An Intelligent Pattern Searching Model with Suffix Structures

A.U. Makolo
Department of Computer Science
University of Ibadan, Ibadan, Nigeria
a.makolo@ui.edu.ng

ABSTRACT

Discovering patterns in genomic sequences possess a lot of challenges to scientists. Pattern discovery is basically a heuristic problem and efficient algorithms are sought for its implementation. In this paper, we present a model for the identification and extraction of biologically significant patterns from a set of sequences using suffix tree data structures with suffix links. Our algorithm achieved an analytical time complexity of $O(n \log n)$ as opposed to a quadratic running time obtained by similar algorithms. We implemented an Intelligent Sequence Analysis Tool (ISAT) incorporating the algorithm which is capable of analyzing sequences offline and does a self-update of its database when online. This additional feature is desirable in developing countries where constant access to the internet is a challenge.

Keywords: Pattern Discovery, Motifs, Suffix Tree, ISAT.

African Journal of Computing & ICT Reference Format:

1. INTRODUCTION

Pattern discovery algorithms are necessary for automated genome annotation needed in making sense and interpreting the large volume of genomic sequence data presently being generated. In its simplest form, the problem of pattern discovery in DNA sequences can be formulated as follows: Given a set of sequences, find an unknown pattern that occurs frequently. If a pattern of $m$ letters long appears exactly in every sequence, a simple enumeration of all $m$-letter patterns that appear in the sequences gives the solution. The challenge is developing efficient algorithms in terms of time and space requirement.

Search trees have been found to be useful for genomic data processing. The suffix data structure is particularly useful because of its dynamic and iterative search methodology. Also the adaptability of suffix-tree to solving many sequence processing problems that are otherwise computationally extremely hard to solve makes it a good candidate for genomic data processing. [2] In addition, the linear time and space complexity of suffix-trees make them attractive for use with large scale sequence processing tasks. It also exposes the internal structure of a biological sequence in a deeper and more meaningful way than any other data structure by showing the relationship between the different patterns in a way suitable to all problems involving pattern matching. [9]. The implementation of the suffix tree incorporating a suffix link results in an appreciable acceleration in the construction and traversal of a suffix tree because the links act as pointers the leaves and sub-trees of the tree. [6].

This work develops and implements an algorithm for searching for repeated patterns in a set of DNA sequences. The algorithm implemented in java on windows operating system provides a user friendly interface for users to submit their sequences and the repeated patterns are returned. It also computes the position specific scoring matrix in which the absolute frequency of occurrence of each nucleotide in the motif is derived. The absolute frequency matrix is then translated into a relative frequency matrix. This relative frequency is the probability of occurrence of each motif character at each position in the returned motif. Usually, high scoring motifs are biologically significant and short sequences varying between length 6 and 15 found in gene promoter regions can be important motifs for gene expression regulation.

2. RELATED WORKS

Overview of Suffix Tree
A Suffix tree is a lexicographically interconnected data structure. It provides efficient access to all substrings of a string over which it is built. A Suffix tree for a string $S$ is a tree whose edges are labeled with strings, such that each suffix of $S$ corresponds to exactly one path from the tree's root to a leaf. [2]. It is particularly useful for finding a small sequence of symbols in a large one, a common ancestor of two different strings and common substrings of two different strings.
The incorporation of suffix links to suffix tree implementation is an important feature crucial for linear time construction of suffix trees. Suffix links are edges or pointers that span across the suffix tree, between two internal nodes which may not be related through a parent-sibling relationship.

The suffix link is an auxiliary structure that accounts for the most important acceleration element in the construction and traversal of a suffix tree. The suffix tree construction algorithm is based on the observation that constructing the suffix tree can be performed by iteratively expanding the leaves of a partially constructed suffix tree. Through the use of suffix links, which provide a mechanism for quickly traversing across sub-trees, the suffix tree can be expanded by simply adding the j+1 character to the leaves of the suffix tree built on the previous j characters. The algorithm thus relies on suffix links to traverse through all of the sub-trees in the main tree, expanding the outer edges for each input character.

Scoring Matrices

In bioinformatics, scoring matrices for computing alignment scores are often based on observed substitution rates, derived from the substitution frequencies seen in multiple alignments of sequences. Every possible identity and substitution is assigned a score based on the observed frequencies of such occurrences in alignments of related proteins. The score is calculated from the frequency of occurrence of a match of the two individual amino acids in evolutionarily related sequences, and provides a measure of a chance alignment of the two amino acids. This score also reflects the frequency that a particular amino acid occurs in nature, as some amino acids are more abundant than others. [5].

Higher scores indicate that the probability that those two amino acids aligned by chance is very small, and lower scores indicate a high probability the two amino acids aligned by chance, and are evolutionarily unrelated. Thus, identities are assigned the most positive scores, frequently observed substitutions also receive positive scores, but matches that are unlikely to have been a result of evolution, and are more likely indicative of unrelatedness at that position, are given negative scores.

After the analysis of sequences, we construct a profile matrix for the repeated patterns identified. A Profile or Position Weight Matrix is a motif descriptor. It attempts to capture the intrinsic variability characteristic of sequence patterns. A Profile is usually derived from a set of aligned sequences functionally related. For instance, if we have these sequences:

Sequence 1:ACCGTACC
Sequence 2:GACCGTAA
Sequence 3:AAGGTTGG
Sequence 4:GCAGCAGT
Sequence 5:TCGGTCAT
Sequence 6:CAGGTACT

A profile can be derived from above set of sequences by tabulating the frequency with which each nucleotide is observed at each position.
A profile matrix for the above sequence can be constructed as below:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Each coefficient in this matrix indicates the number of times that a given nucleotide has been observed at a given position. For instance, the nucleotide "A" has been observed in two of the aligned sequences in position 1, thus this is indicated in the matrix.

More often than the absolute frequencies, the relative frequencies are tabulated in a profile. In such cases, the coefficients of the matrix can be interpreted as probabilities of a given nucleotide occurring at a given position in a functional site. Then, given a sequence of length \( l \), we need to calculate the product of the coefficients from such a matrix corresponding to each nucleotide in each position of the sequence. This can be done by computing the probability of finding such a sequence in a true functional site. For instance, to compute the probability of finding 'C' in the first column, 'A' in the second column, 'G' in the third column until we get to the last motif character 'T'.

### Suffix Tree In Motif Discovery Tools

The choice of algorithmic framework to adopt in development of pattern discovery algorithms is a constant challenge among computational biology researchers. Studies have shown that the suffix tree data structures are a defacto choice for string searching. The first simple motif discovery algorithm to use the suffix tree was developed by [10]. The suffix tree was used to represent the sequences, returning all the traversal from the root node to the leaf node as unique patterns. The use of the suffix tree for preprocessing and organizing the input data resulted in an accelerated search for motifs. This was followed by [2] who developed the VERBUMCULUS algorithm and applied it to protein sequences. [4] used a variant of the suffix tree called a mismatch tree to develop the MITRA algorithm which detected complex motifs with mutations successfully. WEEDER algorithm by [7] also used the suffix tree and identified simple motifs allowing the flexibility of parameter specification by users.

In the work of [8] an improvement on the use of Suffix tree was achieved by incorporated boxlinks data structure with the suffix tree. EXMOTIF by [11] used a variant of the suffix tree was used, consisting of inverted index of symbol positions. This was used to enumerate all structured motifs by positional joins over the index. It used a hash table to store the computed motifs thus facilitating a speedy lookup and extracted all the repeated patterns after statistically validating them. EXMOTIF was reported to outperform RISO in both approximate and exact matching and superior to RISSOTO in showing the actual occurrences of the structured motifs instead of the relative frequency of the occurrence as obtained using RISOTTO.

A common choice among researchers of motif discovery tools is a combination of pattern-driven and statistical-based methods since this approach guarantees that the sensitivity of the statistical based method be complemented with the speed efficiency of pattern-driven techniques. An example of this is the STEME (Suffix Tree and Expectation Maximization for Motif Elicitation ) algorithm by [9]. It combined the suffix tree, a pattern-driven approach with Expectation Maximization, a statistical approach. The incorporation of the Suffix Tree improved the speed limitation of the expectation maximization based algorithms (such as MEME). This also influenced the development of STGEM by [6].

### 3. THE ISAT ARCHITECTURE

The general structure of a motif discovery algorithm includes a system that accept a DNA sequence of any length, an algorithm that implements the suffix tree search by constructing a tree with the sequences, searching for frequently occurring patterns, destroying the constructed tree to output the patterns. These patterns which have a high biological significance depending on the value of the position specific scoring matrix computed are called motifs.

The user specifies the various lengths of sequence repetition interested in. Usually the regulatory motifs which are of high interest occur in lengths between 6 and 14. They are submitted to available online motif database such as TRANSFAC or JASPAR where they will be available for online sequence comparison. The unique feature of ISAT is that it can be used to analyze a sequence offline, and it updates itself whenever it establishes an internet connection so that the user can have access to the latest version of the online databases.

From the biological point of view, sequence comparison is motivated by the fact that all living organisms are related by evolution. That implies that the genes of species that are closer to each other should exhibit similarities at the DNA level.
The model also provides other information about the sequence submitted such as the total number of sequence available, the position of the repeated patterns, the number of times it occurred and the computation of the scoring matrix. A standard motif discovery algorithm by [7] used suffix tree to find sequence of unknown length and developed WEEDER which runs at a quadratic time. ISAT, however, uses suffix tree with suffix links and it offers a better searching scheme since suffix links act as pointers and accelerates the speed of searching for patterns occurring with a high frequency.

4. RESULTS

The dataset used to test the algorithm was downloaded from PlasmoDB (An online gene data bank of Plasmodium Falciparum maintained by National Center for Biotechnology Information (www.ncbi.org)). The running time for ISAT and WEEDER was computed by adding a time stamp to the algorithms. Fig 3 shows the graph of the running time. ISAT in using suffix tree with suffix links offers a better searching scheme since suffix links act as pointers and accelerates the speed of searching for patterns occurring with a high frequency.

ISAT outperforms WEEDER in saving considerable computational time.

The user simply specifies the file name containing the DNA string and the file that contains the length of motifs to be searched for. The scoring matrix for the patterns discovered are computed and the result of the output is shown below:
The first part of figure 4 is the absolute frequency while the second part is the relative frequency. The system can be used for analysing biological sequence of any organism. It also provides other useful information such as number of sequences present, the length of the sequence in addition to outputing repeated patterns.

5. DISCUSSION OF RESULTS

The suffix tree search algorithm was improved by introducing suffix links which accelerated the search scheme since links act as pointers in searching for subsequences or patterns in the constructed tree. This improvement contributed to an improvement in computational time and made ISAT outperform WEEDER that used only suffix tree without incorporating suffix links. The gradation in running time for the different gene sizes is directly proportional to the size of the sequence, but for very long sequences it remained at nlogn which is quite remarkable.

Recurrence equation for the suffix tree is $T(n) = T(2n/3) + Cn$

We prove the running time as follows.

$T(n) \leq dn\log n - d(2n/3)\log(2n/3) + Cn$

The coefficient will not matter any more given an essentially large value. Therefore, $2/3$ can fizzle out leaving us with $n$. So, we can have

$$dn\log n - d(\log 3 - \log 2) + Cn \leq 0 \quad \text{“note that } -dn(\log 3 - \log 2 + Cn \text{ is from the equation 1”}$$

$$-dn(\log 3 - \log 2) \leq - Cn$$

$$d \geq \frac{C}{-n(\log 3 - \log 2)} \quad \{ -n \text{ cancels out } \}$$

$$d \geq \frac{C}{(\log 3 - \log 2)} \quad \{ \text{this is to be constant, still insignificant} \}$$

Hence, we have $n\log n \quad \{ \text{where } C \text{ and } d \text{ are constants but } d > C \}$

Therefore, $T(n) = O(n\log n)$
6. CONCLUSION AND FUTURE PERSPECTIVES
This study briefly describes the development and implementation of a pattern searching algorithm for DNA sequence analysis. This is an important area of research in bioinformatics due to the various insights that analyzing sequences can provide. We have shown how incorporating suffix links into our suffix tree search scheme improved its efficiency and we have also demonstrated the efficiency of our model by comparing its performance with that of a standard sequence analysis algorithm. A possible future work would be to compare ISAT with more search algorithms that used Suffix tree data structures and also use other parameters as evaluation criteria other than speed of computation.

REFERENCES

Author’s Biography
Angela Makolo is a lecturer in Computer Science Department of the University of Ibadan and the Principal Investigator of the University of Ibadan Bioinformatics Research Group (ui.bioinformatics.edu.ng). She has a PhD in Computer Science with Bioinformatics Option from the University of Ibadan. Her research work focuses on Computational Biology and Machine Learning.