JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 12, Number 3, 2005
© Mary Ann Liebert, Inc.
Pp. 314–330

Finding Motifs in Promoter Regions

LIBI HERTZBERG, OR ZUK, GAD GETZ, and EYTAN DOMANY

ABSTRACT

A central issue in molecular biology is understanding the regulatory mechanisms that control gene expression. The availability of whole genome sequences opens the way for computational methods to search for the key elements in transcription regulation. These include methods for discovering the binding sites of DNA-binding proteins, such as transcription factors. A common representation of transcription factor binding sites is a *position specific score matrix* (PSSM). We developed a probabilistic approach for searching for putative binding sites. Given a promoter sequence and a PSSM, we scan the promoter and find the position with the maximal score. Then we calculate the probability to get such a maximal score or higher on a random promoter. This is the p-value of the putative binding site. In this way, we searched for putative binding sites in the upstream sequences of *Saccharomyces cerevisiae*, where some binding sites are known (according to the *Saccharomyces cerevisiae* Promoters Database, SCPD). Our method produces either exact p-values, or a better estimate for them than other methods, and this improves the results of the search. For each gene we found its statistically significant putative binding sites. We measured the rates of true positives, by a comparison to the known binding sites, and also compared our results to these of MatInspector, a commercially available software that looks for putative binding sites in DNA sequences according to PSSMs. Our results were significantly better. In contrast with us, MatInspector doesn't calculate the exact statistical significance of its results.

Key words: promoters, transcription factors, binding sites, PSSM.

1. INTRODUCTION

Many aspects of transcription regulation involve DNA-binding proteins, called *transcription factors*. These factors modulate the expression of genes by binding to specific positions in genes' promoters. Identifying *binding sites*, the locations to which these factors bind, remains a difficult problem in molecular biology. A central reason for this difficulty is that a single transcription factor might bind to regions which vary greatly in their sequence. Although the binding sites for a particular transcription factor share some common pattern, the pattern is not specific, and thus finding it is a difficult task.

One way to deal with this problem is through biological experiments, which are often costly and time consuming. The recent availability of complete genomic sequences (including intergenic regions) motivates

Department of Physics of Complex Systems, Weizmann Institute of Science, Rehovot 76100, Israel.

attempts to understand the regulatory mechanisms through computational analysis. Algorithms and tools for searching regulatory elements can be divided into two major classes: 1) methods that search for known transcription factor binding motifs (for example by using PSSMs, like MatInspector (Quandt *et al.*, 1995; Elkon *et al.*, 2003; and Aerts *et al.*, 2003); 2) methods that try to detect new consensus patterns within a set of DNA sequences. MEME (Bailey and Elkan, 1995) employs an EM technique for building a common consensus from a given set of DNA sequences, while Thijs *et al.* (2002) and Hughes *et al.* (2000) use a Gibbs sampling technique to detect common patterns.

We present here a method that searches for known transcription factor binding sites using their PSSMs. Given a DNA sequence D and a PSSM M, we scan the sequence and find the maximal score on it according to M. Suppose that the value of this maximal score is τ. We then need to decide whether τ is high enough to enable the transcription factor to bind to this position; i.e., we need to decide on a threshold score value, $\overline{\tau}$, such that a position with a score $\tau > \overline{\tau}$ is a "match," and a position with a score $\tau < \overline{\tau}$ is not. All methods that use PSSMs have to deal with this question. In Elkon *et al.* (2003), this threshold value is determined in a way that the number of matches in a big background promoter set (of 12,981 promoters of length 1,200 bp) is approximately 10% of the whole set. In MatInspector, it is chosen in a way that the estimated mean of the number of matches in a random sequence of length 1,000 is not too high.

We improve this by calculating the probability to get such a maximal score τ, or higher, in a random sequence of the same length (of the given sequence D). In some cases, we calculate the exact probability, and in others, we get a tight upper bound on it. This is an improvement in two aspects: 1) we get a reliable p-value; 2) we get more information on the score than just whether it is higher or lower than the threshold score. This is important, since the biological reality of the binding of a transcription factor has more than two possibilities (bound and not bound). The transcription factor can bind to the binding site with a certain affinity. Thus, a higher score (and a lower p-value) might indicate a higher affinity of binding of the transcription factor to this promoter.

In Section 2 we present the algorithm we use to calculate the p-value for the binding of a transcription factor to a promoter sequence.

We first define some notions in 2.1 and then present a sketch of the algorithm in 2.2. The algorithm is divided into two stages. In 2.3 we describe the first stage, of finding for a given PSSM the set A of all sequences whose score exceeds a threshold. This is done either by an exact computation or an approximation. In 2.4, we describe the second stage of the algorithm, where the probability of observing H occurrences of members of A in a random DNA sequence of length N is calculated. In 2.4.2, the exact computation is presented, and in 2.4.3, an approximation is presented. The results are described in 3. In 3.2, we present the comparison with MatInspector's results, and in 3.6, we describe how we use our method to find a common transcription factor in a group of promoters. In 3.6.1, we compare our method with PRIMA (Elkon *et al.*, 2003), a program for searching for common transcription factors in a set of genes.

2. METHODS

2.1. Preliminaries

Let Σ be the alphabet of the four nucleotides of which DNA sequences are composed, i.e., $\Sigma \equiv \{A, C, G, T\}$, and S the size of Σ, $S = |\Sigma| = 4$. Let $p_1, \ldots, p_4$ be the frequencies of the four nucleotides. We use here a random model for DNA sequences with independent nucleotides (not necessarily uniform). We denote the random model by B.

Let R be a DNA sequence of length L ($R = (r_1, \ldots, r_L)$, $r_i \in \Sigma$). Denote by F a given $S \times L$ position weight matrix (PWM). We take the log likelihood ratio to be the score of R:

$$score(R) = \log \left(\frac{\prod_{i=1}^{L} P(x_i = r_i/F)}{\prod_{i=1}^{L} P(x_i = r_i/B)} \right) = \sum_{i=1}^{L} \log \left(\frac{P(x_i = r_i/F)}{P(x_i = r_i/B)} \right) \tag{1}$$

where

$$P(x_i = j/F) = \frac{F_{j,i}}{F_{1,i} + F_{2,i} + F_{3,i} + F_{4,i}}, \quad P(x_i = j/B) = p_j$$

We get that

$$score(R) = \sum_{i=1}^{L} M_{r_i,i}$$

where $M_{j,i} = \log(\frac{P(x_i=j/F)}{P(x_i=j/B)})$. Matrix M will be denoted as the position specific score matrix (PSSM). Our aim is to determine the distribution of scores of R on random promoter regions of length N.

2.2. *A sketch of the algorithm*

Let $D = (d_1, \dots d_N)$ be a promoter region of length N. We scan D and calculate *score*(R) for $R = (d_i \dots d_{i+L-1})$, $i = 1, \dots, N - L + 1$. Let τ be the maximal score we found. We calculate the probability to get a maximal score T higher than τ, $P(T \geq \tau)$ in a random sequence of length N.

We divide this calculation into two parts:

1. Find the set of all sequences of length L, $A = A(\tau) \subset \Sigma^L$, which have a score higher than τ, $A = \{R = (r_1, \dots, r_L), r_i \in \Sigma | score(R) \geq \tau\}$.
2. Let H be the number of occurrences of sequences from A in a DNA sequence of length N. We want to calculate the probability $P(H > 0)$ to observe at least one sequence from the set A in a random sequence of length N; clearly, $P(H > 0) = 1 - P(H = 0)$.

$P(H > 0)$ is the p-value produced by our algorithm, the probability of finding a higher score in a random sequence.

2.3. *Finding A*

Let K be the size of A, $K = |A|$. We first estimate K by a method taken from Staden (1989) (see 2.3.2). This estimation gives very tight lower and upper bounds of K. If K is not too large, we enumerate all sequences in A by a branch and bound algorithm in time $O(K)$ (see 2.3.1). However, sometimes K is huge, and then enumerating K sequences will take too much time. In this case we use an upper bound on K.

2.3.1. Branch and bound algorithm. Here we describe the branch and bound algorithm used to enumerate A, the set of all targets with score above the threshold. For this we need a new definition. Let $R = (r_1, \dots, r_L)$ be a sequence. We define $R' = \cup_{i=1}^{L} (r_1, \dots, r_{i-1}, r'_i, r_{i+1}, \dots, r_L)$, where r'_i is the next best nucleotide to r_i in the $i - th$ position. Coping with ties inside columns of M is done by enumerating them in ascending indexing order. For this, we need a more delicate definition given by

$$r'_i = \begin{cases} r_i, & M_{r_i,i} = \min\{M_{j,i} | j \in \Sigma\}, r_i = \max\{j | M_{r_i,i} = M_{j,i}\} \\ \arg\max\limits_{j \in \Sigma,} \{M_{j,i} | \{M_{j,i} < M_{r_i,i}\} \cup \{M_{j,i} = M_{r_i,i}, j > r_i\}\}, & \text{otherwise.} \end{cases} \quad (2)$$

Notice that if there are multiple maxima, we always assume that arg max takes the smallest index among them.

For a set of sequences B, we define $B' = \cup_{R \in B} R'$. We define also $B(\tau) = \cup_{R \in B, score(R) \geq \tau} R$ (the set of sequences with score above τ). The algorithm is as follows:

1. Start with the consensus sequence $A_0 = R^*$, by taking at each column the best nucleotide. $R^* = (r_1^*, \dots, r_L^*)$ where r_i^* is defined by $r_i^* = \arg\max_{j \in \Sigma} M_{j,i}$.
2. For i from 1 to $3L$ do: $A_i = A_{i-1} \cup A'_{i-1}(\tau)$
3. Output $A \equiv A_{3L}$

Note that the fact that we perform $3L$ steps guarantees that every one of the 4^L sequences might be reached ($3L$ is the maximal possible "distance" between the consensus sequence and the worst scoring sequence if we allow only operations of the kind $r_i \to r_i'$, and clearly this is the maximal such "distance" between any two sequences).

Actually, we usually do not have to do the for-loop until the end. If at some step of the for-loop we did not gain any new sequence (that is $A'_{i-1}(\tau) \subset A_{i-1}$), we can stop immediately.

The algorithm can be implemented using a hash table containing A_{i-1} in every step. In addition, a list of the indices of the occupied cells in the hash table is maintained. At step i, we need to compute $A'_{i-1}(\tau)$. We can enumerate the list of indices and generate the L sequences of R' for each $R \in A_{i-1}$. However, since $A'_{i-1}(\tau) \subset (A_{i-1} \cup (A_{i-1} \setminus A_{i-2})'(\tau))$, we need to enumerate only indices that were added in the previous step. We insert to the hash table only sequences with score above τ, and use the hash to check whether they are already stored. Notice that we enumerate only the K sequences with score above the threshold τ, and each one can be expanded once to at most L sequences. Therefore, the overall time complexity is $O(KL)$.

2.3.2. Calculating bounds on K. This method is taken from Staden (1989). It assumes that the random model of the DNA is composed of independent letters (not necessarily uniformly distributed). Given a score matrix, $M_{4\times L}$, every sequence R of length L ($R = r_1 \ldots r_L$, $r_i \in \{1, 2, 3, 4\}$) gets a score:

$$\mathit{score}(R) = \sum_{i=1}^{L} M_{r_i,i}.$$

We would like to look at the distribution of $\mathit{score}(R)$ over all sequences of length L. It can be described as a sum of L independent random variables, $x_1, \ldots, x_L$, where

$$x_i = \begin{cases} M_{1,i}, & p_1 \\ M_{2,i}, & p_2 \\ M_{3,i}, & p_3 \\ M_{4,i}, & p_4 \end{cases} \tag{3}$$

and where $p_1, \ldots, p_4$ are the given probabilities of the symbols in Σ.

The probability to get a score higher than τ in a specific position is the probability to see there at least one target from A, which is

$$P(A) = \sum_{j=1}^{K} \prod_{n=1}^{L} P(d_{i+n-1} = r_n^j) \tag{4}$$

where $A = \{R^1, \ldots, R^K\}$, $R^j = (r_1^j, \ldots, r_L^j)$. Recall that d_i is the DNA letter in position i (see 2.2). Using (3), it can be calculated by

$$P(A) = P\left(\sum_{i=1}^{L} x_i > \tau\right). \tag{5}$$

Notice that if we change x_i's definition by keeping the $M_{i,j}$'s and setting $p_1 = \cdots = p_4 = \frac{1}{4}$ in Equation (3) (even if the true random model is not uniform), then K, the number of sequences of length L with a score higher than τ, is equal to

$$K = P\left(\sum_{i=1}^{L} x_i > \tau\right) \times 4^L. \tag{6}$$

Assume the score matrix contains only natural values. Then the random variables $x_1, \ldots, x_L$ can have only natural values. We define for each x_i its probability-generating function: $G_i(x) = \sum_{j=1}^{4} p_j x^{M_{j,i}}$. The coefficient of x^N gives the probability that x_i has exactly the value N. Now, since we assume

$x_1, \ldots, x_L$ are independent, the generating function of the probability distribution of $x_1 + \cdots + x_L$ is $G_{x_1+\cdots+x_L}(x) = G_1(x)G_2(x)\ldots G_L(x)$. By multiplying the generating functions, we get the probability of every possible score (we need to calculate the coefficients of the multiplication).

Define for a polynomial $g(x) = \sum_{i=1}^{deg(g)} g_i x^i$ its *sparseness*, or the number of its nonzero coefficients to be $SP(g) = \sum_{i=1}^{deg(g)} 1_{g_i \neq 0}$. Then the computation of the coefficients of the multiplication of two polynomials $g(x), h(x)$ requires $O(SP(g) \cdot SP(h))$ operations. Denote $T = \max\{\sum_{i=1}^{L} M_{r_i,i} | r_i \in \{A, C, G, T\}, i = 1, \ldots, L\}$. Then the complexity of the multiplication is $O(4(L-1)T)$: we have $L-1$ multiplications of a polynomial with degree less than T, with a polynomial with at most four nonzero entries.

However, our score matrix M doesn't necessarily contain only natural values. We would like to approximate K, the number of targets with a score higher than τ. We can have an upper bound and a lower bound for K in the following way:

To get an upper bound for K we can get a natural score matrix M' by $M'_{i,j} = \lceil M_{i,j} \cdot T \rceil$, where T is some large natural number. We use M' instead M in (3), and we calculate (exactly) the probability $p = P(x_1 + \cdots + x_L \geq \lfloor \tau \cdot T \rfloor)$ by the method of generating functions. If we use $p_1 = \cdots = p_4 = \frac{1}{4}$, then $K' = 4^L \cdot p$ is an upper bound for K: for every target $R = (r_1, \ldots, r_L)$ s.t. $score(R) = \sum_{i=1}^{L} M_{r_i,i} \geq \tau$, $\sum_{i=1}^{L} M'_{r_i,i} \geq \lfloor \tau \cdot T \rfloor$. The lower bound can be achieved in a similar way.

2.3.3. Dealing with both strands. Since *DNA* is double stranded, we should look for binding sites on both strands. We do it by simply scanning both strands and recording the best match to our $PSSM$. In order to give a p-value accounting for both strands, we simply assume we scan only the positive (5′) strand, but the set of "legal" sequences is extending by taking $A \cup A^{RC}$, where A^{RC} is defined as the set of all reverse-complements of sequences from A. In case we enumerate A, we simply add, right after step 3 in 2.3.1, another step:

4. $A = A \cup A^{RC}$

Note that here we also have to preform "unique" since both a sequence and its reverse complement could get a score above τ. (This occurs often for palindromic motifs). If we only estimate $P(A)$ or K, we can simply multiply the bounds we got in 2.3.2 by two. This suffices since clearly $|A \cup A^{RC}| \leq |A| + |A^{RC}| = 2K$ and $P(A \cup A^{RC}) \leq P(A) + P(A^{RC}) \leq 2P(A)$. (Remember that A consists of the K sequences with highest probabilities.)

2.4. Calculating $P(H > 0)$

2.4.1. A naive approximation. There are $N - L + 1$ positions in which the target sequence can appear. At each position, the probability of appearance is $P(A)$.

Assuming that all the positions are independent gives the geometric distribution approximation:

$$P(H = 0) \approx (1 - P(A))^{N-L+1}. \tag{7}$$

As is shown by Robin and Daudin (1999), the quality of this approximation varies and depends on the overlapping structure of the sequence.

2.4.2. Exact computation. Suppose K is small enough and we are given the set A of all sequences with score higher than τ, $A = \{R^1, \ldots, R^K\}$. Recall that H counts the number of appearances of any of them in a DNA sequence. We present here a recursive algorithm for calculating $P(H = 0)$. First, we define T_i to be the event of finding any of the K targets at position i and not finding any of them at $l < i$ (e.g., a target was found at i "for the first time"). Let $t_i = P(T_i)$; $P(H > 0) = \sum_{i=1}^{N-L+1} t_i$. The t_i's are calculated recursively; define I_i to be the event of finding any of the K targets at position i (not necessarily for the first time). In addition, for every $j = 1, \ldots, K$, define $I_i(j)$ to be the event of finding R^j at index i, and $T_i(j) = I_i(j) \bigcap T_i$. Let also $t_i(j) = P(T_i(j))$. Clearly, from the above definitions, it follows that

$$T_i = \bigcup_{j=1}^{K} T_i(j) \ , \ \ t_i = \sum_{j=1}^{K} t_i(j). \tag{8}$$

The solution will depend on the overlap patterns of pairs of our K sequences. Define for each pair of sequences R^j and R^m

$$\epsilon_i(m, j) = \epsilon_i(R^m, R^j) = \begin{cases} 1 & R^j \text{ overlaps } R^m \text{ at distance } i\ (r_1^j = r_{1+i}^m, \ldots, r_{L-i}^j = r_L^m) \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Then the following recursion holds (Blom and Thorburn, 1982):

$$t_i(j) = P(I_i(j)) \left(1 - \sum_{n=1}^{i-L} t_n\right) - \sum_{n=1}^{L-1} \sum_{m=1}^{K} t_{i-n}(m) P(I_i(j)/T_{i-n}(m)). \tag{10}$$

Define L vectors of size K, $\vec{Q}_1, \ldots, \vec{Q}_L$ by

$$\vec{Q}_n(m) = \prod_{j=L-n+1}^{L} P(x = r_j^m),\ m = 1, \ldots, K,\ n = 1, \ldots, L$$

where $\vec{Q}_n(m)$ is the probability that an n-letter string matches the n-letter suffix of R^m. Notice that $\vec{Q}_L(j) = P(I_i(j))$. Then $P(I_i(j)/T_{i-n}(m)) = \epsilon(m, j) \cdot \vec{Q}_n(j)$, and Equation (10) can be written in matrix form:

$$\vec{t_i} = diag(\vec{Q}_L) \left(\vec{1}_K - \sum_{n=1}^{i-L} 1_{K \times K} \vec{t_n}\right) - \sum_{n=1}^{L-1} diag(\vec{Q}_n) \epsilon_n^T \vec{t}_{i-n} \tag{11}$$

where ϵ_n is the matrix of overlaps at distance n, $\epsilon_n^T(i, j) = \epsilon_n(j, i)$, $\vec{t_i}$ is the vector of the $t_i(j)$'s, $\vec{1}_K$ and $1_{K \times K}$ are a vector and a matrix of all ones, respectively, and $diag(\vec{Q})$ is a diagonal matrix of size $K \times K$ with the elements of $\vec{Q}$ on the diagonal.

The initial conditions are $(i = 1, \ldots, L)$:

$$\vec{t_i} = \vec{Q}_L - \sum_{n=1}^{i-1} diag(\vec{Q}_n) \epsilon_n^T \vec{t}_{i-n}.$$

The first sum on the right side of Equation (11) can be easily computed, since all the rows of $1_{K \times K}$ are identical. Therefore, the major computational effort, when advancing one step in the recursion, is in calculating the second sum. This can be done efficiently if we compute the matrices $\epsilon_1, \ldots, \epsilon_{L-1}$ once, in advance. The exact performance depends on the structure of the ϵ_n's, but it is bounded by $O(LK)$: suppose ϵ_n^T has m different rows, $\vec{c}_1, \ldots, \vec{c}_m$. Every row specifies for all targets whether their last $L - n$ letters are identical to a specific sequence of length $L - n$ (unique to this row), which is the beginning of one of the targets. If $\vec{c}_j(i) = 1$, it means that target number i ends with the unique sequence of length $L - n$ of row $\vec{c}_j$. Then, we get that for every $k \neq j$, $\vec{c}_k(i) = 0$. Let $n_j,\ j = 1, \ldots, m$ be a count of the number of nonzero entries in each row. Then $\sum_{j=1}^{m} n_j \leq K$. It means that each of the ϵ_n's matrices can be stored in space $O(K)$. We need to multiply $\vec{c}_j$ and $\vec{t}_{i-n}$. The multiplication takes $O(n_j)$, and the multiplication of all different rows takes $O(\sum_{j=1}^{m} n_j) = O(K)$. The result is then multiplied by $diag(\vec{Q}_n)$, which also takes $O(K)$. Since we have $L - 1$ different matrices, the computation of one recursion step takes $O(LK)$, and thus the computation of the whole recursion takes $O(NLK)$.

To calculate the ϵ_n's in the beginning, we need to compare two lists of size K. The first contains the first n letters of all the sequences, and the second contains the last n letters of them. This can be done efficiently using the "meet in the middle" approach (Diffie and Hellman, 1977), often used in cryptology. The time complexity of the beginning step reduces to $O(LKlog(K))$.

The total complexity of the algorithm is $O(LK(N + \log(K)))$.

2.4.3. An upper bound for $P(H > 0)$. Suppose K is large and we are not given A, but a (tight) upper bound on it's probability: $\rho > P(I_i)$. Then to get an upper bound of $P(H > 0)$ we set $\epsilon \equiv 0$, (i.e., no overlaps between all the K sequences); see Section 2.4.4 for proof that this gives an upper bound. Substituting $\epsilon \equiv 0$ in Equation (10) gives

$$t_i = \rho \left(1 - \sum_{n=1}^{i-L} t_n\right), \tag{12}$$

where the initial conditions

$$t_1 = \cdots = t_L = \rho. \tag{13}$$

The solution can be represented using the roots of the characteristic polynomial:

$$p(x) = x^L - x^{L-1} + \rho. \tag{14}$$

The roots can be found numerically, by transforming to an eigenvalue problem. The eigenvalue problem can be solved, for example, using the QR algorithm (Francis, 1962a, 1962b). Usually $p(x)$ has only simple roots. In this case, the solution is

$$P(H = 0) = \sum_{j=1}^{L} u_j g_j^{N-L+1} \tag{15}$$

where $g_1, \ldots, g_L$ are the roots of $p(x)$, and $u_1, \ldots, u_L$ are determined by the initial conditions. If (14) has nonsimple roots, then there is a slight change in the solution (see Zuk *et al.* [2004] for details on solving the recurrence relations). So we have obtained easily an upper bound for $P(H > 0) = \sum_{i=1}^{N-L+1} t_i$. Time complexity: Finding the roots of the polynomial numerically depends only on L. The calculation of $P(H = 0)$ then takes $O(LlogN)$.

2.4.4. Getting an upper bound using $\epsilon \equiv 0$. Let $A \subset \Sigma^L$ be a general set of size K, $A = \{R^1, \ldots, R^K\}$ (A represents the set all sequences with a score higher than a given threshold). Assume there exists a nonoverlapping set $A' \subset \Sigma^L$ of size K, $A' = \{R^{1'}, \ldots, R^{K'}\}$ such that $P(x = r_n^{m'}) = P(x = r_n^m)$, for $m = 1, \ldots, K$, $n = 1, \ldots, L$ (i.e., each symbol of pattern $R^{i'}$ of A' has the same probability as the corresponding symbol of R^i). We want to show that

$$P'(H > 0) \geq P(H > 0) \tag{16}$$

where $P'(H > 0)$ is the probability to have at least one appearance of a sequence from A' in a random DNA sequence of length N and $P(H > 0)$ is the same for A.

Let $B(N), B'(N) \subset \Sigma^N$ be the set of sequences of length N containing at least one sequence from A and A', respectively. Now define the following mapping:

$$\begin{aligned} f &: B'(N) \to B(N) \\ f(\sigma) &= \delta \end{aligned} \tag{17}$$

where σ is a sequence of length N and δ is obtained from σ by replacing the first appearance of some $R^{i'} \in A'$ with $R^i \in A$ (note that the index i is kept). Our goal is to find a function $g : B'(N) \to B(N)$ which is mapped onto $B(N)$. We will build g dynamically, in the following process:

1. First, define g to be

$$g : B'(N) \to B(N)$$

$$g(\sigma) = \begin{cases} \sigma & \text{if some } R^j \text{ appears after the first } R^{i'}; \\ f(\sigma) & \text{otherwise.} \end{cases} \tag{18}$$

Note that here g is not necessarily mapped onto $B(N)$. For example, take $K = 1$, $S = |\Sigma| = 2$ (binary sequences), $L = 5$, $N = 7$. Take $R^{1'} = 01011$ (a nonoverlapping sequence), and $R^1 = 10111$. Then the sequence $\delta \in B(7), \delta = 1011110$ has no source. This is because $\delta \notin B'(7)$, and in its "potential" source, $\sigma = 0101110$, R^1 appears after $R^{1'}$ (and thus $g(\sigma) = \sigma$).

2. While (g is not mapped onto $B(N)$) do
   (a) Find $\delta \in B(N) \cap \overline{g(B'(N))}$.
   (b) Find $\sigma \in B'(N)$ such that $f(\sigma) = \delta$, by replacing the last R^i in δ with $R^{i'}$.
   (c) Change g by setting $g(\sigma) = f(\sigma)(= \delta)$.

If we consider our example, then taking $\delta = 1011110$, and replacing the last (and only) R^1 with $R^{1'}$ gives $\sigma = 0101110$, and we set $g(\sigma) = \delta$. Note that at the beginning σ is the only source of σ; therefore, after step $2(c)$, σ becomes "sourceless." Therefore, we will get to it sometime in step $2(a)$ and name it as the "new" δ. For this $\delta = 0101110$, we will find its own $\sigma = 0010110$, and set again $g(\sigma) = \delta$, and so on.

Clearly, after step 2 is finished, we get that g is mapped onto $B(N)$. Therefore, all we have to prove is that the while-loop in step 2 terminates after a finite number of steps. Note that during the process, in step $2(a)$, we will always find a δ (otherwise the condition of the while-loop is not satisfied).

It is less trivial to show that in step $2(b)$ we always get a σ such that $f(\sigma) = \delta$. For this we look at all the "sourceless" sequences δ which can be found in step $2(a)$. Claim: They all have the property that if they have a sequence from A' before the last R^i, then it overlaps this R^i. We will prove this claim by induction on the algorithm steps. Induction basis: It is easy to see that in the beginning the "sourceless" sequences don't have any sequence from A' before the last R^i (otherwise, they would have had a source: $g(\delta) = \delta$).

Induction step: Notice that all the changes to g are of the type: set $g(\sigma) = f(\sigma)$ instead of $g(\sigma) = \sigma$. Let δ be a sequence that had a source in the beginning (i.e., $g(\delta) = \delta$) and in step $2(c)$ became sourceless. Let $\sigma = f(\delta)$. Then σ was found to be sourceless in $2(a)$, and δ was set to be its source. By the induction assumption, σ has the property that if it has a sequence from A' before the last R^i, then it overlaps this R^i. We know that δ is obtained by replacing the last R^i in σ with $R^{i'}$. We need to show that δ also has the property that the only sequence from A' it has before the last R^i overlaps this R^i. Since the $R^{j'}$s are nonoverlapping, when we replace the last R^i in σ with $R^{i'}$, we get that there are no $R^{j'}$s before $R^{i'}$ in the created sequence (δ). This is because if there was a $R^{j'}$ that overlapped R^i, it had been "ruined" by $R^{i'}$. Now it is enough to show that the last R^i in δ overlaps with $R^{i'}$. If there is no R^i after $R^{i'}$, then we would have had in the beginning $g(\delta) = \sigma$, and σ wasn't sourceless. Since σ has its last R^i in the position of $R^{i'}$ in δ, there cannot be some R^i in δ which is after and not overlapping $R^{i'}$. Thus, we get that the last R^i in δ overlaps $R^{i'}$, and the property holds here. Thus, the claim is proven by induction.

Therefore, it follows that if a sequence becomes "sourceless" during the process, it has the property that the only $R^{j'}$s it has before the last R^i overlap this R^i. Thus, when applying $2(b)$, we get a sequence σ such that $f(\sigma) = \delta$, because after the replacement, $R^{i'}$ is the first appearance of a sequence from A' in σ.

Having confirmed the validity of step $2(b)$, we note also that during the whole process, $g(\sigma)$ is either σ or $f(\sigma)$, $\forall \sigma \in B'(N)$. Each time we perform step $2(c)$, we remove a "sourceless" sequence δ, thus reducing the number of "sourceless" sequences by one. If σ is the only source of σ, than we add a "sourceless" sequence σ. Thus the number of "sourceless" sequences is either reduced by one or kept. However, the number can be kept only if g has a self-loop $g(\sigma) = \sigma$, and this self-loop is ruined in step $2(c)$. Note also that no self-loops can be generated in the process. Therefore, there can be only a finite number of steps on which the number of "sourceless" sequences is kept. But clearly there is only a finite number of steps on which this number is reduced by one. Therefore, we finally reach zero "sourceless" sequences in a finite number of steps, and g is mapped onto $B(N)$.

Note that since we have kept the indices i, we get that $P(\sigma) = P(g(\sigma))$, $\forall \sigma \in B'(N)$, and therefore

$$P(B'(N)) = P'(H > 0) \geq P(H > 0) = P(B(N)). \tag{19}$$

3. EXPERIMENTAL RESULTS AND DISCUSSION

Here we compare the results of our algorithm with MatInspector (Quandt *et al.*, 1995), a commercially available software that looks for putative binding sites in DNA sequences using PSSMs. The comparison is based on the Saccharomyces Cerevisiae Promoters Database (SCPD; Zhu and Zhang, 1999), which contains a list of known transcription factors' binding sites, and their location in the *Saccharomyces* *cerevisiae* genome. In Section 3.1 we describe the data we use and how we measure the performance of our algorithm on it. In Section 3.2 we describe how we compare our results to MatInspector's results. The comparison is done with different interpretations of the output of MatInspector software.

3.1. Saccharomyces cerevisiae *genome*

We used the *Saccharomyces cerevisiae* Promoters Database (SCPD; Zhu and Zhang, 1999), which contains a list of known transcription factors' binding sites and their location in the *Saccharomyces cerevisiae* genome. It contains the PWMs of 24 transcription factors. We extracted the upstream sequences of all genes that are known to be bound by at least one of these 24 transcription factors. There are 135 such genes. We used the upstream regions given in SCPD; all of them end right before the translation start codon (ATG), and their length is between 500 to 800 bases.

We computed the p-values (for $H > 0$) for each of the 24 transcription factors on each of the 135 genes (we use here the upper bounds of the p-values; see Section 2.4.3). However, here we deal with multiple comparisons; it is expected that on average 1.2 of 24 random, identically distributed p-values will have a value below 0.05. Suppose we choose 0.05 as the p-values cutoff value. Thus, on average, at least one transcription factor would be falsely identified as statistically significant; i.e., it will be a false positive. To deal with the problem of multiple comparisons, we used the method of Benjamini and Hochberg (1995), which controls the false discovery rate (FDR method). According to the method, the transcription factors are ordered in an increasing order of their p-values, and a parameter Q, which controls the fraction of false positives, is set. We then identify the minimal index j such that for all $i > j$ we have p-value$(i) \geq \frac{j \cdot Q}{24}$. The list of transcription factors with indices $\leq j$ are the output of the procedure. This procedure ensures that the output list of transcription factors has an expected fraction of false positives which is lower than Q. We applied it (for each of the genes) to the 24 p-values of the transcription factors, to get the statistically significant putative binding sites for every gene (see Fig. 1).

For a specific gene, a transcription factor is said to be *positive* at a certain value of Q if it was found to be statistically significant by the FDR method using this value of Q. It is said to be *true positive* (TP) if it is positive and it is also known to bind this gene (by SCPD). It is *false positive* (FP) if it is positive, but it isn't known to bound this gene. In a similar way we define *true negatives* (TN) and *false negatives* (FN). We present the results in a ROC (receiver operating characteristic) curve. This curve describes the relative amount of correct predictions as a function of the relative amount of incorrect ones, or the tradeoff

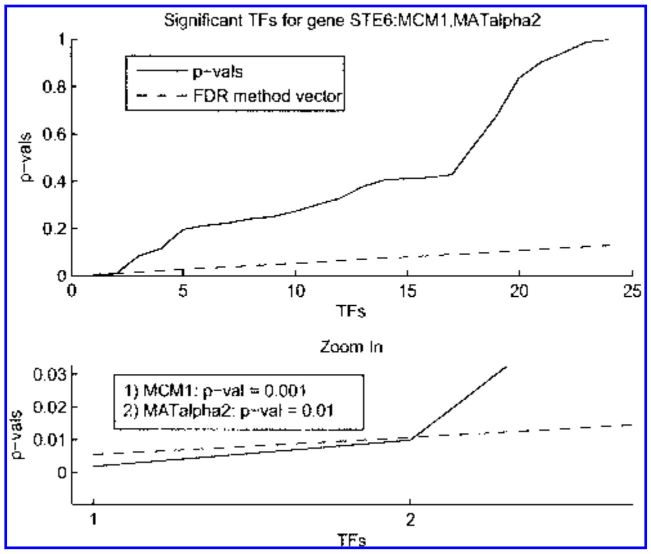


FIG. 1. Upper plot: Applying the FDR method on the p-values of the 24 transcription factors, for gene STE6. Here we choose $Q = 0.13$, which outputs two "real" transcription factors. The two statistically significant transcription factors are MCM1, MATalpha2; these are exactly the transcription factors which are known to bind STE6 (SCPD). **Lower plot:** A zoom-in of the upper plot showing the two statistically significant transcription factors p-values.

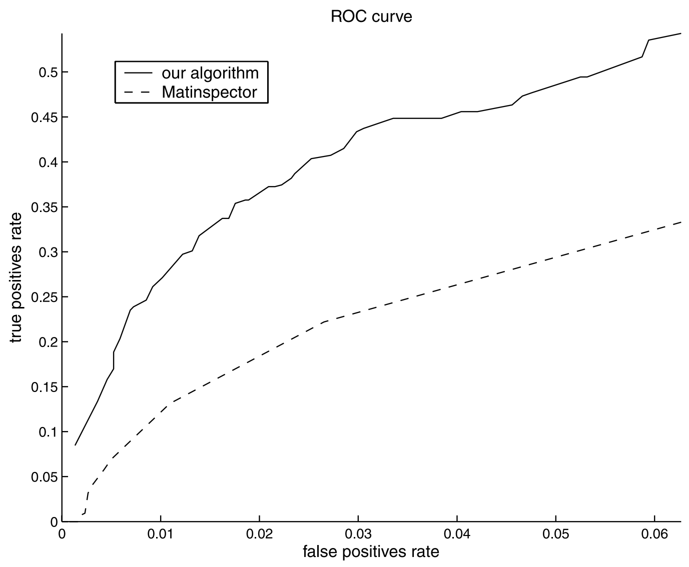


FIG. 2. ROC curves showing ability to identify binding sites in promoters of genes from SCPD. The x-axis shows the *false positives rate* FP/(FP+TN); the y-axis shows the *true positives rate*, TP/(TP+FN). The results shown are average values on all 135 genes. The solid curve shows our results in a range of values of the FDR method parameter, $Q \in [0.01, 0.45]$. The dashed curve shows MatInspector results in a range of thresholds on the number of matches (1–150).

between true positives and false positives, averaged on all 135 genes. Clearly, a low Q value ensures a low number of false positives, but at the cost of a low number of true positives. Hence, we used a range of values of the FDR method parameter, $0.01 < Q < 0.45$, to select an optimal working point. The results are shown in Fig. 2.

3.2. Comparison with MatInspector

Given a PWM and a DNA sequence, MatInspector outputs a list of all the matches of this PWM in the promoter. MatInspector computes a PSSM out of the PWM (not by taking the log likelihood ratio). A match is a position in the sequence in which the score according to the PSSM is above a certain threshold. MatInspector recommends to use its optimized score threshold (a different score threshold for every PSSM), which is computed by MatInspector "in a way that a minimum number of matches is found in non-regulatory test sequences." However, it is not published how this threshold is computed for every PSSM. We use the same 24 PWMs as in our search for transcription factors binding sites (see Section 3.1).

3.2.1. Our $P(H > 0)$ and FDR versus MatInspector's optimized score threshold and a threshold on the number of matches. In order to decide which of the transcription factors has a number of matches which is statistically significant, we decide on a threshold on the number of matches. We used a range of threshold values (which play the role of Q in our method) to find the optimal working point. When a transcription factor has a number of matches which is higher than the threshold value (the threshold on the number of matches), we decide that it is statistically significant. We made this for the same 135 genes and 24 PWMs. A comparison between MatInspector results and our results is shown in Fig. 2.

It can be seen in Fig. 2 that our algorithm is significantly more precise in searching for putative binding sites in genes from SCPD than MatInspector (in the sense of achieving a better true positives versus false positives tradeoff). We repeated our analysis using MatInspector's PSSMs (instead of log-likelihood ratio PSSMs), and our results were very similar. Thus, we conclude that our higher performance is not due to differences in the PSSMs.

3.2.2. Using MatInspector's RE values. MatInspector provides an additional parameter for every PWM. It is called *RE value,* and it is an estimation of the mean number of matches of a PWM in a random

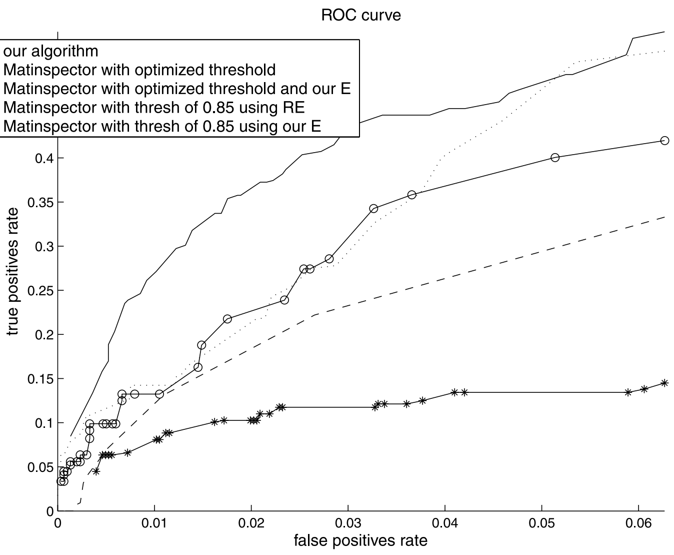


FIG. 3. ROC curves showing ability to identify binding sites in promoters of genes from SCPD. The x-axis shows the *false positives rate* FP/(FP+TN); the y-axis shows the *true positives rate*, TP/(TP+FN). The results shown are average values on all 135 genes. The solid curve shows our results in a range of values of the FDR method parameter, $Q \in [0.01, 0.45]$ (same as in Fig. 2). The dashed curve shows MatInspector results using the optimized score thresholds suggested by MatInspector in a range of thresholds on the number of matches (3–150). The solid curve with stars shows MatInspector results with a constant score threshold of 0.85 using the RE value. The range of values for the threshold of RE/(num matches) to be considered a match is $[0.0067, 0.06]$. The solid curve with circles shows MatInspector results with a constant score threshold of 0.85, using our estimation of E instead of MatInspector's RE value. The range of values for the threshold of E/(num matches) to be considered a match is $[0.0067, 0.11]$. The dotted curve shows MatInspector results with the optimized score thresholds suggested by MatInspector, using our estimation of the mean number of matches. The range of values for the threshold of E/(num matches) to be considered a match is $[0.0067, 0.1]$.

sequence of length 1,000. A match here is a score higher than 0.85 according to MatInspector's PSSMs (MatInspector's PSSMs have scores in the range $[0, 1]$). We evaluated MatInspector results using the RE values, too. We ran the MatInspector software on the same 135 promoters and 24 PWMs. We used a threshold of 0.85 on the PWMs scores in order to use the RE parameter in a meaningful way. In order to decide which of the transcription factors has a number of matches which is statistically significant, we calculated the ratio between the RE value of each PWM (out of the 24) and the number of matches of this PWM (both are given by MatInspector). When this ratio is lower than a certain threshold, we decide that this PWM is statistically significant in the given DNA sequence (we used a range of threshold values).

We repeated this analysis using our computation of the mean of the number of matches of every PWM (see Section 3.2.3). We used the same PSSMs as MatInspector, and the match score thresholds were all 0.85. We used our estimation to the mean of the number of matches instead of the RE values given by MatInspector. As can be seen in Fig. 3, our estimation of the mean number of matches again improves the performance.

3.2.3. Using MatInspector's optimized score thresholds with our calculation of the mean number of matches. Finally, we used MatInspector optimized score thresholds with our estimation of the number of matches of every PWM. For each PWM, we calculated the mean number of matches, where a match is having a score higher than MatInspecor's optimized score threshold for the PSSM (MatInspector's PSSM) of this PWM. This was done because we saw that our estimation of the mean of the number of matches improves the performance in the case of a constant score threshold (0.85). In addition, it is obvious that taking a constant score threshold on all the PSSMs doesn't give the best results. Since different PWMs are defined with different stringency, each PWM should have its own score threshold. However, MatInspector software has a limitation on the definition of the score thresholds. The only possibility to choose different score thresholds for different PWMs is by choosing the optimized score thresholds, which are determined by an unpublished procedure of MatInspector (and there is a possibility to choose the optimized score threshold $\pm$ a constant, the same constant for all the matrices). The other possibility is to choose a constant score threshold for all PWMs (we did this with a score threshold of 0.85). Choosing the optimized score thresholds combined with using our estimation of the mean number of matches gave the highest performance of all MatInspector's runs. Even so, our method shows the highest results in terms of true-positives rate versus false-positives rate tradeoff. A comparison of all the results is shown in Fig. 3.

3.3. Our calculation of the mean number of matches

Let

$$y_i = \begin{cases} 1 & \text{there is a match in position } i \ (\mathit{score}(D_i \ldots D_{i+L-1}) \geq \text{threshold, or } I_i \text{ occurred}). \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

for $i = 1, \ldots, N - L + 1$. Then the mean of the number of matches,

$$E = E(H) = E\left(\sum_{i=1}^{N-L+1} y_i\right) = \sum_{i=1}^{N-L+1} E(y_i) = (N - L + 1) \cdot E(y_1)$$

and

$$E(y_1) = \mathit{Prob}(y_1 = 1) = P(A).$$

Thus, in order to get an accurate estimation of E, we must compute $P(A)$ very accurately. In Section 2.3.2, we describe how it is done.

3.4. Reasons for differences in performance

There is a central difference between our approach and MatInspector's. Given a promoter sequence and a PWM, we look at the best match of this PWM in the sequence and ask what is the probability to find a match of such quality (score) in a random promoter. Intuitively, we ask, "Does the TF bind the promoter region in at least one position, or not?" MatInspector refers to the number of matches of the PWM in the sequence, where a match is defined as a position with a score higher than a certain threshold. MatInspector then uses an estimation of the mean number of matches. We believe that this difference is a main cause of our higher performance. However, it might be also due to MatInspector's choice of the optimal score thresholds (whose calculation is not published).

As was described in 3.2.2, MatInspector uses an estimation of the mean number of matches of a PWM in a random sequence, the RE value. However, the calculation of the RE value is not published. We present in 3.3 our calculation of the mean number of matches. As was shown in Fig. 3, using our calculation instead that of MatInspector improves significantly MatInspector's results. This suggests that we calculate the mean number of matches more accurately than MatInspector. This is another reason for MatInspector's lower performance.

3.5. Synthetic data

We continue with evaluating our methods on synthetic data. It enables us to measure our performance on many (100) datasets and to calculate the mean and standard deviation of our performance (which is measured as the tradeoff between true positives and false positives). It could be that the real dataset we used (SCPD) is exceptional for some reason, so in this way we get a more reliable picture of our results.

We built 100 datasets. Each dataset is similar in its structure to the real dataset we used (SCPD). All datasets consist of 135 promoters of length 500. The promoter sequences were sampled from a 3-order Markov model background distribution, trained on *Saccharomyces cerevisiae* promoter regions. In the promoter regions we planted motifs which are known to be true binding sites of the 24 transcription factors from SCPD. Each motif was sampled uniformly from the list of known binding sites of a specific transcription factor. The number of motifs in every promoter is similar to that in the real dataset. We ran our method on these 100 datasets and compared the results to our results on the real dataset (Fig. 4).

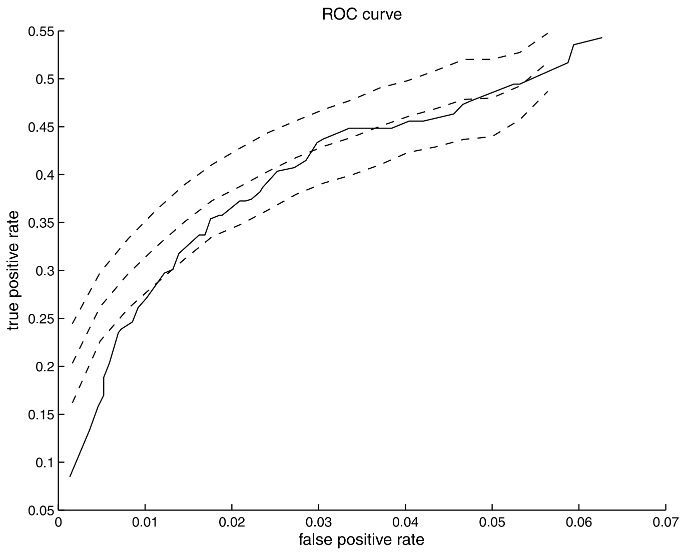


FIG. 4. ROC curves showing ability to identify binding sites in promoters of genes from SCPD. The x-axis shows the *false positives rate* FP/(FP+TN), the y-axis shows the *true positives rate*, TP/(TP+FN). The results shown are average values on all 135 genes. The solid curve shows our results in a range of values of the FDR method parameter, $Q \in [0.01, 0.45]$. The dashed curves show the mean ± standard deviation of the 100 ROC curves of the synthetic datasets.

It can be seen in Fig. 4 that our performance on the SCPD database is similar to our performance on synthetic datasets. It means that our high performance on the SCPD database is not incidental, but it reflects the power of our method. It also indicates that the SCPD database is "normal," for example in the sense that there are not many binding sites which are not listed there (had there been, our performance on it would have been worse than that on the synthetic data).

3.6. Finding a common TF in a group of genes

Suppose we are given a set of promoters of genes, G, which are suspected to be coregulated; i.e., they might have a common transcription factor that regulates them. Let n be the size of G. Let F be a set of PSSMs of known transcription factors. Let m be the size of F. We describe here a method to search for a common transcription factor in this set of genes using our probabilistic approach.

First we compute p-values for every pair of PSSM and gene. Then for every PSSM $i = 1, \ldots, m$ we have n p-values, $p_{i,1}, \ldots, p_{i,n}$. Our null hypothesis is

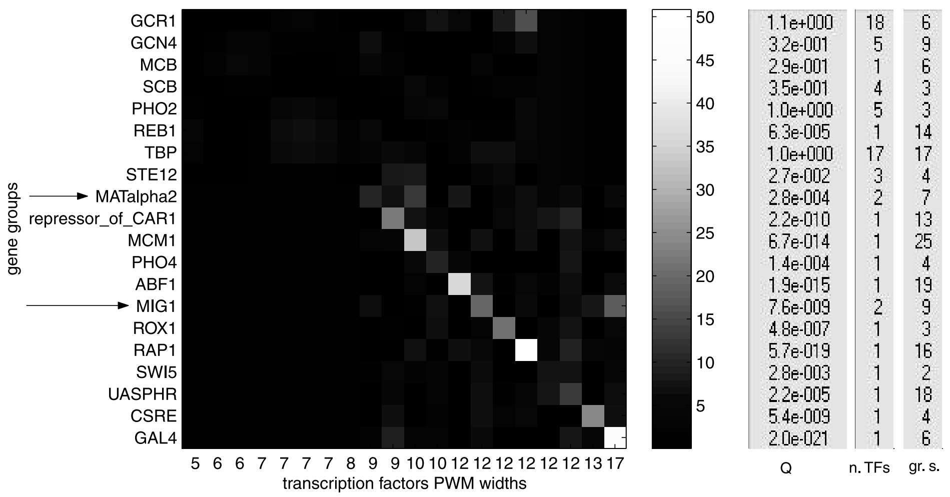
$$\forall i \in \{1, \ldots, m\}.\forall j \in \{1, \ldots, n\}.p_{i,j} \sim U[0, 1].$$

Let $score(i) = \prod_{j=1}^{n} p_{i,j} = s_i$. This score represents the plausibility of the transcription factor to be a common regulator of the gene group. The lower the score, the more plausible it is. We calculate this score for every $M \in F$, $s_1, \ldots, s_m$. We now can compute the probability $p_i = P(score(i) \leq s_i)|$ null model) for every $M \in F$. The density function of the product of n uniform independent random variables on the interval [0, 1], $x_1, \ldots, x_n$ is (Weisstein, 1999)

$$P_{x_1 \ldots x_n}(t) = \frac{(-1)^{n-1}}{(n-1)!}(ln(t))^{n-1}.$$

In this way, we get a p-value for every transcription factor for being a common regulator of the gene group. We use the FDR method on the set of m p-values to get the statistically significant transcription factors for the group of genes. If there are statistically significant transcription factors, they are suspected to be common regulators for this group of genes.

We applied this on groups of coregulated genes taken from SCPD. We took the group of all genes that a specific transcription factor is known to bind. There are 20 transcription factors in SCPD that have known



Q	n. TFs	gr. s.
1.1e+000	18	6
3.2e-001	5	9
2.9e-001	1	6
3.5e-001	4	3
1.0e+000	5	3
6.3e-005	1	14
1.0e+000	17	17
2.7e-002	3	4
2.8e-004	2	7
2.2e-010	1	13
6.7e-014	1	25
1.4e-004	1	4
1.9e-015	1	19
7.6e-009	2	9
4.8e-007	1	3
5.7e-019	1	16
2.8e-003	1	2
2.2e-005	1	18
5.4e-009	1	4
2.0e-021	1	6

FIG. 5. Plot of $-\log(\text{p-values})$ obtained for 20 transcription factors for 20 groups of genes. Each group of genes contains the known targets of one of the transcription factors. Each row corresponds to a gene group, labeled by the name of their common transcription factor. Each column corresponds to a transcription factor; the numbers under the columns are the widths of their PWMs. The columns are placed in an increasing order of the widths, and the gene groups appear in the same order as the transcription factors. The Q-values that are listed give for each gene group the minimal value of the Q parameter needed in order to get the common transcription factor of this gene group as statistically significant (when using the FDR method). The numbers next to the Q-values, labeled "n. TFs" shows the number of transcription factors that are reported as statistically significant by the FDR method when the minimal Q value is used for the corresponding gene group. The size of each gene group is shown in the column labelled by "gr. s."

PWM, which are also known to bind to more than one gene. Thus, we work on 20 groups of coregulated genes. It can be seen in Fig. 5 that in most of the gene groups the transcription factor to which we assign the lowest p-value is their known common transcription factor (the values on the diagonal are usually the highest of all the values in a row). The gene groups for which this doesn't hold can be divided into two classes:

1. Gene groups whose known common transcription factor has a short PWM (of width between 5 and 8). When a PWM is short, it can't get a very low p-value on a specific gene. For example, suppose the width of a PWM, M, is 6, the length of the gene's promoter is 500, and that the promoter contains a position with the maximal possible score for M. A rough estimation for the probability to get such a maximal score is (see Section 2.4.1)

$$1 - \left(\frac{4^6 - 1}{4^6}\right)^{500} = 0.11.$$

   This means that the minimal p-value of this PWM on each gene is approximately 0.1, and we get that $p_{M,1}, \ldots, p_{M,n} \geq 0.11$. It can be seen in Fig. 5 that the left side of the square is darker than the right side (higher p-values). This is because the transcription factors are ordered in an increasing order of their PWM's widths, and the first nine transcription factors have widths between 5 and 9, which result in higher p-values.
2. Gene groups that contain a subgroup of genes with a different additional common transcription factor. This is the case in two groups of the genes (by arrows in Fig. 5): the identified genes whose common regulator is MATalpha2 (all the genes but one are also regulated by the transcription factor MCM1) and the genes whose common regulator is MIG1 (a subgroup of the genes are regulated by the transcription factor GAL4).

This shows that the p-values we calculate for a transcription factor to be bound to different genes can be used to estimate the p-value for a transcription factor to be a common regulator of a group of genes. As was

Table 1. Comparison Between Our Results and PRIMA's, on a Set of 103 Human Promoters Corresponding to E2F Target Genes Reported by Ren *et al.* (2002)[a]

Transcription factor	*PRIMA's analytical score*	*We found as statistically significant*	*Our p-value*	*Q parameter*
E2F	1.9×10^{-10}	$+$	8.5×10^{-6}	7.2×10^{-5}
NF-Y	1.7×10^{-14}	$+$	7.4×10^{-37}	4.8×10^{-35}
NRF	3.1×10^{-4}	$+$	7.2×10^{-5}	5.1×10^{-4}
CREB	2.5×10^{-5}	$-$	0.93	1.13

[a]PRIMA used 107 PWMs, and we used 197 PWMs (both from the TRANSFAC database, Wingender *et al.* [2001]). PRIMA found four significantly enriched PWMs. Three of them, E2F, NF-Y, and NRF where found to be statistically significant also by our method. Indicated for each one are the analytical score given by PRIMA, our p-value for this transcription factor for being a common regulator, and the minimal value of the Q parameter needed in order to get this transcription factor as statistically significant by our method.

shown here, this also gives good results. This analysis can be also applied to clusters of coexpressed genes, resulting from microarray experiments, which are usually suspected to have common regulators.

3.6.1. Comparison with PRIMA. We compared our method with PRIMA, a program for searching for common transcription factors in a set of genes using PWMs. PRIMA uses a big background set of promoters and compares the enrichment of the PWMs in the given set with that in the background set. Elkon *et al.* (2003) present PRIMA's results for several clusters of genes. One of these clusters is a set of 103 human promoters corresponding to E2F target genes reported by Ren *et al.* (2002). PRIMA scanned this set with 107 PWMs from the TRANSFAC database. It found E2F to be significantly enriched, as were three other transcription factors: NF-Y, CREB, and NRF. We scanned the same group of promoters with 197 PWMs from the TRANSFAC database (Wingender *et al.*, 2001). We calculated a p-value for every transcription factor for being a common regulator of the gene group as was described in 3.6. The comparison between our results and PRIMA is presented in Table 1.

As listed in Table 1, we found the transcription factor E2F to be statistically significant for its target genes (with Q parameter of 7.2×10^{-5} and p-value of 8.5×10^{-6}). This means that for this set of genes our method finds the true common transcription factor. In addition, our method found the transcription factors NF-Y and NRF to be statistically significant, in accordance with PRIMA's results. This strengthens the reliability of both methods. Notice that PRIMA uses additional information, a big background promoter set, while we use the given promoter set alone. Our method didn't find the transcription factor CREB to be statistically significant. It might be because its PWM is short (its width is 8). As was described in 3.6, our method might miss transcription factors with short PWMs.

4. SUMMARY

In this paper, we have presented a method for searching for putative transcription factors' binding sites in promoter sequences and estimated their statistical significance. Many of the new methodologies search for a common binding site in a group of genes (for example, Elkon *et al.* [2003], Hughes *et al.* [2000]). A search for a binding site in a single promoter sequence is more suceptible to noise, raising the need for a subtle analysis. The accuracy of the p-values of our method enables us to perform a better search for binding sites in single promoters. Our ability to correctly predict putative binding sites is significantly higher than that of MatInspector, which also searches for binding sites in single promoters. In addition, as was shown in 3.6, given a group of genes which are suspected to be coregulated, our method can be used to estimate the statistical significance of a transcription factor to be a common regulator of this group. Thus we can exploit the sequence information of several genes together.

This work can be extended in several directions. First, the methods we have developed can be extended to searching for multiple occurrences of a motif in a promoter sequence, with calculation of a p-value for the number of occurrences found. Second, we have concentrated on searching for a motif of a single transcription factor. However, it is known that in many cases transcription factors cooperate with each other and form combinatorial transcriptional regulation. Thus, an interesting direction is to search for motif combinations and calculate their statistical significance. Third, an important challenge is to integrate our method with additional data in order to improve the tradeoff between true positives and false positives. For example, it can be combined with gene expression data as was suggested in Section 3.6, or gene homology data, (see, for example, Loots *et al.* [2002]). The binding sites found using our methods can be further investigated using several analysis techniques. For instance, Pilpel *et al.* (2001) use gene expression data to detect cooperation between pairs of transcription factors. In addition, we are developing user-friendly software which will be publicly available.

ACKNOWLEDGMENTS

The authors wish to thank Ido Kanter for many useful conversations and original ideas relating to this work and Michael Lassig for helpful discussions. This work was supported in part by the European Community's Human Potential Programme under contract HPRN-CT-2002-00319, STIPCO; the NIH, under grant #5 P01 CA 65930-06; the Clore Fellowship (for G.G.), the Minerva Foundation, and the Germany–Israel Science Foundation (GIF).

REFERENCES

Aerts, S., Thijs, G., Coessens, B., Staes, M., Moreau, Y., and Moor, B.D. 2003. Toucan: Deciphering the cis-regulatory logic of coregulated genes. *Nucl. Acids. Res.* 31(6), 1753–1764.

Bailey, T., and Elkan, C. 1995. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning* 21, 51–80.

Benjamini, Y., and Hochberg, Y. 1995. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J. R. Statist. Soc.* 57(1), 289–300.

Blom, G., and Thorburn, D. 1982. How many random digits are required until given sequences are obtained? *J. Appl. Prob.* 19, 518–531.

Diffie, W., and Hellman, M.E. 1977. Exhaustive cryptanalysis of the nbs data encryption standard. *IEEE Computer* 10(6), 74–84.

Elkon, R., Linhart, C., Sharan, R., Shamir, R., and Shiloh, Y. 2003. Genome-wide *in silico* identification of transcriptional regulators controlling the cell cycle in human cells. *Genome Res.* 13, 773–780.

Francis, J.G.F. 1962*a*. The qr transformation part i. *Computer J.* 4, 135–148.

Francis, J.G.F. 1962*b*. The qr transformation part ii. *Computer J.* 4, 332–345.

Hughes, J.D., Estep, P.W., Tavazoie, S., and Church, G.M. 2000. Computational identification of cis-regulatory elements associated with groups of functionally related genes in *Saccharomyces cerevisiae*. *J. Mol. Biol.* 296, 1205–1214.

Loots, G., Ovcharenko, I., Pachter, L., Dubchak, I., and Rubin, E. 2002. rVista for comparative sequence-based discovery of functional transcription factor binding sites. *Genome Res.* 12(5), 832–839.

Pilpel, Y., Sudarsanam, P., and Church, G. 2001. Identifying regulatory networks by combinatorial analysis of promoter elements. *Nature Genet.* 29(2), 153–159.

Quandt, K., Frech, K., Karas, H., Wingender, E., and Werner, T. 1995. MatInd and MatInspector: New, fast and versatile tools for detection of consensus matches in nucleotide sequence data. *Nucl. Acids. Res.* 23, 4878–4884.

Ren, B., Cam, H., Takahashi, Y., Volkert, T., Terragni, J., Young, R., and Dynlacht, B. 2002. E2F intergrates cell cycle progression with DNA repair, replication, and G_2/M checkpoints. *Genes and Dev.* 16, 245–256.

Robin, S., and Daudin, J.J. 1999. Exact distribution of word occurrences in a random sequence of letters. *J. Appl. Prob.* 36, 179–193.

Staden, R. 1989. Methods for calculating the probabilities of finding patterns in sequences. *CABIOS* 5(2).

Thijs, G., Marchel, K., Lescot, M., Rombauts, S., Moor, B.D., Rouze, P., and Moreau, Y. 2002. A Gibbs sampling method to detect overrepresented motifs in the upstream regions of coexpressed genes. *J. Comp. Biol.* 9(2), 447–464.

Weisstein, E.W. 1999. *www.mathworld.wolfram.com/uniformproductdistribution.html*. *MathWorld—A Wolfram Web Resource*.

Wingender, E., Chen, X., Fricke, E., Geffers, R., Hehl, R., Liebich, I., Krull, M., Matys, V., Michael, H., Ohnhauser, R., Pruss, M., Schacherer, F., Thiele, S., and Urbach, S. 2001. The TRANSFAC system on gene expression regulation. *Nucl. Acids. Res.* 29, 281–283.

Zhu, J., and Zhang, M.Q. 1999. SCPD: A promoter database of yeast *Saccharomyces cerevisiae*. *Bioinformatics* 15, 607–611.

Zuk, O., Getz, G., Hertzberg, L., Kanter, I., and Domany, E. 2004. On finding patterns in random sequences. Submitted for publication.

Address correspondence to:
Eytan Domany
Department of Physics of Complex Systems
Weizmann Institute of Science
Rehovot 76100
Israel

E-mail: eytan.domany@weizmann.ac.il