VOTING ALGORITHMS FOR THE MOTIF FINDING PROBLEM

Xiaowen Liu and Bin Ma
Department of Computer Science, University of Western Ontario
London, ON, Canada, N6A 5B7
Email: liuxiaowencs@gmail.com, bma@uwo.ca

Lusheng Wang*
Department of Computer Science, City University of Hong Kong
Kowloon, Hong Kong
Email: lwang@cs.cityu.edu.hk

Finding motifs in many sequences is an important problem in computational biology, especially in identification of regulatory motifs in DNA sequences. Let c be a motif sequence. Given a set of sequences, each is planted with a mutated version of c at an unknown position, the motif finding problem is to find these planted motifs and the original c. In this paper, we study the VM model of the planted motif problem, which is proposed by Pevzner and Sze [1]. We give a simple Selecting One Voting algorithm and a more powerful Selecting k Voting algorithm. When the length of motif and the number of input sequences are large enough, we prove that the two algorithms can find the unknown motif consensus with high probability. In the proof, we show why a large number of input sequences is so important for finding motifs, which is believed by most researchers. Experimental results on simulated data also support the claim. Selecting k Voting algorithm is powerful, but computational intensive. To speed up the algorithm, we propose a progressive filtering algorithm, which improves the running time significantly and has good accuracy in finding motifs. Our experimental results show that Selecting k Voting algorithm with progressive filtering performs very well in practice and it outperforms some best known algorithms.
Availability: The software is available upon request.

1. INTRODUCTION

The motif finding problem in molecular biology is to find similar regions common to each sequence in a given set of DNA, RNA, or protein sequences. This problem has many applications, such as locating binding sites and finding conserved regions in unaligned sequences, genetic drug target identification and designing genetic probes.

Since DNA bases and protein residues are subject to mutations, motifs with similar functions in different sequences are not identical. From an algorithmic point of view, the motif finding problem can be considered as the consensus pattern problem. Different variants of this problem are NP-hard [2, 3] and several polynomial time algorithm schemes have been proposed [3–5].

In practice, the motif finding problem has been intensely studied. Many methods and software have been proposed [6–20]. Basically, there are two different approaches for the motif finding problem [21]. The first approach uses the pattern-driven method. For DNA and RNA sequences, all 4^L possible patterns are tried to find the best motif consensus [14, 18], where L is the length of motif. When the length of motif is large, the running time of pattern-driven algorithm is formidable. The other approach uses the sample-driven method. Sample-driven algorithms use all substrings of length L in input sequences (all L-mers) as the set of patterns [1, 3, 9]. These algorithms start from L-mers in input sequences, then use heuristic search to find the motif consensus. Due to mutations, the sample-driven algorithms may miss some good starting patterns and fail to find the real motif consensus. Based on this observation, the extended sample-driven approach is developed. This approach is a hybrid of the pattern-driven method and the sample-driven method. Both L-mers in input sequences and close neighbors of the L-mers are used as the patterns [15, 21].

To test and evaluate different methods, Pevzner and Sze proposed a planted motif model [1]. In the

*Corresponding author.

planted model, the input is a set of random samples, each sample is planted with a motif with mutations (errors). The planted motif problem is to find the planted motifs in the samples and the motif consensus. If we find the correct motif consensus, the planted motifs can be found easily. Therefore, we will focus on finding the motif consensus in this study. There are two different mutation models. The first model is the FM model, where each sequence contains one instance of an (L, D) motif, i.e. each instance of length L contains D mutated positions, where the D positions are randomly selected. The second model is the VM model, where each sequence contains exactly one instance, and each position of the instance is mutated independently with probability p.

This first model has been studied and tested with many algorithms [21–23]. In this paper, we mainly study the second model. In the experiments, our algorithms are tested on both the FM and VM models.

The main contributions of this paper are the following. First, we give a simple Selecting One Voting algorithm and a more powerful Selecting k Voting algorithm. We prove that, when the length of motif and the number of input sequences are large, the two algorithms can find the unknown motif consensus with high probability. Second, most researchers believe that a large number of sequences containing mutated motifs can help us find motifs. When the number of input sequences increases, the probabilities that motifs can be found will increase. In the proof, we show that the number of input sequences can help our algorithms find motifs. Our experiments on simulated data also support the relationship between the number of input sequences and the performance of our algorithms. Select k Voting algorithm is powerful, but the time complexity of the algorithm is too high to be practical for real problems. Finally, we propose a progressive filtering method to speed up Selecting k Voting algorithm. With the filtering method, the time complexity of Selecting k Voting algorithm is improved from $O(Lm^{k+1}n)$ to $O(\alpha Lm(k^2+n))$, where n is the number of input sequences, m is the length of each sequence and α is an input parameter. Our experimental results show that Selecting k Voting algorithm with progressive filtering performs very well in practice and it outperforms some best known algorithms.

2. PROBABILITY MODELS

In this paper, we consider DNA sequences and use a fixed alphabet $\Sigma = \{A, C, G, T\}$. For a string $s \in \Sigma^m$, the i-th letter of s is denoted by $s[i]$. Thus, $s = s[1]s[2]\ldots s[m]$. A string s is called a *uniform random DNA sequence* if for each letter $s[i]$ in s, $Pr(s[i] = A) = Pr(s[i] = C) = Pr(s[i] = G) = Pr(s[i] = T) = \frac{1}{4}$. Let s_1 and s_2 be two strings of the same length. The Hamming distance between s_1 and s_2 is denoted by $d(s_1, s_2)$. For a string $t \in \Sigma^L$ and a string $s \in \Sigma^m$, where $L < m$, the distance between t and s is the minimum distance between t and any L-mer in s, denoted by $d(t, s)$. For a set of strings $S = \{s_1, s_2, \ldots, s_n\}$ and a string s of the same length m, if each letter $s[i]$ in s is a majority letter in $\{s_1[i], s_2[i], \ldots, s_n[i]\}$, s is called a *consensus string* of S.

In the VM probability model, n input strings with planted motifs are generated as follows. We first generate n uniform random DNA sequences $\overline{s_1}, \overline{s_2}, \ldots, \overline{s_n}$, each of length m. Suppose a uniform random DNA sequence $c \in \Sigma^L$ is the original motif consensus. Based on c, we generate n motifs $c_1, c_2, \ldots, c_n \in \Sigma^L$ with mutations (errors) by changing each letter in c independently with probability $p = \frac{3}{4} - \epsilon$. That is, for each letter $c_i[j]$ in c_i, $Pr(c_i[j] = c[j]) = \frac{1}{4} + \epsilon$, and for $u \in \Sigma \backslash \{c[j]\}$, $Pr(c_i[j] = u) = \frac{1}{4} - \frac{\epsilon}{3}$. Then, for each string $\overline{s_i}$, we randomly select a position h in $\{1, 2, \ldots, m - L + 1\}$ and replace $s[h]s[h+1]\ldots s[h+L-1]$ with c_i to get s_i. We say that c_i is planted to s_i. In this way, we get a new set of strings $s_1, s_2, \ldots, s_n$, which is a set of random strings with planted motifs. From now on, we will consider the string set $S = \{s_1, s_2, \ldots, s_n\}$ as the input sequences. For each string s_i, the set of all L-mers of s_i is denoted by P_i.

Since the FM probability model is used in our experiments, we also give the definition of the FM model. In the (L, D) FM model, with n uniform random DNA sequences and a motif consensus $c \in \Sigma^L$, the method of generating n mutated motifs is different from the VM model. In motif consensus c, we randomly select exact D positions and each of the D letters is changed to any of other three letters to get a mutated motif c_i. For each of the D positions, $Pr(c_i[j] = c[j]) = 0$, and for $u \in \Sigma \backslash \{c[j]\}$, $Pr(c_i[j] = u) = \frac{1}{3}$. Finally, the n mutated motifs are planted to the n uniform random DNA sequences to get a set of n input sequences.

With the VM probability model, we give the definition of the planted motif problem.

Definition 2.1. Given a set $S = \{s_1, s_2, \ldots, s_n\}$ of strings each of length m, generated as described in the VM probability model, and an integer L, the *planted motif problem* is to find the unknown motif consensus c.

In some cases, we want to find the closest substrings in the input sequences. Then, we have another similar problem.

Definition 2.2. Given a set $S = \{s_1, s_2, \ldots, s_n\}$ of strings each of length m, generated as described in the VM probability model and an integer L, the *planted closest substring problem* is to find a length L substring t_i for each string $s_i \in S$ and a consensus string t such that $\sum_{1 \le i \le n} d(t, t_i)$ is minimized.

```
                    Algorithm 1
Input    A sequence set $S = \{s_1, s_2, \ldots, s_n\} \subset \Sigma^m$, a starting pattern $t \in \Sigma^L$.
Output   A motif consensus with length $L$.

1.  For each sequence $s_i$, find an $L$-mer $t_i \in P_i$ such that $d(t, t_i) = d(t, s_i)$.
2.  Output a consensus string of $\{t_1, t_2, \ldots, t_n\}$.
```

Fig. 1. The voting algorithm.

3. ALGORITHMS

In this section, we give several algorithms for the planted motif problem.

3.1. Voting Algorithm

For the planted motif problem, our algorithms have two parts. The first part is to find a starting pattern (L-mer). The second part is to use the starting pattern to find the motif consensus and planted motifs. Here, we first give a simple voting algorithm for finding motif consensus from a given starting pattern t. In details, our algorithm has two steps: (1) in each sequence s_i, find an L-mer $t_i \in P_i$ with the minimum distance to t; (2) find a consensus string of $t_1, t_2, \ldots, t_n$ (Fig. 1).

In practice, we can use the resulting consensus string of the voting operation as a new starting pattern, and do voting operation repeatedly until there is no further improvement. Compared with other pattern refinement methods, such as Gibbs sampling [7] and EM [8, 9] methods, the voting algorithm uses a consensus string instead of a profile to represent the pattern and uses a simple voting operation to do pattern refinement. The main advantage of the voting algorithm is its high speed. Since the pattern is represented by a string, the voting algorithm converges faster than the Gibbs sampling and EM methods. In addition, the voting algorithm avoids the time consuming computation of likelihoods in Gibbs sampling and EM methods. So, the voting algorithm is much faster than the Gibbs sampling and EM approaches. With the fast speed, we can try much more starting patterns to find a good motif. Our experimental results also show that the voting algorithm is powerful for finding motifs.

```
                    Algorithm 2
Input    A sequence set $S = \{s_1, s_2, \ldots, s_n\} \subset \Sigma^m$ and integers $L$ and $r$.
Output   A motif consensus with length $L$.
1.  Repeat Steps 2-5 $r$ times.
2.      Randomly select a sequence $s_i \in S$ which has not been selected in previous rounds.
3.      For each $L$-mer $t \in P_i$, do
4.          Use the voting algorithm to find a consensus $t^*$ from starting pattern $t$.
5.          Add $t^*$ to candidate motif consensus set $C$.
6.  Output motif consensus $c_A$ such that $\sum_{1 \le i \le n} d(c_A, s_i)$ is the minimum in all candidates in $C$.
```

Fig. 2. Selecting One Voting algorithm.

3.2. Selecting One Voting Algorithm

The performance of the voting algorithm depends on the qualities of starting patterns. So we need to find good starting patterns for it. Our method follows the sample-driven approach. We use L-mers in input sequences as the set of patterns. We randomly select an input sequence $s_i \in S$ and find the planted motif c_i by enumerating all L-mers in s_i. Then the motif c_i is used as a starting pattern of the voting algorithm. The above procedure is repeated r times and the best motif consensus is output, where r is an input parameter. The algorithm is called Selecting One Voting algorithm (Fig. 2).

We can prove that, when L and n are large enough, Algorithm 2 can correctly find the motif consensus with high probability. In the following analysis, we use an important lemma about Chernoff-Hoeffding bound [24].

Lemma 3.1. *Let $X_1, X_2, \ldots, X_n$ be n independent random binary (0 or 1) variables, where X_i takes on the value of 1 with probability p_i, $0 < p_i < 1$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X]$. Then for any $0 < \lambda < 1$,*

(1) $\mathbf{Pr}(X \geq \mu + \lambda n) \leq e^{-2\lambda^2 n}$,
(2) $\mathbf{Pr}(X \leq \mu - \lambda n) \leq e^{-2\lambda^2 n}$.

From the VM probability model, each planted motif has $(\frac{3}{4} - \epsilon)L$ mutations on average. When r is large, we repeat the voting operation many times. Then, the probability that we can find a planted motif with no more than $(\frac{3}{4} - \epsilon)L$ mutations is high.

Lemma 3.2. *The probability that one planted motif t with $d(c,t) \leq (\frac{3}{4} - \epsilon)L$ is selected in Step 3 of Algorithm 2 is at least $1 - \left(\frac{3}{4}\right)^r$.*

Proof. Based on the VM probabilistic model, the distance between c and a mutated motif c_i fits the binomial distribution $B(L, \frac{3}{4} - \epsilon)$. Therefore, for each c_i, $Pr(d(c_i, c) > \frac{3}{4} - \epsilon) \leq \frac{3}{4}$ (The inequality can be proved by enumerating all possible L's). In Algorithm 2, we randomly select r planted motifs as starting patterns. Since the r motifs are independently generated, the probability that each selected motif c_i has $d(c_i, c) > (\frac{3}{4} - \epsilon)L$ is no more than $\left(\frac{3}{4}\right)^r$. Therefore, the lemma is proved. □

When the length of motif is large enough, with high probability, two planted motifs have smaller distance than two random L-mers. Based on this observation, we have the following lemma.

Lemma 3.3. *When $d(c,t) \leq (\frac{3}{4} - \epsilon)L$ and $L > \frac{9}{8\epsilon^4} \log \frac{3m}{\epsilon}$, for each sequence $s_i \in S$, the probability that c_i is selected in Step 1 of the voting algorithm is no less than $1 - \frac{1}{3}\epsilon$.*

Proof. First, we consider the planted motif c_i in s_i. Let $X_1, X_2, \ldots, X_L$ be the random variables such that $X_j = 1$, if $c_i[j] = t[j]$; $X_j = 0$, otherwise. From the assumption, we have $d(c,t) \leq (\frac{4}{3} - \epsilon)L$. From the generation method, for each letter $c_i[j]$, $Pr(c_i[j] = c[j]) = \frac{1}{4} + \epsilon$. Therefore, $Pr(X_j = 1) = \frac{1}{4} + \frac{4}{3}\epsilon^2$ and $Pr(X_j = 0) = \frac{3}{4} - \frac{4}{3}\epsilon^2$. Let $X = \sum_{1 \leq j \leq L} X_j$. Then $E(X) = (\frac{1}{4} + \frac{4}{3}\epsilon^2)L$. By Lemma 3.1,

$$Pr\left(X \leq \left(\frac{1}{4} + \frac{2}{3}\epsilon^2\right)L\right) \leq Pr\left(X \leq E(X) - \frac{2}{3}\epsilon^2 L\right) \leq e^{-\frac{8}{9}\epsilon^4 L}.$$

When $L > \frac{9}{8\epsilon^4} \log \frac{3m}{\epsilon}$, the probability is

$$Pr\left(X \leq \left(\frac{1}{4} + \frac{2}{3}\epsilon^2\right)L\right) \leq \frac{\epsilon}{3m}. \qquad (1)$$

Second, we consider an L-mer $t' \in P_i \backslash \{c_i\}$. Let $Y_1, Y_2, \ldots, Y_L$ be the random variables such that $Y_j = 1$, if $t'[j] = c[j]$; $Y_j = 0$, otherwise. For each letter $t'[j]$, $Pr(t'[j] = A) = Pr(t'[j] = C) = Pr(t'[j] = G) = Pr(t'[j] = T) = \frac{1}{4}$. Therefore, $Pr(Y_j = 1) = \frac{1}{4}$ and $Pr(Y_j = 0) = \frac{3}{4}$. Let $Y = \sum_{1 \leq j \leq L} Y_j$. We have $E(Y) = \frac{1}{4}L$. By Lemma 3.1,

$$Pr\left(Y \geq \left(\frac{1}{4} + \frac{2}{3}\epsilon^2\right)L\right) \leq Pr\left(Y \geq E(Y) + \frac{2}{3}\epsilon^2 L\right) \leq e^{-\frac{8}{9}\epsilon^4 L}.$$

When $L > \frac{9}{8\epsilon^4} \log \frac{3m}{\epsilon}$, the probability is

$$Pr\left(Y \geq \left(\frac{1}{4} + \frac{2}{3}\epsilon^2\right)L\right) \leq \frac{\epsilon}{3m}.$$

Consider all the L-mers in $P_i \backslash \{c_i\}$, the probability that there is an L-mer $t' \in P_i \backslash \{c_i\}$ such that $d(t', c) \geq (\frac{1}{4} + \frac{2}{3}\epsilon^2)L$ is no more than $\frac{m-L}{3m}\epsilon$.

Together with (1), when $d(c,t) \leq (\frac{3}{4} - \epsilon)L$ and $L > \frac{9}{8\epsilon^4} \log \frac{3m}{\epsilon}$, the probability that c_i is selected in Step 1 of the voting algorithm is no less than $1 - \frac{1}{3}\epsilon$. □

When the planted motifs can be found with high probability, the voting algorithm can find the motif consensus with high probability.

Lemma 3.4. *Suppose $d(c,t) \leq (\frac{3}{4}-\epsilon)L$ and each planted motif c_i is selected in Step 1 of the voting algorithm with probability no less than $1-\frac{1}{3}\epsilon$. When $\frac{n}{\log n} \geq \frac{9}{2\epsilon^2}$, the probability that $t^* = c$ is no less than $1-\frac{4L}{n}$.*

Proof. Consider a position j , $1 \leq j \leq L$, such that $t[j] \neq c[j]$. Let $X_1, X_2, \ldots, X_n$ be the random variables such that $X_i = 1$, if $t_i[j] = c[j]$; $X_i = 0$, otherwise. For all motifs $c_1, c_2, \ldots, c_n$, the expected number of motifs c_i such that $c_i[j] = c[j]$ is $(\frac{1}{4}+\epsilon)n$. Let M^+ and M^- be the sets of planted motifs selected and not selected into $\{t_1, t_2, \ldots, t_n\}$, respectively. The probability that a planted motif is not selected in the voting algorithm is no more than $\frac{1}{3}\epsilon$. Therefore, the expectation of $|M^-|$ is no more than $\frac{1}{3}\epsilon n$. In the worse case, each motif in M^- has the same letter with c at position j. Then, the expected number of t_i's such that $t_i \in M^+$ and $t_i[j] = c[j]$ is no less than $(\frac{1}{4}+\epsilon)n - \frac{\epsilon}{3}n = (\frac{1}{4}+\frac{2}{3}\epsilon)n$. Let $X = \sum_{1\leq i\leq n} X_i$. We have $E(X) \geq (\frac{1}{4}+\frac{2}{3}\epsilon)n$. By Lemma 3.1,

$$\begin{aligned} Pr\left(X \leq \frac{n}{4}+\frac{1}{3}\epsilon\right) &\leq Pr\left(X \leq E(X)-\frac{1}{3}\epsilon n\right) \\ &\leq e^{-\frac{2}{9}\epsilon^2 n}. \end{aligned}$$

When $\frac{n}{\log n} \geq \frac{9}{2\epsilon^2}$, the probability is

$$Pr\left(X \leq \frac{n}{4}+\frac{1}{3}\epsilon\right) \leq \frac{1}{n}.$$

For a letter $u \in \Sigma\backslash\{c[j]\}$, let $Y_1, Y_2, \ldots, Y_n$ be the random variables such that $Y_i = 1$, if $t_i[j] = u$; $Y_i = 0$, otherwise. For all motifs $c_1, c_2, \ldots, c_n$, the expected number of c_i's with $c_i[j] = u$ is $(\frac{1}{4}-\frac{1}{3}\epsilon)n$. Note that the expected number of t_i's not being a planted motif is no more than $\frac{1}{3}\epsilon n$. In the worse case, each t_i not being a planted motif has letter u at position j. Then, the expected number of t_i's with $t_i[j] = u$ is no more than $(\frac{1}{4}-\frac{\epsilon}{3})n + \frac{\epsilon}{3}n = \frac{1}{4}n$. Let $Y = \sum_{1\leq i\leq n} Y_i$. It follows that $E(Y) \leq \frac{1}{4}n$. By Lemma 3.1,

$$\begin{aligned} Pr\left(Y \geq \frac{n}{4}+\frac{1}{3}\epsilon n\right) &\leq Pr\left(Y \geq E(Y)+\frac{1}{3}\epsilon n\right) \\ &\leq e^{-\frac{2}{9}\epsilon^2 n}. \end{aligned}$$

When $\frac{n}{\log n} \geq \frac{9}{2\epsilon^2}$, the probability that $t^*[j] = u$ is no more than n^{-1}. There are three possible letters in $\Sigma\backslash\{c[j]\}$. The probability that none of them has $\frac{n}{4}+\frac{1}{3}\epsilon n$ letters in column j is at least $1-\frac{3}{n}$. Therefore, the probability that $c[j]$ is the majority letter at column j is at least $1-\frac{4}{n}$.

Then, we consider a position j' such that $t[j'] = c[j']$. Similar to the previous case, we can prove that when $\frac{n}{\log n} \geq \frac{9}{2\epsilon^2}$, the probability that $t^*[j'] = c[j']$ is at least $1-\frac{4}{n}$.

Consider all the L positions in c, the probability that $t^* = c$ is no less than $1-\frac{4L}{n}$. □

From the Lemmas 3.2, 3.3 and 3.4, we get the following theorem.

Theorem 3.1. *When $\sum_{1\leq i\leq n} d(c,s_i)$ is the minimum in all strings in Σ^L, $L > \frac{9}{8\epsilon^4}\log\frac{3m}{\epsilon}$, and $\frac{n}{\log n} \geq \frac{9}{2\epsilon^2}$, the probability that Algorithm 2 can find the motif consensus c is no less than $1-\left(\frac{3}{4}\right)^r - \frac{4L}{n}$.*

Theorem 3.1 shows that m, L and n are all important factors of Selecting One Voting algorithm. The increase of n will increase the probability of finding the motif consensus. Many researchers believe that a large number of sequences containing mutated motifs can help us find motifs. This proof shows why motifs can be found more easily in many similar sequences than in a few sequences. That is the reason why multiple alignment of many similar sequences is important for finding useful biological information.

Suppose $\sum_{1\leq i\leq n} d(c,s_i)$ is the minimum in all strings in Σ^L. When the length of motif and the mutation rate can not guarantee that Algorithm 2 outputs the motif consensus with high probability, we can prove the algorithm is a good approximation algorithm for the planted closest substring problem. For output c_A of Algorithm 2,

$$\begin{aligned} E\left(\sum_{1\leq i\leq n} d(c_A,s_i)\right) &\leq E\left(\sum_{1\leq i\leq n} d(c_1,s_i)\right) \\ &= \left(\frac{3}{4}-\frac{4}{3}\epsilon^2\right)nL. \end{aligned}$$

For the optimal solution c,

$$E\left(\sum_{1\le i\le n} d(c,s_i)\right) = \left(\frac{3}{4}-\epsilon\right) nL.$$

Therefore, the expected approximation ratio of Selecting One Voting algorithm is $1+\frac{4}{3}\epsilon$.

In Selecting One Voting algorithm, we try $r(m-L+1)$ different starting patterns. For each starting pattern, the voting operation takes $O(Lmn)$ time. So the time complexity of the whole algorithm is $O(rLm^2n)$.

```
Algorithm 3
Input   A sequence set $S = \{s_1, s_2, \ldots, s_n\} \subset \Sigma^m$ and integers $L$, $r$ and $k$.
Output  A motif consensus with length $L$.
1. Repeat Steps 2-6 $r$ times.
2.   Randomly select $k$ sequences $s_{x_1}, s_{x_2}, \ldots, s_{x_k}$ from $S$ which have not been selected in previous rounds.
3.   For each $L$-mer set $\{a_1, a_2, \ldots, a_k\}$ where $a_1 \in P_{x_1}, a_2 \in P_{x_2}, \ldots, a_k \in P_{x_k}$, do
4.     Find a consensus string $t$ of $a_1, a_2, \ldots, a_k$. (If there are several consensus strings, randomly select one).
5.     Use the voting algorithm to find a consensus $t^*$ from starting pattern $t$.
6.     Add $t^*$ to candidate motif consensus set $C$.
7. Output motif consensus $c_B$ such that $\sum_{1\le i\le n} d(c_B, s_i)$ is the minimum in all candidates in $C$.
```

Fig. 3. Selecting k Voting algorithm.

3.3. Selecting *k* Voting Algorithm

Selecting One Voting algorithm only uses L-mers in input strings as starting patterns of voting. When motifs are short and mutation rate is high, Selecting One Voting algorithm may miss some good patterns and fail to find the motif consensus. To get more good patterns, one simple idea is to use a consensus string of several planted motifs as a starting pattern of voting. Intuitively, when we know k planted motifs, the consensus string of the k mutated motifs will be more similar to the unknown motif consensus than one mutated motif. Based on this observation, we give a new powerful Selecting k Voting algorithm. In details, we first randomly select k sequences $s_{x_1}, s_{x_2}, \ldots, s_{x_k}$ from S and select one L-mer in each of the k sequence to get k L-mers. By enumerating all L-mers in $s_{x_1}, s_{x_2}, \ldots, s_{x_k}$, the k planted motifs can be selected. Then, we find a consensus string of $c_{x_1}, c_{x_2}, \ldots, c_{x_k}$. When there are several consensus strings, we randomly select one. In this way, we can get a consensus string of the k planted motifs and use this consensus string as a starting pattern of voting. Similar to Algorithm 2, the above procedure is repeated r times. The algorithm is shown in Fig. 3.

We can show that Selecting k Voting algorithm is more powerful than the simple Selecting One Voting algorithm. Suppose k planted motifs are selected as $a_1, a_2, \ldots, a_k$. We consider one column in the multiple alignment of the k planted motifs. From the VM probabilistic model, the numbers of occurrences of the letters in Σ fit the multinomial distribution. Suppose $u \in \Sigma$ is selected as the majority letter, the number of occurrences of u is x_1, and the numbers of occurrences of the other three letters are x_2, x_3, x_4, respectively. Obviously, x_1, x_2, x_3 and x_4 are non-negative integers and $\sum_{i=1}^4 x_i = k$. In addition, x_1 is a maximum number in x_1, x_2, x_3, x_4. From the observation, we define a set $Q = \{(x_1,x_2,x_3,x_4) | x_1,x_2,x_3,x_4 \in Z^* \ \& \ \sum_{i=1}^4 x_i = k \ \& \ x_1 = \max_{i=1}^4 x_i\}$, where Z^* is the set of non-negative integers. Set Q contains all possible values of (x_1,x_2,x_3,x_4) such that u can be selected as the majority letter. Sometimes, more than one letter has the maximum number of occurrences. So, set Q can be divided into four disjoint subsets Q_1, Q_2, Q_3 and Q_4. If $(x_1,x_2,x_3,x_4) \in Q_i$, then there are i letters with the maximum number of occurrences. Based on the property of multinomial distribution, we have the following observation.

Observation 3.1. When $\{a_1, a_2, \ldots, a_k\} \subseteq \{c_1, c_2, \ldots, c_n\}$, for each letter $t[j]$ in the consensus string t, the probability that $t[j] = c[j]$ is

$$q = \sum_{i=1}^{4} \sum_{(x_1,x_2,x_3,x_4)\in Q_i} \frac{1}{i} \frac{k!}{x_1!x_2!x_3!x_4!} \left(\frac{1}{4}+\epsilon\right)^{x_1} \left(\frac{1}{4}-\frac{1}{3}\epsilon\right)^{x_2+x_3+x_4}.$$

Let $\frac{1}{4}+\delta = q$. When $k \ge 1$, we have $\delta \ge \epsilon$. For example, suppose the error rate of the VM model is $p = 0.27$ and $\epsilon = \frac{3}{4} - p = 0.48$. (Notice that $4/15 \approx 0.27$ and the error rate 0.27 is corresponding to the $(15, 4)$ FM model in the Motif Challenge Problem.) If $k = 3$, we have $\delta = \frac{5}{4}\epsilon + \frac{2}{3}\epsilon^2 - \frac{4}{3}\epsilon^3 \approx 0.60$. And the error rate of the consensus string of three planted motifs is $1 - q \approx 0.15$. The change from 0.27 to 0.15 can increase the accuracy of the voting algorithm significantly. Our experimental results also support this point. In addition, we can use Chernoff-Hoeffding bound to prove that when $k > \frac{9}{2}\epsilon^2$, $\delta > \frac{1}{4}$.

For Selecting k Voting algorithm, we can show similar result to Theorem 3.1. That is, when L and n are large enough, Selecting k Voting algorithm can find the motif consensus with high probability. The difference is that the condition $L > \frac{9}{8\epsilon^4} \log \frac{3m}{\epsilon}$ is changed to $L > \frac{9}{8\epsilon^2\delta^2} \log \frac{3m}{\epsilon}$, where $\delta \geq \epsilon$. Similar to Lemma 3.2, we have

Lemma 3.5. *The probability that one consensus string t with $d(c, t) \leq (\frac{3}{4} - \delta)L$ is selected in Algorithm 3 is at least $1 - \left(\frac{3}{4}\right)^r$.*

Suppose we find a consensus string t with $d(c, t) \leq (\frac{3}{4} - \delta)L$. For a letter $c_i[j]$ in c_i, the probability that $c_i[j] = t[j]$ is no less than $\frac{1}{4} + \frac{4}{3}\epsilon\delta$. Similar to Lemma 3.3, we have the following lemma.

Lemma 3.6. *When $d(c, t) \leq (\frac{3}{4} - \delta)L$ and $L > \frac{9}{8\epsilon^2\delta^2} \log \frac{3m}{\epsilon}$, for each string $s_i \in S$, the probability that c_i is selected in Step 1 of the voting algorithm is no less than $1 - \frac{1}{3}\epsilon$.*

When the consensus string of k motifs is used, the length of L can be reduced from $L > \frac{9}{8\epsilon^4} \log \frac{3m}{\epsilon}$ to $L > \frac{9}{8\epsilon^2\delta^2} \log \frac{3m}{\epsilon}$ where $\delta > \epsilon$. From this point of view, Selecting k Voting algorithm is more powerful than Selecting One Voting algorithm.

From Lemmas 3.4, 3.5 and 3.6, we get the following theorem.

Theorem 3.2. *When $\sum_{1 \leq i \leq n} d(c, s_i)$ is the minimum in all strings in Σ^L, $L > \frac{9}{8\epsilon^2\delta^2} \log \frac{3m}{\epsilon}$ and $\frac{n}{\log n} \geq \frac{9}{2\epsilon^2}$, the probability that Selecting k Voting algorithm can find the motif consensus c is no less than $1 - \left(\frac{3}{4}\right)^r - \frac{4L}{n}$.*

Suppose $\sum_{1 \leq i \leq n} d(c, s_i)$ is the minimum in all strings in Σ^L. When the mutate rate is so high that Selecting k Voting algorithm can not find the motif consensus, we can prove Selecting k Voting algorithm is a good approximation algorithm for the planted closest substring problem. For output c_B of Algorithm 3,

$$E\left(\sum_{1 \leq i \leq n} d(c_B, s_i)\right) \leq \left(\frac{3}{4} - \frac{4}{3}\epsilon\delta\right) nL.$$

For the optimal solution c,

$$E\left(\sum_{1 \leq i \leq n} d(c, s_i)\right) = \left(\frac{4}{3} - \epsilon\right) nL.$$

Therefore, the expected approximation ratio of Selecting k Voting algorithm is $1 + (1 - \frac{4}{3}\delta)\frac{4\epsilon}{3 - 4\epsilon}$. When k is large enough, δ is approximate to $\frac{3}{4}$, and the ratio is approximate to 1.

We try $r(m - L + 1)^k$ different L-mer sets in Step 3 of Algorithm 3. For each consensus string, the voting operation takes $O(Lmn)$ time. So the time complexity of the whole algorithm is $O(rLm^{k+1}n)$.

In practice, the length of input sequences is from several hundred to thousands. The time complexity of Selecting k Voting algorithm is too high to be practical. Therefore, we will introduce a progressive method to speed up Selecting k Voting algorithm in the next subsection.

3.4. Progressive Filtering Algorithm

In Selecting k Voting algorithm, we need to enumerate all possible L-mer sets of the selected k sequences. Then, we need to do voting operations for $r(m - L + 1)^k$ times. To speed up the algorithm, we can filter out random L-mers to decrease the number of voting operations. If the number of voting operations is decreased, the time complexity of the algorithm will be improved.

Consider two sequences s_i and $s_j \in S$. From the VM probabilistic model, $d(c_i, c_j)$ is a relatively small value. Intuitively, the distance $d(c_i, c_j)$ tends to be less than the distance between two random L-mers. The property of the pairwise distance gives us the inspiration of designing a progressive filtering algorithm.

In the Motif Challenge Problem proposed by Pevzner and Sze [1], the distance between a pair of planted motifs is often not the shortest in all L-mer pairs, because there are $O(m^2)$ random L-mer pairs. Many real data also has the same property. Although the distance between a pair of planted motifs may be not the shortest in all L-mer pairs, the distance is smaller than the distances of a large portion of two random L-mers. With the above analysis, we design a progressive filtering algorithm. In the selected k sequences $\{s_{x_1}, s_{x_2}, \ldots, s_{x_k}\} \subset S$ in Selecting k Voting algorithm, we first consider two sequences s_{x_1} and s_{x_2}. In all pairs of L-mers (t_1, t_2), where $t_1 \in P_{x_1}$ and $t_2 \in P_{x_2}$, we keep the best α pairs of L-mers based on $d(t_1, t_2)$ and delete other pairs, where α is an input parameter. The set of the α pairs of L-mers is represented by $\mathcal{S}_2$. In practice, we can set $\alpha \approx m^{1.5}$. The reason is that if $\alpha \approx m^{1.5}$, the planted motif c_{x_1} is contained in $m^{0.5}$ pairs on average. As a result, the probability that (c_{x_1}, c_{x_2}) is not in the $m^{1.5}$ pairs is small.

Then, we consider the third sequence s_{x_3}. For each $t_3 \in P_{x_3}$ and each pair $(t_1, t_2) \in \mathcal{S}_2$, we compute the sum of pair distance $d(t_1, t_2) + d(t_1, t_3) + d(t_2, t_3)$. Base on this value, we also keep the best α triples in set $\mathcal{S}_3$. Similarly, we do the same operation for $s_{x_4}, \ldots, s_{x_k}$. Finally, we get a set $\mathcal{S}_k$ containing α k-tuples and use the k-tuples to get consensus strings and do voting operations. The algorithm is shown in Fig. 4.

```
Algorithm 4
Input    A set of $k$ sequences $s_{x_1}, s_{x_2}, \ldots, s_{x_k} \in S \subset \Sigma^m$, and integers $L$ and $\alpha$.
Output   A set $\mathcal{S}_k$ with $\alpha$ $k$-tuples.
1. Find set $\mathcal{S}_2$ of the best $\alpha$ pairs of $L$-mers $(t_1, t_2) \in P_{x_1} \times P_{x_2}$ based on $d(t_1, t_2)$.
2. For $i = 3$ to $k$
3.     In $\mathcal{S}_{i-1} \times P_{x_i}$, find set $\mathcal{S}_i$ of the best $\alpha$ $i$-tuples $(t_1, t_2, \ldots, t_i)$ based on $\sum_{t', t'' \in \{t_1, t_2, \ldots, t_i\}} d(t', t'')$.
4. Output set $\mathcal{S}_k$.
```

Fig. 4. Progressive filtering algorithm.

The time complexity of the progressive filtering algorithm is $O(\alpha k^2 Lm)$. If we combine the progressive filtering algorithm and Selecting k Voting algorithm, the time complexity of the new algorithm is $O(\alpha r Lm(k^2 + n))$, which is much better than the original Selecting k Voting algorithm. When $\alpha = m^{1.5}$, the time complexity is $O(rLm^{2.5}(k^2 + n))$.

We note that the special case of the progressive filtering algorithm for $k = n$ can be used directly to find motifs. When $k = n$, the progressive filtering algorithm can output α different n-tuples. Then we can find a consensus string from the L-mers in each n-tuple and output the best consensus string.

3.5. Motif Refinement

In practice, we can use some heuristic methods to improve the accuracy of the voting algorithm. Here we introduce two methods.

First, after we get a consensus string t^* from a voting operation, we do not directly output t^*. We can use the resulting string t^* as a new starting pattern and do voting operation repeatedly until there is no further improvement.

Second, we can do local search based on a candidate motif consensus. For a candidate motif consensus t^*, we can change one letter in t^* to get a new motif consensus t^{**}. There are totally $L(|\Sigma|-1)$ ways to change t^*. From the $L(|\Sigma|-1)$ ways, we select the best way to change t^* based on the score function. The local search can be repeated until there is no further improvement.

There are some techniques in speeding up the local search in implementation. Let a be the selected L-mer with the minimum distance to t^* in s_i, and b another L-mer in s_i. Note that $d(t^{**}, b) \geq d(t^*, b) - 1$ and $d(t^{**}, a) \leq d(t^*, a) + 1$. If $d(t^*, b) \geq d(t^*, a) + 2$, then $d(t^{**}, b) \geq d(t^*, a) + 1 \geq d(t^{**}, a)$. Therefore, when we search an L-mer with the minimum distance to a new motif consensus t^{**}, it is not necessary to compare all L-mers in s_i with t^{**}. We only consider L-mers with distance no more than $d(t^*, s_i) + 1$ to t^*. This technique can increase the speed of local search dramatically. Another technique is to use the bit representation of L-mers. In this way, the distance between L-mers can be computed with bit operations, which are much faster than counting different letters. This is also an advantage of the local search method compared with the EM method, which needs to compute the likelihood of each L-mer and can not use bit operation strategy to speed up.

4. EXPERIMENTS

We implemented the algorithms in Java. The software is available upon request. To get more starting patterns, we used selection with replacement instead of selection without replacement in Step 2 of Algorithm 2 and Algorithm 3 in the implementation. We tested the algorithms on a PC with an AMD 2.0G CPU and 1.0G Memory. In the experiments, we generated several sets of simulated data. Basically, we followed the settings of the Motif Challenge Problem proposed by Pevzner and Sze [1].

First, we tested the algorithms on simulated data of VM model with a small mutation rate. The parameters were $m = 600$, $L = 15$ and $\epsilon = 0.55$. That is, the mutation rate was $p = \frac{3}{4} - 0.55 = 0.2$. To discover the relationship between n and the accuracy rates of the algorithms, we set $n = 3, 5, 10, 20, 40, 100$ and generated six groups of data respectively. In each group of data, we generated 1000 instances. Selecting One Voting algorithm and Selecting $k = 3$ Voting algorithm were tested with parameter $r = 20$. For an instance with $n = 20$, the running time of Selecting One Voting algorithm is 20.7 seconds. The time complexity of Selecting k Voting is too high to finish the tests in reasonable time. In the tests of Selecting k Voting algorithm, instead of using all L-mer sets of selected k sequences, we only used L-mer sets containing planted motifs to do voting operations. The performance of this method is similar to that of Selecting k Voting algorithm. So we use the results of this method for reference. The results are reported in Table 1.

Table 1. The percentages of correct outputs of Selecting One Voting algorithm and Selecting $k = 3$ Voting algorithm with $m = 600$, $L = 15$, $\epsilon = 0.55$ on the VM model.

	Selecting One	Selecting $k = 3$
$n = 3$	8.0	23.4
$n = 5$	34.1	52.7
$n = 10$	86.3	92.6
$n = 20$	99.7	99.9
$n = 40$	100	100
$n = 100$	100	100

Table 1 shows that when the error rate is not high, both algorithms can find the planted motifs in most cases. The results also show that the number of input sequences is an important factor of the accuracy rates of the algorithms. When n increases, the accuracy rates of the algorithms increase, which is consistent with the results in the proof in Section 3. This fact explains why motifs can be found more easily in many similar sequences than in a few sequences.

Second, we increased the error rate from 0.2 to 0.27 ($\epsilon = 0.48$). Notice that $4/15 \approx 0.27$ and the error rate 0.27 is corresponding to the $(15, 4)$ FM model in the Motif Challenge Problem. Similar to the previous tests, we set $m = 600$, $L = 15$ and $\epsilon = 0.48$, and generated six groups of simulated data each containing 1000 instances. The results are reported in Table 2.

Table 2. The percentages of correct outputs of Selecting One Voting algorithm, Selecting $k = 3$ Voting algorithm and Selecting $k = 3$ Voting algorithm with progressive filtering, with $m = 600$, $L = 15$, $\epsilon = 0.48$ on the VM model.

	Selecting One	Selecting $k = 3$	Selecting $k = 3$ with Progressive Filtering
$n = 3$	1.1	8.4	1
$n = 5$	6.5	21.7	5
$n = 10$	38.7	59.9	38
$n = 20$	89.6	94.9	88
$n = 40$	99.9	100	99
$n = 100$	100	100	100

When the error rate increases, the accuracy rates of Selecting One Voting algorithm decrease much faster than those of Selecting k Voting algorithm. The tests show that Selecting k Voting algorithm is more powerful than Selecting One Voting algorithm.

Although Selecting k Voting algorithm is powerful, the time complexity of the algorithm is too high. To speed up Selecting k Voting algorithm, we proposed a progressive filtering algorithm. To evaluate the progressive filtering algorithm, we selected 100 instances from each group of previous simulated data with error rate 0.27, and tested Selecting $k = 3$ Voting algorithm with progressive filtering on the instances. The parameters were set to $r = 20$ and $\alpha = 20000$. For an instance with $n = 20$, the running time of Selected k Voting algorithm with progressive filtering is 725 seconds. The results are also reported in Table 2. The experimental results show that Selecting k Voting algorithm with progressive filtering has good accuracy rates, while the running time is improved significantly compared with the original Selecting k Voting algorithm.

Since Selecting One Voting and Selecting $k = 3$ Voting algorithm with progressive filtering have good performance and short running time, we tested their performance on difficult cases. We compared the algorithms with the well known random projection algorithm [23]. We followed the test method of Table 1 in Ref. 23 and tested on several difficult FM models where $n = 20$, $m = 600$ and $(L, D) = (12, 3), (14, 4), (16, 5), (18, 6)$ and $(19, 6)$. For each model, 100 instances were generated. For Selecting One Voting, we selected all 20 sequences to do voting operations. For Selecting $k = 3$ Voting algorithm with progressive filtering, we set $r = 100$ and $\alpha = 20000$.

In the difficult FM models, some random L-mers in input sequences may have small distances to the motif consensus c and only a part of planted motifs can be found. We did tests on the difficult FM models and found that, in some extreme cases, the motif consensus c does not have the optimal score function, and another length L string c' with $d(c, c') = 1$ has the optimal score function $\sum_{1\le i\le n} d(c', s_i) < \sum_{1\le i\le n} d(c, s_i)$. In this case, even if our algorithm can find the motif consensus c, the motif consensus will not be output as the optimal solution.

In the experiments of FM model, we followed the objective function used in Refs. 1 and 23. We assumed that D was known and counted the number of mutated motifs with distance no more than D to the motif consensus as the score function. Although it is not practical to use a fixed D in real problems, the function is used for comparison.

The accuracy rates of the random projection algorithm are from Table 1 in Ref. 23. The details are listed in Table 3. Since each input instance contains only 20 sequences, Selecting One Voting algorithm can only select 20 planted motifs as the starting patterns. Moreover, the planted motifs have many errors. Therefore, the accuracy rates of Selecting One Voting algorithm are not good for some difficult models such as $(16, 5)$ and $(18, 6)$ models. For Selecting $k = 3$ Voting algorithm with progressive filtering, there are many possible starting patterns and the consensus strings contain less errors. Therefore, it performs well on the difficult models and outperforms the best known random projection algorithm. From the experimental results, we can conclude that, although the ideas of our algorithms are simple, our algorithms are effective and powerful in finding planted motifs.

Table 3. The percentages of correct outputs of the random projection algorithm, Selecting One Voting algorithm and Selecting $k = 3$ Voting algorithm with progressive filtering (The results of PROJECTION are from Table 1 in Ref. 23).

L	D	PROJECTION	Selecting One	Selecting $k = 3$ with progressive filtering
12	3	96	97	100
14	4	86	91	100
16	5	77	53	100
18	6	82	36	99
19	6	98	90	100

5. CONCLUSION

In this paper, we studied the planted motif problem. We proposed Selecting One Voting algorithm and Selecting k Voting algorithm for finding planted motifs. We formally proved the common belief that a large number of input sequences can help us find motifs. To speed up Selecting k Voting algorithm, we also gave a progressive filtering algorithm. The experimental results validated the relationship between the number of input sequences and the accuracy rates of our algorithms, and showed that Selecting k algorithm with progressive filtering is powerful in finding planted motifs in difficult planted motif models.

References

1. Pevzner PA, Sze S-H. Combinatorial approaches to finding subtle signals in DNA sequence. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)* 2000: 269–278.
2. Lanctot JK, Li M, Ma B, Wang S, Zhang L. Distinguishing string selection problems. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* 1999: 633–642.
3. Li M, Ma B, Wang L. Finding similar regions in many sequences. *Journal of Computer and System Sciences* 2002; **65**: 73–96.

4. Andoni A, Indyk P, Patrascu M. On the optimality of the dimensionality reduction method. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)* 2006: 449–458.
5. Ma B, Sun X. More efficient algorithms for closest string and substring problems. In *Proceedings of the 12th Annual International Conference on Research in Computational Molecular Biology (RECOMB)* 2008: 396-406.
6. Hertz GZ, Stormo GD. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics* 1999; **15**(7-8): 563–577.
7. Lawrence CE, Altschul SF, Boguski MS, Liu JS, Neuwald AF, Wootton JC. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* 1993; **262**(5131): 208–214.
8. Bailey TL, Elkan C. 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 (ISMB)* 1994: 28–36.
9. Bailey TL, Elkan C. The value of prior knowledge in discovering motifs with MEME. In *Proceedings of the Third International Conference on Intelligent Systems and Molecular Biology (ISMB)* 1995: 21–29.
10. Bailey TL, Elkan C. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning* 1995; **21**: 51–83.
11. Rocke E, Tompa M. An algorithm for finding novel gapped motifs in DNA sequences. In *Proceedings of the Second Annual International Conference on Research in Computational Molecular Biology (RECOMB)* 1998: 228–233.
12. Eskin E, Pevzner PA. Finding composite regulatory patterns in DNA sequences. *Bioinformatics* 2002; **18 Suppl 1**: S354–S363.
13. Blanchette M, Schwikowski B, Tompa M. An exact algorithm to identify motifs in orthologous sequences from multiple species. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)* 2000: 37–45.
14. Brazma A, Jonassen I, Vilo J, Ukkonen E. Predicting gene regulatory elements *in silico* on a genomic scale. *Genome Research* 1998; **8**(11): 1202–1215.
15. Price A, Ramabhadran S, Pevzner PA. Finding subtle motifs by branching from sample strings. *Bioinformatics* 2003; **19 Suppl. 2**: ii149–ii155.
16. Sinha S, Tompa M. Discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research* 2002; **30**(24): 5549–5560.
17. Prakash A, Blanchette M, Sinha S, Tompa M. Motif discovery in heterogeneous sequence data. *Pacific Symposium on Biocomputing* 2004: 348–359.
18. Tompa M. An exact method for finding short motifs in sequences, with application to the ribosome binding site problem. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB)* 1999: 262–271.
19. Tompa M, Li N, Bailey TL, Church GM, Moor BD, Eskin E, Favorov AV, Frith MC, Fu Y, Kent WJ, Makeev VJ, Mironov AA, Noble WS, Pavesi G, Pesole G, Régnier M, Simonis N, Sinha S, Thijs G, van Helden J, Vandenbogaert M, Weng Z, Workman C, Ye C, Zhu Z. Assessing computational tools for the discovery of transcription factor binding sites. *Nature Biotechnology* 2005; **23**(1): 137–144.
20. van Helden J, André B, Collado-Vides J. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *Journal of Molecular Biology* 1998; **281**(5): 827–842.
21. Keich U, Pevzner PA. Finding motifs in the twilight zone. *Bioinformatics* 2002; **18**(10): 1374–1381.
22. Keich U, Pevzner, PA. Subtle motifs: defining the limits of motif finding algorithms. *Bioinformatics* 2002; **18**(10): 1382–1390.
23. Buhler J, Tompa M. Finding motifs using random projections. *Journal of Computational Biology* 2002; **9**(2): 225–242.
24. Hoeffding W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association* 1963; **58**(301): 13–30.