
An efficient motif discovery algorithm with unknown motif length and number of binding sites

Henry C.M. Leung*

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

E-mail: cmleung2@cs.hku.hk

*Corresponding author

Francis Y.L. Chin

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

E-mail: chin@cs.hku.hk

Abstract: Most motif discovery algorithms from DNA sequences require the motif's length as input. Styczynski et al. introduced the Extended (l,d)-Motif Problem (EMP) where the motif's length is not an input parameter. Unfortunately, their algorithm takes an unacceptably long time to run, e.g. over 3 months to discover a length-14 motif. Since the best motif may not be the longest nor have the largest number of binding sites, in this paper we further eliminate another input parameter about the minimum number of binding sites in order to provide more realistic/robust results. We also develop an efficient algorithm to solve EMP and this redefined problem.

Keywords: motif discovering; transcription factor; binding sites; consensus pattern; DNA sequences; gene regulatory.

Reference to this paper should be made as follows: Leung, H.C.M. and Chin, F.Y.L. (xxxx) 'An efficient motif discovery algorithm with unknown motif length and number of binding sites', *Int. J. Data Mining and Bioinformatics*, Vol. x, No. x, pp.xxx-xxx.

Biographical notes: Henry C.M. Leung is a PhD student in bioinformatics at the Department of Computer Science, the University of Hong Kong. His research interest is in motif discovery in DNA sequences.

Francis Y.L. Chin is the Chair of Computer Science, Taikoo Professor of Engineering and Associate Dean of the Graduate School at The University of Hong Kong. He was the Head of the Computer Science Department for nearly 15 years until 2000. He received his BSc from the University of Toronto in 1972 and PhD from Princeton University in 1976. He became an IEEE Fellow in 1996. Before joining HKU, he taught at the University of Maryland, Baltimore County; University of California, San Diego; University of Alberta; the Chinese University of Hong Kong; and the University of Texas at Dallas.

1 Introduction

A *gene* (protein coding gene) is a segment of DNA that is the blueprint for protein. Genes seldom work alone. In most cases, genes cooperate to produce different proteins to provide particular functions. Understanding how the *gene regulatory network* works is important in molecular biology.

In order to start the decoding process (*gene expression*), a molecule called *transcription factor* will bind to a short region (*binding site*) preceding the gene. One kind of transcription factor can bind to the binding sites of several genes to cause these genes to co-express. These binding sites have similar patterns called *motifs*. Finding motifs from a set of DNA sequences is a critical step for understanding the gene regulatory network.

In order to discover motifs, we must first have a model to represent a motif. There are three common models: matrix representation, regular grammar representation and string representation.

In the matrix representation model, motifs are represented by *Position Weight Matrices* (PWMs) or *Position Specific Scoring Matrices* (PSSMs). For example, we can represent a length-9 motif of the transcription factor MATalpha2 as follows

A	0	0.8	0	0	0	0.9	0.8	0.4	0
C	0.9	0	0	0	0	0	0.1	0	0
G	0	0.2	0	1	0	0	0	0	0
T	0.1	0	1	0	1	0.1	0.1	0.6	1

Both PWMs and PSSMs incorporate probabilities and thus may represent motifs in real biological data better than the other representations. However, since the solution space for PWMs and PSSMs is infinite in size, algorithms generally either rely on local search to produce a sub-optimal motif matrix (e.g., Bailey and Elkan, 1995; Chin et al., 2004; Lawrence and Reilly, 1990; Leung et al., 2005), or take too long to run when the motif is longer than 10 bp (e.g., Leung and Chin (2005) which relies on partitioning).

Some algorithms (Ono and Ng, 2005; Rigoutsos and Floratos, 1998) use regular grammars to represent motifs. They assume that all binding sites are patterns satisfying a set of rules, which cannot be satisfied by sequences in non-binding regions. For example, the MATalpha2 motif can be represented by [CT][AC]TGT [AT][ACT][AT]T. Typically, these algorithms find the optimal grammar from a restricted class of regular grammars by exhaustion and hence the running time of these algorithms tends to be long.

Algorithms using string representation (Brazma et al., 1998; Buhler and Tompa, 2001; Chin and Leung, 2005; Lawrence et al., 1993; Li et al. 2002; Liang, 2003; Pesole et al., 1992; Pevzner and Sze, 2000; Price et al., 2003; Rajasekaran et al., 2005; Rigoutsos and Floratos, 1998; Sagot, 1998; Styczynski et al., 2004; Tompa, 1999; Wolfertsteeter, 1996) assume all binding sites are variants of the motif. For example, the MATalpha2 motif can be represented by CATGTAATT. Pevzner and Sze (2000) give a precise definition of motif discovery problem based on string representation.

Planted (l, d) -Motif Problem (PMP): Suppose there is a fixed but unknown nucleotide sequence M (the motif) of length l . Given t length- n sequences, each of which contains exactly one planted variant (binding site) of M , we want to determine M without knowing the positions of the planted variants. A variant is a length- l string derivable from M with exactly d point substitutions. \square

Many algorithms have been developed to solve PMP. Some (Buhler and Tompa, 2001; Chin and Leung, 2005) work efficiently when l is small (≤ 20). However, PMP is an inadequate model for reality. There are three main weaknesses in PMP. First, biologists seldom get a set of sequences where each contains exactly one planted variant. Due to experimental noise and error, they usually get a set of sequences, some of which contain no variants, some exactly one and some more than one. Second, biologists usually do not know the motif's exact length l ; at best they only know the range for the length. Third, the Hamming distance between each variant and the motif may not be exactly d .

Consequently, researchers (Chin and Leung, 2005; Rajasekaran et al., 2005) have modified the PMP to better model reality. Modifications include allowing each sequence to contain any number of variants and allowing the Hamming distance between a variant and the motif M to be at most d (instead of exactly d).

Based on the assumption that it is easier to estimate the ratio d/l than l , Styczynski et al. (2004) defined the EMP as follows.

Extended (l, d) -Motif Problem (EMP): Suppose there is a fixed but unknown nucleotide sequence M of length L . Given t length- n input sequences, containing a total of at least k planted (l, d) -variants of M where $l \leq L$, we want to determine M without knowing the positions of the planted (l, d) -variants in the input sequences and length L . A (l, d) -variant is a length- L string derivable from M with at most d point substitutions over any window of l nucleotides, where $l \leq L$. \square

In practice, many motifs are similar in that the positions of the (l, d) -variants for these motifs are very close. Therefore, Styczynski et al. (2004) used a 'maximal motif' concept to represent a set of similar motifs. They also developed an algorithm to find all maximal motifs given the input parameters l , d and k .

Definition of maximal motif: A sequence M is a maximal motif if it satisfies the following properties:

- 1 The length of M is at least l .
- 2 M has at least k (l, d) -variants in the input sequences.
- 3 The length of M cannot be increased without producing a motif with fewer (l, d) -variants in the input sequences.
- 4 The positions of all (l, d) -variants of M cannot start earlier without producing a motif with fewer (l, d) -variants in the input sequences.

Example of maximal motif: We are given the following sequences:

0 1 2 3 4 5 6 7 8 9
 S_1 : TACAGTCGGTGC...
 S_2 : GCCAGTCGGCTG...
 S_3 : CGGAGTCGCGAC...

Suppose we want to solve EMP with $l = 7$, $d = 1$ and $k = 2$. We may discover four similar motifs M_1 , M_2 , M_3 and M_4 :

M_1 : GAGTCGG
 M_2 : GAGTCGGG
 M_3 : GCAGTCGC
 M_4 : GCAGTCGCC

M_1 has 3 (7, 1)-variants at positions $S_i[2\dots 8]$, M_2 at $S_i[2\dots 9]$ and M_3 at $S_i[1\dots 8]$, for $i = 1, 2, 3$. M_4 has 2 (7, 1)-variants at positions $S_j[1\dots 9]$ ($j = 2, 3$). However, only M_3 and M_4 are maximal motifs.

M_1 is not a maximal motif because its length can be increased to form motif M_2 with the same number of (7, 1)-variants (Property 3 is not satisfied). M_2 is also not a maximal motif because the positions of its (7, 1)-variants $S_i[2\dots 9]$ can start earlier at $S_i[1\dots 8]$ to form motif M_3 having the same number of (7, 1)-variants (Property 4 is not satisfied). Although M_3 can be extended to form motif M_4 , M_3 has 3 (7, 1)-variants while M_4 has only 2. Thus, both M_3 and M_4 are maximal motifs. \square

EMP is a better model than PMP. However, it is difficult to guess the minimum number of binding sites k in the data set. If the chosen k is too large, we may miss the planted motif. If k is too small, there will be a huge number of outputs (random noise) and we have no idea which one is the planted motif.

Moreover, we have no idea how to compare two maximal motifs: given a short motif with many (l, d)-variants and a longer motif with fewer (l, d)-variants, we cannot determine which one is more likely to be the planted motif.

Another problem is that although Styczynski et al.'s algorithm for EMP does not miss any maximal motifs, the running time of the algorithm is far too long to be useful in practice. For example, when $t = 20$, $n = 600$, $l = 14$, $d = 4$, the running time of their algorithm takes more than 3 months.

In this paper, we make two main contributions. First, we propose a measurement for comparing motifs of different lengths and numbers of (l, d)-variants in the input sequences by calculating the expected number of random strings having similar properties. Based on this measurement, we modify EMP by eliminating k , the minimum number of (l, d)-variants, as input thus introducing the Further Extended (l, d)-Motif Problem (FEMP). Secondly, we introduce **Algorithm exVote**, which runs faster than the algorithm proposed by Styczynski et al., to solve EMP as well as FEMP.

This paper is organised as follows. Section 2 describes the measurement for comparing motifs and how to determine the optimal k automatically. **Algorithm exVote** is described in Section 3. Experimental results on both simulated data and real data are discussed in Section 4, followed by a conclusion in Section 5.

2 comparing different motifs

Given two motifs with different lengths and different numbers of (l, d) -variants, how can we determine which one is more likely to be the planted motif? In order to answer this question, we calculate the expected number of random sequences with properties similar to each of these two motifs. The motif with fewer random sequences having similar properties is more likely to be the planted motif. Similar analyses have also been used for PMP (Buhler and Tompa, 2001) and the motif problem in matrix representation (Chin et al., 2004).

2.1 Expected number of random sequences

Assume we are given t length- n random sequences generated according to a particular probability distribution of 'A', 'C', 'G' and 'T' and σ be an arbitrary length- L substring in one of the sequences. Let M be another random length- L sequence with equal occurrence probabilities of 'A', 'C', 'G' and 'T'. The probability that σ is a (l, d) -variant of M is

$$p(L, l, d) = \frac{N(L, l, d)}{4^L}$$

where $N(L, l, d)$ is the number of (l, d) -variants of a length- L sequence. The method for calculation of $N(L, l, d)$ is given in the Appendix. Note that $N(L, l, d)$ can be computed once and stored in a table for future use by the algorithm.

Given t length- n input sequences, the probability that a length- L sequence M has exactly i (l, d) -variants in the input sequences is

$$\binom{t(n-L+1)}{i} p(L, l, d)^i (1-p(L, l, d))^{t(n-L+1)-i}.$$

By summing up the probability that M has exactly i (l, d) -variants for all $i \geq k$, we can get the probability that M has at least k (l, d) -variants in the t length- n input sequences.

$$P(L, k, l, d) = \sum_{i=k}^{t(n-L+1)} \binom{t(n-L+1)}{i} p(L, l, d)^i (1-p(L, l, d))^{t(n-L+1)-i}.$$

Since there are 4^L possible length- L sequences, the expected number of length- L sequences having at least k (l, d) -variants in t length- n input sequences is approximately

$$E(L, k, l, d) = 4^L P(L, k, l, d)$$

This formula is only an approximation because the probability of a length- L sequence having at least k (l, d) -variants is not mutually independent of the probability of another length- L sequence having similar properties. However, experiments in Section 4 show that the formula is a good approximation.

If $E(L, k, l, d) \gg 1$, there are many random length- L sequences having similar properties as the planted motif and it would be difficult to identify this planted motif from this set of random sequences. If $E(L, k, l, d) \ll 1$, any length- L sequence having at least k (l, d) -variants in the t length- n sequences is unlikely to be random noise and is probably the planted motif.

2.2 Further Extended (l, d) -Motif Problem (FEMP)

When we are given two motifs M_1 and M_2 of length L_1 and L_2 and t length- n input sequences having k_1 and k_2 (l, d) -variants of M_1 and M_2 respectively, M_1 is more likely to be the planted motif if $E(L_1, k_1, l, d) < E(L_2, k_2, l, d)$. However, when both $E(L_1, k_1, l, d)$ and $E(L_2, k_2, l, d)$ are sufficiently small, say 10^{-5} , even when one is larger than the other, we can say both M_1 and M_2 are planted motifs with confidence.

Instead of discovering the longest motif or the motif with the largest number of (l, d) -variants, we should find those motifs with small expected numbers of random sequences with similar properties. Based on this idea, we modify EMP as follows:

Further Extended (l, d) -Motif Problem (FEMP): Suppose there is a fixed but unknown nucleotide sequence M of length L . Given t length- n sequences containing an unknown but much more than expected number of planted (l, d) -variants of M , we want to determine M and the positions of the planted (l, d) -variants with knowledge of l and d only. \square

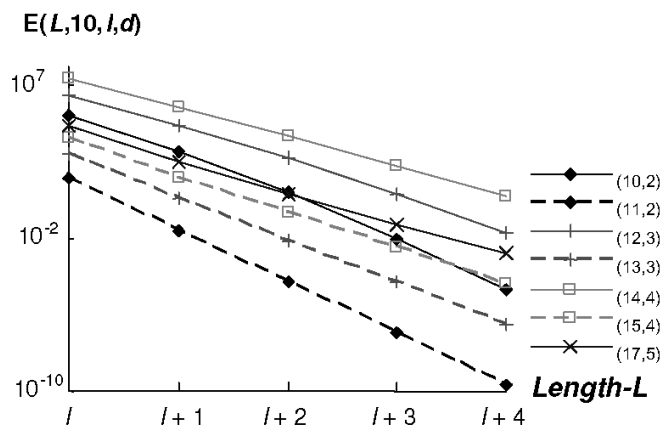
3 Algorithm

Styczynski et al. (2004) developed an algorithm to discover all maximal motifs for EMP. The main idea of their algorithm is based on an observation that if a length- L sequence M has at least k (l, d) -variants in the input sequences, each length- l' substring of M with $l \leq l' < L$ should have at least k (l, d) -variants in the input sequences too. Their algorithm first discovers all length- l motifs with at least k (l, d) -variants in the input sequences (*initial step*) and then merges the length- l (l, d) -variants of these motifs to form the length- $(l + 1)$ (l, d) -variants if any two length- l (l, d) -variants overlap at $l - 1$ consecutive positions. A length- $(l + 1)$ motif will be formed if it has at least k length- $(l + 1)$ (l, d) -variants. This merging step will be repeated again and again to form longer (l, d) -variants and longer motifs.

Although this algorithm guarantees that no maximal motif is missing, the running time is very long. This is especially so when the Hamming distance d is large. Note that even when $E(L, k, l, d)$ is very small, $E(l', k, l, d)$ can be a very large number when l' is slightly larger than l (Figures 1 and 3). As $E(l', k, l, d)^2$ pairs of length- l' motifs have to be studied to form length- $(l' + 1)$ motifs, the merging step, which needs to consider

$$\sum_{l'=l}^{L-1} E(l', k, l, d)^2$$

pairs of length- l' motifs in total, might take a very long time.

Figure 1 Expected number of length- L motifs with at least 10 (l, d) -variants

The values of t and n are 20 and 600 respectively.

As for the initial step, similar to Pevzner and Sze (2000) and Styczynski et al. (2004) reduced the problem of finding all length- l motifs to the clique searching problem. Since the Hamming distance between the motif and its (l, d) -variant is at most d , the Hamming distance between two (l, d) -variants of a particular length- l motif is at most $2d$. Consider a graph G where each length- l substring in the input sequences is represented by a vertex and there is an edge joining two vertices if and only if the Hamming distance between the two corresponding length- l substrings is at most $2d$. Since a length- l maximal motif (a length- l substring of M) has at least k (l, d) -variants in the input sequences, these (l, d) -variants form a maximal clique of size k or more in the graph. By finding all maximal cliques of size at least k , they can discover all length- l motifs, and their corresponding (l, d) -variants (binding sites) in the input sequences.

Graph G has $O(nt)$ vertices and the expected degree of each vertex is $O(ntp)$ where p is the probability that the Hamming distance between the two arbitrary length- l sequences is at most $2d$. Assuming each nucleotide is generated independently according to some background probability distribution, p has minimum value when all nucleotides are equally probable in the sequences. The minimum value of p can be calculated as follows:

$$p = \sum_{i=0}^{2d} \binom{l}{i} \left(\frac{3}{4}\right)^i \left(\frac{1}{4}\right)^{l-i}.$$

Although clique searching is NP-hard, when d is small, p tends to $1/4^l$ and the graph is sparse. Finding all maximal cliques of size at least k for a sparse graph is still tractable. However, when d increases, the number of edges increases and the graph G becomes dense. For example, when $l = 14$ and $d = 4$, the value of p is 0.11 which means that the expected degree of each vertex is $0.11(nt)$ and the number of edges in G is $O((nt)^2)$. To our best knowledge, the fastest algorithm (Makino and Uno, 2004) for finding all maximal cliques in a dense graph is $O((nt)^{2.376})$ per output. Since there are $O((nt)^k)$ maximal cliques in graph G in the worst case, the running time of Styczynski et al.'s algorithm can be $O((nt)^{k+2.376})$, which may be prohibitively long.

3.1 Algorithm exVote

In order to solve EMP (and FEMP), we propose Algorithm exVote, which uses a Voting approach to find all length- l or longer motifs directly. The Voting approach was first introduced by Chin and Leung (2005) for solving PMP. To our best knowledge, it is the fastest algorithm for solving PMP. Here we modify the Voting algorithm to solve EMP.

The Voting algorithm (Chin and Leung, 2005) is based on a simple idea that if a substring σ is a variant of a motif M , M is also a variant of σ . In order to solve PMP, each length- l substring in the input sequences gives one vote to its variants (length- l sequences). For example, when solving the planted (6, 1)-motif problem, a length-6 substring AAAAAA in the input sequences will give one vote to each of the following length-6 string:

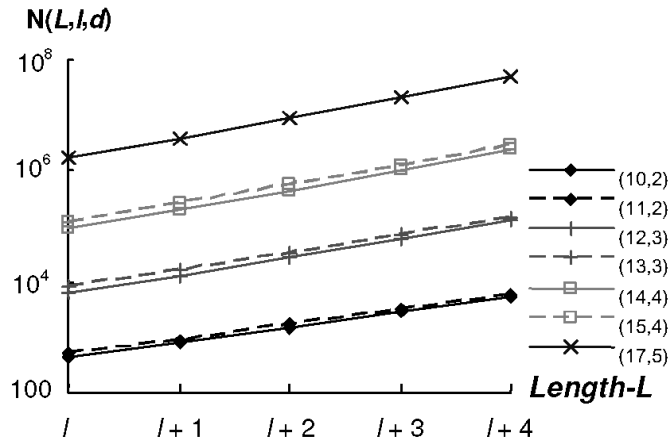
AAAAAA, CAAAAA, GAAAAA, TAAAAA, AAAAAA, ..., AAAAAT.

Under the restriction that each length- l sequence gets at most one vote from each sequence, the motif should get exactly t votes.

Since EMP allows any number of (l, d) -variants, Algorithm exVote gives one vote to the (l, d) -variants of each length- L substring in the input sequences. Those length- L sequences receiving at least k votes are motif candidates. The time complexity of Algorithm exVote is $O(tnN(L, l, d))$ which is no more than $O(tn4^L)$, since $N(L, l, d)$ is the number of (l, d) -variants of a length- L sequence and is always less than 4^L . As $k \approx t$ which is normally much larger than L , the time complexity $O(tn4^L)$ is already much smaller than $O((nt)^{k+2.376})$, the time complexity of the initial step of Styczynski et al.'s algorithm. The improved time complexity rendered by Algorithm exVote is demonstrated in Section 4.

The space complexity for Algorithm exVote is $O(4^L + tn)$ and would not create much problem when L is small. As shown in Figure 2, the value of $N(L, l, d)$ increases exponentially with the motif length L as do the time and space complexities too. In order to handle large L , Section 3.2 describes some simple techniques to reduce the space complexity without increasing the time complexity too much.

Figure 2 Values of $N(L, l, d)$ for different parameters L, l and d



The values of t and n are 20 and 600 respectively.

3.2 Reducing the space required

A straight-forward implementation of Algorithm `exVote` requires $O(4^L)$ space to store the number of votes received by the 4^L length- L sequences. Under normal circumstances where motifs have length $L \leq 18$, $4^L \leq 4^{18} = 64$ GB memory is still feasible. However, $O(4^L)$ space may be too large for some applications. However, the amount of required space can be reduced by partitioning the 4^L length- L sequences into groups and processing each group one after another. For example, we may partition the sequences into 4^m groups according to their length- m prefix. Two sequences s_1 and s_2 are in the same group if and only if their first m nucleotides are the same.

Instead of scanning the input sequences once, we scan the input sequences 4^m times. For each scan, one length- m prefix P is chosen and all length- L substrings give one vote to their (l, d) -variants having prefix P . The space needed to store the votes received by all length- L sequences with prefix P is $O(4^{L-m})$. At the end of each scan, we will find all length- L motifs with prefix P and the space can be reused again for other scans. The time and space complexities of this modified Algorithm `exVote` will then be $O(tnN(L, l, d) + tn4^m)$ and $O(4^{L-m} + tn)$ respectively. Note that when $m \leq \log_4 N(L, l, d)$, the time complexity remains $O(tnN(L, l, d))$.

3.3 Solving FEMP

In order to solve FEMP, we first find all motifs of different lengths L , each with k (l, d) -variants in the input sequences. We then output only those motifs with $E(L, k, l, d)$ values less than some predefined threshold α . We can control the number of output by controlling the value of threshold α . The probability of finding the real motif increases with a lower value of α and with an increasing of the running time and number of outputs (false positive).

In practice, we discover motifs starting from length l to n . Intuitively, when n, t, l, d , and k are fixed, the expected number of length- L motif candidates decreases with L . (This will be shown in Section 4.) When L is large, instead of discovering all length- L motifs directly by voting as described in Section 3.1. We may discover all length- L motifs after knowing the length- $(L-1)$ motifs and checking whether these length- $(L-1)$ motifs can be extended to length- L motifs. This extension step can be done by matching the length- $(L-2)$ prefix of a length- $(L-1)$ motif with the length- $(L-2)$ suffix of another and checking whether their corresponding sets of (l, d) -variants have similar property. Since there are four alphabets in DNA sequences, the expected number of length- L motifs we have to check is $4 \cdot E(L-1, k, l, d)$, which is much less than $(E(L-1, k, l, d))^2$ in Styczynski et al.'s algorithm.

This extension step will be repeated until we cannot find any motifs for a particular length. In the worst case, we have to discover all motifs of length l to n . However, since $E(L, k, l, d)$ decreases with L exponentially, we rarely need to go much beyond the length of the planted motif in practice. Therefore, we can discover the planted motif, whose $E(L, k, l, d)$ value is less than the threshold α , in reasonable time.

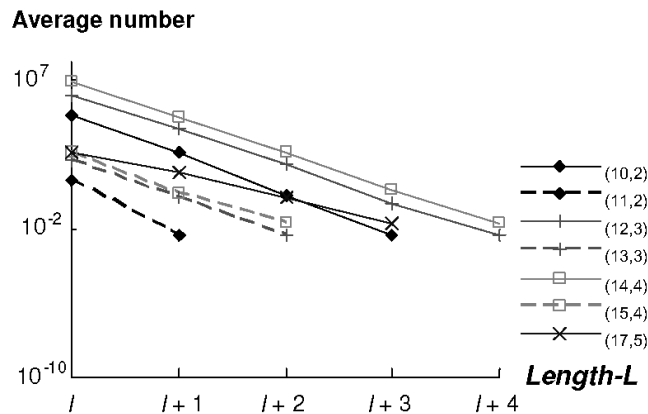
4 Experimental results

We have implemented Algorithm `exVote` in C++. Experimental results of this algorithm on both simulated data and real biological data will be shown below. Even though our experiments were performed on a P3 700 CPU with 4GB memory, our programs used less than 70 MB memory in all experiments.

4.1 Evaluating the formula for the expected number

In order to check whether the estimation of $E(L, k, l, d)$ in Section 2.1 is correct, we generated 20 length-600 sequences with equal occurrence probability for each nucleotide and counted the number of length- L sequences having at least k (l, d) -variants. For each set of parameters (l, d) , we repeated each experiment 100 times. Figures 1 and 3 show the values of $E(L, 10, l, d)$ from the formula given in Section 2.1 and from experiments.

Figure 3 Average number of length- L motifs with at least 10 (l, d) -variants (an experimental verification of $E(L, 10, l, d)$)



The values of t and n are 20 and 600 respectively.

The resemblance of these two figures means that our calculated $E(L, 10, l, d)$ is almost the same as the average number of length- L sequences having at least 10 (l, d) -variants. However, for those situations with $E(L, 10, l, d)$ less than $1/100$ (e.g., $E(13, 10, 11, 2)$ when $(l, d) = (11, 2)$ and $L = l + 2 = 13$), we could not discover any length- L sequences having at least 10 (l, d) -variants in the 100 experiments. Therefore we could not plot the values of $E(L, 10, l, d)$ in these situations in Figure 3.

We also performed the same experiments for different values of k . The average number of length- L sequences having at least k (l, d) -variants were almost the same as those values of $E(L, k, l, d)$ expressed in our formula in Section 2.1.

4.2 Results on simulated data

We are interested in comparing the performance of Algorithm `exVote` against the best performing algorithm up to now. Styczynski et al. (2004) compared the performances of their algorithm with different motif discovering programs like GibbsDNA

(Lawrence et al., 1993), WINNOWER (Pevzner and Sze, 2000), SP-STAR (Pevzner and Sze, 2000) and PROJECTION (Buhler and Tompa, 2001). Styczynski et al.'s algorithm has the best performance as it can discover the planted motifs in all cases and has the best accuracy.

Thus, we compared Algorithm *exVote* with Styczynski et al.'s algorithm. We also compared against PROJECTION, which discovers motifs by heuristic search. The simulated data were generated in the same way as Buhler and Tompa (2001), i.e., a total of 20 length-600 sequences (as in Section 4.1) each with a planted variant of a randomly generated M . We repeated the experiments 20 times for each set of parameters (l, d) on Styczynski et al.'s algorithm, PROJECTION and Algorithm *exVote*. For each data set, each algorithm made 30 predictions of the hidden motif and an algorithm is said to have predicted the hidden motif correctly if it is in one of the 30 predictions. The accuracy of the algorithms were measured by the average number of times the algorithms predicted the hidden motif correctly.

In all cases, both Styczynski et al.'s algorithm and Algorithm *exVote* could discover the planted motif M , i.e., accuracy = 1. However, Algorithm *exVote* was faster than Styczynski et al.'s algorithm. Although PROJECTION (Buhler and Tompa, 2001) had the shortest running time for some input parameters, it could not discover the planted motifs in some cases. Table 1 shows the running time of these three algorithms, in particular, when comparing the running times of Styczynski et al.'s algorithm and Algorithm *exVote*, the time needed for solving the (14, 4)-problem was reduced from 3 months to 197.5 seconds and from 3 weeks to 27 minutes on the (17, 5)-problem.

Table 1 Running times of three algorithms

l	d	<i>PROJECTION</i>		<i>Styczynski et al.'s algorithm</i>		<i>exVote</i>	
		Accuracy	(time)	Accuracy	(time)	Accuracy	(time)
10	2	0.82	(161.1 s)	1.00	(8 minutes)	1.00	(0.1 s)
11	2	0.95	(12.5 s)	1.00	(<1 minutes)	1.00	(0.7 s)
12	3	0.71	(8.7 minutes)	1.00	(10.5 hours)	1.00	(9.8 s)
13	3	0.94	(46.0 s)	1.00	(10 minutes)	1.00	(17.4 s)
14	4	0.65	(15.4 minutes)	1.00	(>3 months)	1.00	(197.5 s)
15	4	0.90	(129.0 s)	1.00	(6 hours)	1.00	(206.1 s)
17	5	0.86	(273.2 s)	1.00	(3 weeks)	1.00	(27 minutes)

The values of t and n are 20 and 600 respectively. The accuracy is the average value for 20 experiments.

4.3 Results on real biological data

We also tested the performance of Algorithm *exVote* on real biological data. SCPD (Zhu and Zhang, 1999) is a database of different transcription factors for yeast. For each transcription factor, the published motif pattern, the positions of the binding sites and the set of sequences containing these binding sites are kept. We used MEME (Bailey and Elkan, 1995), PROJECTION (Buhler and Tompa, 2001), Voting algorithm (Chin and Leung, 2005) (developed for solving the PMP) and Algorithm *exVote* to discover the motif for each data set. Since both PROJECTION and Voting algorithm

required the motif length l and Hamming distance d as input parameters, we set l to be the length of the published motif and tested all values of d from 0 to l . These two algorithms are said to discover the motif if they output the published motif for some d . MEME is a popular program for discovering motif in matrix representation. For each data set, we inputted the length of the published motif as input of MEME. For Algorithm exVote, we let $l = 6$ and $d = 1$ for all data set.

Table 2 shows the results of these algorithms. Even though we did not input the lengths of published motifs, Algorithm exVote could discover all the published motifs. Besides, since the FEMP is more flexible than the PMP, in some cases (ROX1 and UASGABA), Algorithm exVote could discover the correct motifs while PROJECTION and Voting algorithm failed. Although MEME is a powerful program for discovering motif, as some real motifs cannot be modelled by a matrix well, MEME could not discover some motifs.

Table 2 Experimental results on real biological data

<i>Name</i>	<i>Motif pattern</i>	<i>MEME</i>	<i>PROJECTION</i>	<i>Voting</i>	<i>Algorithm exVote</i>	<i>(E(L,k,l,d))</i>
API	TTANTAA	–	–	TTACTAA	TTAATAA	(2.6)
BAS1	TGACTC	–	TGACTC	TGACTC	TGACTC	(0.0577)
GATA	CTTATC	CTTATC	CTTATC	CTTATC	CTTATC	(6.75×10^{-9})
GCR1	C[AT]TCC	CTTCC	CTTCC	CTTCC	CTTCC	(8.98×10^{-4})
HSE,HTSF	TTCTAGAA	TTCTAGAAG	TTCTAGAAG	TTTCTAGAA	TTTCTAGAA	(8.35×10^{-5})
NBF	ATG[CT]G[AG]A [AT][AT]	–	TGTGAAAAG	ATGTGAAAA	ATGTGAAAT	(88.1)
ROX1	[CT][CT]NATTGTT[CT]	GCCTATTGTT	–	–	ATTGTTC	(0.0423)
UASGABA	AAAAACCGCC GGCGGCAAT	–	–	–	AAAAGCCGCC GGCGGCAAT	(0.748)

5 Discussion

In this paper, we have introduced the expected number of sequences with similar properties in a set of random sequences (i.e., with the same number and length) as a measurement for comparing different motifs. Based on this measurement, we define the FEMP which does not need to take the motif's length and the number of planted variants as input parameters.

We have also developed Algorithm exVote to solve the EMP and FEMP. Experiments on simulated data and real biological data show that Algorithm exVote performs better than the popular motif discovering algorithms in terms of flexibility and running time.

The main purpose of the maximal motif is to reduce the number of motifs to be considered by eliminating similar or inferior ones. However, discovering maximal motifs may not be the best way for solving motif problems. For example, given the following sequences

0 1 2 3 4 5 6 7 8 9
 S_1 : TATTCACTGC...
 S_2 : GCCTCACCTG...
 S_3 : CGGACATGAC...

We want to solve the Extended (4, 1)-Motif Problem with $k=3$. ‘GTCAT’ has 3 (4, 1)-variants at $S_i[2\dots 6]$ and ‘ACACG’ at $S_i[3\dots 7]$ for $i = 1, 2, 3$. ‘ACACG’ is not a maximal motif because it is of the same length and has the same number of variants as ‘GTCAT’ and the positions of its variants are one base pair behind ‘GTCAT’ (property 4). However, they are treated as the same even though ‘GTCAT’ and ‘ACACG’ are two motifs with very different patterns. Thus, it is doubtful whether property 4 should be included in the definition of maximal motif. Note that **Algorithm exVote** can discover all maximal motifs with the same time and space complexities after removing property 4.

Acknowledgement

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

References

- Bailey, T. and Elkan, C. (1995) ‘Unsupervised learning of multiple motifs in biopolymers using expectation maximization’, *Machine Learning*, Vol. 21, pp.51–80.
- Brazma, A., Jonassen, I., Eidhammer I. and Gilbert, D. (1998) ‘Approaches to the automatic discovery of patterns in biosequences’, *Journal of Computational Biology*, Vol. 5, pp.279–305.
- Buhler, J. and Tompa, M. (2001) ‘Finding motifs using random projections’, *Proceedings of RECOMB01*, pp.69–76.
- Chin, F. and Leung, H. (2005) ‘Voting algorithm for discovering long motifs’, *Proceedings of Asia-Pacific Bioinformatics Conference*, pp.261–272.
- Chin, F., Leung, H., Yiu, S.M., Lam, T.W., Rosenfeld, R., Tsang, W.W., Smith, D. and Jiang, T. (2004) ‘Finding motifs for insufficient number of sequences with strong binding to transcription factor’, *Proceedings of RECOMB04*, pp.125–132.
- Lawrence, C. and Reilly, A. (1990) ‘An expectation maximization (em) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences’, *Proteins: Structure, Function and Genetics*, Vol. 7, pp.41–51.
- Lawrence, C., Altschul, S., Boguski, M., Liu, J., Neuwald, A. and Wootton, J. (1993) ‘Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment’, *Science*, Vol. 262, pp.208–214.
- Leung, H. and Chin, F. (2005) ‘Finding exact optimal motif in matrix representation by partitioning’, *Bioinformatics*, Vol. 22, Supp. 2, pp.ii86–ii92.
- Leung, H., Chin, F., Yiu, S.M., Rosenfeld, R. and Tsang, W.W. (2005) ‘Finding motifs with insufficient number of strong binding sites’, *Journal of Computational Biology*, Vol. 12, No. 6, pp.686–701.
- Li, M., Ma, B. and Wang, L. (2002) ‘Finding similar regions in many strings’, *Journal of Computer and System Sciences*, Vol. 65, pp.73–96.

- Liang, S. (2003) ‘cWINNOWER algorithm for finding fuzzy DNA motifs’, *Proc. of Computer Society Bioinformatics Conference 2003*, pp.260–265.
- Makino, K. and Uno, T. (2004) ‘New algorithms for enumerating all maximal cliques’, *Proceedings of Scandinavian Workshop on Algorithm Theory*, pp.260–272.
- Ono, H. and Ng, Y.K. (2005) ‘Best fitting-length substring patterns for a set of strings’, *Proceedings of Computing and Combinatorics, 11th Annual International Conference (COCOON)*, pp.240–250.
- Pesole, G., Prunella, N., Liuni, S., Attimonelli, M. and Saccone, C. (1992) ‘Wordup: an efficient algorithm for discovering statistically significant patterns in DNA sequences’, *Nucl. Acids Res.*, Vol. 20, No. 11, pp.2871–2875.
- Pevzner, P. and Sze, S.H. (2000) ‘Combinatorial approaches to finding subtle signals in DNA sequences’, *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pp.269–278.
- Price, A., Ramabhadran, S. and Pevzner, P. (2003) ‘Finding subtle motifs by branching from sample strings’, *Bioinformatics*, Vol. 1, pp.1–7.
- Rajasekaran, S., Balla, S. and Huang, C.H. (2005) ‘Exact algorithm for planted motif challenge problems’, *Proceedings of Asia-Pacific Bioinformatics Conference*, pp.249–259.
- Rigoutsos, I. and Floratos, A. (1998) ‘Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm’, *Bioinformatics*, Vol. 14, pp.55–67.
- Sagot, M.F. (1998) ‘Spelling approximate repeated or common motifs using a suffix tree’, in Lucchesi, C.L. and Moura, A.V. (Eds.): *Latin’98: Theoretical informatics, volume 1380 of Lecture Notes in Computer Science*, pp.111–127.
- Styczynski, M.P., Jensen, K.L., Rigoutsos, I. and Stephanopoulos, G.N. (2004) ‘An extension and novel solution to the (l,d) -motif challenge problem’, *Genome Informatics*, Vol. 15, pp.63–71.
- Tompa, M. (1999) ‘An exact method for finding short motifs in sequences with application to the ribosome binding site problem’, *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology*, pp.262–271.
- Wolfertsteeter, F., Frech, K., Herrmann, G., and Wernet, T. (1996) ‘Identification of functional elements in unaligned nucleic acid sequences by a novel tuple search algorithm’, *Computer Applications in Bio-sciences*, Vol. 12, No. 1, pp.71–80.
- Zhu, J. and Zhang, M. (1999) ‘SCPD: a promoter database of the yeast *Saccharomyces cerevisiae*’, *Bioinformatics*, Vol. 15, pp.563–577, <http://cgsgigma.cshl.org/jian/>.

Appendix

Calculating $N(L, l, d)$

In this section, we will describe how to calculate the number $N(L, l, d)$ of (l, d) -variants of a length- L sequence M . Given two length- L sequences σ and M , $e_{\text{suf}}(\sigma, M)$ is a length- l bit string defined as follows,

$$e_{\text{suf}}(\sigma, M)[i] = \begin{cases} 1 & \sigma[L-l+i] \neq M[L-l+i] \\ 0 & \sigma[L-l+i] = M[L-l+i] \end{cases}$$

$e_{\text{suf}}(\sigma, M)$ is a binary bit string which represents the positions that the length- l suffix of σ and M are different.

Let $C[L, \beta]$ be the number of (l, d) -variants σ of a length- L sequence M satisfying $e_{\text{suf}}(\sigma, M) = \beta$ where β is a binary string of length- l . For example, when $l = 5$, given two length-8 sequences

1 2 3 4 5 6 7 8
 σ : AGCTAACG
 M : CCGTTACT

$e_{\text{suf}}(\sigma, M) = '01001'$ and $C[L, '01001'] = 3^2$ if $d \geq 2$.

By considering all possible bit strings β , we can calculate the number of (l, d) -variants of a length- L sequence

$$N(L, l, d) = \sum_{\beta} C[L, \beta] = \sum_{\beta: |\beta| \leq d} C[L, \beta]$$

where $|\beta|$ be the number of '1' in the bit string β .

We applied dynamic programming to calculate $C[L, \beta]$. When $L = l$, we have

$$C[l, \beta] = \begin{cases} 3^{|\beta|} & |\beta| \leq d \\ 0 & |\beta| > d \end{cases}$$

When $L > l$, we calculate the value of $C[L, \beta]$ by the following recurrence,

$$C[L, \beta] = \begin{cases} 3(C[L-1, 0 \bullet \beta'] + C[L-1, 1 \bullet \beta']) & \beta[l] = 1 \wedge |\beta'| < d \\ C[L-1, 0 \bullet \beta'] + C[L-1, 1 \bullet \beta'] & \beta[l] = 0 \wedge |\beta'| \leq d \\ 0 & |\beta| > d \end{cases}$$

where $\beta' = \beta[1..l-1]$ and \bullet means concatenate

Although there are 2^l possible length- l bit strings β , we only consider the values of those $C[L, \beta]$ with $|\beta| \leq d$. Therefore the total number of $C[L, \beta]$ we have to calculate is at most

$$(L-l+1) \sum_{j=0}^d \binom{l}{j}$$

Since it takes constant time to calculate each $C[L, \beta]$, the total running time for counting $N(L, l, d)$ is

$$O\left((L-l+1) \sum_{j=0}^d \binom{l}{j} + \sum_{j=0}^d \binom{l}{j}\right) = O\left((L-l) \sum_{j=0}^d \binom{l}{j}\right). \quad \square$$