

VOTING ALGORITHMS FOR DISCOVERING LONG MOTIFS*

FRANCIS Y.L. CHIN AND HENRY C.M. LEUNG[†]

*Department of Computer Science
The University of Hong Kong
Pokfulam, Hong Kong*

Pevzner and Sze [14] have introduced the Planted (l,d) -Motif Problem to find the similar patterns (motif) in sequences which represent the promoter region of co-regulated genes. l is the length of the motif and d is the maximum Hamming distance around the similar patterns. Many algorithms have been developed to solve this motif problem. However, these algorithms have the problems that the running times are too long or do not guarantee the motif can be found. In this paper, we introduce new algorithms to solve the motif problem. Our algorithms can find motif in reasonable time not only for the challenging $(9,2)$, $(11,3)$, $(15,5)$ -motif problems but even for longer motifs, say $(20,7)$, $(30,11)$ and $(40,15)$, which has never been seriously attempted by other researchers because of the high time and space complexities.

1 Introduction

Understanding the gene regulatory network, how genes cooperate to perform functions, is an important problem in Bioinformatics. A subproblem is to discover the common pattern of the transcription factor binding sites for co-regulatory genes.

In order to start the gene expression process, a molecule called transcription factor will bind to a short substring in the promoter region of the gene. We call this substring the binding site of the transcription factor. One kind of transcription factor can bind to several binding sites in the promoter regions of different genes to make these genes co-regulating, thus, these binding sites should have similar patterns. The motif discovering problem is to find the common patterns, motif, in the promoter regions of a set of co-regulatory genes.

Many algorithms [1-3,5-18] have been introduced to solve this problem based on different assumptions. Pevzner and Sze [14] define a very precise version of this motif discovery problem which has also been considered in [3,12,15].

Planted (l,d) -Motif Problem: Suppose there is a fixed but unknown nucleotide sequence M (the motif) of length l . Given t length- n nucleotide sequences, and each sequence contains a planted variant of M , we want to determine M without knowing the positions of the planted variants. A variant is a substring derivable from M with at most d point substitutions.

A number of algorithms have been introduced to solve this problem. They can be classified into three categories: brute-force, clique search and heuristic search.

Brute-force algorithms [2,7,13,16-18] try to test all 4^l possible motifs. Although these algorithms guarantee that the motif can be found, their running times increase exponentially with l . Therefore, these algorithms are not suitable for discovering long motifs in practice.

* The research was supported in parts by the RGC grant HKU 7135/04E

[†] email address : {chin,cmleung2}@cs.hku.hk

Algorithms using clique search approach [12,14] construct a t -partite graph G . Each partite contains $n - l + 1$ nodes which represent all length- l substrings in an input sequence. Two nodes in different partites will be joined by an edge if the Hamming distance between the two corresponding length- l substrings is less than or equal to $2d$. The Planted (l,d) -Motif Problem is reduced to finding a clique of size t in graph G . These algorithms can handle longer motif than the brute-force algorithms can. However, since the number of edges increases with the value of d , these algorithms fail to find the motif when the number of edges in the graph is large, which includes the (9,2), (11,3), (15,5)-motif problems.

Algorithms based on heuristic search [1,3,5-6,8-10] first find out a small set of length- l sequences with high probability to be the motif, then refine these sequences by some local optimization searching techniques, e.g., EM-algorithm, Gibbs Sampling, etc. Although these algorithms may solve the challenging (9,2), (11,3), (15,5)-motif problems in practice, there is no guarantee that the motif can be discovered even for a short motif.

Our contribution includes a new Voting Algorithm to find the motif. Similar to the brute-force algorithms, it guarantees the finding of the motif. However, the running time of this algorithm increases exponentially with d instead of l . A problem of this algorithm is that its storage requirement increases with l . Modifications of the Voting Algorithm have been introduced to reduce the storage requirement at the expense of a slight increase of running time. Thus, the modified Voting Algorithm works well with long motif as long as d is small, say 4 or 5 and can handle longer motif than those brute-force algorithms, e.g., the challenging problems like (9,2), (11,3), (15,5)-motif problems. When $l > 15$, say the (20,7), (30,11) and (40,15)-motif problem, even the modified Voting Algorithm will fail because its high time and space complexities. As far as we know, no known software can discover motifs for these large l and d . In this paper, we have introduced two improvements on our Voting Algorithm. Instead of considering all positions of the motif, our Voting Algorithm only works at a *subset* of these l positions, the “projection” onto l' positions. Based on the voting results on these l' positions, we have high probability to find the motif of length l . In fact, the l' positions can be chosen randomly and the probability of success can increase tremendously if different sets of positions are tried. Besides choosing the sets of positions at random, we can have a better result if these positions are the complement set of the previous l' positions. Experiments on simulated data show that the Voting Algorithm with projection can discover long motifs, e.g., the (40,15)-motif problem with over 95% successful rate. Note that Buhler et al [3] has shown that no algorithms can discover the motif when the value of l is small while the value of d is large because there are many random length- l sequences which can be taken as motifs. Examples of unsolvable cases include (9,3), (11,4), (15,6)-motif problems.

This paper is organized as follows. We describe the idea of the Voting Algorithm in Section 2 and the heuristic improvements in Section 3. Experimental results of the algorithm on both real data and simulated data are shown in Section 4, followed by a discussion in Section 5.

2 Voting Algorithm

In this section, we will first describe the basic idea of the Voting Algorithm which runs faster than the brute-force algorithms without compromising its effectiveness.

First, we define a length- l sequence (substring) s' to be a d -variant (or simply variant) of another length- l sequence (substring) s if the Hamming distance between s' and s is at most d . $N(s,d)$ is a set that contains all d -variants of a length- l sequence s . Note that all planted variants m_i of the motif M in the input sequences are in the set $N(M,d)$. At the same time, M is also in $N(m_i,d)$ for all planted variants m_i of M . The idea of the Basic Voting Algorithm is that each length- l substring σ in the input sequences gives one mark to all length- l sequences in $N(\sigma,d)$ with the restriction that each length- l sequence can get at most one mark from each length- n input sequence, the motif M will get exactly t marks. Algorithm 1 outlines the procedure for the Basic Voting Algorithm, where $S_i[j]$ is the j -th character in the i -th input sequence and $H(s)$ is the hash value of a length- l sequence s .

Algorithm 1: Basic Voting Algorithm

```

1: Create two hash tables V and R and set the value of each entry be 0
   {Table V keeps the number of marks received by each length- $l$  sequence}
   {Table R ensures that each length- $l$  sequence receives at most one mark from each input
   sequence}
2:  $C \leftarrow \phi$                                      {set of motifs}
3: for  $i \leftarrow 1$  to  $t$ 
4:   do for  $j \leftarrow 1$  to  $n - l + 1$ 
5:     do for each length- $l$  sequence  $s$  in  $N(S_i[j \dots j + l - 1], d)$ 
6:       do if  $R[H(s)] < i$ 
7:         then  $V[H(s)] \leftarrow V[H(s)] + 1$ 
8:            $R[H(s)] \leftarrow i$ 
9:   for each length- $l$  sequence  $s$ 
10:  do if  $v[H(s)] = t$ 
11:    then insert  $s$  into  $C$ 

```

According to the definition of the Planted (l,d) -Motif Problem, each input sequence contains a variant of motif M . If a length- l sequence does not have any variant on one of the input sequence, it will not be the motif and will not be stored in the tables, therefore the storing space can be reduced. It is easy to verify the correctness of the Basic Voting Algorithm, so we shall omit the proof here. Theorem 1 prove that the time and space complexities of the algorithm are $O(nt(3l)^d)$ and $O(n(3l)^d + nt)$ respectively.

Theorem 1: The time and space complexities of the Basic Voting Algorithm are $O(nt(3l)^d)$ and $O(n(3l)^d + nt)$ respectively.

Proof: Let $K(l,d)$ be the size of $N(\sigma,d)$ for any length- l substring σ . $K(l,d) = \sum_{i=0}^d C_i^l 3^i = O((3l)^d)$. We need to access the tables $O(ntK(l,d))$ times and scan the input sequences once. As we can find the hash value of each variant of a length- l substring $S_i[j \dots j + l - 1]$ in constant time, the time complexity of the algorithm is $O(ntK(l,d) + nt) = O(ntK(l,d)) = O(nt(3l)^d)$. For each length- l substring in the first input sequence, it has $K(l,d)$ variants. Therefore, at most $(n-l+1)K(l,d)$ sequences will get one mark after the first iteration of i . The size of the two tables are at most $(n-l+1)K(l,d)$ because we will not store marks of the rest sequences which have not a variant in the first input sequence. Since the space need to store the input sequences is $O(nt)$, the space complexity of the algorithm is $O(n(3l)^d + nt)$. \square

On the other hand, the brute-force algorithms take $O(nt4^l)$ time and $O(nt)$ space. Although the Basic Voting Algorithm runs faster than the brute-force algorithms do, the space needed grows exponentially with d . Thus, it cannot handle long motif with large Hamming distance d because its running time and space requirement increase exponentially with d . A method to reduce the space complexity is to divide the 4^l length- l sequences into groups. We group the 4^l length- l sequences s according to their suffixes of length l' . Two length- l sequences are in the same group if and only if their suffixes are the same. At each iteration, for each substring σ in the input sequences, we give one mark to its variants with a particular suffix. Theorem 2 prove that the time and space complexity of this modified algorithm are $O(nt(3l)^d + nt4^{l'})$ and $O(n(3(l-l'))^d + nt)$ respectively when l' is smaller than $\log_4(3l)^d$.

Theorem 2: The time and space complexity of the modified Voting Algorithm are $O(nt(3l)^d + nt4^{l'})$ and $O(n(3(l-l'))^d + nt)$ respectively when l' is smaller than $\log_4(3l)^d$.

Proof: Although we have divided the length- l sequences into $4^{l'}$ groups, the total number of marks received by the tables remains $O(nK(l,d))$. However, the input sequences will be scanned $4^{l'}$ times, so the modified Voting Algorithm will take $O(nt(3l)^d + nt4^{l'})$. Since for any length- l substring σ , the maximum number of variants with a particular suffix is $K(l-l',d)$, the space complexity for the two tables decreases from $O(n(3l)^d + nt)$ to $O(n(3(l-l'))^d + nt)$. \square

3 Heuristic Improvement

Although the Voting Algorithm can solve the Planted (l,d) -Motif Problem for many l and d including the challenging (9,2), (11,3), (15,5)-motif problems, its running time increases exponentially with d and the length of suffix. Therefore, it cannot handle problem with large l and d . In order to handle longer motif, we introduce two heuristic improvements for the Voting Algorithm.

3.1 Random Projection

When l is large, say $l > 15$, the time required for discovering the motif becomes prohibitively long when $d > 5$. We try to reduce the size of l by projecting all length- l substrings onto a subset of these l positions. This subset of positions can be randomly chosen and the size of the subset, say l' , should be small enough to be solvable by the previous Voting Algorithm. Similar projection idea was used by Buhler et al[3] in which only the number of length- l substrings projected to each length- l' sequence is used for selection of seed sequences. However, in our algorithm, the Voting Algorithm is applied to the projected length- l' sequences for saving time and space.

Denote $\text{HD}(s,s')$ be the Hamming distance between sequences s and s' . Let B be a subset of l' positions from $\{1, \dots, l\}$. A projection $\text{proj}(s,B)$ of a length- l sequence s is the length- l' sequence constructed by projecting the l' characters from s at the positions specified by B .

Our approach is to perform voting on these length- l' projected sequences. For each length- l substring σ in the input sequences, one mark will be given to a length- l' sequence s if $\text{HD}(\text{proj}(\sigma, B), s) \leq \lceil dl'/l \rceil$. In general, for a length- l variant m_i of M , i.e. $\text{HD}(m_i, M) \leq d$, it is expected that the length- l' sequence $\text{proj}(m_i, B)$ is also a $\lceil dl'/l \rceil$ -variant of $\text{proj}(M, B)$, i.e.

$\text{HD}(\text{proj}(m_i, B), \text{proj}(M, B)) \leq \lceil dl'/l \rceil$, these $\text{proj}(M, B)$ will be marked. However, even if M has t various $\{m_i\}$, $\text{proj}(M, B)$ may not get exactly t marks because of the following cases.

- (i) $\text{proj}(M, B)$ is not marked by some planted variant m_i because $\text{HD}(\text{proj}(m_i, B), \text{proj}(M, B)) > \lceil dl'/l \rceil$.
- (ii) $\text{proj}(M, B)$ is marked by a substring σ even though $\text{HD}(\sigma, M) > d$ because $\text{HD}(\text{proj}(\sigma, B), \text{proj}(M, B)) \leq \lceil dl'/l \rceil$.

We shall show later that when l' is comparatively large with respect to l , say $l' \sim 2l/3$, it is highly probable that $\text{proj}(M, B)$ will receive marks from the plant variants m_i of motif M .

Let $\{v_i\}$ be the set of length- l substrings which mark $\text{proj}(M, B)$. Although $\{m_i\}$ and $\{v_i\}$ may be different because of the above cases, large proportion of substrings $\{m_i\}$, say t' length- l variants of M , are in $\{v_i\}$ and t' should slightly less than t . Thus, $\text{proj}(M, B)$ will receive high marks and will be used to identify $\{v_i\}$ in the input sequences. In order to discover M from $\{v_i\}$, one simple method is to give one mark to each variants of v_i again, motif M should get at least t' marks and can be discovered. However, when d increases, this method may fail as the number of variants for v_i increases exponentially with d . For example, even the number of v_i of a (40,15)-motif problem is small, the time needed for discovering M from $\{v_i\}$ may also be prohibitively long.

Another approach to find the motif from $\{v_i\}$ is reducing it to the clique search problem. We construct a graph G where each node represents a length- l sequence in $\{v_i\}$. An edge exists between two nodes if the Hamming distance between the corresponding sequences is at most $2d$. Since there are t' variants m_i of M in $\{v_i\}$, these variants will form a clique of at least size t' . Thus the motif M can be discovered. In practice, the running time for finding the maximum clique is acceptable [23] as the size of the graph G is usually very small.

Theorem 3: Given a random set B of size l' and t length- l planted variants of a motif M with at most d substitutions, the probability $P_t(l, l', d, t, t')$ that “at least t' out of the t variants give mark to $\text{proj}(M, B)$ after performed projection according to B ” is at least

$$\sum_{i=t'}^t C_i^t p(l, l', d)^i (1 - p(l, l', d))^{t-i} \quad \text{where } p(l, l', d) = \sum_{i=0}^{\lceil dl'/l \rceil} \frac{C_i^d \cdot C_{l'-i}^{l-d}}{C_{l'}^l}$$

Proof: Let $p(l, l', d)$ be the probability that $\text{HD}(\text{proj}(m_i, B), \text{proj}(M, B)) \leq \lceil dl'/l \rceil$ for a variant m_i of M with exactly d substitutions. Since there are $C_i^d \cdot C_{l'-i}^{l-d}$ out of $C_{l'}^l$ possible B such that $\text{HD}(\text{proj}(m_i, B), \text{proj}(M, B)) = i$, $p(l, l', d) = \sum_{i=0}^{\lceil dl'/l \rceil} C_i^d \cdot C_{l'-i}^{l-d} / C_{l'}^l$. $P_t(l, l', d, t, t')$ has the minimum value when $\text{HD}(m_i, M) = d$ for all variants m_i , which is equal to $\sum_{i=t'}^t C_i^t p(l, l', d)^i (1 - p(l, l', d))^{t-i}$ using binomial distribution, Therefore $P_t(l, l', d, t, t')$ is at least $\sum_{i=t'}^t C_i^t p(l, l', d)^i (1 - p(l, l', d))^{t-i}$.

□

Table 1: $P_t(l, l', d, t, t')$ for different l, d and t' when $t = 20$ and $l' = 2l/3$

L	d	t'	$P_t(l, l', d, t, t')$	l	d	t'	$P_t(l, l', d, t, t')$
15	3	13	0.869217	24	9	13	0.664495
		14	0.740677			14	0.482605
		15	0.561257			15	0.300016
18	6	13	0.754415	27	9	13	0.641744
		14	0.585880			14	0.458359

		15	0.394541			15	0.279448
21	6	13	0.729136	30	12	13	0.600934
		14	0.555538			14	0.416533
		15	0.365551			15	0.245322

Table 1 shows the value of $P_r(l, l', d, t, t')$ for different values of l , d and t' when $t = 20$ and $l' = 2l/3$, e.g., the probability that there are at least 14 variants of M in the variants set of $\text{proj}(M, B)$ is larger than 0.4165 when $l = 30$, $d = 12$ and $t' = 14$ which is much larger than the probability when the sequences are randomly generated. Although this probability $P_r(l, l', d, t, t')$ might not be large enough to guarantee the finding of M , we can repeat the process with different sets of positions B to increase the probability of discovering M . With respect to the above example, if we repeat this process 10 times for $t = 20$, the probability that 14 or more variants of motif M are in $\{v_i\}$ will increase to $1 - (1 - 0.4165)^{10} = 0.9954$.

Table 2: $P_h(l, d, t, t')$ for different l , d and t' when $t = 20$

l	d	t'	$P_h(l, d, t, t')$	l	d	t'	$P_h(l, d, t, t')$
20	6	13	0.9772	26	8	13	0.9522
		14	0.8773			14	0.8008
		15	0.6563			15	0.5432
22	6	13	0.9743	28	10	13	0.9337
		14	0.8672			14	0.7551
		15	0.6397			15	0.4871
24	8	13	0.9565	30	10	13	0.9285
		14	0.8123			14	0.7433
		15	0.5585			15	0.4737

3.2 Improved Random Projection

Although we have high probability to discover the motif using Random Projection, we can further increase this probability by considering the complement of the set B of positions.

Consider a set B of $\lfloor l/2 \rfloor$ positions, define B^c be the complement of B , i.e. $\{1, \dots, l\} - B$. If m_i is a length- l planted variant of the motif M , then either $\text{HD}(\text{proj}(m_i, B), \text{proj}(M, B)) \leq \lfloor d/2 \rfloor$ or $\text{HD}(\text{proj}(m_i, B^c), \text{proj}(M, B^c)) \leq \lfloor d/2 \rfloor$. Let $\{v_i\}$ and $\{v_i^c\}$ be the set of length- l variants obtained from $\text{proj}(M, B)$ and $\text{proj}(M, B^c)$ respectively. At least half of the variants of motif M should be in $\{v_i\}$ or $\{v_i^c\}$. Calculation of the probability $P_h(l, d, t, t')$ that at least t' of the t variants of a length- l motif M are in $\{v_i\}$ or in $\{v_i^c\}$ is shown in Theorem 4.

Theorem 4: Given a random set B of size $l/2$ and t length- l planted variants of a motif M with at most d substitutions, the probability $P_h(l, d, t, t')$ that either $\text{proj}(M, B)$ or $\text{proj}(M, B^c)$ gets at least t' marks from the t variants when performing random projection is at least

$$\sum_{j=0}^{t'} \left[\binom{t}{j} p_e^j (1-p_e)^{t-j} \cdot \left(\sum_{k=\{0, \dots, t-t', t'-j, \dots, t-j\}} \binom{t-j}{k} \left(\frac{1}{2}\right)^{t-j} \right) \right] \text{ where } p_e = \frac{C_{d/2}^d \cdot C_{(l-d)/2}^{(l-d)}}{C_{l/2}^l}$$

Proof: Without loss of generality, assume both l and d are even. Let p_e be the probability that a variant m_i of M with exactly d substitutions gives mark to both $\text{proj}(M, B)$ and $\text{proj}(M, B^c)$. It would happen only when $\text{HD}(\text{proj}(m_i, B), \text{proj}(M, B)) = \text{HD}(\text{proj}(m_i, B^c), \text{proj}(M, B^c)) = d/2$. It means that the set B contains exactly $(l-d)/2$ positions that m_i equals M . Since there are $C_{d/2}^d \cdot C_{(l-d)/2}^{l-d}$ out of $C_{l/2}^l$ possible B satisfy this requirement, $p_e = C_{d/2}^d \cdot C_{(l-d)/2}^{l-d} / C_{l/2}^l$.

$$\begin{aligned}
& P_h(l, d, t, t') \\
\geq & \text{The probability that there are } t' \text{ or more } d\text{-variants } m_i \text{ satisfy } \text{HD}(\text{proj}(m_i, B), \text{proj}(M, B)) \leq d/2 \\
& \text{or there are } t' \text{ or more } d\text{-variants } m_i \text{ satisfy } \text{HD}(\text{proj}(m_i, B^c), \text{proj}(M, B^c)) \leq d/2 \text{ given that} \\
& \text{HD}(m_i, M) = d \text{ for all } d\text{-variants } m_i \\
= & \sum_{j=0}^t P(\exists j \text{ variants } m_i \text{ s.t. } \text{HD}(\text{proj}(m_i, B), \text{proj}(M, B)) = \text{HD}(\text{proj}(m_i, B^c), \text{proj}(M, B^c)) = d/2) \\
& \bullet P(\exists t' - j \text{ or more variants } m_i \text{ in the rest } t - j \text{ variants s.t. } \text{HD}(\text{proj}(m_i, B), \text{proj}(M, B)) \leq d/2 \\
& \text{or } \exists t - t' \text{ or less variants } m_i \text{ in the rest } t - j \text{ variants s.t. } \text{HD}(\text{proj}(m_i, B), \text{proj}(M, B)) \leq d/2) \\
= & \sum_{j=0}^t \left[C_j^t p_e^j (1-p_e)^{t-j} \bullet \left(\sum_{k=\{0, \dots, t-t', t'-j, \dots, t-j\}} C_k^{t-j} \left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{t-j-k} \right) \right] \\
= & \sum_{j=0}^t \left[C_j^t p_e^j (1-p_e)^{t-j} \bullet \left(\sum_{k=\{0, \dots, t-t', t'-j, \dots, t-j\}} C_k^{t-j} \left(\frac{1}{2}\right)^{t-j} \right) \right]
\end{aligned}$$

□

The values of $P(l, d, t, t')$ for different l , d and t' when $t = 20$ are shown in Table 2. Although the probability $P(l, d, t, t')$ decreases with l , the probability “at least 14 of the 20 variants of motif M are in $\{v_i\}$ derived from $\text{proj}(M, B)$ or in set $\{v_i^c\}$ derived from $\text{proj}(M, B^c)$ ” is larger than 0.7433 (an increase from 0.4165 of the random projection method). Note that $P(l, d, t, t') = 1$ for all $t' \leq t/2$, therefore at least $t/2$ variants $\{m_i\}$ are in $\{v_i\}$ or $\{v_i^c\}$. Similar to the Random Projection, we can take different random sets B so as to increase this probability. For example, if we repeat the process 5 times, the probability of at least 14 out of 20 variants in $\{v_i\}$ or $\{v_i^c\}$ will be $1 - (1 - 0.7433)^5 = 0.9989$.

When using this improved Random Projection approach, we should be careful that the motif must be long. If the length of the motif is short, say 16bp, the length of the short motif (8bp) is too short that there will be many random sequences having a lot of variants. The running time of the Voting Algorithm will increase, as we have to find the length- l variants for a huge number of short motifs and to discover the corresponding candidate motif.

On the other hand, if the length of the motif is sufficient long, say 40, the improved Random Projection Algorithm should be applied to reduce to a motif problem of length 20, which can further be solved by the Random Projection Algorithm.

Table 3: Suggested heuristic improvement used in different situations “S” means using the Voting Algorithm without heuristic improvement. “RP” means using the random projection with $l' = 2l/3$. “RPH” means using the random projection with the complement set ($l' = l/2$)

	$l < 15$	$15 \leq l \leq 20$	$20 < l$
$d \leq 3$	S	S	S

$3 < d \leq 5$	S	S	RPH
$d > 5$	S	RP	RPH

4 Experimental results

In this section, we describe the test results of the Voting Algorithm for both simulated and real biological data. All the experiments were taken on a 2.4GHz P4 CPU with 512Mb memory.

4.1 Simulated Data

We tested the performances of brute-force algorithm, Voting Algorithm and Voting Algorithm with heuristic improvement on different Planted (l,d) -Motif Problem. Table 3 shows the suggested heuristic improvement with respected to different l and d .

All input instances contain $t = 20$ sequences each of length 600. Each nucleotide ('A', 'C', 'G' and 'T') of the input sequences was generated independently with the same occurrence probability. A motif M of length- l was randomly picked and a variant was planted to each input sequence. Each algorithm could output at most 20 solutions (the length- l sequences with at least one variant in each input sequence). The minimum Hamming distance between the planted motif and the 20 solutions are records. For each set of parameter l and d , we ran 50 test cases, recorded the average Hamming distance and the average running time of each algorithm.

Table 4 shows the results of the experiments. The third column is the maximum value of d for the corresponding l such that the Planted (l,d) -Motif Problem can still be solved theoretically. Buhler et al[3] introduced the expected number $E(l,d)$ of length- l random sequences that have one variant in each input sequence. When d is too large such that $E(l,d)$ is large, no algorithm can determine the motif from the set of random sequences with a variant in each input sequence. In other words, $\max\{d \mid E(l,d) < \text{a small constant}\}$ gives the maximum d that the (l,d) -motif problem can be solved with high probability.

Table 4: Experimental results on simulated data of the brute-force algorithm, Voting Algorithm and Voting Algorithm with heuristic improvement We run 50 test cases for each set of parameters and record the average running time and the hamming distance between the planted motif and the solution output by the three programs. “-“ means that the running time of the program is too long (at least more than one day).

l	d	Max d for	Brute-force		Voting		Voting with Heuristic Improvement	
		$E(l,d) < 10$	HD	time	HD	time	HD	time
7	1	1	0	61.6 s	0	<1 s	The results are the same as	
9	2	2	0	17.9 m	0	0.4 s	the Voting algorithm as no	
11	3	3	0	4.8 h	0	8.6 s	heuristic improvement is	
13	4	4	-	-	0	108.s	performed when $l < 15$	
15	5	5	-	-	0.2	22 m	0.2	113.6 s
20	7	7	-	-	-	-	0	111.4 s
30	11	13	-	-	-	-	0.11	124.1 s
40	15	18	-	-	-	-	0.1	125.2 s

Since the brute-force algorithm takes $O(nt^d)$ time, the running time for discovering a length-11 motif is over 4.8h and it cannot handle longer motif in reasonable time. For the Voting Algorithm, although it can solve the Planted (l,d) -Motif Problem for longer motif than the brute-force algorithm can, it cannot handle those problem when d is larger than 5 as its running time increases exponentially with d . With heuristic improvement, the Voting Algorithm can handle longer motif with large d even for the (40,15)-motif problem in one minute.

4.2 Real Biological Data

SCPD [19] contains different transcription factors for yeast. For each set of genes regulated by the same transcription factor, we chose the 600 bp in the upstream of the genes as the input sequences T . The lengths of the motifs were same as those of the published motifs and d was either 0 or 1. Experimental results are showed in Table 5. The Voting Algorithm could discover the motifs for these data sets. Besides, since the lengths of all these motifs were within 10bp, the running time of the Voting Algorithm was within one second for each data set.

Table 5: Experiment result on real biological data The data are collected from the SCPD. For each set of data, we look for the motifs with length equals to the published motif and d equals to 1.

Transcription Factor	Published Motif pattern	Motif Pattern Found
GCR1	CWTCC	CTTCC
GATA	CTTATC	CTTAT
CCBF,SCB,SWI6	CNCGAAA	CGCGAAA
CuRE,MAC1	TTTGCTC	TTTGCTC
GCFAR	CCCGGG	CCCGGG
GCN1	TAATCTAATC	TAATCTAATC

5 Discussion

In this paper, we have introduced the Voting Algorithm for solving the Planted (l,d) -Motif Problem. It guarantees that the motif can be found when d is small and with high probability for large l and d . Experimental results have indicated that our algorithm works quite well for both simulated data and real data.

An open problem of interest is to extend the Voting Algorithm to handle those variants within d from motif M in edit distance instead of Hamming distance. When d is small, this problem can be solved by redefining the variant set $N(\sigma,d)$ of a length- l substring σ . However, the heuristic improvement may not work when both l and d are large and new methods should be needed to handle these cases.

References

1. Timothy Bailey and Charles Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21:51-80, 1995
2. Alvis Brazma, Inge Jonassen, Ingvar Eidhammer, and David Gilbert. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, 5:279-305, 1998.
3. Jeremy Buhler and Martin Tompa. Finding motifs using random projections. *RECOMB01*, p69-76, 2001.
4. Norishige Chiba and Takao Nishizeki. Arboricity and Subgraph Listing Algorithm. *SIAM Journal on Computing*, 14:210-223, 1985
5. Y. Fraenkel, Y. Mandel, D. Friedberg, and H. Margalit. Identification of common motifs in unaligned dna sequences: application to *Escherichia coli* Irp regulon. *Bioinformatics*, 11:379-387, 1995.
6. M. Gelfand, E. Koonin, and A. Mironov. Prediction of transcription regulatory sites in archaea by a comparative genomic approach. *Nucleic Acids Res.*, 28:695-705, 2000.
7. J. van Helden, B. Andre, and J. Collado-Vides. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *Journal of Molecular Biology*, 281(5):827-842, 1998.
8. G. Z. Hertz and G. D. Stormo. Identification of consensus patterns in unaligned dna and protein sequences: a large-deviation statistical basis for penalizing gaps. *The Third International Conference on Bioinformatics and Genome Research*, p201-216, 1995
9. Charles Lawrence, Stephen Altschul, Mark Boguski, Jun Liu, Andrew Neuwald and John Wootton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262:208-214, 1993
10. C. Lawrence and A. Reilly. An expectation maximization (em) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins: Structure, Function and Genetics*, 7:41-51, 1990
11. Ming Li, Bin Ma, and Lusheng Wang. Finding similar regions in many strings. *Journal of Computer and System Sciences* 65:73-96, 2002
12. Shoudan Liang. cWINNOWER Algorithm for Finding Fuzzy DNA Motifs. *Computer Society Bioinformatics Conference*, p260-265, 2003
13. G. Pesole, N. Prunella, S. Liuni, M. Attimonelli, and C. Saccone. Wordup: an efficient algorithm for discovering statistically significant patterns in dna sequences. *Nucleic Acids Research*, 20(11):2871-2875, 1992.
14. Pavel Pevzner and Sing-Hoi Sze. Combinatorial approaches to finding subtle signals in dna sequences. *The Eighth International Conference on Intelligent Systems for Molecular Biology*, p269-278, 2000.
15. Marie-France Sagot. Spelling approximate repeated or common motifs using a suffix tree. In C.L. Lucchesi and A.V. Moura editors, *Latin'98: Theoretical informatics*, volume 1380 of *Lecture Notes in Computer Science*, p111-127, 1998.
16. Roger Staden. Methods for discovering novel motifs in nucleic acid sequences. *Computer Applications in Biosciences*, 5(4):293-298, 1989
17. Martin Tompa. 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*, p262-271, 1999.
18. F. Wolfertsteeter, Kornelie Frech, Grit Herrmann, and Thomas Wernet. Identification of functional elements in unaligned nucleic acid sequences by a novel tuple search algorithm. *Computer Applications in Bio-sciences*, 12(1):71-80, 1996.

19. Jian Zhu and Michael Zhang. SCPD: a promoter database of the yeast *Saccharomyces cerevisiae*. *Bioinformatics* 15:563-577, 1999. <http://cgsigma.cshl.org/jian/>