Prediction of *methanobacterium* using suffixtree

***Satyasaivani B.*[1], Kaladhar DSVGK[2], Shashi M.[3] and Kesavareddy J.[4]**

1. Department of Computer Science, GITAM University, Visakhapatnam, Andhra Pradesh, Email: vedulvani@yahoo.co.in
2. Department of Bioinformatics, GITAM University, Visakhapatnam, Andhra Pradesh
3.Head Of The Department, Department Of Computer and Systems Engineering, Andhra University, Visakhapatnam, Andhra Pradesh
4. Department of Computer science, GITAM University, Visakhapatnam, Andhra Pradesh

Abstract- Prediction of *Methanobacterium* using suffix tree is a server designed to know the query sequence related to *Methanococus thermophilus* and also to execute the maximum length of the string related to *Methanococcus*. MEGA 4.0 is used to know the conserved sites aligned from 70 sequences related to 16S rRNA nucleotide sequence (*Methanococcus thermophilus*) from NCBI database. There are 12 strings aligned in all sequences that are highly conserved in the aligned sequences. Ukkonen's algorithm is used to find the suffix tree for the given patterns (conserved sites). If the query sequence is submitted to the PMST (Prediction of *Methanobacterium* using Suffix Tree), the results will give the maximum sequence length and the suffix tree based on Ukkonen's algorithm.

Keywords: Ukkonen's algorithm, *Methanobacterium*, MEGA, 16S rRNA

INTRODUCTION

The comparison of two or more sequences of numbers or letters is common in several fields such as molecular biology, speech recognition, and computer science. Sequence comparison is particular important in molecular biology where it has been critical in study of evolution, the control of gene expression, and in the analysis of protein structure/function relationship [1]. The Ukkonen algorithm has the desirable property of processing the string symbol by symbol from left to right. It always has the suffix tree for the scanned part of the string ready [2]. A repeating pattern is an important feature of music objects, which commonly used in analyzing the repeated part of music data and looking for themes. Most of the repeating patterns are key melodies or easy to familiarize and remember for people. Yu-Lung Lo *et al.*, 2007 used the themes or the repeating patterns to construct indices that can speedup query execution for music retrievals [3]. Common queries over such datasets include both exact and approximate string matches. These queries can be evaluated very efficiently by using a suffix tree index on the string dataset. Although suffix trees can be constructed quickly in memory for small input datasets, constructing persistent trees for large datasets has been challenging. Sandeep tata (2004) explored suffix tree construction algorithms over a wide spectrum of data sources and sizes [4]. Ukkonen's algorithm for on-line approximate string matching is based on the simulation of a nondeterministic finite automaton built from the pattern and using the text as input. This simulation uses bit operations on a RAM machine with word length $w = \Omega(\log n)$ bits, where n is the text size [5]. A 42% speedup was achieved in an application involving name comparisons. Even greater improvements are possible when comparing longer and more dissimilar strings. Although the speed of the algorithm under consideration is comparable to other fast ASM algorithms, it has greater effectiveness in text-processing applications because it supports all four basic Damerau-type editing operations [6]. The problem of searching a pattern in a text allowing errors in the pattern or in the text. The algorithm is based on a very fast kernel, which is able to search short patterns using a nondeterministic finite automation, which is simulated using bit-parallelism. A number of techniques to extend this kernel for longer patterns are presented in that work [7]. Ricardo A. Baeza-Yates and Chris H. Perleberg in 1999 presented an algorithm for string matching with mismatches based in arithmetical operations that runs in linear worst-case time for most practical cases. This is a new approach to string searching. They also presented an algorithm for string matching with errors based on partitioning the pattern that requires linear expected time for typical inputs [8]

Distance metric can be applied to pattern recognition, syntactic tree comparison, classification tree comparison and other applications. The problem of approximate ordered tree matching is also considered [9]. Ukkonen Algorithms, whose time complexities are close to the known upper bounds without transposition invariance, and for some we achieve these upper bounds. In particular, Mäkinen showed how sparse dynamic programming could be used to solve transposition invariant problems, and its connection with multidimensional range-minimum search. As a byproduct, Mäkinen gave improved sparse dynamic programming algorithms to compute LCS and Levenshtein distance [10]. Methanococus thermophilus belongs to the family Methano-bacteriaceae. A phylogenic tree based on rRNA or protein sequences shows that all organisms are related, which suggests a common ancestor [11, 12]. *Methanococcus, Methanothermus* and *Methanopyrus*, can be classified as hyperthermophiles [13].

METHODOLOGY

Methanobacterium sequences are collected from NCBI website. Care has taken while collecting the huge amount of data in FASTA format. This information is submitted to MEGA (Molecular Evolutionary Genetic Analysis) v4.0, to align the sequences and get conserved sites. Conserved sites indicate the relativity of a particular organism. Hence by collecting more relative sites into consideration, the user can get required strings for further process.

The process of aligning sequences by using the software MEGA is:

Step 1: all the methanobacterium sequences are taken in a notepad. Care should be taken to collect required information in the form of FASTA.

Step 2: This information is saved with extension < filename>. fasta

Step 3: By considering all the sequences the resulting data i.e. (.fasta) file is given input to MEGA.

Step 4: Then choose option Clustalw alignment.

Step 5: After aligning the sequences, the user can observes formation of conserved sites i.e *(Astick symbol) on the top of individual columns.

Step 6: we look for occurrences of gaps or unaligned rows or columns then we proceed next step by deleting all those irrelative sequences.

Step 7: This process continues until we get more conserved sequences.

After getting conserved sequences, note the most relative sequences i.e. sequences that are having more conserved sites together without gap into consideration and these conserved sites are searched in input given by the user. Then the system searches the occurrences of conserved sites in the given input and tells whether the given input is relative to methanobacterium or not. Finally the system finds maximum length frequent repeating pattern and generates tree based on ukkonen's suffix tree algorithm. Ukkonen's suffixtree algorithm gives level-by-level implementation of suffix tree by using the maximum length repeating patterns. The resulted tree tells whether the tree consisting of required field or not by using Ukkonen's algorithm. User can find directories and also trivial and nontrivial repeating patterns of obtained maximum length pattern and frequencies of their occurrences. The algorithm begins with an implicit suffix tree containing the first character of the string. Then it steps through the string adding successive characters until the tree is complete. This order addition of characters gives Ukkonen's algorithm its "on-line" property; earlier algorithms proceeded backward from the last character. The naive implementation of this algorithm requires $O(n^2)$ or even $O(n^3)$ time, where n is the length of the string, but by exploiting a number of algorithmic techniques this can be reduced to $O(n)$ (linear) time.

RESULTS AND DISCUSSION

Finding strings using Molecular evolutionary genetic analysis (MEGA) can found the conserved sites for particular organism. The input of this is taken in the form of FASTA format. The sequences are in combination of A, T, G, C and was presented in Table1. In the table, we considered only smaller sequences for readers convenient. The input may consist of combinations having one or more lines of a sequence. The outputs of conserved sequences are represented in Table 2. By using the above information, we can find the relative ness of submitted organism with specified one and the results are displayed on screen in the form of messages as your sequence is relative to methanobacterium, for relative input, otherwise your sequence is not relative to methanobacterium. Also the maximum length-repeating pattern is found and displayed on screen. After finding repeating pattern the next step goes for the finding trivial repeating patterns that are presented in the maximum length repeating pattern and their corresponding frequencies are displayed on the screen and is represented in Table 3. These trivial patterns are used for further research. We can also find the places of their occurrence in corresponding pattern. This in formation becomes useful to the researchers of different platforms. The result of suffix tree formation is represented in console by step-by-step procedure. This can be represented in Figure 1. The output of the program tells, how the tree is constructed and how the nodes are included in tree like Childs and also their frequencies below their positions. These frequencies are taken into consideration only when they are repeating more than two times. The above results are used for further analysis such as finding relativity between two or more organisms and also to compare one DNA image with other to find out the differences among those two images.

ACKNOWLEDGMENT

The author would like to express thanks to Departments of computer science and Bioinformatics, GITAM Institute of science, GITAM University, Visakhapatnam for providing research material and necessary facilities.

REFERENCES

[1] Carrillo H. and Lipman, D. (1988) The multiple sequence alignment problem in biology, SIAM J. Appl. Math., 48(5), 1073-1082.

[2] Ukkonen E. (2005) On-line construction of suffix trees, Algorithmica, 14(3), 249-260.

[3] Yu-Lung Lo, Wen-Ling Lee and Lin-huang Chang (2008) True suffix tree approach for discovering non-trivial repeating patterns in a music object, Springer Science, 37(2), 169-187.
[4] Sandeep Tata, Richard A Hankins and Jignesh M.Patel (2004) Practical suffix tree construction, VLDB Endowment, 30, 36-47.

[5] Baeza-Yates R. and Navarro G. (1999) Faster Approximate String Matching, Springer, New York, 23(2), 127-158.

[6] Hal Berghel and David Roach (1996) An extension of Ukkonen's enhanced dynamic programming ASM algorithm, ACM, New York, 14(1), 94 – 106.

[7] Navarro G. and Baeza-Yates R. (2001) Improving an Algorithm for Approximate Pattern Matching, Algorithmica, 30(4), 473-502.

[8] Ricardo A. Baeza-Yates and Chris H. Perleberg (1999) Fast and practical approximate string matching, Inf. Process. Lett., 59(1), 21-27.

[9] Kaizhong Zhang (1995) Algorithms for the constrained editing distance between ordered labeled trees and related problems, ScienceDirect, 28(3), 463-474.

[10] Veli Mäkinen, Gonzalo Navarro and Esko Ukkonen (2005) Transposition invariant string matching, J. Algorithms, 56(2), 124-153.

[11] Woese CR and Fox GE (1977) Phylogenetic structure of the prokaryotic domain : the primary kingdoms, Proc Natl Acad Sci USA, 74, 5088-5090.
[12] Woese CR. Kandler O and Wheelis ML (1990) Towards a nature system of organisms: proposal for the domains, Archaea, Bacteria, and Eucarya. Proc. Natl Acad Sci USA, 87, 4576-4579.
[13] Boonsri Jongsareejit (2004) Hyperthermophilic Archaea, Silpakorn University International Journal, 4(1-2), 166-180.

Table 1: Retrieved sequences

Number	Sequence
FJ888651	TGCGAGCAATCGTGATAAGGGAACTCCGAGTGCCCGTAAAATCGGGCTGTCCGCCAGTGTAAATAACTGGTGAAGAAAGGGCCGGGCAAGACCGGTGCCAGCCGCCGCGGTAATACCGGCGGCTCGAGTGGTGGCCACTATTACTGGGCTTAAAGCGTTCGTAGCTGGTCTGTTAAGTCTCTGGGGAAATCTTCCGGCTTAACCGGAAGGCGTCTCAGGGATACTGGCAG
FJ560741	CCTACGGGGCGCAGCAGGCGCGAAAACTTTACAATGCGAGAAATCGTGATAAGGGAACCCCGAGTGCCCGTAAATTCGGGCTGTCCATCAGCGTAAAAAACTGGTGAAGAAAGGGCCGGGCCAAGACCGGTGCCAGCCGCCGCGGTAA
FJ560740	CCTACGGGGCGCAGCAGGCGCGAAAACTTTACAATGCGGGCAACCGTGATAAGGGAACCTCGAGTGCCTGTAAATGCAGGCTGTTCAGGTGCCTAAAACACACCTGAAGAAAGGGCCGGGCAAGACCGGTGCCAGCCGCCGCGGTAA
FJ560739	CCTACGGGGTGCAGCAGGCGCGCAAACCTCCGCAATGCAGGAAAGTGCGACGGGGGAAACCCAAGTGCCACTCTTAACGGGGTGGCTTTTCTTAAGTTTAAAAAAGCTTTTGGAATAAGAGTGGGCAAGACCGGTGCCAGCCGCCGCGGTAA
AM747295	GCTCAGTAACACGTGGATAACCTAACCTTAGGACCGGGATAACCCTGGGAAACTGGGGATAATACCGGATATGTAGGGCTGCCTGGAATTGGTTCCCTATTGAAATGTTCCGANGCCTAAGGATGGATCTGCGGCAGATTAGGTAGTTGGCGGGGTAACTGCCCACCAAGCCTGTAATCTGTACGGGNTGTGAGAGCAAGAGCCCGGAGATGGAACCTGAGACATGAATCCAGGCCCTACGGGG

Table 2: Conserved sequences from MEGA

String number	STRING
1	GGGAAACTGG
2	TGGATCTGCGGC
3	GCCCACCAAGCC
4	ACGGGTTGTGAGAGCAAGACCCGG
5	TGAAGACAAGGTTCCAGGCCCTACGGGGCGC
6	CAGGCGCGAAACCTCCGC
7	GCTGGGCAAGACCGGTGCCAGCCGCCGCGGTAACACCG
8	CAAGTGGTGGCC
9	TTATTGGGCCTAAAGCGTT
10	GGGTAGGGGTGAAATC
11	CGAACCGGATTAGAT
12	CCGGGTAGTCCTGGCCGTAAACGATG

Table 3: Bi- and Tri- veal repeating patterns

Patterns	Frequencies
at	2
ag	3
agg	2
tgg	3
aga	4
gga	2
tgg	4

Figure 1: Ukkonen suffix tree

```
The tree at iteration 3:3 is:
1.txt; leaf = false; suf = null; position = -1; childrenMap =  {1=leaf = true; suf = 1
null, -1
  |
  +---- 1.txt, 0
  |
  +---- .txt, 0
  |
  +---- txt, 0
  |
  +---- xt, 0

       i = 3 ; j = 4   beta =
null, -1
  |
  +---- 1.txt, 0
  |
  +---- .txt, 0
  |
  +---- txt, 0
  |
  +---- xt, 0
```