BIOINFORMATICS ORIGINAL PAPER

Vol. 23 no. 5 2007, pages 531–537
doi:10.1093/bioinformatics/btl662

Sequence analysis

Computing exact *P*-values for DNA motifs

Jing Zhang[1], Bo Jiang[2], Ming Li[1,3,*], John Tromp[4], Xuegong Zhang[2] and Michael Q. Zhang[5,2,*]

[1]State Key Laboratory of Intelligent Technology & System, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China, [2]Bioinformatics Division, TNLIST and Department of Antomation, Tsinghua University, Beijing 100084, China, [3]School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, [4]CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands and [5]Cold Spring Harbor Laboratory, Cold Spring Harbor, NY 11274, USA

Received on July 7, 2006; revised on December 1, 2006; accepted on December 22, 2006

Advance Access publication January 18, 2007

Associate Editor: Dmitrij Frishman

ABSTRACT

Motivation: Many heuristic algorithms have been designed to approximate *P*-values of DNA motifs described by position weight matrices, for evaluating their statistical significance. They often significantly deviate from the true *P*-value by orders of magnitude. Exact *P*-value computation is needed for ranking the motifs. Furthermore, surprisingly, the complexity of the problem is unknown.

Results: We show the problem to be NP-hard, and present MotifRank, software based on dynamic programming, to calculate exact *P*-values of motifs. We define the exact *P*-value on a general and more precise model. Asymptotically, MotifRank is faster than the best exact *P*-value computing algorithm, and is in fact practical. Our experiments clearly demonstrate that MotifRank significantly improves the accuracy of existing approximation algorithms.

Availability: MotifRank is available from http://bio.dlg.cn

Contact: mzhang@cshl.edu, mli@uwaterloo.ca

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 INTRODUCTION

Transcription factors (TFs) play a prominent role in gene regulation; identifying and characterizing their binding sites is central to annotating genomic regulatory regions and understanding gene-regulatory networks. An important aspect of this is to determine the statistical significance of the occurrences of transcription factor binding site (TFBS), also called motifs, in a DNA sequence. Statistical measures used for evaluating overabundance of patterns in sequences have been studied extensively, among which the *z-score* and *P-value* are most popular. Between them, the *P*-value is more reliable in statistical significance evaluation (Denise *et al*., 2001).

In previous studies, statisticians and computational biologists have devoted considerable efforts to solving these problems. Some are concerned with theoretical asymptotic word count distributions using either Gaussian (Brendel *et al*., 1986; Leung *et al*., 1996; Waterman, 1995), (Compound) Poisson (Chryssaphinou and Papastavridis, 1988; Godbole, 1991; Schbath, 1995) or large deviation approximations (Denise *et al*., 2001). The problem of pattern autocorrelation has been well studied theoretically, beginning with its introduction by Guibas and Odlyzko (1981). Other works concerned with the exact distribution of word counts using generating functions (Gentleman and Mullin, 1989; Goulden and Jackson, 1983; Hertzberg *et al*., 2005; Kleffe and Langbecker, 1990; Régnier, 2001) or other methods (Beckstette *et al*., 2006; Bejerano *et al*., 2004; Staden, 1998) soon followed. However, these methods are too theoretical to be of practical use, especially when dealing with motifs described by a position weight matrix (PWM) (Stormo, 2000). In practice, a variety of motif discovery tools evaluates results based on various approximate statistic criteria, such as Ahab (Rajewsky *et al*., 2002), Clover (Frith *et al*., 2004), MotifScanner (Thijs *et al*., 2001), MEME (Bailey and Gribskov, 1998; Bailey and Elkan, 1994), Consensus (Hertz and Stormo, 1999) and more recently the elegant approximate *P*-value calculation for multiple alignment (Nagarajan *et al*., 2005). These tools take less time to compute but give less accuracy in varying degrees or on different models.

Inspite of all these efforts over many years, it is surprising that we do not even know the complexity of this problem. Is it really difficult? Neither approximation approaches nor the current inefficient exact *P*-value algorithms can be justified if there exists an efficient algorithm for computing exact *P*-values. This article answers all these questions.

It turns out that the answers can be derived from a different field of bioinformatics: optimal spaced seeds for homology search introduced by Ma *et al*. (2002). By connecting the two fields, we not only obtain a proof that the *P*-value calculation problem is NP-hard, but also a practical algorithm, based on dynamic programming, to calculate the exact *P*-value of a PWM motif given in the most general form as a PWM. We use first-order Markov model to characterize the promoter region instead of i.i.d model. It is more general and contains more information. Its time complexity is good enough to provide a real-time calculation on typical biological data.

We have implemented the algorithm as the program MotifRank, to calculate exact *P*-values and to rank

*To whom correspondence should be addressed.

candidate motifs. The ability of MotifRank in discriminating known TFBSs from their backgrounds is validated on a simulated data set. Moreover, MotifRank compares favorably with MotifScanner, which estimates *P*-values, on a data set from *Saccharomyces cerevisiae* (yeast) Promoter Database (SCPD) (Zhu and Zhang, 1999). Our extensive tests show that, as expected, ranking of motifs according to exact *P*-values is better than using the approximate values given by MotifScanner.

2 PROBLEM DESCRIPTION

Given a set of genomic DNA sequences and a set of candidate motifs, the general problem of evaluating statistical significance of DNA motifs is to rank such motifs according to a underlying model. From a theoretical point of view, regulatory regions can be divided into two parts: the binding sites which play an important role in regulating gene expression, and the background which is not bound by TFs of interest. The key point for discriminating the signals from the background is to estimate if the motif is over-represented under the null hypothesis. We use a Markov chain to model the background and PWMs with appropriate thresholds to describe motifs; we take *P*-value as our statistic to rank the potential biological relevance of different motifs. Assuming the observed number of occurrences of motif m is k, its *P*-value is defined to be the probability that m hits (matches) the Markov chain at least k times. The central problem here is how to calculate the exact *P*-value efficiently.

Without loss of generality, we will use first-order Markov chains in this article to model the genetic sequences. Generalization to higher order is straightforward. For any string (or Markov chain) S, we use $S[i]$ to denote the ith position of S, and $S[i,j]$ to denote the substring of S from $S[i]$ to $S[j]$ inclusive, i.e. $S[i]S[i+1]\ldots S[j]$. Consider a first-order Markov chain R of length n. In the basic problem, let us consider word motif $\{A, C, G, T\}$. A motif m is considered to hit a string if the string contains the motif as a substring. That is, there $\exists i, j$, such that $m = R[i,j]$. The number of hits is the number of such different i, regardless of overlaps. Now we give the formal definition of the basic problem.

Input: A word m over alphabet $\Omega = \{A, C, G, T\}$, an integer $k \geq 1$, the stationary probability and transition probability of a first-order Markov region R
Output: Pr(m hits region R at least k times)

We next consider the multiple-word motif, which hits a string if any of its words does. This gives the following intermediate problem:

Input: A set of words $M = \{m_1, m_2, \ldots, m_t\}$ over alphabet Ω, an integer k, the stationary probability and transition probability of a first-order Markov region R
Output: Pr(M hits region R at least k times)

When we use a PWM to describe a motif, not only is the sequence region probabilistic, but the motif itself is as well. The PWM representation of a motif m of length l is a $4 \times l$ PWM matrix P, in which $P_{i,j}$ is the frequency for the ith character at the jth position of m. For a string s of the same length l, we define the score of s as the product of corresponding frequency scores in P (alternatively, we will sometimes use log odds ratios, the logarithm of the ratio of a letter's frequency at a position in the motif to its average frequency in the entire background). The meaning of 'hit' here is different from the word motif model. We say a motif m hits a string s when there is a substring s' of length l whose score is at least some threshold c_0. In biological applications, there are different methods to define c_0 and here it is regarded as an input parameter. Now, we define our third and central problem.

Input: A motif m with PWM P over alphabet $\Omega = \{A, C, G, T\}$, a threshold c_0, an integer k, the stationary probability and transition probability of a first-order Markov region R
Output: Pr(m hits region R at least k times)

We will give a linear time DP algorithm to calculate the *P*-value for a word motif in Section 3.1. We will adapt the DP algorithm to solve the word set problem in Section 3.2. The time and space complexity analysis will be given in Section 4.2. Calculating the *P*-value of a matrix motif is the main problem of this article. In Section 4.1, we will prove that it is an NP-hard problem and give an efficient algorithm for it by reducing it to the second problem in Section 3.3.

3 ALGORITHMS

3.1 Calculating the *P*-value for a word motif

We first give a simple algorithm for the case of $k = 1$ (i.e. the motif hits the region at least once) and then sketch an algorithm for the general k. The algorithms and ideas for computing the optimal spaced seeds (Keich *et al.*, 2004; Ma *et al.*, 2002) can be readily borrowed and generalized to our situation.

The basic idea is to calculate a series of conditional probabilities instead of the target probability. The conditional probabilities can be calculated by DP and the number of such probabilities is polynomial to n. We will give the formal definition of the conditional probability first and then constrain the domain of the parameters of the conditional probability.

For a string w over alphabet $\Omega = \{A, C, G, T\}$ and $|w| \leq n - i$, the conditional probability is defined to be

$$f(i, w) = \Pr(m \text{ hits } R[i,n] \mid w \text{ is a prefix of } R[i,n]) \quad (1)$$

In other words, the conditional probability is the hit probability of motif word m in a subregion of R under the condition that the prefix of the subregion is w. We can see that the target hit probability of m in region R equals $f(1, \epsilon)$. We will try to compute $f(i, w)$ in terms of other $f(i', w')$ computed earlier and limit the set of w we need to consider in the process.

For any $f(i, w)$, we decompose it according to the character following w in region $R[i,n]$, and then the following relation holds:

$$f(i, w) = \sum_{c \in \Omega} (f(i, wc) \times \Pr(R[i + |w|] = c \mid w \text{ is a prefix of } R[i,n]))$$

$\Pr(R[i + |w|] = c \mid w$ is a prefix of $R[i,n])$ expresses the probability that the character following w in region $R[i,n]$ is c.

It can be easily derived from the transition matrix of the first-order Markov chain when $|w|>0$. The case $|w|=0$ requires additional information about the last character before the region. Similar to Equation (1), for any $i \leq n$, we define

$$F(i,q) = \Pr(m \text{ hits } R[i,n] \mid R[i-1] = q) \tag{2}$$

We have, for $i>0$,

$$F(i,q) = \sum_{c\in\Omega} f(i,c) \times \Pr(R[i] = c \mid R[i-1] = q) \tag{3}$$

We next determine the domain of w. Clearly, if w is not a prefix of m, $\Pr(m = R[i, i+|m|-1] \mid w \text{ is a prefix of } R[i,n]) = 0$, which means that m cannot start at $R[i]$. We can see that, when $w[j,|w|-1]$ is not a prefix of m, m cannot start at $R[j]$ which has a prefix $w[j,|w|-1]$. So we have to find the longest suffix of w which is also a prefix of m. This leads us to the following definition.

DEFINITION 1. *Let $P(m)$ be the set of all prefixes of a word m. For any string w, let $S_m(w)$ denote the longest suffix of w which is in $P(m)$.*

For example, if $m=ACCAC$ and $w=CCAC$, then $S_m(w)=AC$ which is a suffix of w and a prefix of m. It is the longest string that satisfies the two requirements simultaneously.

The following observation helps to constrain the domain of w in $P(m)$. For w does not belong to $P(m)$,

$$f(i,w) = \begin{cases} F(i', w_{-1}) & \text{if } S_m(w) = \epsilon \\ f(i', S_m(w)) & \text{otherwise} \end{cases} \tag{4}$$

where $i' = i + |w| - |S_m(w)|$ and w_{-1} denotes the last letter of w. Thus, we only have to compute w which is a prefix of m. Algorithm 1 shows that $f(i,w)$ and $F(i,q)$ can be computed by DP in polynomial time.

Algorithm 1 Dynamic programming for $k=1$

Input: A motif word m over $\Omega=\{A,C,G,T\}$, the stationary probability $I_{1\times4}$ and transition matrix $T_{4\times4}$ of a first-order Markov region R.
Output: Probability that m hits R at least once.

```
Calculate map g : P(m) × Ω ⟼ P(m), ∀w ∈ P(m) and c ∈ Ω, g(w,c) = S_m(wc) using suffix tree
for i = n → 1 do
  for w ∈ P(m) from longest to shortest do
    if |w| = 0 then
      calculate F(i,q), ∀q ∈ Ω
    else
      calculate f(i,w)
    end if
  end for
end for
output Σ_{q∈Ω} I(q)F(1,q)
```

We show how to calculate $f(i,w)$ in Algorithm 2. $F(i,q)$ can be calculted similarly

Algorithm 2 Calculate $f(i,w)$

```
if |m| > n+1−i then
  f(i,w) = 0
else if w = m then
  f(i,w) = 1
else
  for c ∈ Ω do
    r_c = T[w_{-1}][c] which is the transition probability from w_{-1} to c, where w_{-1} denotes the last character of w
    i' = i + |wc| − |S_m(wc)|
    if |S_m(wc)| = 0 then
      f_c = F(i', c)
    else
      f_c = f(i', S_m(wc))
    end if
  end for
  f(i,w) = Σ_{c∈Ω} f_c × r_c
end if
```

We generalize Algorithms 1 and 2 to arbitrary k by defining a series of probabilities $\{f^{(1)}(i,w), f^{(2)}(i,w), \ldots, f^{(k)}(i,w)\}$ where

$$f^{(j)}(i,w) = \Pr(m \text{ hits } R[i,n] \text{ at least } j \text{ times} \mid w \text{ is a prefix of } R[i,n])$$

for $1 \leq j \leq k$. $f^{(k)}(1,\varepsilon)$ is exactly the P-value we want to calculate. $F^{(j)}(i,q)$ is defined similarly.

The recursion formulae here are

- When $w=m$, $f^{(j)}(i,w) = \sum_{c\in\Omega}(f^{(j-1)}(i,wc) \times \Pr(R[i+|w|] = c \mid w \text{ is a prefix of } R[i,n]))$
- When $w \neq m$, $f^{(j)}(i,w) = \sum_{c\in\Omega}(f^{(j)}(i,wc) \times \Pr(R[i+|w|] = c \mid w \text{ is a prefix of } R[i,n]))$

In other words, we use the conditional probabilities in a smaller subregion with fewer hits and a prefix constructed from w to calculate $f^{(j)}(i,w)$.

Algorithm 3 Computing P-value of a motif word

```
Calculate the map g using suffix tree
for i = n → 1 do
  for j = 1 → k do
    for w ∈ P(m) from longest to shortest do
      Calculate f^{(j)}(i,w)
    end for
  end for
end for
```

The details of calculating $f^{(j)}(i,w)$ are the same as Algorithm 1 and 2 with appropriate modification of the

recursion formulae. The complete time complexity analysis will be given in Section 4.2.

3.2 Calculating *P*-value for multiple-word motif

The basic idea is similar to Algorithm 1. We just extend all the definitions in Section 3.1 from a single word to a set of words. Two main definitions are conditional probability and prefix set.

The conditional probability for multiple-word motif M is

$$\mathbb{f}^{(j)}(i,w) = \Pr(M \text{ hits region } R[i,n] \text{ at least } j \text{ times} \mid b \text{ is a prefix of } R[i,n])$$

The definition of $\mathbb{F}^{(j)}(i,q)$ is similar and we omit it here. The prefix set of M is the union of the prefix sets of all the motif words in M, that is $P(M) = \bigcup_{s=1}^{t} P(m_s)$. The recursion formulae for multiple-word motif are

- if w matches any word in M, $\mathbb{f}^{(j)}(i,w) = \sum_{c\in\Omega} (\mathbb{f}^{(j-1)}(i,wc) \times \Pr(R[i+|w|] = c \mid w \text{ is a prefix of } R[i,n]))$
- otherwise, $\mathbb{f}^{(j)}(i,w) = \sum_{c\in\Omega}(\mathbb{f}^{(j)}(i,wc) \times \Pr(R[i+|w|\,] = c \mid w \text{ is a prefix of } R[i,n]))$

Replacing $P(m)$, $f^{(j)}(i,w)$ and $F^{(j)}(i,q)$ by $P(M)$, $\mathbb{f}^{(j)}(i,q)$ and $\mathbb{F}^{(j)}(i,q)$ in algorithms in Section 3.1, we can get the DP algorithm to calculate P-value for multiple-word motif. We will analyze time and space complexity of the algorithm in Section 4.2.

3.3 Computing one PWM motif hit probability

We will prove that calculating P-value for a PWM motif over a Markov region is NP-hard in Section 4.1. This explains the prevalence of approximation and exponential algorithms in the field. We present a DP algorithm by constructing a multiple-words motif from a matrix motif with threshold c_0 and then use the algorithms described in Section 3.2.

Given a threshold c_0 and a motif m represented by PWM P, we say a string of length $|m|$ is *compatible* with P if its motif score is at least c_0. The main idea of the algorithm is to rank the letters in each position of the motif in decreasing order of score.

In position i of motif m, call the highest score letter $m_{i,0}$, the second most $m_{i,1}$, the third most $m_{i,2}$ and the lowest score letter $m_{i,3}$. So each position independently ranks the letters from 0 through 3. We enumerate sequences of length $l = |m|$ in rank-lexicographic order, skipping incompatible strings whenever possible. In particular, if a string $s = m_{1,s_1} m_{2,s_2} \ldots m_{l,s_l}$ is compatible with m, then its successor, the next string to be considered, is simply the rank-lexicographically next string $m_{1,s_1} \ldots m_{i-1,s_{i-1}}\, m_{i,s_i+1}\, m_{i+1,0} \ldots m_{l,0}$, where i is the largest position with letter rank less than 3. If, on the other hand, s is not compatible with m, then, letting j be the largest position with non-zero letter rank, we skip all 4^{l-j} sequences of the form $m_{1,s_1} \ldots m_{j,s_j}\, m_{j+1,*} \ldots m_{l,*}$. These sequences can obviously score no better than s and are therefore also incompatible. The successor of s is then $m_{1,s_1} \ldots m_{i-1,s_{i-1}}\, m_{i,s_i+1}\, m_{i+1,0} \ldots m_{l,s_0}$, with i the largest position before j with letter rank less than 3. Since no $l = |m|$ consecutive successors can be incompatible, we can enumerate the set of compatible strings in time $O(K \times |m|)$, where K is the size of the set of all strings compatible with m. If we consider the double-stranded structure of DNA, we have to add the sequences reserve-complementary to those found.

Then we can use the algorithm for multiple-word motifs to calculate the P-value. Although it is an exponential time algorithm, in real biological applications c_0 is often large enough for MotifRank to be practical.

4 COMPLEXITY

4.1 The complexity of computing one PWM motif hit probability

In this section, we will prove that calculating P-value for a PWM motif over a Markov region is NP-hard by reducing the spaced seed problem to it. It indicates there does not exist a polynomial time complexity algorithm to calculate one PWM motif hit probability unless $P = NP$.

THEOREM 1. *Calculating the probability that a given PWM motif m over a constant size alphabet hits a first-order Markov region at least once at a given threshold is NP-hard.*

PROOF. Consider the simple case of a binary alphabet and an i.i.d. region. We reduce the *spaced seed* problem to this case. We describe the spaced seed problem here briefly. A spaced seed is a string over $\{0,1\}$. There is an i.i.d. sequence R, each bit with probability P (similarity level) to be 0, and $1-P$ to 1. A spaced seed s hits a string r when r has a substring r' of the same length as s that is 'implied' by s, i.e. wherever s has a 1, r' also has a 1. The spaced seed sensitivity problem is to calculate the hit probability of a spaced seed on a binary i.i.d. sequence R of a given similarity level. It has been recently shown to be NP-hard for similarity 1/2 in (Li *et al.*, 2006).

We reduce the spaced seed problem to our problem. Given a spaced seed s of length l with z 0s, we construct a motif m of length l over alphabet $\{0,1\}$ as follows: for $1 \le i \le l$, if $s_i = 1$, we let $\Pr(m_i = 1) = 1$ and $\Pr(m_i = 0) = 0$, otherwise $\Pr(m_i = 1) = \Pr(m_i = 0) = 1/2$.

Then a string r of length l is implied by s iff r has a non-zero motif score with respect to m. Thus, with a sufficiently small threshold like 2^{-z}, the probability of m passing the threshold in a random region is precisely the hit probability of spaced seed s. □

4.2 Complexity of algorithms in Section 3

We will give the time and space complexity analysis for the algorithms described in Section 3. Lemma 1 and Theorem 2 are about Algorithm 3 for calculating the P-value for a word motif given in Section 3.1. Theorem 3 is about the algorithm for calculating the P-value for multiple-word motif given in Section 3.3.

The line 1 of Algorithm 3 is to calculate the mapping $g : P(m) \times \Omega \longmapsto P(m)$, which means to find the longest suffix of the string composed of a prefix string in the prefix set $P(m)$ appending a character c and the suffix must be a prefix in $P(m)$, too. We have the following lemma about the time to calculate $S_m(w)$ for arbitrary string w.

LEMMA 1. *Given a word m and a string w over a constant alphabet, $S_m(w)$ can be calculated in $O(m)$.*

PROOF. Let $\overleftarrow{m}$ denote the reverse of string m. Construct suffix tree for $\overleftarrow{m}$ and search $\overleftarrow{w}$ on it. The result path is $S_m(w)$. The time complexity of constructing a suffix tree is $O(m)$ and searching is $O(m)$, so the total time complexity is $O(m)$. □

We have the following theorem about the time and space complexity of Algorithm 3.

THEOREM 2. *The probability that a word motif m over alphabet $\Omega = \{A, C, G, T\}$ hits a first-order Markov region R for at least k times can be calculated in polynomial time. The time complexity is $O(k \times n \times |m|)$ and the space complexity is $O(k \times |m|^2)$, where n is the length of R.*

PROOF. According to Lemma 1, line 1 takes time $O(|m|^2)$. Consider how many $f^{(j)}(i, w)$ will be calculated in Algorithm 3 (this will also bound the number of $F^{(j)}(i, q)$ computed). The three nested loops in Algorithm 3 produce $O(k \times n \times |m|)$ iterations. Each iteration calculates a $f^{(j)}(i, w)$ in constant time. So the time to calculate all f is $O(k \times n \times |m|)$. The space complexity is $O(k \times |m|^2)$ since when calculating $f^{(j)}(i, w)$, we only have to maintain $f^{(j)}(j, w)$ for $i \le j \le i + m$ of size at most $O(k \times |m|^2)$. □

The algorithm for multiple-word motif is an extension to Algorithm 3, so the time and space analysis are similar to Theorem 2. We give the results directly.

THEOREM 3. *The probability that a set of word motifs $M = \{m_1, m_2, \ldots, m_t\}$ over alphabet $\Omega = \{A, C, G, T\}$ hits a first-order Markov region R at least k times can be calculated in polynomial time. Let $||M|| = \sum_{i=1}^{t} |m_i|$. The time complexity is $O(k \times n \times ||M||)$ and the space complexity is $O(k \times ||M|| \times (\max_i |m_i|))$.*

Now we analyze the time complexity of the algorithms described in Section 3.3. The algorithm to extract multiple words from PWM motif has time complexity $O(K \times |m|)$ as already shown in the algorithm. Adding the time to calculate the *P*-value for multiple words, the total time complexity of our exact *P*-value calculating algorithm is $O(K \times |m| + k \times n \times ||M||)$, in which $M = \{m_1, m_2, \ldots, m_K\}$ is the set of compatible strings, $||M|| = \sum_{i=1}^{K} |m_i|$ and k is the hit time of M in the region of length n.

5 EXPERIMENTS AND RESULTS

5.1 Experiment methods

We apply our algorithm given in Section 3.3 to the statistical significance evaluation of PWM motifs and rank them according to their exact *P*-values.

Consider a first-order Markov model whose elements in transition matrix Π are estimated from a promoter region. The initial probability of the Markov model is chosen to be the stationary distribution μ defined by $\mu = \mu\Pi$.

There are many strategies for choosing the threshold of a PWM motif on a specific sequence. Here we adopt a simple way by allowing a user-defined multiplying factor, which was introduced in Tronche *et al.* (1997). We then multiply the maximum of scores of all $|m|$ long strings on PWM motif m by the factor to define the threshold c_0. This factor allows the user to define the type of motif to search for. It controls the balance between specificity and sensitivity. When the factor is reduced, more degeneracy in the extraction of motif words will be allowed, so sensitivity will increase but specificity will decrease. Besides, we remove the words which are not substrings of the promoter sequence from the word set extracted from m according to c_0. Furthermore, since DNA is double stranded, we need to look for binding sites on both strands.

We use the algorithm introduced in Section 3.3 to calculate the exact *P*-values for motifs in promoter regions.

5.2 Comparison with MotifScanner on SCPD data

The SCPD contains 130 promoter regions of different genes with known binding sites coming from 21 PWM motifs. We extract upstream (of ATG) sequences, of length 1001 bp, for all 130 genes from SCPD as our experimental data set. For each promoter region, we estimate the parameters of Markov model, and calculate the exact *P*-values for each of the 21 PWM motifs by using MotifRank on double strands and with specific factor.

We compare our results on SCPD data with a similar tool, MotifScanner (Thijs *et al.*, 2001), which estimates *P*-values. MotifScanner is a program that can be used to screen DNA sequences with pre-compiled motif models. The input of MotifScanner consists of four parts: DNA sequences, candidate motifs represented by position frequency matrices, a prior probability, which allows the user to specify the degeneracy in the found instances of motifs and a background model. We input the same SCPD data set and choose first-order Markov model on double strands with prior probability 0.2, which is suggested by MotifScanner for yeast data. MotifScanner outputs the approximate *P*-values of PWM motifs which are regarded to be over-represented in the region. As a result, the outputs do not always contain the *P*-values for all the known motifs reported by biologists.

We rank the PWM motifs in each promoter region in increasing order of *P*-values, and compare the ranks of known motifs annotated in SCPD. As aforementioned, the value of multiplying factor in MotifRank determines the degeneracy of instances retrieved from PWM motifs, and so does the prior probability in MotifScanner. Both tools will fail to report the instances of some known motifs when a relatively strict constraint on degeneracy is imposed. To ensure that the comparison with MotifScanner is under similar degrees of degeneracy, we adopt a multiplying factor of 0.65 in MotifRank, with which the instances of known motifs reported by both tools are nearly the same. To better assess the results, we divide the promoter regions into different groups. Majority of promoter regions contain only one known motif, but some contain more. For example, the CDC2 promoter region only has MCB motif reported but CDC9 promoter region has two motifs: MCB and REB1. We thus divide the regions into several groups according to the reported number of distinct motifs and compare ranks given by different tools within each group. We use the average rank of the known motifs within each region group to evaluate the rank results. As a criterion reflecting the overall performance, a lower average rank

Table 1. Average and standard deviation of the rank of known TFs with *P*-values reported by different tools

Number of reported motifs per region	Number of promoter regions	MotifRank		MotifScanner	
		Average rank	Standard deviation	Average rank	Standard deviation
1	78	2.33	1.46	3.09	1.84
2	22	2.95	1.79	3.34	2.12
3	7	3.24	1.95	3.38	1.64
4	1	5.50	3.42	6.75	1.71
Total	108	2.73	1.78	3.30	2.06

The promoter regions are divided into four groups according to the number of TFs reported to bind in them.

of reported motifs means that the tool will assign a comparatively lower *P*-value to the TFs known to bind in promoter regions.

The average rank and its standard deviation using different tools are given in Table 1. Here we only consider the ranks of known motifs with *P*-values given by both tools, so there are totally 108 promoter regions in Table 1. From the average rank of known motifs, we can demonstrate the advantage of exact *P*-values over approximate ones (in the sense of giving a lower average rank for the motifs which are reported to bind in corresponding promoter region). Remarkably, our method has a higher performance on words that tend to have self-overlaps than MotifScanner does. For example, the motif of RAP1-binding sites 'R(A/G)M(C/A)ACCCANACAY(C/T)' which has several self-overlaps could be ranked first by MotifRank when it is within most of its binding promoters, whereas MotifScanner would give a worse rank for it. More details are available in the Supplementary Material.

We also use the ranks given by MotifRank and MotifScanner to predict putative TFs that bind in the promoter region. Given an integer *N*, a PWM motif is said to be positive in a specific promoter region if its rank is among the top *N*, and the fraction of true positives and false positives can be calculated according to the known motifs reported by biologists. Thus we get a receiver operating characteristic (ROC) curve over a range of possible *N*s. The ROC curves of MotifScanner and MotifRank with multiplying factors ranging from 0.6 to 0.8, a range that allows moderate degeneracy in the extraction of motif words, are illustrated in Figure 1. We can see that the area under the curves of MotifRank is significantly larger than that of MotifScanner, which shows the advantage of exact *P*-values over approximate ones in predicting the binding of TFs.

5.3 Performance on simulated data

We test our method on simulated data to further validate its ability to discriminate known TFBSs from backgrounds. We simulated a data set with 100 sequences according to the first-order Markov model and parameters extracted from SCPD. We randomly choose instances of motifs from the frequency matrix of 20 PWM motifs in SCPD and uniformly implanted them into foreground sequences. The number of

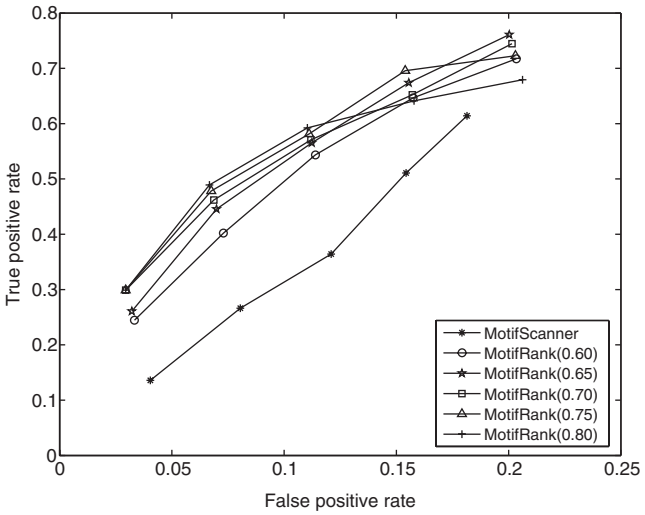

Fig. 1. ROC curves manifest the ability of MotifRank and MotifScanner to identify TFs that bind in promoter regions. Different points on a ROC curve correspond to cutoffs on the ranking list varying from 1 to 5. X-axis denotes false-positive rate, i.e. the number of false positives to the total number of TFs that are not reported to bind in the promoter, while y-axis denotes true positive rate, i.e. the number of true positives to the total number of TFs that are reported to bind in the promoter.

motifs and their occurrences in each sequence is similar to that in real data sets. Simultaneously, we generate 100 background sequences under the same model but without any implanted motif, and calculate the average *P*-value of each PWM motif. The average *P*-values of implanted PWM motifs on foreground and background sequences are given in Table 2.

There is a >30-fold difference relative to the background. Our method is indeed powerful to distinguish sites that contain true motifs from those that do not. Its high performance on the SCPD data is consistent both in this Section and Section 5.2.

5.4 Time performance

We have proved that the worst-case time complexity of the algorithm introduced in Section 3.3 is exponential and there is no polynomial time algorithm for exact *P*-value calculation of the PWM motifs, unless NP = P. The algorithm is implemented using C++ and compiled under Microsoft Visual Studio 6.0. The test PC is equipped with a Pentium 4 running at 2.8 GHz and 512 MB of RAM. The average time to compute the *P*-value of one motif with a 1001 bp promoter region is 0.7 ms. This indicates that our DP algorithm is efficient enough for biological data in practice.

6 CONCLUSION AND FUTURE WORK

Accurate and efficient methods for calculating the *P*-value of DNA motifs are critical in aiding the discovery of TFBS in promoter regions. Although existing tools provide approximate estimates of *P*-values, they are not accurate enough sometimes and there is a need for efficient algorithms to calculate exact *P*-values.

Table 2. Average *P*-values on foreground and background simulated sequences

TF name	Length	Foreground	Background
ABF1	12	1.85E−08	0.94
CSRE	13	1.21E−13	1.00
SCB	7	2.38E−06	0.69
GCN4	6	4.42E−03	0.17
GCR1	5	1.86E−01	0.47
MCB	6	1.37E−04	0.59
MCM1	10	6.01E−09	0.74
MATalpha2	9	1.08E−04	0.84
MIG1	12	3.26E−09	0.93
PHO2	7	3.20E−02	0.12
PHO4	10	2.19E−10	0.91
RAP1	12	1.38E−05	0.85
REB1	7	5.65E−05	0.70
ROX1	12	8.34E−02	1.00
SWI5	12	3.82E−05	0.78
STE12	8	9.05E−06	0.69
TBP	7	4.10E−03	0.32
UASPHR	12	1.33E−01	0.79
XBP1	12	1.50E−05	0.96
repressor_of_ CAR1	9	2.79E−03	0.65

The first two columns show the name and length of the TFs whose PWM motifs are implanted into foreground sequences. The entries in the foreground column are the average *P*-values over foreground sequences with corresponding PWM motifs implanted. In contrast, we have also run 100 sequences as background without implanting any PWM motifs and calculated the average *P*-values on them. The results are shown in the background column.

A rapid algorithm to compute exact *P*-values for DNA motifs is presented in this article. We have validated the performance of the algorithm on both real and simulated biological data. Comparison with other tools confirms that the performance of exact *P*-values is better than approximate *P*-values in terms of evaluating the statistical significance of motifs.

Our algorithm for calculating exact *P*-values is useful and feasible wherever the evaluation of statistical significance for candidate motifs is desired. Furthermore, our method could be integrated into discovery of TFBSs in given promoter regions.

In future work, we will study the algorithms and experiments on evaluating statistical significance of synergistic motifs and TFBSs common to multiple sequences.

7 ACKNOWLEDGEMENTS

J.Z.'s work is supported by the National Natural Science Foundation of China Grant 60553001 and the National Basic Research Program of China Grant 2007CB807900, 2007CB807901. B.J. and X.Z. are supported in part by NSFC grants 60540420569, 30625012. M.L.'s work was partially supported by the Chang Jiang Scholarship Program, NSERC and Canada Research Chair program. M.Z. is partially supported by the Chang Jiang Scholarship Program and a NIH grant. J.T.'s work has been partially funded by the Dutch BSIK/BRICKS project.

Conflict of Interest: none declared.

REFERENCES

Bailey,T.L. and Gribskov,M. (1998) Combining evidence using *P*-values: application to sequence homology searches. *Bioinformatics*, **14**, 48–54.

Bailey,T.L. and Elkan,C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*. pp. 28–36.

Beckstette,M. *et al*. (2006) Fast index based algorithms and software for matching position specific scoring matrices. *BMC Bioinformatics*, **7**, 389.

Bejerano,G. *et al*. (2004) Efficient exact *P*-value computation for small sample, sparse, and surprising categorical data. *J. Comp. Biol*., **11**, 867–886.

Brendel,V. *et al*. (1986) Linguistics of nucleotide sequences: morphology and comparison of vocabularies. *J. Biomol. Struct. Dyn*., **4**.

Chryssaphinou,O. and Papastavridis,S. (1988) A limit theorem for the number of non-overlapping occurrences of a pattern in a sequence of independent trials. *J. Appl. Prob*., **25**, 428–431.

Denise,A. *et al*. (2001) Assessing statistical significance of overrepresented oligonucleotides. In *WABI'01*. pp. 85–97.

Frith,M.C. *et al*. (2004) Detection of functional DNA motifs via statistical over-representation. *Nucleic Acids Res*., **32**, 1372–1381.

Gentleman,J. and Mullin,R. (1989) The distribution of the frequency of occurence of nucleotide subsequence, based on their overlap capability. *Biometrics*, **45**, 35–52.

Godbole,A.P. (1991) Poisson approximations for runs and patterns of rare events. *Adv. Appl. Prob*., **23**, 851–865.

Goulden,I.P. and Jackson,D.M. (1983) Combinatorial Enumeration. Wiley.

Guibas,L.J. and Odlyzko,A.M. (1981) Strings overlaps, pattern matching and nontransitive games. *J. Combinat. Theory*, Series A **30**.

Hertz,G.Z. and Stormo,G.D. (1999) Identifying DNA and protein patterns with statistically significant alignment of multiple sequences. *Bioinformatics*, **15**.

Hertzberg,L. *et al*. (2005) Finding motifs in promoter regions. *J. Comp. Biol*., **12**, 314–330.

Keich,U. *et al*. (2004) On spaced seeds for similarity search. *Discrete Appli. Math,* **138**, 253–263.

Kleffe,J. and Langbecker,U. (1990) Exact computation of pattern probabilities in random sequences generated by Markov chains. *Comp. Appl. Biosci*., **6**, 347–353.

Leung,M.Y. *et al*. (1996) Over and underrepresentation of short DNA words in Herpes virus genomes. *J. Comp. Biol*., **3**, 345–360.

Li,M. Superiority and complexity of the spaced seeds. In *SODA*'06.

Ma,B. *et al*. (2002) PatternHunter: faster and more sensitive homology search. *Bioinformatics*, **18**, 440–445.

Nagarajan,N. *et al*. (2005) Computing the *P*-value of the information content from an alignment of multiple sequences. *Bioinformatics*, **21** (Suppl. 1), *ISMB '05*, pp. i311–i318.

Rajewsky,N. *et al*. (2002) Computational detection of genomic cis regulatory modules, applied to body patterning in the early Drosophila embryo. *BMC Bioinformatics*, **3**, 30.

Régnier,M. (2001) A unified approach to word occurrence probabilities. *Discrete Appl Math*., **104**, 259–280.

Staden,R. (1998) Methods for calculating the probabilities of finding patterns in sequences. *Comput. Appl. Biosci*., **5**, 89–96.

Schbath,S. (1995) Compound Poisson approximation of word counts in DNA sequences. *ESAIM: Probab. Stati*., **1**, 1–16.

Stormo,G.D. (2000) DNA binding sites: representation and discovery. *Bioinformatics*, **16**, 16–23.

Tronche,F. *et al*. (1997) Analysis of the distribution of binding sites for a tissue-specific transcription factor in the vertebrate genome. *J. Mol. Biol.,* **266**, 234–245.

Thijs,G. *et al*. (2001) A Gibbs sampling method to detect over-represented motifs in upstream regions of coexpressed genes. In *Proceedings Recomb'2001*. pp. 305–312.

Waterman,M.S. (1995) *Introduction to Computational Biology*. Chapman & Hall.

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