ALGORITHMS FOR CHALLENGING MOTIF PROBLEMS [*]

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

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

Pevzner and Sze [19] have introduced the Planted (*l*,*d*)-Motif Problem to find similar patterns (motifs) in sequences which represent the promoter regions of co-regulated genes, where *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 either have long running times or do not guarantee the motif can be found. In this paper, we introduce new algorithms to solve this motif problem. Our algorithms can find motifs in reasonable time for not only the challenging (9,2), (11,3), (15,5)-motif problems but for even longer motifs, say (20,7), (30,11) and (40,15), which have never been seriously attempted by other researchers because of large time and space requirements. Besides, our algorithms can be extended to find more complicated motifs structure called cis-regulatory modules (CRM).

1 Introduction

Understanding the gene regulatory network, i.e. how genes cooperate to perform functions, is an important problem in Bioinformatics. An important subproblem is to finding motifs for co-regulatory genes.

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

Many algorithms [1-3,7-12,14-15,17-20,22-23,25] have been introduced to solve this problem based on different assumptions. Pevzner and Sze [19] define a very precise version of this motif discovery problem which has also been considered in [3,17,20].

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.

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

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

Buhler and Tompa [3] had studied the limitation of this model. They found that when l, t and n are fixed, d must not be larger than some threshold. Otherwise, there are many random patterns (sequences) P such that each input sequence contains a variant of P. No algorithm can distinguish the motif M from these random patterns P and therefore it is very unlikely that the motif can be found. They determined the threshold by considering the expected number E(l,d) of random patterns P with at least one variant in each sequence using the following equation.

$$E(l,d) = 4^l[1-(1-p(l,d))^{n-l+1}]^t$$

where $p(l,d) = \sum_{i=0}^{d} C_i^l (3/4)^i (1/4)^{l-i}$ is the probability that the Hamming distance between a length-l pattern P and a randomly generated length-l string is less than or equal to d. Theoretically, when E(l,d) is large, no algorithm can find the motif M. They further introduced some "challenging" problems, e.g. (9,2), (11,3), (15,5)-motif problems, for t = 20 and n = 600, in which E(l,d) are slightly larger than 1. Many algorithms cannot solve these challenging problems.

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

Brute-force algorithms [2,9,18,22-23,25] 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, they are not suitable for finding long motifs.

Algorithms using clique search approach [17,19] 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 at most $2d$. 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 when the number of edges in the graph is large, as in the case of the (9,2), (11,3), (15,5)-motif problems (The running time can be $O((nt)^{t+2.376})$ in these cases).

Algorithms based on heuristic search [1,3,7-8,10-12] first find out a set of length-l sequences with high probability of being the motif, then refine these sequences by some local searching techniques, e.g., EM-algorithm [1], Gibbs Sampling [11], 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 found even when the motif is short.

As far as we know, until now, no known software can find motifs for large l and d. Our contribution includes:

1) Waterman et al. [24] designed an algorithm for finding consensus patterns in a set of sequences by enumerating the *neighbors* (will be discussed in Section 2) of each length-l substring in the input sequences. Their algorithm takes $O(4^l nt(3l)^d)$ time when applied to solve the Planted (l,d)-Motif Problem directly. We have designed a Voting Algorithm based on similar idea as Waterman et al. with time complexity $O(nt(3l)^d)$. Since the Voting Algorithm runs faster than the brute-force algorithms, it

can handle longer motifs than brute-force algorithms, e.g., the challenging (9,2), (11,3), (15,5)-motif problems. However, when $l > 15$, e.g. (20,7), (30,11) and (40,15)-motif problem, even the Voting Algorithm will fail because of large time and space requirements.

2) We have designed a Voting Algorithm with projection. Instead of considering all positions, our improved Voting Algorithm considers only *l*' of the *l* positions of the motif. Based on the voting results on these *l*' positions, we can with high probability find the motif of length *l*. In fact, the *l*' positions can be chosen randomly and the probability of success can be increased 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.
3) We have extended the Voting algorithm for finding cis-regulatory patterns which run faster than previous algorithms.

Depending on the sizes of *l* and *d*, the appropriate algorithm should be applied to find the motif. Experiments on simulated data show that the improved Voting Algorithm with projection can find long motifs, e.g., the (40,15)-motif problem with over 95% successful rate. Note that Buhler et al [3] have shown that no algorithms can find 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), (20,8), (30,14) and (40,19)-motif problems. Thus, our algorithms can solve the Planted (*l*,*d*)-Motif Problem with almost the maximum solvable *d* especially for short motif length *l*.

This paper is organized as follows. We describe the Voting Algorithm in Section 2 and the heuristic improvements in Section 3. Experimental results on both real data and simulated data are shown in Section 4. An extension of the Voting Algorithm on cis-regulatory patterns is presented in Section 5 followed by a discussion in Section 6.

2 Voting Algorithms

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

First, we define a length-*l* sequence *s*' to be a *d-variant* (or simply variant) of another length-*l* sequence *s* if the Hamming distance between *s*' and *s* is at most *d*. Let N(*s*,*d*) be the 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*.

Algorithm 1 outlines the procedure for the Basic Voting Algorithm. The idea of the Basic Voting Algorithm is that each length-*l* substring σ in the input sequences gives one vote to all length-*l* sequences *s* in N(σ,*d*), which is recorded in hash table V in Algorithm 1. If each length-*l* sequence *s* can get at most one vote from each input sequence, the motif *M* will get exactly *t* votes because of the assumption that each input sequence has exactly one planted variant of *M*. Hash table R in Algorithm 1 is to ensure that each length-*l* sequence *s* receives at most one vote from each input sequence.

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 votes received by each length-$l$ sequence $s$. Each length-$l$ sequence,
    received $t$ votes is a candidate for motif. Hash table R ensures that each length-$l$ sequence receives at most
    one vote from each input sequence. $S_i[j]$ is the $j$-th character in the $i$-th input sequence $S_i$ and H($s$) is the
    hash value of a length-$l$ sequence $s$.}
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 $N(S_i[j \ldots j + l - 1], d)$
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 $j \leftarrow 1$ to $n - l + 1$
10:     do for each length-$l$ sequence $s$ in $N(S_i[j \ldots j + l - 1], d)$
11:            do if V[H($s$)] = $t$
12:                   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 in one of the input sequence, it will not be the motif and will not be stored in the hash tables. Therefore the storage space can be reduced from $O(nt(3l)^d)$ to $O(n(3l)^d)$. The correctness of the Basic Voting Algorithm is straightforward and thus omitted. Theorem 1 proves that the time and space complexities of the algorithm are $O(nt(3l)^d)$ and $O(n(3l)^d + nt)$ respectively. On the other hand, the brute-force algorithm takes $O(nt4^l)$ time and $O(nt)$ space. Waterman et al. [24] have designed an algorithm for finding consensus in the input sequences based on an idea similar to voting. However, since their method is not designed for solving the Planted (*l*,*d*)-Motif Problem, it takes $O(4^l)$ time to fill in an entry of the hash table and another $O(4^l)$ time to scanning the hash table for finding the motif. As a result, it has a time complexity of $O(4^l nt(3l)^d)$ instead of $O(nt(3l)^d)$.

Although the basic Voting Algorithm runs faster than the brute-force algorithm does, the required space grows exponentially with *d*. Thus, the basic Voting algorithm cannot handle long motifs with large Hamming distance *d*. A method to reduce space complexity is to divide the 4^l length-*l* sequences into groups and to process each group one by one. We group the 4^l length-*l* sequences *s* according to their suffixes of length $l - l'$. Two length-*l* sequences are in the same group if and only if their suffixes are the same. Instead of scanning the input sequences once, we scan the input sequences $4^{l-l'}$ times for the $4^{l-l'}$ groups of sequences. For each iteration, each substring σ in the input sequences will be processed and one vote will be given to its variants with a particular suffix. Theorem 2 proves that the time and space complexities of this modified algorithm are $O(nt(3l)^d +$

$nt4^{l-l'}$) and $O(n(3l')^d + nt)$ respectively. Note that when $l - l'$ is $\Omega(\log_4(3l)^d)$, $O(nt(3l)^d + nt4^{l-l'}) = O(nt(3l)^d)$.

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. Substring σ starting at position j in the i-th input sequence s_i is represented by $S_i[j \ldots j + l - 1]$ (line 5 of Algorithm 1) where $S_i[j]$ is the j-th character of S_i.

$$K(l,d) = \sum_{i=0}^{d} C_i^l 3^i = O((3l)^d)$$

where C_i^l is the number of ways of choosing i objects from l. Since we can access each entry in the hash tables V and R in constant time, the voting on all d-variants of a length-l substring σ in an input sequence (lines 5 to 8 of Algorithm 1) takes $O(K(l,d))$ time. Therefore, when all the length-l substrings of the t length-n sequences are considered, it will take $O(ntK(l,d))$ time in total (the two for-loops of i and j in lines 3 to 8). Finally we have to check $K(l,d)$ entries for each of the $n - 1 + 1$ substring s for motif candidates, i.e. those entries which have received t votes (line 9 to 12 of Algorithm 1). This checking will take $O(nK(l,d))$ time. Thus the running time of the basic Voting Algorithm is $O(ntK(l,d)) + O(nK(l,d)) = O(ntK(l,d)) = O(nt(3l)^d)$.

Each length-l substring in the first input sequence has $K(l,d)$ variants. Therefore, at most $(n - l + 1)K(l,d)$ length-l sequences will get one vote after processing the first input sequence S_1. Since only those length-l sequences that get a vote when processing S_1 can possibly be the motif, we need only to keep track of votes for these length-l sequences when processing the remaining sequences $S_2 \ldots S_t$. So, the maximum number of possible candidates for the motif, i.e. size of the two tables V and R in Algorithm 1, is at most $(n - l + 1)K(l,d)$ or $O(n(3l)^d)$. As the space needed to store the input sequences is $O(nt)$, the space complexity of the algorithm is $O(n(3l)^d + nt)$. □

Theorem 2: The time and space complexities of the modified Voting Algorithm are $O(nt(3l)^d + nt4^{l-l'})$ and $O(n(3l')^d + nt)$ respectively.
Proof: Although we have divided the length-l sequences into $4^{l-l'}$ groups, the total number of votes remains $O(ntK(l,d))$ issued by the length-l substrings. However, since we have to scan the input sequences $4^{l-l'}$ times, the modified Voting Algorithm will take $O(ntK(l,d) + nt4^{l-l'}) = O(nt(3l)^d + nt4^{l-l'})$ time.
At each iteration, we need to store the votes for a group of length-l sequences with a particular length-$(l - l')$ suffix only, the space needed (i.e. tables V and R in Algorithm 1) decreases from $O(nK(l,d))$ to $O(nK(l',d))$. Thus, the space complexity of the modified Voting Algorithm is $O(nK(l', d) + nt) = O(n(3l')^d + nt)$. □

3 Heuristic Improvements

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 motifs, we introduce two heuristic improvements for the Voting Algorithm.

3.1 Random Projection

When l is large, say $l > 15$, the time required for finding 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. A similar projection idea was used by Buhler et al [3] in which each length-l substring in the input sequences gives only one vote to the length-l' projected sequence (thus a total of $t(n - l + 1)$ votes will be issued) and those length-l' substrings with votes larger than some predefined threshold are used as seeds for finding the motif by an EM algorithm. Their algorithm is successful only when a sufficiently large number of d-variants of the motif having the same nucleotides at the l' projected positions. In our algorithm, each length-l substring in the input sequences gives votes to several length-l' substrings instead of one ($O(3l')^{dl'/l}$) time more votes than Buhler et al's algorithm). As a result, our algorithm can find the motif even when the number of d-variants of the motif having the same nucleotides at the l' projected positions is small. Besides, our algorithm can find the motif directly instead of further applying a local searching algorithm.

Denote HD(s,s') be the Hamming distance between sequences s and s'. Let B be a subset of l' positions from $\{1, \ldots, l\}$. A projection 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 vote will be given to a length-l' sequence s if $\mathrm{HD}(\mathrm{proj}(\sigma,B),s) \leq \lceil dl'/l \rceil$. In general, for a length-l variant m_i of M, i.e. HD(m_i,M) $\leq d$, it is expected that the length-l' sequence proj(m_i,B) is also a $\lceil dl'/l \rceil$-variant of proj(M,B), i.e. $\mathrm{HD}(\mathrm{proj}(m_i,B),\mathrm{proj}(M,B)) \leq \lceil dl'/l \rceil$, and proj($M$,$B$) will be voted. However, even if M has t variants $\{m_i\}$, proj(M,B) may not get exactly t votes in the following cases:

(i) proj(M,B) is not voted by some planted variant m_i because $\mathrm{HD}(\mathrm{proj}(m_i,B),\mathrm{proj}(M,B)) > \lceil dl'/l \rceil$.

(ii) proj(M,B) is voted by a substring σ even though HD(σ, M) > d because $\mathrm{HD}(\mathrm{proj}(\sigma,B),\mathrm{proj}(M,B)) \leq \lceil dl'/l \rceil$.

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

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 $P_r(l,l',d,t,t')$ that "at least *t'* out of the *t* variants give vote to proj(*M*,*B*) after performed projection according to *B*" is at least

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

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

Let $\{v_i\}$ be the set of length-*l* substrings which vote 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 be slightly less than *t*. Thus, proj(*M*,*B*) will receive high votes and will be used to identify $\{v_i\}$ in the input sequences. The last procedure is to finding the motif from this set of length-*l* substrings. We choose to find the motif using clique search method. In practice, the running time for finding the maximum clique is acceptable [4] as the size of the graph is usually very small.

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

l	*d*	*t'*	$P_r(l,l',d,t,t')$	*l*	*d*	*t'*	$P_r(l,l',d,t,t')$
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 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

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'* ≈ 2*l*/3, e.g., the probability that there are at least 14 variants of *M* in the variants set of proj(*M*,*B*) is larger than 0.4165 (when *l* = 30, *d* = 12) which is much larger than the probability for a set of randomly-generated sequences. 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 finding *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 $1-(1-0.4165)^{10}=0.9954$.

Although there may be some length-*l'* sequences getting *t'* votes (false positive), proj(*M*,*B*) should get relatively more votes than other sequences because it will received votes by some substring s even HD(s, M) > d (case ii) and also from the variants of M. Besides, even when these false positive sequences have received enough votes to be considered, their corresponding length-l sequences will eventually be determined not the motif because they do not have one variant in each input sequence. Thus these false positive sequences would not introduce extra errors in our algorithm. However, the time complexity will be increased because of these false positive sequences. As the number of false positive sequences depends very much on the value of t', when t' increases, the number of false positive sequences and the value of $P_r(l,l',d,t,t')$ will decrease. Thus, the time needed to process the false positive sequences will decrease as well as the probability of finding the motif. In practice as shown in Section 4, we can set the value of t' as described above such that the motif can be found with high probability in reasonable time.

3.2 Improved Random Projection

Although we have high probability to find 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. B^c = {1, …, l} – B. If m_i is a length-l planted variant of the motif M, then either $\mathrm{HD}(\mathrm{proj}(m_i,B),\mathrm{proj}(M,B)) \le \lfloor d/2 \rfloor$ or $\mathrm{HD}(\mathrm{proj}(m_i,B^c),\mathrm{proj}(M,B^c)) \le \lceil d/2 \rceil$. Let $\{v_i\}$ and $\{v_i^c\}$ be the set of length-l variants obtained from proj(M,B) and 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 proj(M,B) or proj(M,B^c) gets at least t' votes from the t variants when performing random projection is at least

$$\sum_{j=0}^{t}\left[\left(C^t_j\, p_e^{\,j}(1-p_e)^{t-j}\right)\bullet\left(\sum_{k=\{0,\ldots,t-t',t'-j,\ldots,t-j\}} C^{t-j}_k\left(\frac{1}{2}\right)^{t-j}\right)\right] \text{ where } p_e=\frac{C^d_{d/2}\bullet C^{(l-d)}_{(l-d)/2}}{C^l_{l/2}}$$

Proof: 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 vote to both proj(M,B) and proj(M,B^c). It would happen only when $\mathrm{HD}(\mathrm{proj}(m_i,B),\mathrm{proj}(M,B))$ = $\mathrm{HD}(\mathrm{proj}(m_i,B^c),\mathrm{proj}(M,B^c))$ = d/2. It means that set B contains exactly (l – d)/2 positions that m_i equals M. Since there are $C^d_{d/2}\bullet C^{l-d}_{(l-d)/2}$ out of $C^l_{l/2}$ possible B satisfying this requirement, $p_e = C^d_{d/2}\bullet C^{(l-d)}_{(l-d)/2}/C^l_{l/2}$.

$P_h(l,d,t,t')$
$\ge$The probability that there are t' or more variants m_i satisfying $\mathrm{HD}(\mathrm{proj}(m_i,B),\mathrm{proj}(M,B)) \le d/2$
or there are t' or more variants m_i satisfy $\mathrm{HD}(\mathrm{proj}(m_i,B^c),\mathrm{proj}(M,B^c)) \le d/2$ given tha
$\mathrm{HD}(m_i,M)=d$ for all variants m_i

$$
\begin{aligned}
&= \sum_{j=0}^{t} \mathrm{P}\Big(\exists j \text{ variants } m_i \text{ s.t. } \mathrm{HD}\big(\mathrm{proj}(m_i,B),\mathrm{proj}(M,B)\big) = \mathrm{HD}\big(\mathrm{proj}(m_i,B^c),\mathrm{proj}(M,B^c)\big) = d/2\Big) \\
&\quad \bullet \mathrm{P}\Big(\exists\, t' - j \text{ or more variants } m_i \text{ in the rest } t - j \text{ variants s.t. } \mathrm{HD}\big(\mathrm{proj}(m_i,B),\mathrm{proj}(M,B)\big) \le d/2 \\
&\qquad \text{or } \exists\, t - t' \text{ or less variants } m_i \text{ in the rest } t - j \text{ variants s.t. } \mathrm{HD}\big(\mathrm{proj}(m_i,B),\mathrm{proj}(M,B)\big) \le d/2\Big) \text{a} \\
&= \sum_{j=0}^{t}\left[\left(\mathrm{C}^{t}_{j}\, p_e^{\,j}(1-p_e)^{t-j}\right) \bullet \left(\sum_{k=\{0,\dots,t-t',t'-j,\dots,t-j\}} \mathrm{C}^{t-j}_{k}\left(\frac{1}{2}\right)^{k}\left(\frac{1}{2}\right)^{t-j-k}\right)\right] \\
&= \sum_{j=0}^{t}\left[\left(\mathrm{C}^{t}_{j}\, p_e^{\,j}(1-p_e)^{t-j}\right) \bullet \left(\sum_{k=\{0,\dots,t-t',t'-j,\dots,t-j\}} \mathrm{C}^{t-j}_{k}\left(\frac{1}{2}\right)^{t-j}\right)\right]
\end{aligned}
$$

□

The values of $\mathrm{P}_h(l,d,t,t')$ for different l, d and t' when $t = 20$ are shown in Table 2. Although the probability $\mathrm{P}_h\,(l,d,t,t')$ decreases with l, the probability "at least 14 of the 20 variants of motif M are in $\{v_i\}$ derived from proj(M,B) or in set $\{v_i^c\}$ derived from proj(M,B^c)" is larger than 0.7433 (an increase from 0.4165 of the random projection method). Note that $\mathrm{P}_h\,(l,d,t,t') = 1$ for all $t' \le t/2$, 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 in order to find 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.

4 Experimental results

In this section, we describe the test results of the Voting Algorithm for both simulated and real biological data. The experiments were taken on a 2.4GHz 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 respect to different l and d.

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 ($l' = l/2$)

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

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

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 candidates for possible solutions (the length-l sequences with at least one variant in each input sequence). The pattern with the minimum Hamming distance with the planted motif will be considered as the solution of the algorithm. For each set of parameter l and d, we ran 50 test cases and recorded the average running time of each algorithm along with the average Hamming distance between the solution and the planted motif.

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 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, $\max\{d \mid E(l,d) < \text{some threshold}\}$ gives the maximum d that the (l,d)-motif problem can be solved.

Since the brute-force algorithm takes $O(nt4^l)$ time, the running time for finding a length-11 motif is over 4.8h and it cannot handle longer motif in reasonable time. Although Voting Algorithm 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.

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 recorded 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 E(l,d) < 10** *Buhler et al* [3]	**Brute-force**		**Voting**		**Voting with Heurisitc Improvement**	
			HD	time	HD	time	HD	time
7	1	1	0	61.6 s	0	<1 s	The results are the same as the Voting algorithm as no heuristic improvement is performed when l < 15	
9	2	2	0	17.9 m	0	0.4 s		
11	3	3	0	4.8 h	0	8.6 s		
13	4	4	-	-	0	108.s		
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

4.2 Real Biological Data

SCPD [26] 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 1 or 2. We have compared the performance of the Voting algorithms with a common motif discovery algorithm called MEME [1]. Experimental results are shown in Table 5. MEME [1] uses a $4 \times l$ probability matrix to represent the motif. Although matrix representation [1,5,6,13] can model a motif better than using a string, the Voting algorithm outperformed MEME in some cases because no algorithm can guarantee finding a motif in matrix representation in reasonable time especially when the motif is long. For example, MIME cannot find the motif for the transcription factor IRE whose length is 32. The existence of such long motifs can provide some justification for their study beyond pure academic interest. On the other hand, the Voting Algorithm can find the motif quickly (the running time of the Voting Algorithm was within one second for each data set) if the motif can be represented by a string.

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	Voting	MEME
AP1	TTANTAA	TTACTAA	-
CCBF,SCB,SWI6	CNCGAAA	CACGAAA	-
CuRE,MAC1	TTTGCTC	TTTGCTC	-
GATA	CTTATC	CTTAT	CTTAT
GCFAR	CCCGGG	CCCGGG	-
GCR1	CWTCC	CTTCC	CTTCC
GCN1	TAATCTAATC	TAATCTAATC	TAATCTAATC
IRE	TTTTCGTCTTCGAGGG GAAGGATCAAAGGCGC	TTTTCGTCTTCGAGGG GAAGGATCAAAGGCGC	-
LEU	CCGNNNNCGG	CCGGAACCGG	CCGGAACCGG
NBF	ATGYGRAWW	ATGTGAAAA	-

5 Extension

Some genes cannot be regulated by a single transcription factor. A gene may express only when several types of transcription factors are bound to a set of binding sites in the promoter region of the gene. These binding sites are organized into short sequence units called *cis-regulatory modules* (CRM). A CRM cannot be represented by a single motif but by a set of motifs M_i. The Planted CRM Problem which has been considered in [17,21] is defined as follows.

Planted CRM Problem: Suppose there is a CRM represented by K motifs. The length of each motif M_i is l_i. Each input sequence contains a planted d_i-variant of each motif M_i where the d_{i+1}-variant of M_{i+1} occurs no less than min_i and no more than max_i nucleotides after the end of the d_i-variant of M_i. Given

t length-n input sequences, we want to determine the K motifs M_i without knowing the positions of the planted variants.

The Voting Algorithm can solve this problem with some modifications. We concatenate on the planted CPM problem with K motifs, represented by a length-$\sum_{i=1}^{K} l_i$ sequence. Instead of constructing the hash tables V and R for 4^l possible length-l candidate motifs, we construct the hash tables for $4^{\Sigma l_i}$ possible length-$\sum_{i=1}^{K} l_i$ sequences. For each substring σ of length in the range $[\sum_{i=1}^{K} l_i + \sum_{i=1}^{K-1} min_i, \sum_{i=1}^{K} l_i + \sum_{i=1}^{K-1} max_i]$, we give one vote to those length-$\sum_{i=1}^{K} l_i$ sequences that have variants in σ. The length-$\sum_{i=1}^{K} l_i$ sequence with exactly t marks represents the set of motifs.

From my best knowledge, RISO [21] is the fastest algorithm and it takes about 17000 seconds on a 2.4GHz CPU to solve the planted CRM problem when $l_1 = l_2 = 7$, $d_1 = 2$, $d_2 = 1$, $min_1 = max_1 = 15$ and $n = t = 2000$ (Please refer [21] for more details). We repeated the same experiment by the Voting Algorithm and found the correct motifs in only 7400 seconds. Besides, all motifs found by RISO on real data set from TRANSFAC (http://www.gene-regulation.com) can be found by the Voting Algorithm.

6 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.

Since some binding sites have slightly different length, 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. Besides substitution, edit distance allows insertion and deletion which can model binding sites with different lengths. 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. Another problem is considering a set of sequences with no binding site as an additional input [7]. With this additional set, we may find motif even when the Hamming distance d is large than the threshold, e.g. (9,3), (11,4), (15,6)-motif problem.

Acknowledgement

The authors are grateful to Dr M.Y. Chan for reading the draft of this paper and the anonymous reviewers for their constructive comment.

References

1. T. Bailey and C. Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21:51-80, 1995.
2. A. Brazma, I. Jonassen, I. Eidhammer, and D. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology* (*JCB*), 5:279-305, 1998.
3. J. Buhler and M. Tompa. Finding motifs using random projections. *Journal of Computational Biology* (*JCB*), 9(2):225-242, 2002.
4. N. Chiba and T. Nishizeki. Arboricity and Subgraph Listing Algorithm. *SIAM Journal on Computing*, 14:210-223, 1985.
5. F. Chin, H. Leung, S.M. Yiu, R. Rosenfeld and W.W. Tsang. Finding Motifs with Insufficient Number of Strong Binding Sites. *Journal of Computational Biology* (*JCB*), 2005. (will appear)
6. F. Chin, H. Leung, S.M. Yiu, T.W. Lam, R. Rosenfeld, W.W. Tsang, D. Smith and Y. Jiang. Finding Motifs for Insufficient Number of Sequences with Strong Binding to Transcription Factor. *RECOMB04*, p125-132, 2004.
7. Y. Fraenkel, Y. Mandel, D. Friedberg, and H. Margalit. Identification of common motifs in unaligned dna sequences: application to Escherichia coli Lrp regulon. *Bioinformatics*, 11:379-387, 1995.
8. M. Gelfand, E. Koonin, and A. Mironov. Prediction of transcription regulatory sites in archaea by a comparative genomic approach. *Nucl. Acids Res.*, 28:695-705, 2000.
9. J. van Helden, B. Andre, and J. C. 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.
10. 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 3rd International Conference on Bioinformatics and Genome Research*, p201-216, 1995
11. C. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald and J. Wootton. Detecting subtule sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262:208-214, 1993.
12. 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.
13. H. Leung and F. Chin. Finding Exact Optimal Motif in Matrix Representation by Partitioning. *European Conference of Computational Biology* (*ECCB*), 2005. (will appear)
14. H. Leung and F. Chin. Generalized Planted (*l*,*d*)-Motif Problem with Negative Set. *Workshop on Algorithms in Bioinformatics* (*WABI*), 2005. (will appear)
15. M. Li, B. Ma, and L. Wang. Finding similar regions in many strings. *Journal of Computer and System Sciences*, 65:73-96,2002.
16. S. Liang. cWINNOWER Algorithm for Finding Fuzzy DNA Motifs. *Computer Society Bioinformatics Conference*, p260-265, 2003.
17. L. Marsan and M.F. Sagot. Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. *Journal of Computerational Biology* (*JCB*), 7(3-4):345-362,2000.

18. G. Pesole, N. Prunella, S. Liuni, M. Attimonelli, and C. Saccone. Wordup: an efficient algorithm for discovering statistically significant patterns in dna sequences. *Nucl. Acids Res.*, 20(11):2871-2875,1992.
19. P. Pevzner and S.H. Sze. Combinatorial approaches to finding subtle signals in dna sequences. *In Proc. of the Eighth International Conference on Intelligent Systems for Molecular Biology*, p269-278, 2000.
20. M.F. 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.
21. M.F. Sagot. A Highly scalable algorithm for the extraction of cis-regulatory regions. *In Proc. of the 3-rd Asia-Pacific Bioinformatics Conference* (*APBC*), p271-282, 2005.
22. R. Staden. Methods for discovering novel motifs in nucleic acid sequences. *Computer Applications in Biosciences*, 5(4):293-298, 1989.
23. M. Tompa. An exact method for finding short motifs in sequences with application to the ribosome binding site problem. *In Proc. of the 7th International Conference on Intelligent Systems for Molecular Biology*, p262-271, 1999.
24. M.S. Waterman, R. Arratia and D.J. Galas. Pattern recognition in several sequences: consensus and alignment. Bull. Math. Biol., 46:515-527, 1984.
25. F. Wolfertsteeter, K. Frech, G. Herrmann, and T. 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.
26. J. Zhu and M. Zhang. SCPD: a promoter database of the yeast Saccha-romyces cerevisiae. *Bioinformatics* 15:563-577, 1999. http://cgsigma.cshl.org/jian/