Tools for *Cis*-element Recognition and Phylogenetic Tree Construction Based on Conserved Patterns

T. Ji, K. Gopavarapu, D. Ranjan
Department of Computer Science
New Mexico State University
Las Cruces, NM, 88001

B. Vasudevan, C. S. Gopalan, M. O'Connell
Department of Plant and Environmental Sciences
New Mexico State University
Las Cruces, NM, 88001

Abstract

A major challenge in biological research today is extraction of useful information from large amount of raw biological data. We developed three tools - *CisFind*, *CisGibbs* and *CisTree* - to analyze genomic data based on conserved patterns. *CisFind* is used for finding common conserved patterns in multiple DNA sequences. *CisGibbs* and *CisTree* , which use *CisFind* as a core, are tools for *cis*-regulatory element discovery of the co-expressed genes and phylogenetic tree construction respectively.

CisFind utilizes a modified suffix tree algorithm to extract un-extendable common patterns of multiple DNA sequences. *CisGibbs* takes the output of *CisFind* as the input and utilizes Markov Chain Monte Carlo (MCMC) strategy to find possible *cis*-regulatory elements. When tested on data that was already verified through biological experiments, *CisGibbs* predicted *cis*-elements with good accuracy and its running time was low. *CisTree* is used to build phylogenetic tree of DNA fragments which are responsible for the same function in different species. *CisTree* measures the distances between sequences based on conserved patterns found by *CisFind* and builds trees by hierarchical clustering method. The tool was tested on data pertaining to Glutamine Synthetases (GS) and yielded promising results.

Keywords: *cis*-regulatory element recognition, phylogenetic tree construction

1 Introduction

Cis-regulatory elements (*cis*-elements) are specific short DNA segments that regulate the transcription of genes. Through transcription and translation, genes are finally expressed into various proteins which are the main functional components of living bodies. A basic requirement for understanding gene regulation is identification of *cis*-elements. Several algorithms and software tools have been developed to find potential *cis*-elements, such as Gibbs Sampling[1, 2], MEME[3], AlignACE[4], ANN-Spec[5], Random Projection[6], MITRA[7], Suffix Tree[8] and so on. These methods can be categorized into two basic types those that use machine learning algorithms, and those that use string searching and comparison. This paper presents a new method, *CisGibbs*, for recognizing potential *cis*-elements, which utilizes MCMC strategy based on a quick string searching. The method is based on the basic biological intuition that similar or similarly expressed genes will have similar *cis*-elements. The method is built on top of *CisFind*, which is a tool that provides an efficient way of detecting highly conserved patterns in multiple DNA sequences. The tool *CisFind*, that was also developed by our group and is useful in bioinformatics analysis in its own right, utilizes a modified suffix tree to find un-extendable common patterns. These un-extendable common patterns are viewed by *CisGibbs* as "core regions" for the putative *cis*-elements. The length of these core regions are then extended by using Bayesian models and Monte Carlo methods. We select the most possible *cis*-elements based on log-likelihood using a greedy algorithm. We used LexA, ABF1, NF-Y, and GRE protein binding sites, which have been experimentally tested, to test our approach. Our method improves the Gibbs Sampling method developed by Lawrence group[1, 2] in the way that we keep increasing the weights of positions in predicted *cis*-elements which show more conservation in training cycles. Besides, *CisGibbs* can identify all potential *cis*-elements whose "goodness" is above a certain user-specified threshold. Results show that these alternative possible binding sites are good alternative choices for prediction.

This paper also presents a tool *CisTree* for building phylogenetic trees. Construction of phylogenetic trees based on DNA sequences from different species is a commonly used and meaningful technique for evo-

lution analysis and sequence comparison. There are some algorithms for phylogenetic tree building, such as neighbor-joining[9], maximum parsimony[10] and maximum likelihood[10]. However, previously, the construction of trees is usually based on the whole DNA segments. This encounters one basic problem that the trees constructed may not be meaningful if only small parts of sequences and not the whole sequence is relevant for building the trees. *CisTree* measures distances among DNA sequences based on conserved patterns found by *CisFind* , and construct the tree by hierarchical clustering method. We used nine GS sequences to test our tool. The result gives us suggestions of basic clustering and sequence relationship among them. We found some possible *cis*-elements in GS sequences, and biological experiments on testing them are in progress at the department of Plant and Environmental Science of New Mexico State University.

2 Methods

2.1 *CisFind*

A DNA sequence is a string over the alphabet Σ={A,C,G,T}. String X is a substring of a string W if there exist strings Y and Z such that $W = YXZ$. Given a set of strings $S = \{S_1, S_2, \ldots, S_N\}$,$(K, L)$-conserved patterns are a set of substrings defined as $C = \{P|P$ is a substring in at least K strings in S, and has length at least $L\}$. We further define the notation of un-extendable conserved patterns. For a set of strings S, we say that a (K, L)-conserved pattern $P_1 \in C$ is extendable if $\exists P_2 \in C$ such that P_1 is a substring of P_2, and wherever P_1 occurs in any string in S, P_2 also occurs at the same place. After removing extendable patterns in C, we can get the set of (K, L) un-extendable conserved patterns $U = \{P_1, P_2, \ldots, P_M\}$ for set S.

CisFind is a tool to find un-extendable conserved patterns. *CisGibbs* and *CisTree* are based on *CisFind* shown in figure 1. We utilize a modified suffix tree to first get RU which is a set of right un-extendable patterns, and then get the set of un-extendable patterns. Given set C defined above, set RU based on C is a set that remove P_1 from C if there is pattern P_2 in C such that $P_1Y = P_2$ and string Y is not empty. For set $S = \{S_1, S_2, \ldots, S_N\}$, a suffix tree[11] is a tree that contains all suffix strings of all strings in S. We implemented the suffix tree by Ukkonen's linear time method[11] and modified this original suffix tree to get set RU. The way to modify a suffix tree is to form new strings based on S. Let $S' = \{S'_1, S'_2, \ldots, S'_N \,|\, S'_1 = S_1q_1, S'_2 = S_2q_2, \ldots, S'_N = S_Nq_N,\ q_1, q_2, \ldots, q_N$ are

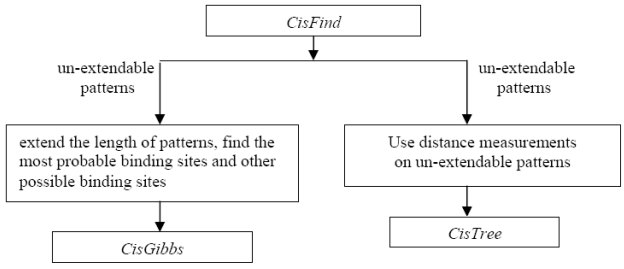

Figure 1: A brief diagram for *CisFind*, *CisGibbs* and *CisTree*

different from each other, and not in the alphabet $\Sigma\}$. Then, build a suffix tree for strings $S'_1, S'_2, \ldots, S'_N$. At each node, we record the number of strings in set S' reaching this node. If from the root to a node, the length of the pattern lying in this path is equal or longer than L, and this is the deepest node in this branch where at least K strings in set S' reaching it, then this pattern should be a $(K, L)-$ right un-extendable pattern.

To get set U, we need to check whether patterns in RU are also left un-extendable. One obvious method is to check each pattern in RU to see whether it is a substring of another pattern in RU. However, there is a faster way. First, reverse the patterns in RU, and build a tree for these reversed patterns. In this tree, each node represents a character in a pattern. If the end of a reversed pattern is an internal node, then this pattern is a prefix of another pattern. That is, the un-reversed pattern is not left un-extendable in set RU. After removing these patterns, we get set U.

The total time to get U contains the time to find set RU and the time to get U based on RU. The order of complexity for building the modified suffix tree is $\mathcal{O}(n)$ where n is the total length of all input strings. To get right un-extendable set, we need to traverse the tree which is also $\mathcal{O}(n)$. Suppose there are N patterns in set RU, and the total length of all these N patterns is L, then it takes $\mathcal{O}(L)$ to get U from RU. Thus, the order of complexity to get un-extendable patterns is $\mathcal{O}(n + L)$.

2.2 *CisGibbs*

Multiple DNA sequences from regulatory regions of likely co-expressed genes are first sent to *CisFind*. Since the un-extendable conserved patterns found by *CisFind* are exact matches, these short highly conserved patterns should be viewed as "core regions" of putative *cis*-elements. "Core regions" are more conserved during evolution, and expected to contribute

more to the binding function than other positions in a *cis*-element. Besides, these patterns are the ones only appear in at least K sequences of the total N input sequences. Nevertheless, all of the N input sequences are expected to contain the same binding site or sites. Thus, we need to extend these patterns into the whole proper promoters. To achieve this goal, we choose Gibbs Sampling method which is an example of MCMC algorithm. Set U of $S = \{S_1, S_2, \ldots, S_N\}$ by *CisFind* is the input for *CisGibbs*. We use greedy algorithm to do the optimization, that is, we will accept the predicted *cis*-elements in the next cycle of Gibbs Sampling training only if the alignment score (4) is higher than its current value.

Suppose pattern $P_k (P_k \in U)$ has length L_k, and will be extended to length L'_k for potential *cis*-elements. During the training, a "state" in the Markov Chain is defined as the set of predicted *cis*-elements. The initial "state" is the output of *CisFind*. To decide which state should be transited to from the current state, we randomly pick one sequence from the input DNA sequences, and randomly try three different new locations for the *cis*-element in this picked sequence. Each new location together with predicted *cis*-elements in other sequences yields a new state. Three alignment scores are calculated by (4) respectively for each new state. If the highest alignment score is higher than that of the current state, the current state will transit to this new state. Otherwise, the current state will transit to itself. And then, do the same updating to another sequence. Until all input DNA sequences have been updated once, this inner loop cycle of training is finished and the next begins.

For a state, we have following notations: $C_{i,j,n}$ is the count of letter $j (j \in \{A, C, G, T\})$ at position i in the predicted *cis*-element in $S_n (S_n \in S)$; The complement of predicted *cis*-elements is called "background"; $pseu_j$ is the pseudo count for letter $j (j \in \{A, C, G, T\})$ which is usually set as 0.1 of total number of letter j in background; $C_{0,j}$ is the count of letter $j (j \in \{A, C, G, T\})$ in the background.

$$C_{i,j} = \sum_{n=1}^{N} C_{i,j,n} \tag{1}$$

$$p_{i,j} = \frac{C_{i,j} + pseu_j}{N - 1 + \sum_{j \in \{A,C,G,T\}} pseu_j} \tag{2}$$

$$p_{0,j} = \frac{C_{0,j} + pseu_j}{\sum_{k \in A,C,G,T} C_{0,k} + \sum_{k \in \{A,C,G,T\}} pseu_k} \tag{3}$$

We use w_i to represent the weight of position i in a *cis*-element; w_i is set to 1 at the beginning. Because positions in a promoter do not contribute equally likely to binding function and we assume that the more conserved a position is, the more related it is to the binding function, and vice versa. As soon as the entropy of position i reaches a higher (or lower) threshold during the Markov Chain transition w_i will be increased (or decreased). Based on these notations, the alignment score of a state s is defined as follows:

$$AS(s) = \sum_{i=1}^{L'_k} \sum_{j \in A,C,G,T} w_i C_{i,j} log \frac{p_{i,j}}{p_{0,j}} \tag{4}$$

The high level algorithm for Gibbs Sampling training is described as follows. After getting the most possible *cis*-elements, we want to see whether we have alternative choices of predictions whose "goodness" is higher than a user specific threshold. We adapted Nucleic Weight Matrix (NWM) method[12] to find all other possible sites in the input sequences as long as the "similarity" of a pattern to the alignment matrix $M(P_k)$ exceeds an entropy threshold. $M(P_k)$ is the alignment matrix got from Gibbs Sampling viewing P_k as the core region. We construct it by calculating frequencies of four residues at each position. We first utilize (5) to evaluate the weight of each position, and then utilize (6) to evaluate the goodness of a pattern to this alignment matrix. Some notations are: $p_i(b)$ represents frequency of letter $b (b \in \{\text{A,C,G,T}\})$ at position $i (1 \leq i \leq L'_k)$ in the alignment matrix; $count_i(b)$ is count of letter $b (b \in \{\text{A,C,G,T}\})$ at position i in the alignment matrix; $MaxCount(i) = max_{b \in \{A,C,G,T\}}\{count(i, b)\}$. The weight of position i $W(i)$ is:

```
1   global_alignment_score ← 0.00
2   for outer_loop ← 1 to outer_loop_times
3       do set the output of CisFind as the current state
4          alignment_score ← weighted log likelihood value of the current state
5          local_alignment_score ← alignment_score
6          for inner_loop ← 1 to inner_loop_times
7              do update each sequence in the data set, set it as the current state
8                 alignment_score ← weighted log likelihood value of the current state
9                 if alignment_score > local_alignment_score
10                    do local_alignment_score ← alignment_score
11                       local alignment state ← current state
12                else if alignment_score < local_alignment_score
13                       do current state ← local alignment state
14             inner_loop++
15         if local_alignment_score > global_alignment_score
16             do global_alignment_score ← local_alignment_score
17                global alignment state ← local alignment state
18      outer_loop++
19  output global alignment state and global_alignment_score
```

$$W(i) = \frac{1}{log5} \times [\sum_{b \in \{A,C,G,T\}} p_i(b) \times \log p_i(b) + \log 5] \tag{5}$$

Goodness of a pattern P $G(P, M(P_k))$ with respect to the alignment matrix $M(P_k)$ is given by:

$$G(P, M(P_k)) = \frac{\sum_{i=1}^{L'_k} W(i) \times count_i(b)}{\sum_{i=1}^{L'_k} W(i) \times MaxCount(i)} \tag{6}$$

Total running time for *CisGibbs* is the sum of time for predicting the most possible *cis*-element in each sequence and finding all other possible sites. With respect to the first item, suppose the outer loop and inner loop times are O and I times respectively, the number of input DNA sequences is N, and the length of an un-extendable pattern is L which will be enlarged to length L', then the order of complexity of finding the most possible *cis*-element in each sequence is $\mathcal{O}(L' \times O \times I \times N)$. If the total length of the input DNA sequences is n, then the order of complexity of finding all other possible sites is $\mathcal{O}(n)$.

2.3 *CisTree*

To build a phylogenetic tree for DNA sequences, one important problem is how to measure distances among sequences. Here, we measure the distances based on the un-extendable conserved patterns found by *CisFind*.

Given set of strings $S = \{S_1, S_2, \ldots, S_N\}$, suppose the set of (K, L) un-extendable conserved patterns is $U = \{P_1, P_2, \ldots, P_M$ with length $L_1, L_2, \ldots, L_M$ respectively$\}$. We use $P_{k,n}$ to represent the pattern $P_k (P_k \in U)$ in the string $S_n (S_n \in S)$. If pattern P_k does not appear in S_n, then pattern $P_{k,n}$ is the most similar pattern with P_k found in S_n. If there are several patterns equally similar to P_k, and they are not the same, then $P_{k,n}$ is a probability combination of them. Those $P_{k,n}$'s form a set Ψ_k. However, $P_{k,n}$ and P_k may be quite different, we cannot directly use Ψ_k to measure the distances. We need to properly measure the "similarity" between two patterns, and deselect those which are far from P_k. Before introducing how to measure the similarity between two patterns, we first introduce some notations. $M(\Psi_k)$ is an alignment matrix based on Ψ_k. $p_{i,j}$ represents frequency of letter j at position i of $P_{k,n}$. $f_{i,j}$ is the frequency of letter j at position i of $M(\Psi_k)$. Thus, similarity between $P_{k,n}$ and $M(\Psi_k)$ is given by (7)[13]:

$$S(P_{k,n}, M(\Psi_k)) = \sum_{i=1}^{L_k} \sum_{j \in \{A,C,G,T\}} p_{i,j} ln \frac{p_{i,j}}{f_{i,j}} \quad (7)$$

Use $\Psi'_k (\Psi'_k \subset \Psi_k)$ to represent the selected ones from Ψ_k given a user-specific threshold using (7). We believe Ψ'_k are ready to be used. We build a frequency matrix based on Ψ'_k. For pattern $P_{k,n} \in \Psi_k$, which appears in sequence S_n, we give it a score by (8), where the frequency of letter b at position i in the alignment matrix based on Ψ'_k is denoted by $p_{i,b}$.

$$Score_n(\Psi'_k) = \prod_{i=1}^{L_k} p_{i,b} \quad (8)$$

where, $b \in \{A, C, G, T\}, 1 \le n \le N, 1 \le k \le M$

For each pattern $P_k (1 \le k \le M)$ in set U, construct the alignment matrix based on Ψ'_k in the same way, and then calculate $Score(P, \Psi'_k)$ for each pattern P in set Ψ_k. We define the normalized distance between strings S_i and $S_j (S_i \in S, S_j \in S)$ as follows.

$$Dis(S_i, S_j) = \frac{\sqrt[2]{\sum_{m=1}^{M} (Score_i(\Psi'_m) - Score_j(\Psi'_m))^2}}{\sqrt[2]{\sum_{m=1}^{M} (MaxScore_m - MinScore_m)^2}} \quad (9)$$

where, $MaxScore_m = Max\{Score_j(\Psi'_m)\}$

$MinScore_m = Min\{Score_j(\Psi'_m)\}$

$j \in \{1, 2, \ldots, N\}$ This distance can also be calculated using other distance functions, such as Manhattan distance, Minkowski distance (with different λ) and so on. We build the tree by Hierarchical clustering approach. In the tree, each leaf represents a DNA sequence. We keep joining the most adjacent two DNA sequences. After joining them, they are viewed as "one" DNA sequence. There are several strategies to recalculate distances between this newly joined "DNA sequence" and others, such as complete linkage, single linkage, average linkage, centroid and so on. We implemented seven methods. One is allowed to choose a strategy from the menu.

3 Results and Discussion

3.1 Results and discussion for *CisGibbs*

To test *CisGibbs*, we used four kinds of protein binding sites: lexA, NF-Y, GRE and ABF1 which are retrieved from the Institute of Experimental Genetics. These DNA sequences are from the promoter regions ranging from 200 nucleic acids to 500 nucleic acids long.

With respect to lexA protein binding sites, we found pattern ACTGTATAT by *CisFind* from 13 sequences containing this protein binding site. Then, send it to the Gibbs Sampling and enlarge the length to 19. Compared with the real *cis*-elements of lexA tested through biological experiments, our prediction of the promoter sites is only one nucleic acid position shifted. However, if we examine the log likelihood, our prediction has a higher value of log likelihood. With respect to NF-Y, our prediction can get all the correct sites except for one sequence. However, as described in section 2, *CisGibbs* can detect other possible binding sites in each sequence. And the correct binding site of the wrongly predicted is in the alternative possible *cis*-elements' group. With respect to GRE protein binding sites which are bipartite binding sites with a

short spacer, we can correctly find binding sites of 6 or 7 sequences among the total 17 sequences. However, the correct binding sites can usually be found in the "other possible binding sites" group. These alternative possible binding sites can well make up the flaws after training from Gibbs Sampling. The result for the ABF1 binding site is less positive. The reason for this is because the core region for ABF1 is very short. Nearly every position in ABF1 is flexible. It is difficult to detect this short conserved pattern for *CisFind*.

3.2 Results and discussion for *CisTree*

Promoter sequences for Alfalfa cytosolic GS - MsGS100 and MsGS13 were isolated from Alfalfa genomic library using Universal genome walker kit. The isolated DNA fragments were confirmed by sequencing and used for insilico analysis. MsGS100 is more constitutive and is expressed everywhere in plant whereas MsGS13 is though expressed in other parts of plant but is more nodule enhanced. The promoter regions for other GS were extracted from GenBank database. The goal of our computational method is to discover binding sites by clustering genes having similar expression patterns and then identifying common patterns in promoter regions of those genes. This approach has been made to study transcription factor profile in Plant GS genes.

From analysis and observation, the intuition of clustering result of nine GS promoter sequences we collected is that five of them should be in one group which are more nodule enhanced, and the left four should be in the other group which are expressed less in nodules. Meanwhile, *CisTree* also gives us a possible grouping and relationship among them. The result is similar with our intuition. However, one sequence is grouped differently. Now, we have used *CisFind* to detect distinct conserved patterns between these two groups. And we expect these specific patterns play as regulatory elements in controlling the root nodule function. Biological experiments are in progress to test the predicted *cis*-elements in the department of Plant and Environmental Science at New Mexico State University. Besides, we believe if we can have more sequences from these two groups, the grouping and pattern extraction can be more reliable.

4 Software User Interface

4.1 User interface for *CisFind*

The inputs for *CisFind* are multiple DNA sequences which should be stored in FASTA format. In the main menu of *CisFind*, choose the folder containing these sequence files, then choose the minimum length of patterns and the minimum number of sequences they want these patterns appear in the total sequences. Then *CisFind* will display all patterns it finds. Figure 2 is a typical output which shows the un-extendable conserved patterns found in 15 GS sequences from promoter regions. DNA sequences are shown as lines. Patterns are shown as colored ovals on the DNA sequences. Each color represents a different pattern. We added mouse events to it. When user click a pattern, all other sites of this pattern will turn the color into black and the positions of these patterns will be showed. Meanwhile, at the place the mouse clicked, the position of the DNA segment, the pattern and the length will be shown. The right most list on the screen is the list of all un-extendable patterns got by *CisFind*.

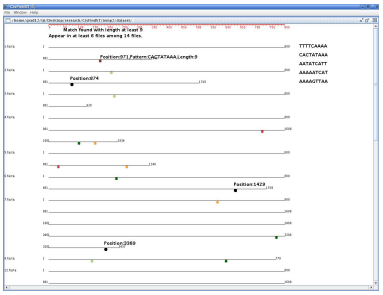

Figure 2: A typical output of un-extendable conserved patterns from 15 GS sequences

4.2 User interface for *CisGibbs*

We can select the patterns we are interested in from the output of *CisFind* as the input of *CisGibbs*. Take lexA binding site as an example. We found pattern ACTGTATAT from *CisFind*, and extend its length to 19. The output after training is showed in figure 3. The left columns in the output contain the sequences and the positions of these *cis*-elements in each sequence. In the right columns, the small letters are the most possible *cis*-elements in each sequence. The ten capital letters before and after the small letters are five nucleic acids respectively before and after the predicted *cis*-elements in the DNA sequences. By setting a threshold of 0.800, other possible binding sites are showed in figure 4. The most right column shows the goodness of this predicted promoter to the alignment matrix from Gibbs Sampling result.

4.3 User interface for *CisTree*

A typical output of *CisTree* is showed in figure 5. Files whose names begin with "1" are sequences

Figure 3: Output of *lexA* binding sites after training by *CisGibbs*

```
                *TACTGTATATNHAHHCAGN
Seq 5 125:  TACAGTATTTATTTTTAAC  0.825424
Seq 5 145:  TATTGTTTTAAAAGTCAAA  0.812725
Seq 6  83:  TGCTGTATATACTCACAGC  0.84729
Seq 8  93:  TACTGTACACAATAACAGT  0.834168
```

Figure 4: Other possible sites for *lexA* binding sites

having lexA binding sites, and those beginning with "3" have NF-Y binding sites.

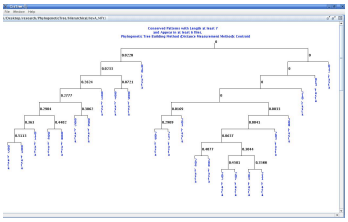

Figure 5: A typical output of *CisTree* using promoter sequences containing lexA and NF-Y binding sites

5 Conclusion

We developed algorithms for recognition of *cis*-elements and phylogenetic tree construction based on conserved patterns in multiple DNA sequences with low time complexity. These algorithms were implemented, and tested by biological data of several known binding sites. The results are promising. We also used them to detect new *cis*-elements in GS sequences. Biological experiments for testing our predictions are going on in the Department of Plant and Environmental Sciences at New Mexico State University.

Acknowledgments

This work was partially supported through NSF grant # HRD-0420407.

References

[1] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, J. C. Wootton, "Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment," Science, vol.262, pp.208-214, 1993.

[2] A. F. Neuwald, J. S. Liu, C. E. Lawrence, "Gibbs motif Sampling: detecting of bacterial outer membrane protein repeats," Protein Science, vol.4, pp.1618-1632, 1995.

[3] T. L. Bailey, C. Elkan, "Fitting a mixture model by expectation maximization to discover motifs in biopolymers," Proc. Int. Conf. Intell. Syst. Mol. Biol., vol.2, pp.28-36, 1994.

[4] J. D. Hughes, P. W. Estep, S. Tavazoie, G. M. Church, "Computational identification of *cis*-regulatory elements associataed with groups of functionally related genes in *saccharomyces cerevisiae*," JMB, vol. 296, pp.1205-1214, 2000.

[5] C. T. Workman, G. D. Stormo, "ANN-Spec: a method for discovering transcription factor binding sites with imporoved specificity," Pac. Symp. Biocomput., vol.5,pp.467-478, 2000.

[6] J. Buhler, M. Tompa, "Finding motifs using random projections," J. Comput. Biol., vol.9,pp.225-242, 2002.

[7] E. Eskin, P. A. Pevzner, "Finding composite regulatory patterns in DNA sequences," Bioinformatics, vol.18,pp.354-363,2002.

[8] L. Marsan, M. Sagot, "Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification," J. Comput. Biol., vol.7,pp.345-362, 2000.

[9] N. Saitou, M. Nei, "The neighbor-joining method: a new method for reconstructing phylogenetic trees," Mol. Biol. Evol., vol.4,pp.406-425,1987.

[10] R. Durbin, S. Eddy, A. Krogh, G. Mitchison, Biological sequence analysis: probabilistic models of proteins and nucleic acids, Cambridge Univeristy Press, New York, 1998.

[11] E. Ukkonen, "On-line construction of suffix-trees,"Algorithmica, vol.14, pp.249-260, 1995.

[12] K. Quandt, K. frech, H. Karas, E. Wingender and T. Werner, "MatInd and MatInspector: new fast and versatile tools for detection of consensus matches in nucleotide and sequence data," Nucleic Acids Research, vol.23, pp.4878-4884, 1995.

[13] 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,"Proc of the 3rd International Conference on Bioinformatics and Genome Research, 1995.