 500 biclusters for $(\alpha, \beta) = (0.5, 0.6)$, 498 were GO enriched, and for $(\alpha, \beta) = (0.6, 0.7)$ we obtained a similar result with 484 GO enriched biclusters. Figure 9C shows a slight shift in enrichment towards clusters extracted later on. In Figure 9D, we also see a peak at the end, which is due to the fact that *BicBin* continues extracting biclusters until the matrix is completely empty. We checked the last biclusters, and most of them consisted of just one motif group and the genes they bind—a trivial bicluster which one expects to be GO enriched.

*Sequence (dis)similarity.* A third validation involves the sequence similarity of the motifs. Even though the most similar motifs have been merged in motif groups, we still expect the transcription factor motif groups in a bicluster to have fairly similar sequences, which serves as a straight-forward validation of our algorithm. Even more interesting, we investigate whether we obtain biclusters containing dissimilar transcription factor motif groups. The latter might reveal novel biological insights in transcriptional networks. We use the Tomtom algorithm (Gupta et al., 2007) to calculate the dissimilarity between two motifs, and compute the pairwise motif group dissimilarities using the Hausdorff distance. We define a bicluster to be highly dissimilar if it contains at least two motif groups whose Hausdorff distance exceeds the 95% quantile of all the pairwise motif scores between individual motifs. More details can be found in the appendix. In Figure 9, we plotted the densities of the highly dissimilar biclusters (dark gray line). In all panels we see that most of the highly dissimilar biclusters are extracted first. The light gray line depicts the density of the biclusters that are both GO enriched and highly dissimilar. It clearly shows that *BicBin* finds these most interesting biclusters first, indicating that *BicBin* is very effective in finding GO enriched and highly dissimilar biclusters. In addition, we marked with ticks on the horizontal axis the biclusters that have a proportion of ones smaller than one. Interestingly, peaks in the density of highly dissimilar GO enriched biclusters coincide with the ticks, indicating that the most interesting biclusters are not fully dense with ones. We remark that these biclusters cannot be found by *Bimax*. Based on these results, we conclude that *BicBin* is an effective biclustering algorithm.

TABLE 1. SUMMARY STATISTICS OF THE BICLUSTERS PER $\alpha$ AND $\beta$ COMBINATION

| $\alpha$ | $\beta$ | *No. of biclusters* | *No. of # GO enriched* | *No. of GO enriched in first 500 biclusters* | *GO enriched and highly dissimilar biclusters* |
|---|---|---|---|---|---|
| 0.5 | 0.6 | 2508 | 2188 | 498 | **1**, **2**, **3**, **5**, **6**, **7**, **8**, **9**, **10**, **13**, 15, 17, **20**, 21, **22**, **26**, 28, **29**, **31**, 32, **39**, 49, 51, **53**, **56**, **60**, 64, 70, 76, 77, 78, **83**, **91**, 100, 101, 116, 121, 128, 184, 191, 203, 475, 651, 1133 |
| 0.6 | 0.7 | 3797 | 2890 | 484 | **1**, **2**, **3**, 4, **5**, **6**, 8, **9**, 10, **12**, **16**, **17**, **18**, **19**, 20, **22**, 24, 25**26**, 27, **33**, 37, 40, **41**, 52, 56, 66, 67, 68, 81, 86, **89**, **91**, 98, 133, 134, 151, 164, **169**, 176, 180, 185, **188**, 214, 224, 320, 688, 894, 925, 1337 |
| 0.5 | 0.7 | 6428 | 2020 | 131 | **1**, **2**, **3**, **4**, **6**, **8**, **9**, **10**, **12**, **13**, **14**, 15, **16**, **17**, **18**, **19**, **21**, **22**, **24**, **29**, **30**, **32**, **34**, **38**, **40**, **45**, **46**, **53**, **54**, **57**, **67**, 86, **87**, **105**, **126**, **149**, **150**, **156**, **157**, 167, 184, 219, **230**, **253**, **262**, **409**, **449**, 484, **bf 500**, **517**, 531, 572, 632, **762**, 852, 1056, 1182, 1294, 1620 |
| 0.7 | 0.8 | 8114 | 1204 | 359 | **1**, 2, 3, 5, 7, 9, 10, 15, 20, 22, 26, 29, 30, 31, 38, 39, 42, 49, 51, 73, 85, 99, 107, 115, 118, 136, 138, 141, 142, 190, 210, 221, 248, 259, 292, 293, 332, 337, 356, 399, 412, 437, 497, 559, 618, 657, 799, 823, 841, 906, **976**, 1037, 1063, 1091, 1245, 1296, 1327, 1377, 1480, 1741, 1789, 1839, 1994, 2083, 2297, 2843, 2968, 3181, 3287, 3575, 3957, 4308, **4494**, 4530, 4667, 4848, 4868, 5693, 6690 |

The bicluster numbers in boldface indicate that the proportion of ones for these biclusters was smaller than 1 (and $> 0.9$). The non-boldface bicluster numbers indicate a proportion of ones $p_1 = 1$.

*Overview of the results.* In Table 1, we provide a detailed overview of the output obtained by *BicBin*. The last column indicates the biclusters whose genes are enriched for at least one GO category, *and* contain at least two motif groups that are highly dissimilar. This shows again that *BicBin* finds the most interesting biclusters first, which is a very desirable property of our biclustering algorithm. We remark that the biclusters whose numbers appear in boldface in the sixth column are not fully dense with ones, but contain some zeros. Hence, we conclude that *BicBin* is very effective in finding GO enriched and highly dissimilar biclusters.

## 5. DISCUSSION

We presented an algorithm, *BicBin*, that is able to find biclusters in large-scale sparse binary datasets. While many approaches have been proposed to bicluster gene expression data, only two methods exist to bicluster binary matrices. The first is *Bimax* (Prelić et al., 2006), which produces an unmanageable, unprioritized list of biclusters, and cannot detect biclusters that have some elements equal to zero. *BicBin* is a major improvement over Bimax since it produces a manageable, prioritized list of biclusters as output, and is capable of finding biclusters that are not fully dense with ones. The second is *Cmnk*, the algorithm proposed in Koyutürk et al. (2004), which returns biclusters that contain a large number of predicted elements, where most of these predicted elements are not part of the true bicluster. *BicBin* clearly improves on this algorithm, since it is able to detect biclusters, even in a noisy background with a much lower false positive rate. We extensively demonstrated this on a series of artificial experiments. More importantly, we show that we can find biclusters in a TFBS dataset such as TRANSFAC, in a purely data-driven way. We correct for sequence similarities of the motifs and employ motif merging to ensure that these biclusters could not have been found by simple one-way hierarchical clustering of the sequences. The genes of the biclusters that *BicBin* detects are enriched for GO categories, and contain both known and novel interacting transcription factors.

## 6. APPENDIX

### 6.1. Additional information on the Bimax comparison

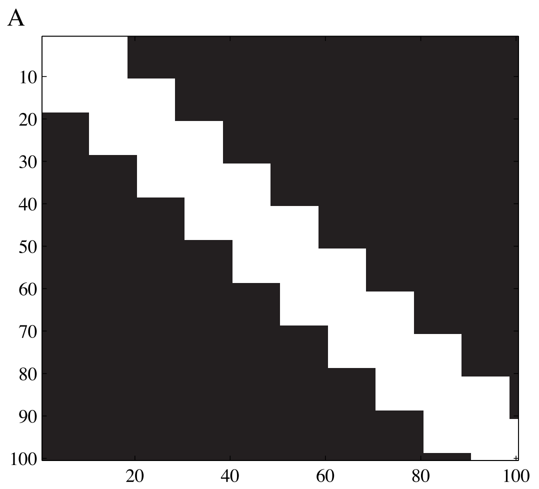
We obtain the binary matrix $E$ by setting the values $n = 100$, $m = 100$, and $t = 10$, and choosing a maximal overlap of $d = 8$, as proposed in Prelić et al. (2006). This results in the binary matrix $E$ (Fig. 10A). Then we randomly change ten of the ones into zeros. The resulting matrix, $E'$ ($E$ with ten additional zeros), is presented in Figure 10B. The results for *BicBin* applied to the original matrix $E$ are given in Figure 11. Note that we used $\alpha = 0.5$ and $\beta = 0.5$ because the biggest overlapping biclusters are all square.

### 6.2. Transforming TRANSFAC into a binary matrix

The TRANSFAC Professional database (release 10.3) contains 815 motifs of transcription factors, of which we selected the 585 vertebrate motifs. With the TFBS Perl module (Lenhard and Wasserman, 2002), we scanned the upstream regions of all genes on an Operon 35K mouse array against all these vertebrate motifs. We used the Ensembl gene ids (NCBI build m36) for the oligos, and retrieved the sequences of the basepairs $-1$ to $-1000$ in the upstream region of the genes with BioMart (release 41). We chose a scanning accuracy of 75%, meaning that a motif needs to correspond to a sequence for at least 75% in order for the upstream region of a gene to be a putative binding site. Both the sense and anti-sense strand of the sequences in the upstream regions of the genes were considered. This procedure yields for every motif–gene pair a so-called *matching number*, that indicates how often the transcription factor motif matches a sequence in the upstream region of a gene. A transcription factor is said to have a putative relation with a gene if the corresponding matching number is at least 1.

Because these putative relations are likely to contain false positives, we took the following two measures to only retain the high confidence putative relationships.

1. Under the null hypothesis that the motif is found by chance in the upstream region of the gene, the matching number follows a binomial distribution with parameters $n = 2(1000-\ell)$, with $\ell$ the length of a motif, and $p = 2^{-\mathrm{IC}}$, with IC the information content of the motif (D'haeseleer, 2006) for a concise



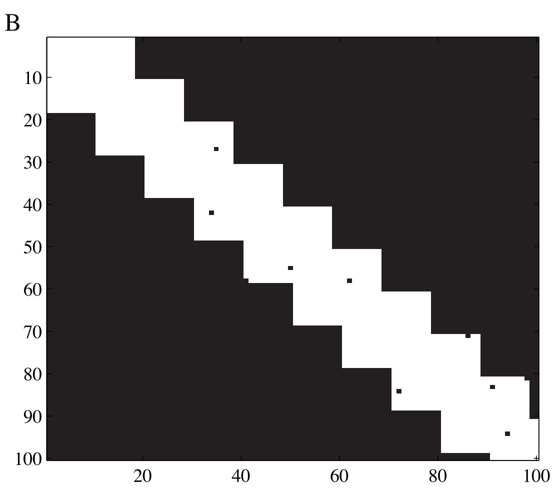


**FIG. 10.** **(A)** Artificial matrix $E$ from Prelić et al. (2006). **(B)** Artificial matrix $E'$, which is $E$ with ten ones randomly changed into ten zeros.

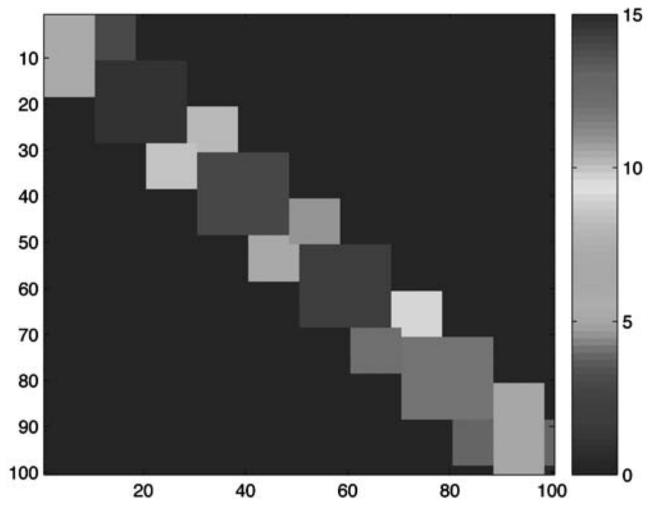


**FIG. 11.** Resulting biclusters when *BicBin* is applied to the original matrix $E$. The biclusters are ranked according to their $C_{\alpha,\beta}$, with the shading indicating the ranks.

overview. We used a threshold of 0.05 to decide whether the motif has a significant binding affinity with the gene, and if so, this motif–gene pair gets a 1 in the binary matrix.

2. The remaining putative relations between transcription factors and genes were checked for conservation across species (17 species), because *functional* binding sites are more likely to be conserved. We used the UCSC genome browser (release mm8, NCBI build m36) to obtain the most conserved regions. If the hit region of a motif–gene pair was found outside a most conserved region, the motif–gene element in the binary matrix was set to 0.

Our last concern is the redundancy of the motifs. The sequences of some of the motifs are so alike that their binding profiles are almost identical. Hence, the most prevalent biclusters will contain groups of motifs that are very similar. These biclusters are clearly uninteresting since they could have been found by one-way hierarchical clustering of the sequences. Therefore we decided to merge similar motifs together in a so-called *motif group* in the following way. First we used the Tomtom algorithm (Gupta et al., 2007) to calculate the similarity between motifs (we used the default setting Euclidian distance). The similarity is expressed in terms of so-called *$E$-values* which range from 0 to 585. With complete-linkage hierarchical clustering on the $E$-values (using Euclidean distance and a cutoff of 0.01), we create 337 nonredundant motif groups. Since each of the motifs has its own unique binding profile, we decided to merge the genes of the motifs that are in the same motif group. This way we keep the genes that make a motif unique, even though not all motifs in the motif group have a binding affinity to all genes. Now we have a binary matrix with 337 rows, representing the motif groups, and 10,801 columns, representing the genes, where an element is 1 if at least one of the motifs in the motif group has a significant binding affinity to the gene.

### 6.3. Motif group dissimilarity

The Tomtom $E$-values are Bonferroni multiple testing corrected $p$-values, without restricting the $p$-values to 1, so the Tomtom $E$-values range from 0 to 585 (we have 585 transcription factor motifs). To calculate the dissimilarity between two *motif groups*, we use the Hausdorff distance. We define a bicluster to be highly dissimilar if it contains at least two motif groups whose Hausdorff distance exceeds the 95% quantile of all the pairwise motif scores between individual motifs, which is 584.98.

## ACKNOWLEDGMENTS

We thank J. Kool, A.G. Uren, C. Lieftink, J. Calbo Angrill, C. Klijn, J. de Ridder, and M.H. van Vliet for useful discussions.

## DISCLOSURE STATEMENT

No conflicting financial interests exist.

## REFERENCES

Angluin, D., and Valiant, L.G. 1979. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *J. Comput. Syst. Sci.* 18, 155–193.

D'haeseleer, P. 2006. What are DNA sequence motifs? *Nat. Biotechnol.* 24, 423–425.

Dhollander, T., Sheng, Q., Lemmens, K., et al. 2007. Query-driven module discovery in microarray data. *Bioinformatics* 23, 2573–2580.

Elnitski, L., Jin, V.X., Farnham, P.J., et al. 2006. Locating mammalian transcription factor binding sites: a survey of computational and experimental techniques. *Genome Res.* 16, 1455–1464.

Erdös, P., and Spencer, J. 1974. *Probabilistic Methods in Combinatorics*, Academic Press, New York.

Gupta, S., Stamatoyannopoulos, J.A., Bailey, T.L., et al. 2007. Quantifying similarity between motifs. *Genome Biol.* 8, R24.

Harbison, C.T., Gordon, D.B., Lee, T.I., et al. 2004. Transcriptional regulatory code of a eukaryotic genome. *Nature* 431, 99–104.

Koyutürk, M., Szpankowski, W., and Grama, A. 2004. Biclustering gene-feature matrices for statistically significant patterns. *Proc. 2004 IEEE Comput. Syst. Bioinform. Conf.* 480–484.

Lenhard, B., and Wasserman, W.W. 2002. TFBS: computational framework for transcription factor binding site analysis. *Bioinformatics* 18, 1135–1136.

Lund, A.H., Turner, G., Trubetskoy, A., et al. 2002. Genome-wide retroviral insertional tagging of genes involved in cancer in Cdkn2a-deficient mice. *Nat. Genet.* 32, 160–165.

Madeira, S.C., and Oliveira, A.L. 2004. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 1, 24–45.

Mikkers, H., Allen, J., Knipscheer, P., et al. 2002. High-throughput retroviral tagging to identify components of specific signaling pathways in cancer. *Nat. Genet.* 32, 153–159.

Prelić, A., Bleuler, S., Zimmermann, P., et al. 2006. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics* 22, 1122–1129.

Sandelin, A., Alkema, W., Engström, P., et al. 2004. JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Res.* 32, D91–D94.

Shamir, R., Maron-Katz, A., Tanay, A., et al. 2005. EXPANDER—an integrative program suite for microarray data analysis. *BMC Bioinform.* 6, 232.

Sheng, Q., Moreau, Y., and De Moor, B. 2003. Biclustering microarray data by Gibbs sampling. *Bioinformatics* 19, 196–205.

Turner, H.L., Bailey, T.C., Krzanowski, W.J., et al. 2005. Biclustering models for structured microarray data. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 2, 316–329.

von Mering, C., Jensen, L.J., Kuhn, M., et al. 2007. STRING 7–recent developments in the integration and prediction of protein interactions. *Nucleic Acids Res.* 35, 358–362.

Wasserman, W.W., and Sandelin, A. 2004. Applied bioinformatics for the identification of regulatory elements. *Nat. Rev. Genet.* 5, 276–287.

Wingender, E., Chen, X., Hehl, R., et al. 2000. TRANSFAC: an integrated system for gene expression regulation. *Nucleic Acids Res.* 28, 316—319.

Address reprint requests to:
*Dr. Lodewyk Wessels*
*Bioinformatics and Statistics*
*Division of Molecular Biology*
*The Netherlands Cancer Institute*
*Plesmanlaan 121*
*1066 CX Amsterdam, The Netherlands*

*E-mail:* l.wessels@nki.nl