

# Graphical Approach to Weak Motif Recognition

Xiao Yang

xyang@pmail.ntu.edu.sg

Jagath C. Rajapakse

asjagath@ntu.edu.sg

BioInformatics Research Center, School of Computer Engineering, Nanyang Technological University, 639798, Singapore

## Abstract

We address the weak motif recognition problem in DNA sequences, which extends the general motif recognition to more difficult cases, allowing more degenerations in motif instances. Several algorithms have earlier attempted to find weak motifs in DNA sequences but with limitations. In this paper, we propose a graph-based algorithm for weak motif detection, which uses dynamic programming approach to find cliques indicating motif instances. The experiments on synthetic datasets show that the algorithm finds weak motif instances more accurately and efficiently compared to earlier approaches. Its performances on real datasets in finding transcription factor binding sites are comparable with the existing techniques.

**Keywords:** bioinformatics, dynamic programming, genomic sequences, motifs, weak motif detection

## 1 Introduction

Over past decades, many efforts have been put together to resolve the motif recognition problem in DNA sequences. Most of the works focused on searching for the Transcription Factor Binding Sites (TFBSs) in the upstream regions of co-regulated genes [1, 4, 5, 7, 8, 9, 12] or a set of orthologous genes [2, 10]. The motif sites detected by computational means were further validated by wet lab experiments.

In genomic sequences, a *motif* is a set of short functional DNA strands that preserve a certain nucleotides composition, relating to some important biological processes, for instance, gene expression process. In practice, true motifs are present as *motif instances* due to mutations and substitutions at a few nucleotide positions. The pattern shared by a group of motif instances is referred to as the *motif consensus*. The *motif recognition* is the problem of finding motif instances in a set of unannotated sequences. Due to the cost and time involved in the experimental approaches, computational methods have recently gained increased attention in motif detection.

The approaches to detect motifs can be broadly categorized into enumeration and heuristic approaches. Oligo-Analysis [4] and Yeast Motif Detection (YMF) [12] are popular enumeration methods, which use a consensus model (CM) to represent motifs. The well-known heuristic approaches for motif recognition include CONSENSUS [5], Multiple Expected Maximization Elicitation (MEME) [1], Gibbs Sampler [7], Bioproscpector [8] and MDscan [9]. The heuristic approaches adopt probability weight matrix (PWM) model. MDscan provides a hybrid of enumeration and heuristic approaches. Although the above algorithms succeeded in detecting motifs in specific occasions, such as TFBSs detection in yeast genomes, none of them has been satisfactory when the motif instances present in a weak form or in solving the weak motif recognition problem.

The limitations of most motif recognition problems were highlighted when Pevzner and Sze defined the *challenge problem* for motif detection [11]. They showed that most existing methods capable of finding motifs of length 6 with no degeneration failed to detect motif instances of length 15 with 4 degenerate positions in a random sample containing 20 sequences of 600 nucleotides long. In order to

address the challenge problem, they introduced two algorithms, namely WINNOWER and SP-STAR [11]. Recently, other techniques have been developed for detecting weak motifs [3, 6].

This paper addresses the weak motif recognition problem and presents a novel detection algorithm based on dynamic programming for clique finding on graphs. The algorithm finds weak motifs at a global optimum when one motif per sequence in the datasets are present. The performance and the running time is better compared to previous methods in weak motif detection.

## 2 Motif Recognition

In this section, we present the motif recognition problem and the drawbacks in existing approaches to weak motif recognition.

Let us denote a motif instance as  $(l, g, d)$ , where  $l$  is the motif length,  $g$  is the number of gaps within the motif, and  $d$  is the number of degenerate positions. The number of degenerate positions represents the number of mutations present in the motif instance. When there are no gaps, a motif is denoted by  $(l, d)$ ,

**Definition 1:** the *motif recognition problem* is that of finding motif instances without the knowledge of the position and their consensus form, given a dataset that contains  $m$  number of sequences with length  $n > m$  in average, each containing  $(l, g, d)$  motif instances.

### 2.1 Enumeration Methods

The obvious way for detecting motifs of length  $l$  is to enumerate all  $4^l$  possible seeds with length  $l$ , count the occurrence and calculate a significance value for each seed. The seeds with large significant scores are taken as potential motifs. However, it becomes intractable when  $l$  is large, for instance  $l = 15$ . Therefore, sample driven approaches considering only oligomers present in the datasets have been introduced.

Oligo-analysis calculates the significance value for each oligomer, for instance  $l=6$ , present in the datasets [4]. The YMF approach has successfully detected motifs in the yeast genomes by calculating a  $z$ -score for each oligomer (with length  $l$ ,  $g=0-11$  and  $d \leq 2$ ). The success of these algorithms in finding the TFBSs in yeast genome is due to the regular appearances of motifs, the short length and small number of degenerate positions. However, they are not suitable to find weak motifs such as  $(15, 4)$ .

Although MDscan [9] has detected longer motifs, such as  $l \in [5, 15]$ , and more degenerate positions, weak motifs are still beyond its capabilities. For example, to detect  $(15, 4)$  motif, when a certain oligomer is taken as a seed, a randomly generated  $l$ -mers with the probability  $\varepsilon$  for having at least  $\gamma$  matches with the seed oligomer should satisfy  $\varepsilon < 0.15\%$  [9]. When  $\gamma = 7$  for a  $(15, 4)$  motif,  $\varepsilon = 0.0566 \gg 0.15\%$  and only when  $\gamma = 9$  ( $\varepsilon = 0.0042$ ) or  $10$  ( $\varepsilon = 0.0008$ ), the requirement  $\varepsilon < 0.15\%$  could be satisfied, meaning  $d \leq 3$ .

### 2.2 Heuristic Methods

The heuristic methods adopt PWM models and local refinement of scoring function for motif detection. CONSENSUS is a greedy algorithm that saves the instances with the best information content score in each step [5]. However, for a random seed with length  $l=15$ , This approach can find more than 2 random motif instances that differ from the seed within 6 positions in a sequence of length  $n=600$ . Therefore, the real instances of the  $(15, 4)$  motif are almost undetectable with CONSENSUS.

MEME uses Expectation Maximization (EM) algorithm [1] to find the maximum likelihood estimation of the motif instance while Gibbs Sampler [7] adopts predictive update step to estimate the most possible instance present in a certain sequence. When the sequence becomes longer, both ap-

proaches have the difficulty in escaping from a large number of local minimum and often trapped in finding similar motif instances or none at all.

### 2.3 Weak Motif Recognition

Both heuristic and enumeration methods have shown acceptable performances in many occasions, such as TFBSs detection in yeast genome, where the motifs are usually short and have good preservation [13] and, especially, when the motif consensus is usually present in the sequence.

**Definition 2:** the *weak motif recognition* represents the motif recognition problem when the number of degenerate positions  $d$  is large relative to the motif length  $l$ .

Enumeration methods are not applicable when  $l$  is large; also, the heuristic methods and sample driven approaches often get trapped in local optima when the motif consensus is not present or weak in the dataset. Recently, a few algorithms have been developed specifically for weak motif detection: WINNOWER [11] represent motif instances as vertices in a graph, then try to delete spurious edges and recover the motif with the remaining vertices; SP-STAR [11] is a local sum of pairwise score improvement algorithm, which considers only the subsequences present in the dataset and iteratively update the scores of the motifs; Random projection (RP) [3] adopts an effective initialization step that clustered the possible motif instances together, derives different consensus for each cluster and uses EM for refinement.

SP-STAR found the instances of a (15, 4) motif in the datasets ( $m=20$ ) with sequence length no more than 700, but collapsed down in longer sequences [11]. WINNOWER performed better than SP-STAR with the same datasets. Recently, LIANG et al. [6] improved the performance of WINNOWER by a stronger filtering constraint. When applied to the same problem as WINNOWER, less spurious edges left after filtering, and the RP proved to be high toleration of degenerate motifs and improved the running time and performance. However, it is still a local searching algorithm.

## 3 Method

In this section, we present our method for weak motif detection. Firstly, we represent the input dataset in a set of graphs and show, with some restrictions, that the true motif instances are given by the vertices of certain cliques of these graphs. The search for the cliques is done using dynamic programming and the motif consensus is recovered from the motif instances found.

Let the dataset of DNA sequences be denoted by  $\mathbf{x}=\{\mathbf{x}_i: i=1, \dots, m\}$ , where  $m$  is the number of sequences and each sequence  $\mathbf{x}_i = \{\sigma_{i1}, \sigma_{i2}, \dots, \sigma_{in_i}\}$ , where  $\sigma_{ij} \in \Omega_{DNA}$ ,  $n_i$  is the length of the sequence  $\mathbf{x}_i$  and  $\Omega_{DNA}$  is the alphabet containing four nucleotides. For simplicity, in this paper, we consider only a single motif instance in a sequence and  $g = 0$ , and therefore represent the motif by  $(l, d)$ . That is, there exist  $m$  motif instances in total in the dataset. Each motif instance differs from the motif consensus, formed by the alignment of all  $m$  instances, at  $d$  positions.

In graphical representation of the dataset, each subsequence is represented by a vertex. Let vertex  $v_{ij}$  represent the subsequence of length  $l$  starting at position  $j$  of the  $i$ th sequence:  $\{\sigma_{ij}, \sigma_{ij+1}, \dots, \sigma_{ij+l-1}\}$ . Therefore,  $m$  number of real motif instances in the dataset are assigned to certain vertices and are sought among total  $m(n-l+1)$  number of vertices. Let us denote the distance between two vertices  $v_{ij}$  and  $v_{i'j'}$  as  $dis(v_{ij}, v_{i'j'})$ , which equals to the Hamming Distance between two subsequences represented by the two vertices. Therefore, for a  $(l, d)$  motif  $\psi$  in the dataset, suppose two of its instances  $\psi_1$  and  $\psi_2$  are represented by vertices  $v_{ij}$  and  $v_{i'j'}$ , respectively, they should satisfy the inequality:  $dis(v_{ij}, v_{i'j'}) \leq 2d$  as any two instances of the motif  $(l, d)$  differ at most  $2d$  positions.

The construction of the graph should make sure that the motif instances represented by vertices in the graph are connected to each other and form a clique of size  $m$ . Then, the motif recognition problem is converted to finding cliques with size  $m$  in a set of graphs. Our algorithm involves three steps: graph construction, clique finding, and rescanning.

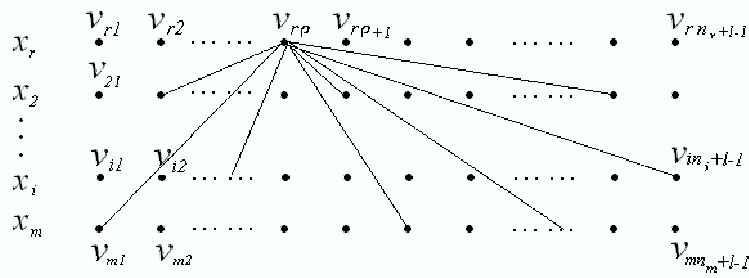


Figure 1: Illustration of graph  $G_\rho$ :  $\rho$  indicates the starting position of the subsequence  $\mathbf{s}_\rho = \{\sigma_{r\rho}, \sigma_{r\rho+1}, \dots, \sigma_{r\rho+l-1}\}$  in the reference sequence  $\mathbf{x}_r$ ; each vertex connected to  $v_{r\rho}$  represents a subsequence whose distance with  $\mathbf{s}_\rho$  is no more than  $2d$ . When  $v_{r\rho}$  represents a real motif instance in the sequence  $\mathbf{x}_r$ , all other motif instances are represented by the vertices of the graph  $G_\rho$ .

### 3.1 Graph Construction

Let us randomly select a sequence  $\mathbf{x}_r$  where  $1 \leq r \leq m$  from the dataset and refer it as *reference* sequence. Since each sequence contains a motif instance, suppose that the potential motif instance in the reference sequence is represented by the vertex  $v_{r\rho}$  where  $\rho$  indicates its starting position. As we are looking for  $l$ -length motifs, for each position  $\rho = 1, \dots, n_r - l + 1$  in the reference sequence, we build a graph  $G_\rho$  as follows:

1. For any other sequence in the dataset <sup>1</sup>,  $\mathbf{x}_i \in \mathbf{x}$ , find a subsequence  $\{\sigma_{ij}, \sigma_{ij+1}, \dots, \sigma_{ij+l-1}\}$ , represented by the vertices  $v_{ij}$  where  $i = 1, 2, \dots, m \neq r$  and  $j = 1, 2, \dots, n_i - l + 1$ ;
2. Connect  $v_{r\rho}$  and  $v_{ij}$  with edge  $e_{v_{r\rho}, v_{ij}}$  if  $dis(v_{r\rho}, v_{ij}) \leq 2d$ . For example, the figure 1 shows  $\mathbf{x}_i$  where  $i = 1, \dots, m$ , are the original sequences from the dataset and each sequence is converted to a set of vertices denoted by black points on the right side correspond to each sequence; the graph  $G_\rho$  is built by connecting  $v_{r\rho}$  to all the vertices corresponding to other sequences if the distances between them are no more than  $2d$ .

After constructing the graph, if  $v_{r\rho}$  represents a real motif instance in the reference sequence  $\mathbf{x}_r$ , the motif instances in other sequences should then be represented by the vertices in the same graph  $G_\rho$ . As such, the tenet of our approach is to convert the given dataset into a set of graphs  $G_\rho$  where  $\rho = 1, \dots, n_r - l + 1$  and look for cliques of size  $m$  such that each of the vertices in the clique represents an actual motif instance.

### 3.2 Clique Finding

After the construction of the set of graphs from the given dataset, a potential motif should appear in one of the graphs: the motif instances are represented by some vertices in this graph, which form a clique of size  $m$ . In what follows, we present a dynamic programming (DP) approach to search each graph for cliques of size  $m$ , if available.

1. Let the set  $S = \{s_r, s_2, \dots, s_m\}$  represents the subsets of vertices in the graph  $G_\rho$  such that  $s_r = \{v_{r\rho}\}$ ,  $s_2 = \{v_{21'}, v_{22'}, \dots, v_{2c_2}\}, \dots, s_i = \{v_{i1'}, v_{i2'}, \dots, v_{ic_i}\}, \dots, s_m = \{v_{m1'}, v_{m2'}, \dots, v_{mc_m}\}$ . For example, in figure 2, the subset  $s_i : i \neq r$ , contains the vertices correspond to sequence  $\mathbf{x}_i$  that have connection to  $v_{r\rho}$ , and  $c_i$  denotes the total number of such vertices. Note that  $s_r$  contains only one element as  $\mathbf{x}_r$  is considered as the reference sequence ( $r = 1$ ).

<sup>1</sup>We can take the reference sequence as the first one, i.e.,  $r=1$ , and the rest sequences are from the second to the  $m^{th}$  in the dataset.

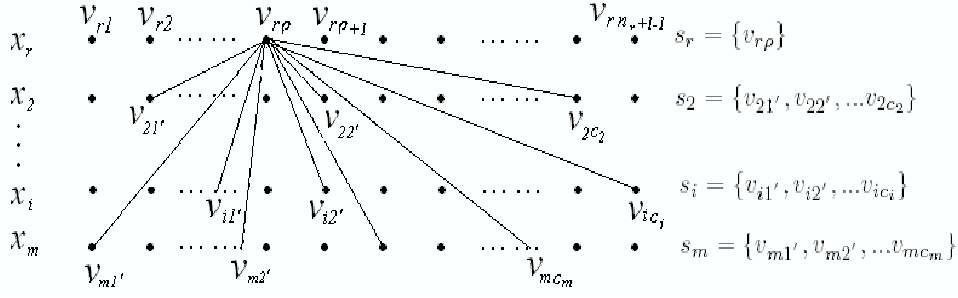


Figure 2: Illustration of clique finding in  $G_\rho$ : sets  $s_i$  for  $i = 1, \dots, m$  forms subsets of vertices in  $G_\rho$ , that are connected to the reference vertex.

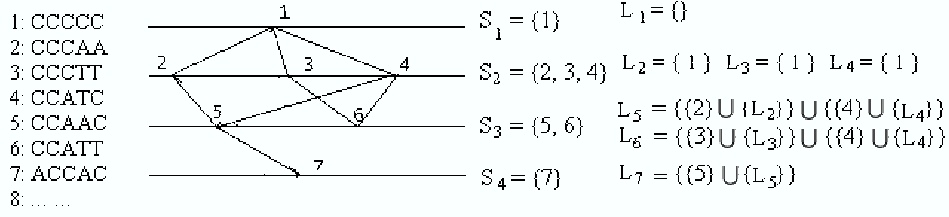


Figure 3: A simple example for clique finding for a  $(5, 1)$  motif. The numbers in the graph denote vertices, which represent subsequences indicated on the left.  $L$  denotes the lists created for each vertex during the clique finding process.

2. Lists  $L_{ij'}$ ,  $j = 1, \dots, c_i$ , are created as follows:

- (a) Set  $L_{r\rho} = \{\}$ .
- (b)  $L_{2j'} = \{v_{r\rho}\}$  for vertex  $v_{2j'} \in s_2$ .
- (c) for  $i = 3, \dots, m$ 
  - Set  $L_{ij'} = \{\}$  for  $v_{ij'} \in s_i$
  - for each vertex  $v_{i-1k'} \in s_{i-1}$ ,  $k = 1, \dots, c_{i-1}$ 
    - if  $dis(v_{ij'}, v_{i-1k'}) \leq 2d$
    - $L_{ij'} = L_{ij'} \cup \{\{v_{i-1k'}\} \cup L_{i-1k'}\}$
  - Repeat

3. By maintaining the above list for each vertex  $v_{ij'} \in s_i$ , if a clique with size  $m$  exists in a certain graph  $G_\rho$ , there must exist a list  $L_{mj'}$  for vertices  $v_{mj'} \in s_m$ , that contains vertices from each set  $s_i$  for  $i = 1, \dots, m-1$ . In other words, these vertices form a clique with size  $m$ .

In practice, if in a particular graph,  $G_\rho$ , most vertices are spurious, i.e., unrelated, and have been included in the list repeatedly, it could cost memory and time for maintaining such lists. As indicated by Pevzner et al. [11], when clique size becomes larger, less spurious vertices could be included. Therefore, we further process the list for each  $v_{ij'}$  to make sure that each vertex in  $L_{ij'}$  is within  $2d$  to  $v_{ij'}$  by deleting spurious subsets  $\{v_{st'}, L_{st'}\}$  ( $s < i$ ) from  $L_{ij'}$  whenever  $dis(v_{ij'}, v_{st'}) > 2d$ . This process is very effective because most of the lists become empty as the lists grow to form the cliques. The above process guarantees that any instances for a  $(l, d)$  motif will be found in the list of vertices in the set  $s_m$ .

For example, in figure 3, we look for a  $(l = 5, d = 1)$  motif. For simplicity, we use numbers to denote vertices in the graph and each vertex represents a subsequence with length 5 shown on the left.

Table 1: Illustration of a (8, 2) motif with 7 instances.

Motif Consensus	Motif Instances	Serial Number
AAAAAAAA	AAAAACAC	1
	CAAAACAA	2
	ACACAAAA	3
	CAAAAAAC	4
	AAAGAACA	5
	GACAAAAA	6
	AAGAGAAA	7

During the process of clique finding, list  $L_i$  is created for each vertex  $i$ . As an example,  $L_7$  is set to be NULL initially, there are two vertices, 5 and 6, in the former set  $s_3$ . Compare 7 with 5 and 6, only  $dis(5, 7) < 2$  then,  $L_7 = \{5\} \cup L_5 = \{5\} \cup \{2\} \cup L_2 \cup \{4\} \cup L_4$  as shown in the figure. As we mentioned earlier, a further check is applied to  $L_7$  and since  $dis(7, 4) > 2$ , then subset  $\{4\} \cup L_4$  is deleted from  $L_7$ , and we have  $L_7 = \{5\} \cup \{2\} \cup L_2$ . From  $L_7$ , we find a clique comprised of vertices  $\{7, 5, 2, 1\}$ .

### 3.3 Rescanning

After obtaining the motif instances from  $s_m$ , motif consensus  $\psi$  could be formed by alignment of the instances. As the sequences in the dataset becomes longer, spurious cliques could be found. However, a motif instance, say  $\psi_i$ , should satisfy the inequality  $dis(\psi, \psi_i) \leq d$  where  $\psi$  denotes the motif consensus formed by the alignment of all of its instances. Therefore, steps are taken for rescanning the dataset with the motif consensus derived from the earlier steps and saving those instances satisfying the inequality  $dis(\psi, \psi_i) \leq d$ . This guarantees that all the possible motif instances are found exactly in each sequence including the spurious instances which preserve as good as the real instances.

### 3.4 Weak Motifs

The more instances of a certain motif  $(l, d)$  are included in the dataset, the motifs can be recovered more confidently. For instance, table 1 gives seven instance of a  $l = 8$  motif. If we only get the first four instances of the (8, 2) motif, the first and the sixth position may not be recovered correctly, where we may get four kinds of different motifs. However, as more instances, 5, 6, or 7, are included, motif consensus can be recovered without ambiguity. In the following, we derive the relationship between the number of instances and the unrecovery rate of a motif.

For a  $(l, d)$  motif  $\psi$ , the probability for a nucleotide to mutate in a certain position is given by  $p_d = d/l$ ; similarly, the probability for it to conserve is  $p_s = 1 - d/l$ . Given  $N_\psi$  instances, the probability that a certain position of the motif could not be recovered from the alignment of the instances,  $p_{unrec}$  is given by:

$$p_{unrec} = \sum_{i=\lceil N_\psi/2 \rceil}^{N_\psi} \binom{N_\psi}{i} (p_d)^i (p_s)^{N_\psi-i} \quad (1)$$

which is a binomial distribution.

**Definition 3:** The *unrecovery rate* ( $UR$ ) of a motif is the number of positions that could not be recovered correctly from a given number of motif instances.

For a dataset containing  $N_\psi$  instances of a motif, as the inequality  $p_{unrec} \leq 1/l$  is satisfied, each position of the motif could be recovered correctly on average. Therefore,  $UR = lp_{unrec}$ .

**Definition 4:** The *random occurrence rate* ( $ROR$ ) for a motif is the probability to observe a motif instance randomly at a certain position in a sequence.

Table 2: Unrecovery rate (UR) and random occurrence rate (ROR) computed for representative weak motifs  $\psi$  with different number,  $N_\psi$ , of occurrence.

Motif ( $l, d$ )	$N_\psi$	UR	ROR	1/ROR
(15, 4)	12	1.0816	1.15e-4	8670
	13	0.5131		
(8, 1)	3	0.3438	3.8e-4	2621
(10,2)	7	0.3334	4.2e-4	2405
(12,3)	9	0.5871	3.9e-4	2553
(14,4)	15	0.5318	3.4e-4	2925
(17,5)	17	0.6056	9.9e-5	10011
(20,6)	19	0.6511	2.9e-5	33885

ROR for a  $(l, d)$  motif is given by:  $\sum_{i=0}^d \binom{l}{d} 3^i / 4^l$  and an indication of the strength of the motif.

We have simulated a set of weak motifs for our experimentation, as shown in the Table 2. The UR and ROR values for different motifs for given number of instances were computed. For example, a  $(15, 4)$  motif, when  $N_\psi = 12$ , one position in average may not be correctly recovered for the i.i.d. model, however, when  $N_\psi$  reached 13, UR (=0.5) becomes less than 1, i.e., each position could be recovered correctly. In our application, the calculation actually corresponded to the VM model [11]. where each position of the motif instance is mutated with a probability  $d/l$ .

The results indicate that as more instances of a motif are included, more easily the motif consensus could be recovered. As seen in the experiments, the performance of our algorithm, as sequence length increased for a  $(l=15, d=4)$  motif in the dataset, was better than other algorithms because our algorithm search for a global optimum when one occurrence of motif per sequence is present in the dataset.

### 3.5 Running Time

The estimated running time of WINNOWER [11] algorithm is about  $O(\binom{t}{k} n^k p^{\binom{k}{2}})$  for a  $t$   $n$ -vertex multipartite graph where  $k$  is the clique size and  $p$  denotes the probability of a connection between two vertices. When  $k=3$ , running time is approximately  $O((tD)^4)$ , where  $D=np$  is the expected edges between a vertex and a given part of the graph. cWINNOWER [6] improved this running time by a stronger constraint function and estimated the running time is about  $O(N^3)$  when  $k=2$  and  $O(N^4)$  when  $k=3$ .

In what follows, we approximately derive the running time of our algorithm. Let  $A = n \sum_{i=0}^{2d} \binom{l}{i} (3/4)^i (1/4)^{l-i}$  denote the random number instances of a motif  $(l, d)$  existing in a sequence with length  $n$ . The running time contains two parts: vertices comparison and list maintenance. Theorems 1-3 give the bounds of computation time of our algorithm.

**Theorem 1:** The vertices comparison time related to the  $i$  th sequence is bounded by  $A(A + (Ap)^{i-3}p^{i-4})$ .

*Proof:* for each vertex  $v$  related the current sequence  $\mathbf{x}_i$ , it costs time  $O(A)$  to compare with all the vertices in the former sequence, and in average there are  $O(Ap)$  vertices that satisfy  $dis(v, u) \leq 2d$ . In this case, all the vertices belong to  $L_u$  should be compared for the processing of the list, which cost  $(Ap)^{i-3}p^{i-4}$  to complete. Therefore, for the current sequence that contain  $A$  vertices in average, the total time is bounded by  $A(A + (Ap)^{i-3}p^{i-4})$ .

**Theorem 2:** The list maintaining time is bounded by  $A((Ap)^{i-2}p^{i-3})$ .

*Proof:* for each vertex  $v$  related to the current sequence, there are at most  $(Ap)^{i-2}p^{i-3}$  vertices in the former sequence that could be added into the  $L_v$ , therefore, since there are  $A$  vertices in average

for current sequence, the total time is at most  $A((Ap)^{i-2}p^{i-3})$ .

**Theorem 3:** for a dataset of  $m$  sequences each containing a single motif, the running time is bounded by  $O(n(mA^2 + A^{m-1}p^{2m-5}))$ .

*Proof:* for a dataset that containing  $m \geq 4$  sequences, the running time is given by:

$$\begin{aligned} & n \sum_{i=4}^m (A(A + (Ap)^{i-3}p^{i-4}) + A((Ap)^{i-2}p^{i-3})) \\ &= n((m-2)A^2 + \frac{2pA^2[1 - (nA^2)^{m-3}]}{1 - nA^2} + A^{m-1}p^{2m-5}) \\ &\approx n((m-2)A^2 + 2A^{m-2}p^{2m-7} + A^{m-1}p^{2m-5}) \\ &\approx n(mA^2 + A^{m-1}p^{2m-5}) \end{aligned} \tag{2}$$

The running time depends on parameter  $n$  as well as  $d$ . The number of vertices in a graph is proportional to  $n$ . Parameter  $d$  specifies how degenerate the motif is and the graph size is proportional to  $A$ . Also, the probability  $p$  increases rapidly as  $d$  increases. For instance, we simulated  $p$  by Monte Carlo simulations and when  $d$  was set to 4, 5, and 6 for a (15,  $d$ ) motif,  $p = 0.169$ ,  $p = 0.389$ , and  $p = 0.771$ , respectively.

Another trait of our algorithm is that the increase of the number of sequences in the dataset hardly affects the running time after sequence number  $m$  reaches certain value since the additional time cost is on graph construction and the maintaining of sparse cliques in the graph. As motif instances become more, only true cliques representing the motifs are left. This could also be seen from the running time derived above, as  $m$  becomes larger,  $A^{m-1}p^{2m-5}$  becomes very small and the running time could be approximated as  $O(nmA^2)$ .

## 4 Experiments

It is implausible to define a weak motif quantitatively as the weakness of a certain motif is influenced by the sequence length, the number of degenerated positions of a motif, etc. Therefore, testing different algorithms with synthetic datasets is an important way to compare the performance of our algorithm. In the following, our algorithm is first tested with synthetic data and then applied to real datasets to show its feasibility.

### 4.1 Synthetic Data

Firstly, we show that a certain number of motif instances need to be introduced for recovering a motif and as more instances are included in the dataset, the motif consensus could be recovered more confidently and correctly. We generated ten samples for each of the  $N_\psi$  value in a dataset with sequence length 600 nucleotides long. All the motifs in table 2 were tested and the results are shown in figure 4. The performance is evaluated by the measure  $|K \cap P|/|K \cup P|$  [11] where  $K$  is the known motif positions and  $P$  is the set of positions predicted by the algorithm.

We can see from figure 4 that, as motif instance number increased, the performance reaches 100%. This is because when the instances increases, the appearance of random cliques is eliminated and the unrecovery rate decreases. When  $N_\psi$  was small, the performance was low due to fact that the algorithms usually find spurious motifs as good as the embedded one.

The running time of our algorithm in all the testings above is below 10s, except in the case of (14, 4), it took around 5 mins to complete. This is comparable with RP method but much faster than WINNOWER and cWINNOWER, which took hours<sup>2</sup> to complete for one setting of parameters.

<sup>2</sup>The improved case of cWINNOWER in the setting of  $N=2000$ ,  $k=2$ ,  $q=32$  for recognition of a (15, 4) motif takes 3 hours 45 minutes [6].

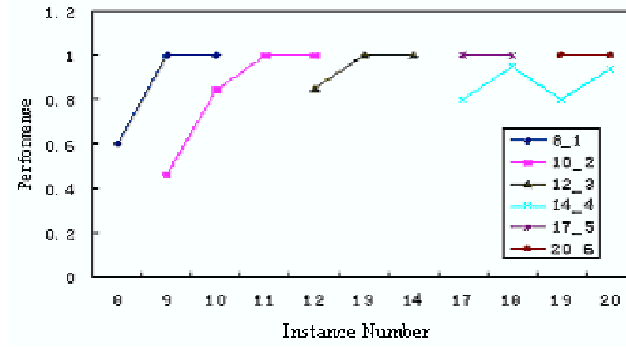


Figure 4: The performance of motif detection: the relationship between the motif instance number  $N_\psi$  and the performance of different  $(l, d)$  motifs. Each point in the graphs were derived by the average value from ten synthetic datasets.

Theoretically, if the sequence is long enough, random cliques could appear. The number of random  $k$ -cliques in the dataset is  $A^k * p^{\binom{k}{2}-k+1}$  where  $p^{\binom{k}{2}}$  is the probability to form a random  $k$ -clique. For example, a  $(15, 4)$  motif, when  $n=2000$ , we could get about 1.65 random 8-cliques; however, the random 9-cliques decrease to 0.0007, no more than 1.

We compared the performance for different algorithms as sequence length increased from 600 to 2000. Following Pevzner et al. [11], we generate eight samples of dataset for each sequence length. Every dataset contains 20 i.i.d. sequences, and every sequence was inserted with a randomly generated instances of the  $(l=15, d=4)$  motif.

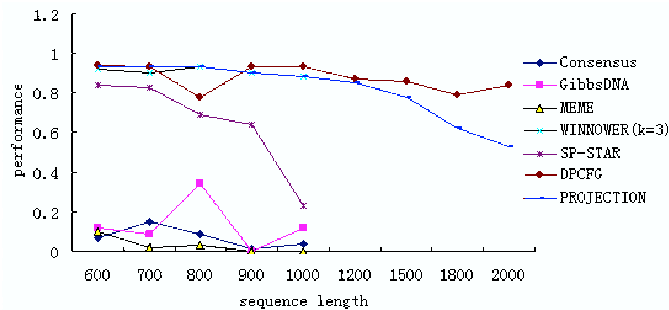


Figure 5: Performances of motif detection algorithms in detecting a  $(15, 4)$  motif at different sequence lengths; each point is averaged from 8 samples except for Random Projection algorithm which is averaged from 100 samples.

As seen, the results from figure 5 indicates that the performance of WINNOWER with  $k=3$  is as good as RP algorithm when the sequence length is no more than 1000. However, RP algorithm deteriorates beyond 1600. The other algorithms deteriorate rapidly as sequence length grows. However, all the instances that within  $d$  positions from motif consensus have been found by our algorithm; the decrease in performance of our algorithm at large sequence length is because we take the random instances as good as embedded motif instances as noise, which have been found by the rescanning step of our algorithm. However, the performance deterioration of other algorithms are mainly due to the failure of finding the real motifs. The sensitivity of our algorithm keeps almost at 100% at all sequence lengths with no false negatives.

## 4.2 Real Data

Here, we applied our algorithm to detect motifs in real datasets to show its feasibility. We first applied it to LexA dataset [5] without including the two sequences, *himA* and *uvrC*, which did not contain any LexA binding sites. The dataset contains 16 sequences, in which 12 of them are more than 200bps long, 4 of them contain more than one reported binding sites and two sequences do not contain any (*himA* and *uvrC*). The consensus TACTGTATATATACAGTA was successfully found with setting ( $l=20$ ,  $d=5$ ). Some of the other consensuses were reported, which corresponded with the shifting version of the real consensus.

We also applied the algorithm to the datasets used by Blanchette et al. [2] in phylogenetic footprinting. For the  $\gamma$  - *actin* 3'UTR dataset that contains six orthologous sequences range from 428 to 1667 nucleotides long, we found the motif TCATGCTAGCCTC, reported by them. We also found an even stronger motif TACACCTCATGCTAGCCTC when allowing for  $d=1$  degenerate position in a motif with length  $l=19$ . For the  $\beta$  - *actin* 3'UTR dataset that contains ten orthologous sequences range from 636 to 1107 nucleotides long, we found the motif CCTGTACTACTGAC by setting ( $l=13$ ,  $d=0$ ), which also reported by them. However, our algorithm found the stronger motif AAAAAGTGAACGGTGAAGG when setting ( $l=20$ ,  $d=2$ ), which means that there exist one motif instance in every orthologous sequence that has two different positions with this consensus. These longer motifs are beyond the detection capability of the methods used by Blanchette et al., however, since these motifs are well preserved ( $d$  is small relative to  $l$ ), our algorithm performs quite good on such datasets.

## 5 Discussion

Weak motif detection problem is important because there are potentially more degenerate functional sites in the non-coding regions of higher organisms. And in protein sequences, the motifs vary more than in DNA sequences. In this paper, we analyzed the weak motif detection problem, highlight the drawbacks in previous approaches, and proposed a new algorithm that showed better performance with synthetic data.

We showed that most earlier methods fail to detect weak motifs for several reasons: (1) when the motif length is long, that enumeration of all possible combinations is implausible; (2) the number of degenerate positions within the motif is so large that the heuristic methods usually get trapped in local optima; (3) sample driven approaches hardly succeed if motif consensus does not present in the dataset. The previous algorithms for weak motif detection, such as WINNOWER, are time-consuming. The proposed algorithm adopts the same idea of converting the motif finding problem to clique finding in graphs. However, our algorithm differs from WINNOWER in that WINNOWER builds a large graph initially, then deletes spurious edges and vertices, it recovers the motif in the rest parts of the graph; the current approach construct a set of graphs correspond to a reference sequence, then look for cliques with size equal to sequence number using dynamic programming, which is effective since the clique size used for motif finding is much bigger than WINNOWER. Other algorithm, such as Random Projection [3] could endure more degenerate motifs and has better performance, but is still a local improvement method.

Our approach is related to CONSENSUS [5] which adopts a greedy method and provides a global optimum algorithm finding all the motif instances at the end of dynamic programming. Since  $l$  and  $d$  should be specified initially for a given dataset without any prior information, a series values could be tested for our algorithm. The present algorithm is optimum when each sequence contains at least one motif instance. However, when certain number of random sequences are included in the dataset, the motifs could still be found by increasing  $d$  to accommodate more degenerate motif instances, for instance as in the dataset of LexA. When two sequences not containing any motif instances are included, our algorithm could still find the consensus with more degenerate setting ( $l=14$ ,  $d=4$ ).

In conclusion, we introduced a novel graph-based algorithm for weak motif detection, which used

DP for finding cliques representing the motifs. The proposed method outperformed earlier methods on synthetic data. It has better running times compared to WINNOWER and SP-STAR methods. The method successfully detected the motif instances in the real datasets. Further work is needed to extend the present method for the detection of motifs when multiple or no instances of motifs are present in the sequences.

## References

- [1] Bailey, T.L. and Elkan, C., Fitting a mixture model by expectation maximization to discover motifs in biopolymers, *2nd ISMB.*, 28–36, 1994.
- [2] Blanchette, M., Schwikowski, B., and Tompa, M., Algorithms for phylogenetic footprinting, *J. Comput. Biol.*, 9(2):211–223, 2002.
- [3] Buhler, J. and Tompa, M., Finding motifs using random projections, *J. Comput. Biol.*, 9(2):225–242, 2002.
- [4] Helden, J.V., Andre, B., and Collado-Vides, J., Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies, *J. Mol. Biol.*, 281:827–842, 1998.
- [5] Hertz, G.Z., Hartzell, G.W. 3rd, and Stormo, G.D., Identification of consensus patterns in unaligned DNA sequences known to be functionally related, *Comput. Appl. Biosci.*, 6(2):81–92, 1990.
- [6] Liang, S., Samanta, M.P., and Biegel, B.A., cWINNOWER algorithm for finding fuzzy dna motifs, *J. Bioinfo. Comput. Biol.*, 2(1):47–60, 2004.
- [7] Liu, J.S., Neuwald, A.F., and Lawrence, C.E., Bayesian models for multiple local sequence alignment and Gibbs sampling strategies, *J. Amer. Statist. Assoc.*, 90:1156–1170, 1995.
- [8] Liu, X., Brutlag, D.L., and Liu, J., Bioprospector: Discovering conserved DNA motifs in upstream regulatory regions of co-expressed gene, *6th Pac. Symp. Biocomputing.*, 127–138, 2001.
- [9] Liu, X.S., Brutlag, D.L., Liu, J.S., An algorithm for finding protein-DNA binding sites with applications to chromatin-immunoprecipitation microarray experiments, *Nat. Biotechnol.*, 20(8):835–839, July, 2002.
- [10] McCue, L., Thompson, W., Carmack, C., Ryan, M.P., Liu, J.S., Derbyshire, V., Lawrence, C.E., Phylogenetic footprinting of transcription factor binding sites in proteobacterial genomes, *Nucleic Acids Res.*, 29(3):774–782, 2001.
- [11] Pevzner, P. and Sze, S., Combinatorial approaches to finding subtle signals in DNA sequences, *Intelligent Systems for Mol. Biol.*, 269–278, 2000.
- [12] Sinha, S. and Tompa, M., A statistical method for finding transcription factor binding sites, *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 8:344–354, 2000.
- [13] Zhu, J. and Zhang, M.Q., SCPD: A promoter database of the yeast *Saccharomyces cerevisiae*, *Bioinformatics*, 15(7-8):607–611, 1999.