Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2007)

Efficient and Scalable Motif Discovery using Graph-based Search

Amit U Sinha and Raj Bhatnagar
Dept. of ECECS, University of Cincinnati, Cincinnati, OH 45221, USA

***Abstract*—Identification of short repeated patterns (motifs) in genomic sequences is the key to many problems in bioinformatics. The promoter regions of genes are an important target of search for such motifs (Transcription Factor Binding Sites). We present a new algorithm, Mottice, for detecting potential binding sites which are present in a given set of genomic sequences. An informed search is performed by organizing the input patterns and their variants in a graph. Such a strategy efficiently leads to the desired solutions. The background is modeled as a Markov process and a composite score function is used. We demonstrate the performance of our algorithm by testing it on real-life data sets from yeast and human promoter sequences. We compared the performance with several popular algorithms and found that other algorithms work well with lower organisms like yeast but only a couple of them work well with human data. We show that our algorithm scales linearly with the size of input dataset. We compare the computational efficiency of our algorithm with other algorithms and show that it performs faster for different datasets and motif sizes.**

I. INTRODUCTION

The completion of many genomic projects has generated huge amount of sequence data. Such data needs to be functionally annotated for it to be of any practical use. On the other end, computational intelligence techniques have been used for more than a decade to find useful information from large datasets. However, biological data provides unique challenges which give an impetus to generating novel data mining paradigms for assigning functional roles to raw DNA (and protein) sequences.

In addition to the long functional regions like genes, there are many short sequences in the genome which play an important role in the cellular processes. Search for such short repeated patterns is known as the motif discovery problem. Though there are many variants of this problem, here we focus on detection of Transcription Factor Binding Sites (TFBS).

The transcription of a gene is initiated by proteins called transcription factors (TFs). These proteins bind to specific locations (TFBS) on the DNA to regulate the transcription of the gene. Typically, a number of TFs act in combination to activate or repress transcription. Discovery of these individual or modular TFBSs is important as it helps in understanding the gene regulatory mechanism.

A number of bioinformatics approaches are in vogue for identification of the TFBSs. Conventionally, a sequence is scanned for motifs that match known TFBSs which have been experimentally identified from promoters or other regulatory sites. Experimental data of the specific binding sites of most well-characterized TFs have been compiled in databases such as TRANSFAC [1] or JASPAR [2]. Examples of such programs include MatInspector [3] or MATCH [4] which are used to compare a genomic sequence input to all the binding site data in the TFBS libraries, and return a list of potential TFBSs based on a statistical match between a region in the sequence and a binding site. However, these approaches result in a large number of false positives. Additionally, the completeness of the TFBSs libraries is a major limitation. Not all TFBSs have been identified, and even for some of the known TFs, their binding specificity has not been fully characterized yet.

An alternative approach to overcome this problem involves a search for common or over represented sequence motifs within the upstream regions of a group of genes. The main advantage of these approaches is the potential to discover novel TFBSs. Several programs are currently available for this purpose – each with its own advantages and caveats (see section II on Related Work). There is clearly room for improvement, especially when predicting regulatory regions in higher metazoans wherein TF cooperativity is much more widespread than in lower eukaryotes or yeast.

The identification of TFBSs – the short functional motifs - is challenging because these binding sites are small and degenerate. For example, if the experimentally validated binding site for a specific transcription factor is ACGTACGT, the transcription factor can still bind to sites that are slightly different than ACGTACGT (AGGTACTT, CCATACGT, etc.). An example is show in Fig. 1.

atcgctatctgtctatccAgGTACtTaggtcct
atttgatcCaTACGTacaccggcaacctgaa
aaACGTAaGTgcaccctctcttcgtggctctg
tgccacccctattacatcttACGTcCaTataca

Fig. 1. Four instances of motif ACGTACGT

The motif ACGTACGT is present in all the 4 sequences, albeit with one or more variations. These patterns are called the *instances* of the motif in the input sequences. The instances are marked in bold and the base which does not match the known motif is shown in lower case.

A recent review [5] brings forth two computational (or statistical) tasks in the *de novo* motif discovery process; the *search* step and the *scoring* step. The search step is a comprehensive identification of patterns which are overrepresented in the input sequences. We call such patterns *candidate motifs*. The scoring step is separating likely motifs from patterns which occur just by chance. Usually a score is assigned to each candidate motif based on some statistics. The motifs are then ranked according to the score and the top ranking motifs are reported to the user.

For the search step, the definition of a binding site motif is itself disputed but the commonly accepted formulation of the

motif discovery problem is defined in the following way – given t sequences of length n (usually promoter regions of co-regulated genes), find the set of binding sites of size l present in all (or some) of the sequences with at most d mutations (mismatches) [6], [7]. Such a search for binding sites in genomic sequences is found to be NP-Hard [8]. So a complete polynomial solution is *not likely* to exist. Moreover, a naïve search may discover thousands of candidate motifs which satisfy the constraints but not all of them may be true motifs.

For the scoring step, the challenge is to predict the likely motifs from the large number of false positives. Since the size of these motifs is very small (6-15 base pairs), the probability of multiple occurrences of a pattern by chance is very high. There is no direct way of assessing if a transcription factor indeed binds at a predicted site. Usually the predicted results are experimentally verified which may take a few months. Statistical measures are used to find the chance that the match is not just a random occurrence.

We present the Mottice algorithm for de novo sequence motif discovery. We address both the search and the scoring aspects of the problem. We develop an exhaustive yet efficient search method coupled with a composite scoring function. The scoring function is also used for pruning the candidates during the search process. The use of auxiliary information such as orthologous sequences, phylogenetic footprinting, etc. is known to improve the prediction of the motifs. However, [5] suggests the use of such information, if available, as a post-processing step. In this paper the focus is on the computational challenges of the problem.

II. Related work

The motif discovery problem has been studied extensively over the last decade. There are a large number of tools available; we will focus on some of the most important ones. MEME [9], one of the earliest tools, is profile-based method using expectation maximization for selecting the motifs. Consensus [10] is also based on profiles. It starts by creating a profile from the n-mers (strings of size n) in the input sequence and then does pair-wise merging of profiles to construct more prevalent motifs. These algorithms make approximations while searching for frequently occurring patterns and may not get the right output in each case. Quite a few tools are based on Gibbs sampling strategy such as AlignAce [11] and BioProspector [12]. There are some other algorithms based on oligo/dyad-analysis [13], etc. The search is based on local search techniques so such methods are often trapped in local minima. Ref. [14] compared some of these methods and found that they work well with low complexity genomes such as yeast but perform poorly on human genomes, etc.

Pattern based enumerative algorithms have performed better in recent comparisons [14], [16]. Weeder [7] stores the input sequences in a suffix tree. All the possible motifs are enumerated and searched in the suffix tree for finding their occurrences in the input sequences [17]. Such an approach is computationally expensive. Since the number of possible patterns is exponential in the length of the pattern, heuristics are applied to speed up the search. Another recent pattern based algorithm, MaMF [16], searches the input space by starting with pairs of patterns which share a short subsequence of length 4-6. These pairs are then combined to get larger sets of related patterns. It has been shown to work better than other algorithms but the greedy search strategy is not guaranteed to find the best solution.

We propose the Mottice algorithm which is deterministic and exact. Unlike enumerative algorithms, all possible solutions are not generated; the patterns present in the input are organized in a graph to quickly find the solutions.

Graph based search methods such as WINNOWER [6] and [18] have been proposed earlier. These algorithms perform either a local optimization search or an almost exhaustive search of possible solutions. In order to conquer the exponential nature of the search space we need to do an as informed a search as possible so that non promising parts of the search space may be pruned away early. Our proposed approach uses a scoring function at each stage of the search process to successively narrow down the space of possible hypotheses. Most other approaches use the scoring function only after the exhaustive search to evaluate the motifs. Our robust search technique works well with noise in data; the performance remains good even when motifs are only present in only some of the input sequences, a situation often encountered in real-life. Further, these methods have been tried on synthetic challenge data sets or low complexity bacterial or fungal genomes. We couple our efficient search method with a hybrid scoring function which performs well even with real-life datasets of more complex eukaryotic genomes apart from prokaryotic genomes.

In addition to the different search strategies, a large number of scoring schemes have also been proposed by different groups. A popular method is to train a Markov model on the background and then predict the probability of occurrence of the motif based on the model. Consensus [10] uses the p-value of motif profile, [19] proposes the use of z-score, etc. These scores capture different properties of the motif. Methods based solely on one of these do not perform well. On the other hand, methods which perform well use a combination of these statistics. Weeder [7] uses the z-score and posterior probabilities while MaMF [16] uses the conservation of bases between the different instances of the motif and its probability of occurrence based on third order Markov model.

Most of the above algorithms focus very little on computational efficiency and scalability. As more data is being made available, these issues will become more important. In this paper, we focus on these aspects of the motif discovery process.

Further, the current algorithms work like a black box as they only report the instances of the motifs found. Our graph based framework enables us to not only find a motif but also provides the relationship between its different instances.

III. Methods

The motif discovery problem is formally defined as following. Given a set of t sequences $S = \{S_1, S_2, \cdots, S_t\}$, each

of length n, find a pattern p of length l, such that there exists a pattern q_i in each S_i, i = 1, 2, ···, t, and $d_H(p, q_i) \leq d$. The distance measure, $d_H(a,b)$ is the hamming distance which measures the number of mismatches between patterns a and b. If such a pattern p exists, it is called a (l,d) motif. The parameters l and d are usually provided by the user; typical values for l are 6-15 and d is usually 0-30% of l. Since there is a chance that the motif may not be present in all the input sequences, the *quorum* condition is relaxed, i.e., all motifs which are present in q (out of t) sequences are reported. It may be noted that the pattern p may or may not occur explicitly in S as shown in Fig. 1. Typical value of n, the length of the promoter region is 500-1000 base pairs. Finally, the candidate motifs are ranked based on the likelihood of their being a non-random pattern.

In this case the similarity measure is the hamming distance which is the number of mutations between two patterns. If patterns a and b differ at exactly k positions ($d_H(a,b) = k$) then pattern a is called a k-mutant of b and vice-versa.

A. Search using Multi-level Graph

The patterns, the motif and the intermediate patterns are arranged as nodes in a graph. (*Intermediate patterns* are the patterns which are generated during the search for motifs). The l-mers derived from the input sequence form the nodes in the bottom level (level 0) of the graph. Patterns in level 1 are 1-mutants of patterns in level 0. Similarly, patterns in level k are 1-mutants of patterns in level k-1. In the graph, each pattern is connected to its 1-mutants. A pattern a in level i is connected to a pattern b in level j if a is a |j-i|-mutant of b. The (possible) motifs are nodes of the graph which are connected to at least one node from q sequences in level 0. The connection may be through intermediate nodes.

Seq 1: GCGCTA
Seq 2: GCTCGA
Seq 3: GCGCGA

Fig. 2. A sample dataset with 3 sequences of length 6 each

A sample dataset is shown in Fig. 2. The dataset contains 3 sequences of length 6 each to illustrate the construction of the graph structure. Fig. 3 shows part of the graph for the dataset in Fig. 2. We search for motifs (6,2), i.e., find patterns of length 6 which has at most 2 mismatches from a pattern in each input sequence. The patterns from the 3 sequences form level 0 of the graph. The 1-mutants of each node in level 0 are added to level 1 as a new node and a connection is made between the nodes. (Not all the 1-mutants are shown in the figure). Similarly, 1-mutants of nodes in level 1 are added to level 2. Before adding a 1-mutant to a higher level, this (higher) level is searched for the presence of that pattern. If it already exists, simply a new connection is made. For example, pattern GCGCTA in level 0 adds its 1-mutant GCTCTA to level 1. Now the pattern GCTCGA in level 0 also generates a 1-mutant GCTCTA. Since GCTCTA already exists in level 1, a new node is not created but a connection is made from the parent node (GCTCGA) to the existing node (GCTCTA). The patterns themselves are also directly added to the next higher level. The top-most level is used to create the next higher level in each iteration of the algorithm. The method is similar to Consensus [10]. However, instead of generating all possible Position Weight Matrices (PWM), only 1-mutants are generated so it works much faster.

In this example there is (6,1) motif and a (6,2) motif which are indicated by a thick oval. The node GCGCGA in level 1 is connected to one node from each sequence in level 0 so it is a (6,1) motif. The node GCTCAA in level 2 is connected to one node from each sequence in level 0 (through intermediate nodes in level 1) so it is a (6,2) motif. A motif in level k is always a (l,k) motif as it is at most k apart from the patterns in the input sequence in level 0.

B. Search Algorithm

The Mottice algorithm is based on exhaustive enumeration of all 1-mutants of a pattern to generate the next higher level. The algorithm is shown in Fig. 4.

Step 1 of the algorithm gets all the l-mers from the input sequences and constructs the bottom level of the graph. An input sequence of length n will have (n-l+1) overlapping l-mers. All the nodes of a level are expanded by generating its 1-mutants (patterns p). If the pattern already exists in the next higher level (step 5), then an edge is drawn between the two nodes else a new node is created (step 6). Once the graph is constructed, the nodes in the graph which meet the quorum are reported as candidate motifs for measuring their statistical significance and ranking. (The quorum is computed by counting the number of input sequences in which a motif is present. This is found out by looking at the nodes at level 0 to which a node is connected.)

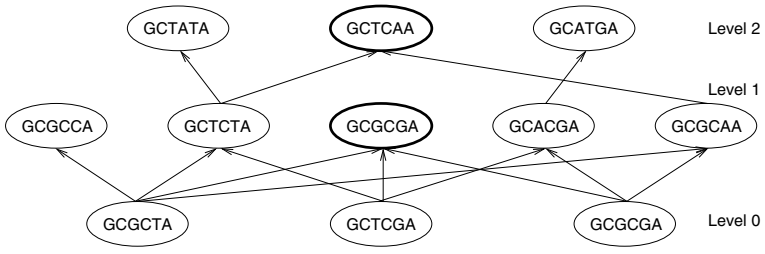

Fig. 3. The graph for a sample dataset

Input: *t* sequences, *l*, *d*, *q*

1 Form level 0 with all l-mers of all the *t* sequences
2 **for** each level L_i, *i* = *0* to (*d* – *1*)
3 **for** each pattern a_{ij}, a_{ij} ε L_i
4 Generate all 1-mutants of a_{ij} (patterns *p*)
5 **if** *p* already exists in level L_{i+1}, form a connection between the two nodes
6 **else** create a new node in level L_{i+1} and connect the new node to a_{ij}
7 Nodes in the graph which are connected to at least one node from *q* sequence in level 0 are the motifs.

Fig. 4. Algorithm Mottice

t input sequences will give rise to $t(n-l+1)$ input patterns. As $n >> l$, number of nodes in Level 0 is almost equal to *tn*. Each pattern will produce $3l$ 1-mutants, not all of which will be different from the existing patterns. However, in the worst case, the number of nodes in Level 1 will be $tn(3l)$. Similarly, the number of nodes in Level 2 will be $tn(3l)^2$.

Generalizing this, number of nodes in level $k = tn(3l)^k$. Total number of nodes in the graph with *d* levels = $tn + tn(3l) + tn(3l)^2 + \ldots + tn(3l)^d = tn((3l)^{d+1} - 1)/(3l - 1) = O(tn(3l)^d)$.

Similar argument can be made for finding the time-complexity of Mottice. In the first iteration of the algorithm, each of the *tn* nodes of Level 0 has to be processed. Time required will be proportional to *tn*. When processing Level *k*, $nl(3l)^k$ nodes have to be processed. So time required = $O(tn(3l)^d)$

So, the worst case complexity is a linear function of the input size (*tn*) and a polynomial function of the length of the motif but exponential in the number of allowable mismatches. This is not surprising as the problem is NP-Hard. However, the worst case scenario is not encountered in practice as many duplicate patterns are generated at each level as we start with only the patterns in the input sequences.

C. Modeling the Background

The promoter regions are not just random sequences of the four bases; they often have strong biases. For example, human promoter sequences of genes regulated by the transcription factor E2F have 29% 'G's and 'C's while they have only 21% 'A's and 'T's. Further there are many repetitive patterns like CG repeats, etc. A lot of patterns generated by the algorithm may simply be an artifact of the bias of the input sequences rather than true signals. So we need a way to discriminate between a signal and the background (the input sequences).

First, the known repetitive elements in DNA (such as 'A' repeats) are discarded using RepeatMasker [21]. Second, the input sequences are modeled as a third order Markov process. Though the choice of order is debatable, third order Markov model is ideal as many common genomic signals are 4 bases long, e.g., TATA boxes. Using such a model, the probability of a sequence *s* in the context of its background is calculated as follows.

$$P(s) = P(s_1s_2s_3)P(s_4|s_1s_2s_3)\,P(s_5|s_2s_3s_4)\ldots P(s_l|s_{l-3}s_{l-2}s_{l-1})$$

where P(*s*) is the probability of the occurrence of sequence s of length *l* and s_i is the *i*th letter of *s*. Such a model may be constructed locally or globally for a data set. In the former, the background model is created using only the promoter sequences in which motif is being searched while in the latter all the promoter sequences of the genome are used to construct the model. We use a local model for two reasons. Firstly, tools like BioProspector [12] allow both options but they perform better with local models. This is reasonable as a local model will capture the fine nuances in the input sequences. Secondly, it is not always possible to construct a global model as data of all the promoter regions of the genome may not be easily available, especially for newly sequenced genomes.

Having constructed the model, the probability of each of the candidate motifs is calculated. The motifs which have low values of probability signify that they represent a potentially true signal, and they are not merely an artifact of the sequence biases.

D. Ranking the Candidates

The algorithm described in the section III.B generates a large number of candidates, very few of which are likely to be true motifs. Simply measuring the probability of a candidate sequence using the background model (as described in section III.C) does not suffice to discover true motifs. Various statistical measures have been proposed to measure the likelihood of a candidate being a true motif [9], [10], [19], [11]. Each of these measures captures a different type of information about the motif. E.g., for a given set of occurrences of a motif, relative entropy measures the level of conservation of a base at each position of the motif pattern. It is found that the relative entropy score is usually high for known motifs. However, such measures fail to discriminate the true motifs when used alone. So we propose a composite scoring function *S(M)* given as:

$$S(M) = S_p \times S_{ic} \times S_s$$

$$S_p = -\frac{1}{q}\sum_{i=1}^{q}\log(P_o(M_i))$$

$$S_{ic} = \sum_{i=1}^{l}\sum_{j=A}^{T} p_{ij}\log\frac{p_{ij}}{P_{Bj}}$$

$$S_s = \sum_{i=1}^{q}\sum_{j=i+1}^{q}[l - d_H(M_i, M_j)]$$

where *M* is the motif, M_i is the *i*th occurrence of the motif, *q* is the total number of occurrences of the motif in all the input sequences, $P_o(M_i)$ is the probability of observing pattern M_i by the background model, p_{ij} is the probability of observing base *j* at position *i* in the motif profile, P_{Bj} is the background probability of observing base *j*, *l* is the length of the motif and $d_H(M_i,M_j)$ is the hamming distance between patterns M_i and M_j.

Each component captures a different type of information about a candidate motif. The first component, S_p, measures the average of logs of the probability of occurrence of the instances of the motif. The second component, S_{ic}, compares the information content (or relative entropy) of all the instances of the motif with the known background probability of each base. The third component, S_s, measures the mutual similarity between the different instances of the motif. Together they should be more helpful in picking the true motifs as a true motif will score high in each measure.

After the candidates are generated by the algorithm, a score is assigned to each candidate motif. They are ranked based on decreasing order of the score and a few top ranking motifs are reported to the user.

E. Pruning the Search Space

Modeling the background also helps in pruning the graph for speeding up the search. The idea is that a pattern which has a high probability of occurrence is not likely to be an instance of a motif. So the nodes in the graph which have a probability higher than a fixed threshold are not expanded further. As a heuristic, we fix this threshold to be the mean of the probability of the patterns in the input. The probability of true motifs is much less than the mean so such a pruning strategy will discard patterns which occur by chance. We see in the following sections that it improves the efficiency of the search without decreasing the sensitivity.

IV. RESULTS

The algorithms were implemented in C++ and tested on a wide variety of datasets for correctness and efficiency. The most time-consuming part of the program was searching for existing patterns in a level while adding a new node. We found that storing the level as a hash table improved the performance.

A. Performance Evaluation

We tested the performance of the algorithm on real-life datasets of yeast and human promoter sequences.

Yeast dataset: The gene regulation mechanism is simpler and well studied in lower organisms like yeast (when compared to mammalian genomes). Therefore it makes a good starting point for testing new algorithms.

The yeast transcription factor protein Mcm1 acts as activator or repressor for genes involved in cell-type determination, mating, cell-cycle control, etc. [22]. We ran our algorithm on a set of promoter sequences of genes which are known to be regulated by Mcm1. There were 17 such promoter sequences [16], each of length 1000 bases. We searched for motifs of length 11, allowing at most 3 mismatches. Since all the 17 genes were known to be regulated by Mcm1, only motifs present in all the sequences were reported.

The top ranking motif reported by our algorithm is shown in Fig. 5 as a sequence logo. The eleven stacks of letters correspond to the 11 bases of the motif. The overall height of the stack indicates the degree of conservation of the base while the tallest letter of a stack indicates the most frequent base at that position. It matches very well with the known binding motif for Mcm1 (CCGAATTAGGA) [1].

Fig. 5. Binding motif of Mcm1 found by our algorithm

This motif was also recognized by other programs such as Weeder [7], MaMF [16], AlignAce [11], Consensus [10], MEME [9] and BioProspector [12]. This suggests that lower organisms set the bar for accuracy of motif search algorithms.

Human dataset: The regulatory modules in higher species like human are much more complex. The discovery of binding sites is more challenging; most of the motif discovery algorithms perform poorly. Recent studies have found that even the best methods fail to find the known motifs in more than 50% cases [14], [16]. Even when the correct motif is identified, it may not be the highest ranked motif returned by the algorithm. Tompa, et al. Ref. [14] suggests the use of top N motifs returned by the algorithm for assessing the performance of the algorithm. Hon and Jain [16] use $N = 30$ and further suggest the use of a similarity threshold (8 out of 11 bases, 73% identity) for classifying a reported motif as a true match. In summary, if a motif in the top 30 motifs returned by the algorithm matches a known motif at 75% positions (bases), it is said to be a true match. We use the same criteria for testing the performance of our algorithm.

We tested our algorithm on the promoter sequences of 8 genes [16], each of length 1200 bases, which are known to be regulated by the transcription factor E2F. The E2F family of transcription factors is well studied for its role in regulation of cell proliferation [24]. The search step identified over a thousand candidate motifs. The 10th ranking motif reported by our algorithm is shown in Fig. 6. It matches the known motif (TTCGCGCCAAA) [1] at 8/11 positions. The first 9 hits may be just noise or they could be novel motifs which need to be experimentally verified.

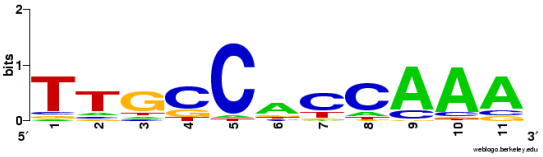

Fig. 6. Binding motif of E2F found by our algorithm

This motif was also recognized by Weeder and MaMF (both match at 9/11 positions) but it was not recognized by AlignAce, Consensus, MEME and BioProspector. This suggests that motif search is more challenging in mammalian genomes and many existing methods fail to discover the known motifs.

B. Comparison of Run-time with other Algorithms

In a recent survey [14], it was found that the Weeder

algorithm [15] outperformed other algorithms in accuracy by a clear margin in a wide variety of tests. It has been demonstrated that the performance of MaMF [16] is comparable to Weeder but the software is not available for comparison. Further, other methods do not even find the right motifs for mammalian genomes. Therefore we restricted the time comparison of our algorithm with Weeder.

The source code for Weeder was downloaded from the author's website [15] and compiled on our machine for a fair comparison. Again, for fairness, we compared the programs' performance on the following data sets of promoter sequences obtained from Weeder website:

1. 5 yeast promoter sequences of length 1000 bp each
2. 11 human promoter sequences of length 500 bp each

Our algorithm was able to extract all the motifs detected by Weeder. Fig. 7 shows the runtime comparison of the two algorithms. The figure clearly shows that Mottice takes one fifth the time as Weeder for all motif sizes and for both yeast and human datasets.

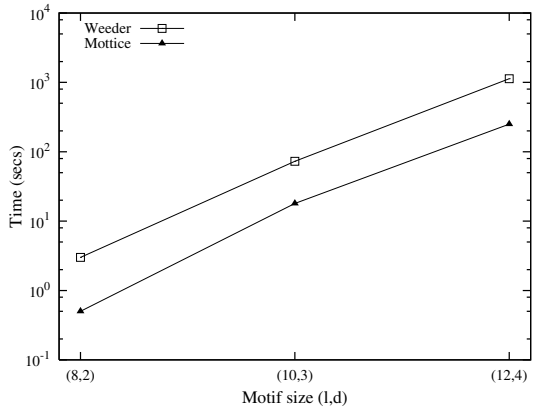

Fig. 7. Comparison of runtime of Weeder and Mottice

Weeder is based on the idea of enumerating all 4^l patterns of length l and then searching them in a generalized suffix trees built using the input sequences [17]. To speed up the computation, some approximations are made about the position of the mismatches [7] so that the results are obtained in a reasonable time. On the other hand, we use the input patterns and generate their variants, changing one base at a time, mimicking the mutations of the binding site. This way the search space grows in a controlled fashion and the results are found much more quickly.

It may be noted that the solution found by Mottice is complete; it finds all the motifs, without making any approximation. On the other hand, Weeder uses a heuristic to speed up the search and may miss parts of the search space. Still, Mottice outperforms Weeder by a wide margin. This is possible because of efficient structure of input patterns. The performance of Mottice can be further improved by applying pruning of the search space.

C. *Pruning the Search Space*

We measured the improvement in performance of the search after applying pruning (as discussed in section III.E). In particular, we tested the effect on the number of nodes added to the search graph and the number of candidates generated at the end of the search process after applying pruning at every level. To assess the effect of pruning, we ran the algorithm with different input dataset sizes. We created 12 secondary datasets (from the Mcm1 dataset) with 2 to 13 sequences in each. Since each promoter was of length 1000 bases, the total input size varied from 2000 bases to 13000 bases. We first ran the program without pruning and then with pruning. For each input size, the number of nodes in the graph and the number of candidate motifs were observed.

Fig. 8 shows the variation of the total number of nodes examined by the search process for different input sizes. Fig. 9 shows the variation of the total number of candidates generated as the final output of the search process. A logarithmic scale is used for the y-axis in Fig. 9 to demonstrate the difference when number of candidates is low.

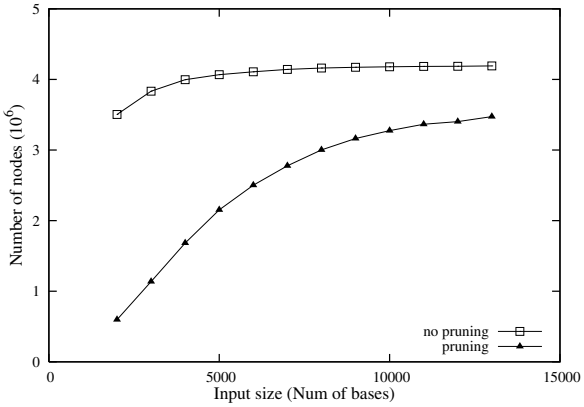

Fig. 8. Effect of pruning on total number of nodes in the graph

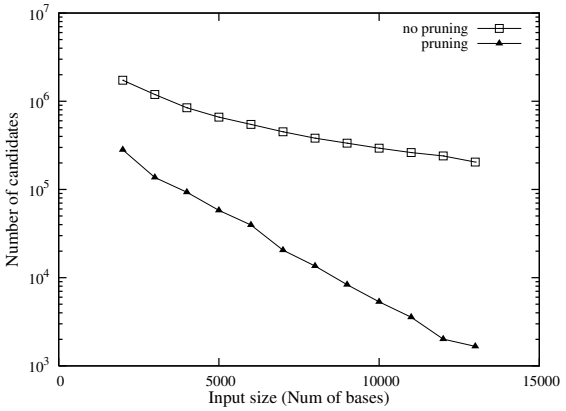

Fig. 9. Effect of pruning on total number of candidates generated

In Fig. 8, the number of nodes expanded increases at a slower rate as input size increases. The reduction in number of nodes due to pruning decreases as input size increases. This happens as the upper limit for maximum number of patterns (4^l) is reached. We observe that pruning is more effective for smaller input sizes. On the other hand, Fig. 9 shows that as the number of sequences increases, number of candidates decreases. This happens as fewer patterns are able to meet the quorum requirements (the pattern must be present

in q out of the t sequences). In this case, pruning is more effective for larger input sizes. When using 13 input sequences (13000 bases), the number of candidates generated with pruning is less than 2 orders of magnitude compared to the number of candidates without pruning. Since the total number of candidates is of interest, pruning is more effective for large input sizes.

Such a pruning improves the efficiency without compromising the sensitivity of the search. This is because only the nodes having a low probability of being a motif are pruned. The experiments in section IV.A were done with pruning and yet the correct motifs were discovered in both the cases.

D. Scalability

Since the time and space complexity is linear with respect to the input size, the algorithm should scale linearly with increasing input size. To verify this, we ran the algorithm for different input sizes from the Mcm1 dataset. We created 16 secondary datasets with 2 to 17 sequences in each. Since each promoter was of length 1000 bases, the total input size varied from 2000 bases to 17000 bases. For each input size, the time required to find the motifs was calculated. The results are shown in Fig. 10. It is evident that the algorithm does scale linearly. Recent studies [20] have conjectured that the length of the active promoter region may be as high as 90 kb. Our algorithm will be useful for mining such datasets.

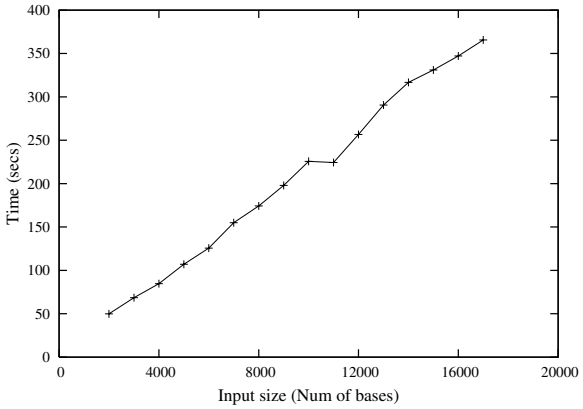

Fig. 10. Scalability of the search for motifs

V. DISCUSSION

Our algorithm is able to identify the motifs in data with noise, i.e., motifs which are present in any subset of the input sequences are also identified in the process and may be reported based on user preference. Typically, binding site identification programs are tested on promoters of a group of related genes which are expected to be under the influence of a common regulatory mechanism. However, it is difficult to judge *a priori* the fraction of the gene-promoters which share the common TFBSs or motifs. Mottice is especially useful in such cases as it can report all motifs which are present in a (user-given) fraction of sequences without any extra computational overhead. The algorithm scales linearly with the input size so it will be useful for large sequences. It can also trace the evolutionary profile of a motif by simply tracing back the path in the graph. If the promoter region corresponds to say, same gene in different species, then such profiles readily facilitate construction of phylogenetic trees.

The use of auxiliary information is known to improve the accuracy of motif prediction. One of the most popular approaches for enhancing motif prediction is searching in promoters of similar genes from other related organisms. If the reported pattern is a true motif, then it is likely to be conserved in promoters of such orthologs. Such ideas have been successfully applied to improve the prediction of motifs [25]. However, we have deliberately avoided a comparative genomics based approach for two reasons. First, there is no known correlation between sequence conservation and function results, in part, from the presence of a large amount of highly conserved non-coding sequences in the human genome (for e.g. the ultra conserved regions, some of which are longer than 1 kb). Second and most importantly, not all TFBSs are conserved among species. For instance, it has been estimated that roughly one third of TFBSs are not conserved between human and rodents. This could be either due to the degeneracy of TFBSs (the same TF may bind to sequence variants of the TFBS that are present in different species) or the redundancy of regulatory elements (a single TFBS could be gained or lost without affecting the overall expression of the gene). Use of structural class of the transcription factor (basic leucine zipper, forkhead, basic helix loop helix, etc) also aids in the de novo discovery of motifs [26]. So we propose use of auxiliary information, if available, only as a post-processing step.

VI. CONCLUSION

We have presented Mottice, a new efficient and scalable method for finding motifs in genomic sequences. We have illustrated the idea by identifying transcription factor binding sites in promoter sequences but the method may be readily applied to other problems as well. The search space is structured as a graph which leads to efficient discovery of motifs. The algorithm does not make any assumption about the motif, it is deterministic and complete; it exhaustively finds all motifs which meet the constraints provided by the user. The background genomic sequence is modeled by a third order Markov model. The candidate motifs are ranked based on a composite scoring function.

The Mottice algorithm was implemented and applied to a wide spectrum of real-life data sets. It was able to detect the motifs in promoter regions of both yeast and human. All the popular algorithms tested worked well with yeast data but most of them failed for human promoters. Run time comparisons were done with algorithms which worked well with human promoters and Mottice produced the results more than five times faster. The use of pruning further helped speed up the search by producing fewer candidates. The method scales linearly with input size (number of sequences and their length) which makes it useful for analyzing large datasets. Even though the worst-case complexity is exponential in d

(the number of allowable mismatches), the value of *d* is very small (1-4) so the performance does not suffer in case of real data. Further, the use of graph structures provides more meaningful insight into the evolution of a binding site.

Currently we are developing more efficient representation of the patterns and the graph. We are working on identifying additional heuristics to prune the tree and speed up the search process allowing quick approximations. We are also working on dividing the search space into equivalence classes for an efficient parallelization of the search process.

ACKNOWLEDGMENT

We would like to thank Anil Jegga and Haiyun Bian for many useful discussions and comments on the manuscript.

REFERENCES

[1] V. Matys, E. Fricke, R. Geffers, et al., "TRANSFAC: transcriptional regulation, from patterns to profiles," *Nucleic Acids Res.*, 31, 374-378, 2003.

[2] A. Sandelin, W. Alkema, P. Engstrom, et al., "JASPAR: an open-access database for eukaryotic transcription factor binding profiles," *Nucleic Acids Res.*, 32, D91-94, 2004.

[3] K. Cartharius, K. Frech, K. Grote, et al., "MatInspector and beyond: promoter analysis based on transcription factor binding sites," *Bioinformatics*, 21, 2933-42, 2005.

[4] Match. Gene-Regulation.com, http://www.gene-regulation.com/cgi-bin/pub/programs/match/bin/match.cgi, (2006).

[5] K.D. MacIsaac, and E. Fraenkel, "Practical strategies for discovering regulatory DNA sequence motifs," *PLoS Comput Biol*, 2(4), e36, 2006.

[6] P. Pevzner, and S.H. Sze, "Combinatorial approaches to finding subtle signals in dna sequences," in *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, 269-278, 2000.

[7] G. Pavesi, G. Mauri, and G. Pesole, "An algorithm for finding signals of unknown length in DNA sequences," *Bioinformatics*, 17 Suppl. , S207-214, 2001.

[8] M. Li, B. Ma, and L. Wang, "Finding similar regions in many strings," in *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, 473-482, 1999.

[9] T.L. Bailey, and C. Elkan, "Fitting a mixture model by expectation maximization to discover motifs in biopolymers," in *Proceedings of the 2nd International Conference on Intelligent Systems for Molecular Biology*, pp. 28-36, AAAI Press, Menlo Park, California, 1994.

[10] G.Z. Hertz, and G.D. Stormo, "Identifying DNA and protein patterns with statistically significant alignments of multiple sequences," *Bioinformatics*, 15, 563-577, 1999.

[11] J.D. Hughes, P.W. Estemp, S. Tavazoie, and G.M. Church, "Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae," *J. Mol. Biol.*, 296, 1205-1214, 2000.

[12] X. Liu, D.L. Brutlag, and J.S. Liu, "BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes," in *Proceedings of the Pac Symp Biocomput.*, 127-38, 2001.

[13] J. Van Helden, A.F. Rios, and J. Collado-Vides, "Discovering regulatory elements in non-coding sequences by analysis of spaced dyads," *Nucleic Acids Res.*, 28, 1808-1818, 2000.

[14] M. Tompa, et al., "Assessing computational tools for the discovery of transcriptional factor binding sites," *Nature Biotechnology*, 23(1), 137-144, 2005.

[15] G. Pavesi, P. Mereghetti, G. Mauri, and G. Pesole, "Weeder Web: Discovery of Transcription Factor Binding Sites in a Set of DNA Sequences from Related Genes," *Nucleic Acids Research*, 32(Web Server issue), W199-203, 2004.

[16] L.S. Hon, and A.N. Jain, "A deterministic motif finding algorithm with application to the human genome," *Bioinformatics*, 22(9), 1047-1054, 2006.

[17] L. Marsan, and M. Sagot, "Algorithms for extracting structured motifs using a suffix tree with application to promoter and regulatory site consensus identification," *J. Comp. Biol.*, 7, 345-360, 2000.

[18] X. Yang, and J. Rajapakse, "Graphical approach for motif recognition in DNA sequences," in *Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 147-152, 2004.

[19] S. Sinha, and M. Tompa, "A Statistical Method for Finding Transcription Factor Binding Sites in Yeast," in *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, 344-54, 2000.

[20] Q. Peng, P.A. Pevzner, and G. Tesler, "The fragile breakage versus random breakage models of chromosome evolution," *PLoS Comp. Biol.*, 2(2): e14, 100-111, 2006.

[21] A.F.A Smit, R. Hubley, and P. Green, "RepeatMasker" at http://repeatmasker.org, 2006.

[22] D.S. Abraham, and A.K. Vershon, "N-terminal arm of Mcm1 is required for transcription of a subset of genes involved in maintenance of the cell wall," *Eukaryot Cell.*, 4(11), 1808-19, 2005.

[23] G.E. Crooks, G. Hon, J.M. Chandonia, and S.E. Brenner, "WebLogo: A sequence logo generator," *Genome Research*, 14:1188-1190, 2004.

[24] K. Helin, "Regulation of cell proliferation by the E2F transcription factors," *Curr Opin Genet Dev.*, 8(1):28-35, 1998.

[25] A.G. Jegga, S.P. Sherwood, J.W. Carman, et al., "Detection and visualization of compositionally similar cis-regulatory element clusters in orthologous and coordinately controlled genes," *Genome Research*, 12(9), 1408-17, 2002.

[26] L. Narlikar, R. Gordan, U. Ohler, and A.J. Hartemink, "Informative priors based on transcription factor structural class improve de novo motif discovery," *Bioinformatics*, 22(14), e384-92, 2006.