MULTITHREADED KMP: A PROPOSED DNA SEQUENCING ALGORITHM

[1]SAMARJEET BORAH, [2]DEBASHREE BHATTACHARJEE, [3]KRISHNA VIJAY KR. SINGH, [4]BIKASH RAI

[2,4]Department of Computer Science & Engineering
[1,3]Department of Computer Applications, East Sikkim Sikkim Manipal University,
Sikkim Manipal Institute of Technology
E-mail: [1]samarjeetborah@gmail.com

Abstract- DNA sequencing is the process of decoding the nucleotide bases in a DNA molecule. The advent of DNA sequencing has significantly accelerated biological research and discovery. There are a number of string searching algorithms as well which are widely used for DNA sequencing namely Knuth-Morris- Pratt (KMP), Boyer Moore, Rabin-Karp, Longest Common Subsequence Problem etc. In this paper a multithreaded KMP algorithm is proposed for DNA sequencing. This work primarily aims at reducing the pattern searching time of the original KMP algorithm by using the concepts of divide and conquer and parallel processing techniques and also enables it to make a search on all the available matches of the source DNA thereby increasing its flexibility.

Keywords- KMP Algorithm, DNA Sequencing, Alignment, String Matching, Time.

I. INTRODUCTION

This project implements KMP algorithm by using the concepts of multithreading and divide and conquer. Here the given source string is divided into two parts based on the size pattern string which is to be looked for in the source string. Then the two parts are given to the thread for the computation of the pattern matching the source. Then the relative percentage of each match and its positions are computed.
In this research work a study on the KMP algorithm is done which is a fast string matching algorithm that is primarily used for matching huge disease DNA sequence or any huge pattern of DNA sequence with a given source DNA sequence. It is a forward linear sequence matching algorithm, which computes the new position from where to begin the search based on the information and bypass reexamination of previously matched characters; this is the reason why it is also called as an intelligent search algorithm.

A. Deoxyribo Nucleic Acid (DNA)

DNA is the hereditary material in humans and almost all other organisms. Nearly every cell in a person's body has the same DNA. Most DNA is located in the cell nucleus, but a small amount of DNA can also be found in the mitochondria. The information in DNA is stored as a code made up of four chemical bases: adenine (A), guanine (G), cytosine (C), and thymine (T). DNA bases pair up with each other, A with T and C with G, to form units called base pairs. Each base is also attached to a sugar molecule and a phosphate molecule. Together, a base, sugar, and phosphate are called a nucleotide. Nucleotides are arranged in two long strands that form a spiral called a double helix. An important property of DNA is that it can replicate, or make copies of itself. Each strand of DNA in the double helix can serve as a pattern for duplicating the sequence of bases. This is critical when cells divide because each new cell needs to have an exact copy of the DNA present in the old cell.

B. DNA Sequencing

The fundamental motive of sequence analysis is to align two sequences in a pair-wise manner. We compare the sequences using various techniques to reveal whether the two sequences are related or not. Sequence alignment is a way of arranging the primary sequence of DNA or RNA. This alignment provides a powerful tool to compare related sequences from where mismatches can be interpreted as mutation. Pair-wise sequence alignment methods are used to find the best-matching of two query sequences. Generally, the DNA sequence for a specific disease is known. To affirm if that particular disease is present in a human or any other organism, the diseased DNA sequence is searched in the organism's DNA sequence and if there exists a presence of the diseased DNA sequence or a trace of that in the organism's DNA sequence then that determines the presence of the disease and the count of the repetitions determines the intensity of the particular disease in the organism.
DNA sequencing may be used to determine the sequence of individual genes, larger genetic regions (i.e. clusters of genes or operand), full chromosomes or entire genomes. The resulting sequences may be used by researchers in molecular biology or genetics for further scientific progress or may be used by medical personnel to make treatment decisions. Many practical applications of DNA sequencing such as clinical applications have already begun.

We here try to compute the relative positions and the percentage of the presence of such disease DNA sequences in an organism and reduce the time complexity of the original Knuth-Morris & Pratt Algorithm.

C. KMP Algorithm

The KMP algorithm was first conceived by Donald Knuth & Vaughan Pratt and independently by J.H Morris in 1977 [6]. But the three published it jointly to form the existing Knutt-Morris-Pratt algorithm. This algorithm is the first linear time string matching algorithm by analysis of the Naive algorithm and plays an important role in bioinformatics, in DNA sequence matching, disease detection, finding intensity of disease etc. This algorithm does a left to right scanning of strings because of which it is also called forward linear search algorithm. This algorithm makes the matching computation in two parts- firstly it computes KMP table, secondly it does linear sequential search. While making comparisons for a sub DNA sequence in a DNA sequence, if the current character doesn't matches with that of the sequence, it doesn't discards all the information gained so far, it calculates the new position from where to begin the search based on the information and bypass re-examination of previously matched characters, this is the very reason why it is also called intelligent search algorithm. The pre-processing phase has time complexity O(m) and searching phase has complexity O (n+m) where n is the length of the DNA sequence and m is the length of the DNA sub sequence(key).

D. KMP Table

The KMP table is also called the Partial Match Table or Failure Function. The goal of the table is to allow the algorithm not to match any character of source string more than once. The key observation about the nature of a linear search that allows this to happen is that in having checked some segment of the main string against an *initial segment* of the pattern, we know exactly at which places a new potential match which could continue to the current position could begin prior to the current position. In other words, we "pre-search" the pattern itself and compile a list of all possible fallback positions that bypass a maximum of hopeless characters while not sacrificing any potential matches in doing so.

There are several extensions in literature proposed by various researchers to enhance the KMP algorithm. The method described in [1] uses the concept of parallel processing to optimize the given algorithm. The given source string is divided into two parts and parallel searching of the disease sequence is done in both the divided strings. Robert W. P. Luk [5] modified the KMP algorithm for string searching of Chinese text. Here the proper decoding of the input as sequences of characters instead of bytes is necessary. The finite-automaton implementation can achieve worst-case time complexity of O (2n) but constructing the transition table depends on the size of the alphabet, Z, which is large for Chinese (for Big-5, Z > 13,000). A mapping technique reduces the size the alphabet to at most |P| where P is the pattern string. In [3] Ajay Daptardar and Dana Shaphira, performs compressed pattern matching in Huffman encoded texts.

Table I. Summary Of Various KMP Extensions

	ALGORITHMS				
PROCESS		KNUTTH-MORRIS-PRATT	Multithreaded Hybrid String Matching Algorithm	Chinese String Searching Using the KMP Algorithm	KMP+Huffman Encoded Text
	Input Parameters	A human DNA sequence and a diseased DNA sequence	A source string and a pattern string	A source Chinese string and a pattern Chinese string	A source string and pattern string compressed or uncompressed.
	Data Structures	KMP Table	KMP Table	KMP Table	KMP Table
	Result	Shows the occurrence of the diseased DNA sequence along with its position	Shows the occurrence of the pattern in the string	Shows the occurrence of the pattern in the string.	Shows the compressed patterns.

II. MULTITHREADED KMP: PROPOSED METHODOLOGY

We perform the computation by dividing the organism's original DNA sequence into two parts as per the length of the disease DNA sequence. Each of these sequences is given to two separate threads for searching the disease DNA sequence in the original DNA sequence. If a match is found then a count on the matches is made and then the relative percentage of the match along with the relative position is computed.

The modified KMP algorithm consists of two parts namely as given below:

- A table that is used for preprocessing the pattern string. It has two parameters namely, the pattern, which is the pattern string, and m, length of the pattern string. It returns an integer array (KMP table).
- KMP string search algorithm that finds the first occurrence of pattern string in text. It requires several parameters like- text: the text string, n: length of the text string, pattern: the pattern string, m: length of the pattern string, pos : position to start searching from in the text string, table : array of integers returned after calling Create Table() etc. It returns either result kmp structure, which will hold the position and the

percentage of partial matches or NULL if not found.

A. Algorithm for KMP Table Generation

Algorithm Create Table()
Input: pattern - the pattern string
m - length of the pattern string
Output: KMP table

1. START
2. table := allocate(sizeof(m));
3. table[0] := 0;
4. table[1] := 0;
5. i := 2, j := 0;
6. while (i<m),do
   6.1. if pattern[j] = pattern[i-1] then
      6.1.1. table[i++] := ++j;
      else if j>0
      6.1.2. j := table[j-1];
      else
      6.1.3. table[i++] := 0;
      End of if structure
   End of while loop
7. Return table;
8. STOP

B. Algorithm for KMP Search Function

Inputs: text - the text string
n - length of the text string
Pattern - the pattern string
m - length of the pattern string
pos - position to start
Output: table - Array of integers returned on calling Create Table

Algorithm KMP_Search()
1. START
2. j:= 0, i:=pos;
3. result_kmp *result:= allocate(sizeof(result_kmp))
4. match:=0;
5. while (i< n) do
   5.1. if pattern[j]=text[i]
      5.1.1. match++;
      5.1.2. if j=m-1
      5.1.3. result->percentage:= match/m*100.0;
      5.1.4. result->position := i-m+1;
      5.1.5. Return result;
      5.1.6. End of if
      5.1.7. i:=i+1;
      5.1.8. j:=j+1;
   5.2. else if match>0
      5.2.1. result-percentage := match/m*100.0;
      5.2.2. result-position := i-match;
      5.2.3. Return result;
      End of if
   5.3. match:=0;
   5.4. j := table[j];
   5.5. if j = 0
      5.5.1. if text[i] != pattern[0]
      5.5.2. I := i+1;
      5.5.3. End of if
      End of if
   End of while loop
6. Return NULL;
7. STOP

III. RESULTS AND DISCUSSION

The proposed algorithm is implemented in C++. It is tested on 4 source DNA strings with 4 DNA pattern strings. This gave us a total of 16 different combinations of test/source strings along with the pattern strings of whose KMP search time without the thread and with the use of thread was recorded. The 4 pattern strings are of size 200 bytes, 400, 800 bytes and 1024 bytes. Similarly the four test/source strings are of size 100 KB, 600 KB, 800 KB and 1024 KB. For both the algorithms namely the proposed Multithread KMP and the Normal KMP searching time are recorded in each case. The results are represented in the form of graphs given below:

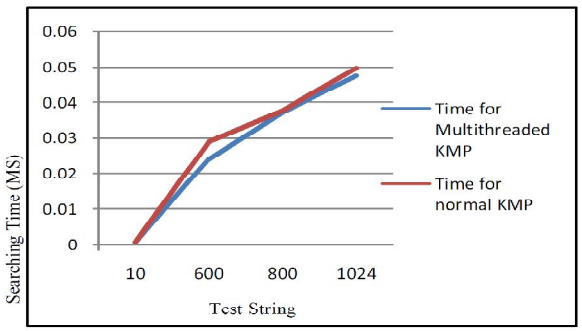

Fig. 1 Graph Showing Sequencing Time for all four Test Strings with Pattern 1(200 bytes)

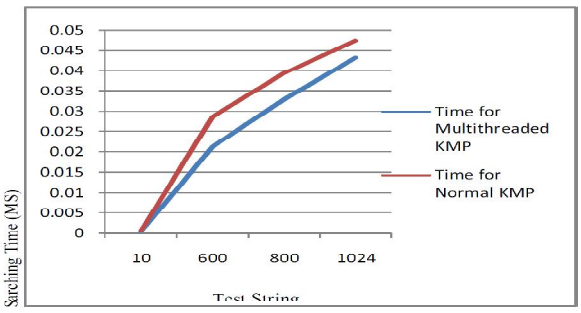

Fig. 2 Graph Showing Sequencing Time for all four Test Strings with Pattern 2(400 bytes)

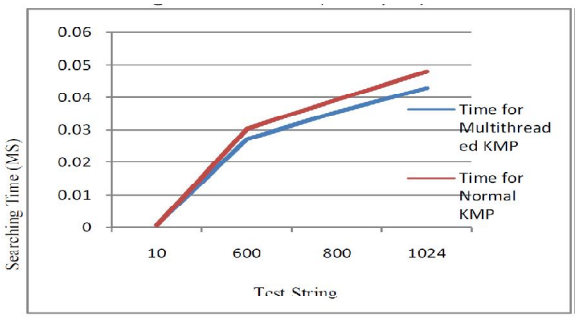

Fig. 3 Graph Showing Sequencing Time for all four Test Strings with Pattern 3(800 bytes)

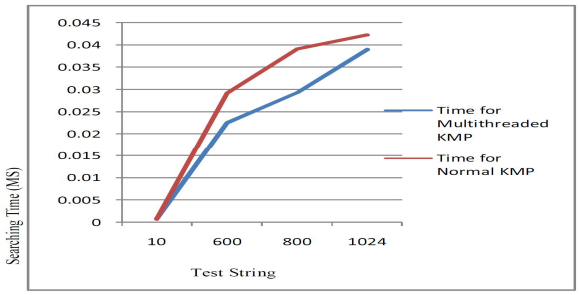

Fig. 4 Graph Showing Sequencing Time for all four Test Strings with Pattern 4(1024 bytes)

Based on the experiments carried out, it is clear that the multithreaded version of the KMP algorithm executes a bit better than the normal KMP algorithm.
After every search completed by the normal KMP and Multithreaded KMP, it is seen that the records written by the normal KMP is lesser than that of the Multithreaded KMP. This tells us that the Multithreaded KMP looks for all the possible patterns that are available in the sequence in a much smaller time thereby increasing its efficiency.

CONCLUSION

The percentage of match of each pattern string and their relative positions in the source string are obtained accurately and stored in text files. Also the searching speed of the multithreaded algorithm is much faster than the normal KMP algorithm.
This application can be used to match patterns of various strings of any huge length to find their relative percentages and positions. The size of the text string often gave difference in results. The Multithreaded KMP gave the desired results when the size of the text string was increased around 10kb and more. Similarly the pattern string was accordingly increased. The Multithreaded KMP requires high speed processors to produce the efficient results and so the implementation was carried out at HP-Z400 Workstation in our lab.
The algorithm has few limitations like, it can work accurately only on large text strings and pattern strings. It also needs adequate hardware and software platforms. The application can be used only in operating systems that support thread applications.
In the experiments carried out it is found that space complexity of the application is really high so future improvements can be made on this area to reduce this requirement. It can be further extended by developing a web based application, displaying all the extracted information like characters of the particular match, mismatch percentage, etc.

REFERENCES

[1] Ajay Daptardar and Dana Shaphira, "Adapting the Knuth-Morris-Pratt Algorithm for Pattern Matching in Huffman Encoded Text Computer Science", Department, Brandeis University,Wattham,MA,2006
[2] Text Available (Online):Http://Dm-Dingwang.Blogspot.Com/2007/05/ Supervised-Versus-Unsupervised-Methods.Html
[3] Akhtar Rasool, Dr. Nilay Khare, Himanshu Arora ,Amit Varshney , Gaurav Kumar, "Multithreaded Implementation of Hybrid String Matching Algorithm" Maulana Azad National Institute of Technology, International Journal on Computer Science and Engineering,2012
[4] Text Available (Online) : Gene Myers Whole – Genome DNA Sequencing, IEEE
[5] Computer Society, 1999, Pp33 –43 Occurrences Algorithm for String Searching Based
[6] on Brute- Force Algorithm, Journal of Computer Science, 82– 86, 2006.
[7] Robert W.P.Luk, Hong Kong "Chinese String Searching Using the KMP Algorithm" Polytechnic University, Kowloon, Hong Kong.1996
[8] Text Available (Online):Http://En.Wikipedia.Org/Wiki/Knuth-Morris-Pratt_Algorithm
[9] Text Available (Online):Http://Dna_Sequencing\Online Analysis Tools.Htm
[10] Text Available (Online):Http://En.Wikipedia.Org/Wiki/Sequence_Analysis
[11] Knuth D., Morris J.And Pratt V, Fast Pattern Matching In Strings, SIAM Journal Of Computer Science, 1977, Pp323 – 350 Samanrahman, Ahmad, Osman. A Minimum Cost Process In Searching For A Set Of Similar DNA Sequence, International Conference On Telecommunications And Informatics, May 2006, Pp348–3 53
[12] Text Available (Online): http://ghr.nlm.nih.gov/handbook/illustrations/dnastructure.jpg.

★★★