Research Journal of Applied Sciences, Engineering and Technology 4(9): 1198-1204, 2012
ISSN: 2040-7467

Submitted: January 11, 2012 Accepted: February 06, 2012 Published: May 01, 2012

Development of Cache Oblivious Based Fast Multiple Longest Common Subsequence Technique(CMLCS) for Biological Sequences Prediction

A. Sumathi and N. K. Sakthivel
School of Computing, SASTRA University, Thanjavur-613 401, TN, India

Abstract: A biological sequence is a single, continuous molecule of nucleic acid or protein. Classical methods for the Multiple Longest Common Subsequence problem (MLCS) problem are based on dynamic programming. The Multiple Longest Common Subsequence problem (MLCS) is used to find the longest subsequence shared between two or more strings. For over 30 years, significant efforts have been made to find efficient algorithms for the MLCS problem. Many of them have been proposed for the general case of any given number of strings. They could benefit greatly from improving their computation times. (Qingguo *et al*.,) proposed a new algorithm for the general case of Multiple LCS problem, which is finding the LCS of any number of strings. This algorithm is based on the dominant point approach and employs a fast divide-and-conquers technique to compute the dominant points. From this existing work, it is observed that, when this approach is applied to a case of three strings, this algorithm demonstrates the same performance as the fastest existing MLCS algorithm. When applied to more than three strings, this technique is significantly faster than the existing sequential methods, reaching up to 2-3 orders of magnitude faster speed on large-size problems. However, from our experimental results, it is observed that as the size of the Data Set is increasing, its performance decreases in terms of execution time. To overcome this major issue, we have developed an efficient model called Cache Oblivious based Multiple Longest Common Subsequence (CMLCS). From our experimental results, it is observed that our proposed work performs better as compared with the existing system in terms of Execution Time and Memory Usage.

Key words: Dominant point method, dynamic programming, LCS, MLCS

INTRODUCTION

The Longest Common Subsequence (LCS) (Qingguo, 2011). is to find the longest subsequence common to all sequences (often just two). Multiple Longest Common Subsequence problems (MLCS) (Qingguo, 2011). is to find the longest subsequence shared between two or more strings. Finding MLCS is an important problem in the areas of bioinformatics and Computational genomics. A method that solves the general MLCS problem efficiently can be applied to many computational biology and computational genomics problems that deal with biological sequences (Dmitry, 2008). A biological sequence is a single, continuous molecule of nucleic acid or protein. Classical methods for the MLCS problem are based on dynamic programming. It is straightforward and it calculates all the entries. It has time and space complexity. Various approaches have been introduced to reduce the complexity of dynamic programming. The new method is based on the dominant point approach. The main idea behind the dominant point approach is to identify exclusively the dominant point values instead of identifying values of all positions in matrix. The dominant point approach first computes the set of all 1-dominants.

As a general iteration step, a set of (k+1)-dominants is calculated from a set of k-dominants. Thus, by advancing from one contour to another, all dominant points can be obtained. Methods that solve MLCS using dominant points are found to be more efficient, reducing the size of search space by orders of magnitude, the MLCS method can be applied to many areas of bio-informatics and computational genomic as well outside the biological domain.

The major drawback of the existing Fast Multiple Longest Common Subsequence is its complexity. ie From our work, it is observed that as the size of the Data Set is increasing, its execution time and memory usage are also increases, which degrading the system performance. To address this issue, we have proposed an efficient model which is the improved version of the existing model. It is called as the Cache based Multiple Longest Common Subsequence problem (CMLCS). From our experimental results, it is noted that this proposed work performs better and it is improving the performance of the existing system in terms of Execution Time and Memory usage.

LITERATURE REVIEW

In this section, we are describing the LCS, MLCS, Dynamic Programming and Dominant-point approach.

Corresponding Author: A. Sumathi, School of Computing, SASTRA University, Thanjavur -613 401, Tn, India

Introduction to LCS: Biological sequence (Chen, 2006), can be represented as a sequence of symbols. For instance, a protein is a sequence of 20 different letters (amino acids), and DNA sequences (genes) can be represented as the combination of four letters A, C, G and T. The corresponding four sub-molecules forming DNA sequence. When a new bio sequence is found, we want to know, it is similar to what other sequences. Sequence comparison has been used successfully to establish the link between cancer-causing genes and a gene evolved in normal growth and development. One way of detecting the similarity of two or more sequences is to find their longest common subsequence. The LCS problem is to find a substring that is common to two or more given strings and is the longest one of such strings. LCS can be used for many applications.

- Molecular biology
- File comparison
- Screen redisplays

MLCS: The Multiple Longest Common Subsequence problem (MLCS) is to find the longest subsequence shared between two or more sequences. It is a classical computer science problem with important applications in many fields such as information retrieval and computational biology (Temple, 1981). A method that solves the general MLCS problem efficiently can be applied to many computational biology and computational genomics problems that deal with biological sequences, with the increasing volume of biological data and prevalent usage of computational sequence analysis tools, we expect that the general MLCS algorithm will have a significant impact on computational biology methods and their applications. It is widely used in bioinformatics and computational biology, mostly in DNA and protein sequence analysis. One of MLCS most direct implementation in a protein sequence analysis is a search for a motif or sets of motifs given a protein family (motif is a short conserved region in a protein sequence with known or implied function). A few best LCS and MLCS based Techniques are Parallel algorithm, Hsu and Su New algorithms, Smith and waterman algorithms and Basic Local Alignment System Tool (BLAST).

Dynamic programming methods: A dynamic Programming approach consists of sequence of four steps:

- Characterize the structure of an optimal solution
- Recursively define the value of an optimal solution
- Compute the value of an optimal solution in a bottom-up fashion
- Construct an optimal solution from computed information

		G	T	A	A	T	C	T	A	A	C
	0	0	0	0	0	0	0	0	0	0	0
G	0	①	1	1	1	1	1	1	1	1	1
A	0	1	1	②	2	2	2	2	2	2	2
T	0	1	②	2	2	③	3	3	3	3	3
T	0	1	2	2	2	3	3	④	4	4	4
A	0	1	2	③	3	3	3	4	⑤	5	5
C	0	1	2	3	3	3	④	4	5	5	⑥
A	0	1	2	3	④	4	4	4	5	⑥	6

Fig. 1: Score matrix L for two sequences, a1: GATTACA; a2: GTAATCTAAC

Classical methods for the MLCS problem are based on dynamic programming (Sankoff, 1972). Given two sequences X and Y of length of a1 and a2, respectively, a Dynamic Programming Algorithm iteratively builds an n1 × n2 score matrix L in which L[i, j], 0≤=i≤=n1, 0≤=j≤=n2. The length of an LCS between two prefixes ×[1,…i] and Y[1,…,j]. Specifically, the score matrix L is defined as follows:

$$L[i,j]=\begin{cases}0 & \text{if } i \text{ or } j=0,\\ L[i-1,j-1]+1 & \text{if } x[i]=y[j].\\ \max\left(L[i,j-1],L[i-1,j]\right) & \text{otherwise}\end{cases} \quad (1)$$

The definition of the matrix L can be naturally generalized to a case of N sequences: for each position L[i1,i2,…iN], its value is defined through the immediately preceding positions. Once the score matrix L is computed, we can extract MLCS by tracing back from the end point [n1, n2] to the starting point [0, 0]. It is straightforward to calculate all entries in L using dynamic programming. It is straightforward to calculate all entries in L using dynamic programming. Figure 1 shows the example of score matrix L for two sequences a1 = GATTACA and a2 = GTAATCTAAC.

The matrix L computed using (1) for two sequences a1: GATTACA and a2: GTAATCTAAC. The regions of the same entry values are bounded by contours; the corner points of contours are dominant points and are circled.

Dominant point methods: Various approaches have been introduced to reduce the complexity of dynamic programming (Qingguo, 2011; Chowdhury, 2010). Dominant points (Korkin, 2001) are minimal points in a multidimensional search space. Knowing those points allows reducing the search space size by orders of magnitude, hence significantly the computation time.

The main idea behind the dominant point approach (Chowdhury, 2010). is to identify exclusively the dominant point values instead of identifying values of all positions in matrix L. It consists of the following two parts:

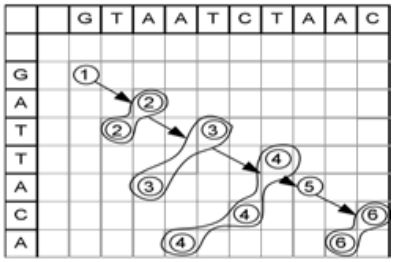

Fig. 2: The dominant point approach for two sequences a1: GATTACA; a2: GTAATCTAAC

- **First part:** Specifically, the Dominant Point Approach with iterative steps to compute all (k+1) dominant points, where 1<=k<=|MLCS| -1

- **Second part:** By tracing back through set of dominant points obtained in the first part of the algorithm and starting with an element from the last dominant set.

Figure 2 shows the process of dominant point approach for two sequences a1: GATTACA and a2: GTAATCTAAC. Broken lines are contours or boundary of different level of points. Dominant points at the same level are encircled together. Arrows in the figure point from the set of K-dominants to the set of (K+1) dominants, 1≤=k≤=|MLCS|-1.

Many parallel (Babu, 1997; Chen, 2006). MLCS algorithms have been developed to speed up the computation. The majority of them are designed for LCS problem of two sequences. Only a few parallel algorithms have been proposed for MLCS problems of three or more sequences. The first parallel approach to tackle the general MLCS problem could not achieve a consistent speedup on the test set of multiple sequences.

Identified problems: From our Related Works, it is observed that various Mechanisms (Xu, 2005; Yap, 1998). such as PLCS (Parallel algorithm for LCS problem) and Parallel Speculative Berger-Munson algorithm have been proposed recently to improve the performance of System Performance in terms of Time and Space Complexity. The Fast Multiple Longest Common Subsequence Technique (Fast MLCS) is the best scheme to improve the performance as compared with above said existing Parallel (Hunt, J.W. and T.G. Szymanski, 1997). MLCS methods.

However, from our study, we revealed that this work has its own draw backs,

- Lengths of the string have some limitations
- As number of sequences increases, its complexity increases

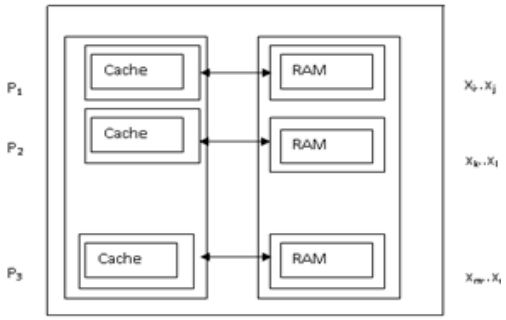

Fig. 3: Cache with parallelization

To address this identified major issue, we have developed an efficient model called Cache based MLCS (CMLCS), which is improving the System Performance in terms of Execution Time and Memory Usage. In the following section, we have presented our proposed work for MLCS problem of any number of sequences.

PROPOSED TECHNIQUE

As stated in the previous section, to improve the performance of Fast MLCS, this research work is proposed an efficient Cache-Oblivious based MLCS Model (CMLCS), which performs better and it will improve the performance of the existing system in terms of Execution Time and Memory Usage. The external architecture of CMLCS is shown in the Fig. 3.

Use of CMLCS: Cache-Oblivious approach (CMLCS) improves the space usage and Execution Time of the traditional dynamic programming.

Algorithm CMLCS:
// C: Cache and R: RAM
begin
C = {(c_1,c_2…c_n)}; R = {(r_1,r_2…r_n)}
C <- R
Processor P:
{(P_a, P_b, P_c … P_n)}
while D^k not empty do
for P ∈ P_a … P_n do
for P_a ∈ p_1…p_n
c = p_1
p_1= p_2
p_2= p_3
end for
end for
P = P_a U P_b U …P_n
end

The pseudo code of Cache-Oblivious based MLCS Cache-oblivious approach performs the following step:

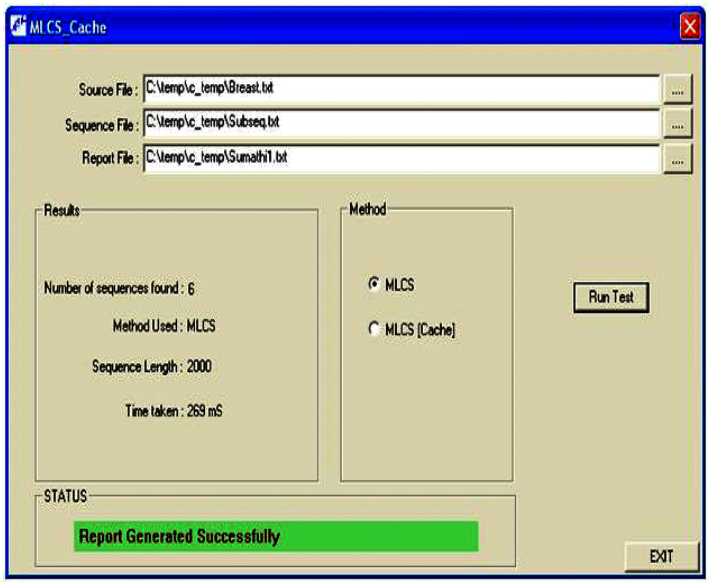

Fig. 4: Execution of MLCS

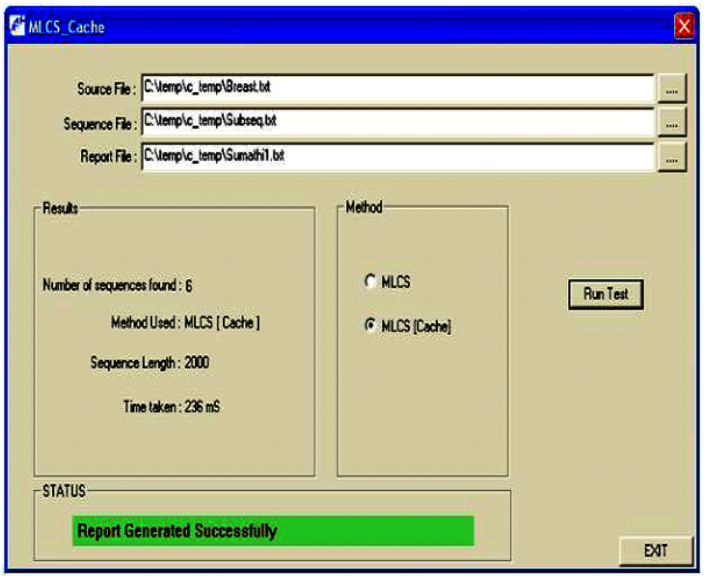

Fig. 5: Execution of CMLCS

- Initialize Cache memory
- Invoke the first gene sequence
- Invoke the second sequence when the first sequence is running
- Invoke the third sequence when the first sequence enters the completion state
- Do the same process until search completes

Based on this Cache Oblivious approach (CMLCS), which is shown in the Fig. 4, this work implemented this procedure and MLCS with VC++ Programming Language. The GUI of the developed tool is shown in the Fig. 4 and 5. As shown in the Fig. 4 and 5, this tool is implemented both the MLCS (Fig. 4) and Cache based CMLCS (Fig. 5) and these methods could be executed and its performance is measured. i.e.,

- The Existing System MLCS and the Proposed System CMLCS can be compared in terms of the execution time and memory.
- It also could be identified the sequence of the given string

PERFORMANCE ANALYSIS

This study has studied the both the existing and proposed technique thoroughly. For this study,

Report Time: 04-01-2012-:: 12:19:34	Report Time: 04-01-2012:: 12:19:38
Source File : C:\temp \c_temp\Bladder. txt	Source File : C:\temp \c_temp\Bladder. txt
Sequence File : C:\temp \c_temp\Subseq. txt	Sequence File : C:\temp \c_temp\Subseq. txt
Sequence Length: 2000	Sequence Length: 2000
Number of Subsequence Found: 7	Number of Subsequence Found: 7
Method Used: MLCS	Method Used: CMLCS
Time taken: 262 ms	Time taken: 211 ms

Fig. 6: Execution details of bladder cancer

Report Time: 04-01-2012-:: 12:19:47	Report Time: 04-01-2012:: 12:19:56
Source File : C:\temp \c_temp\Bladder. txt	Source File : C:\temp \c_temp\Bladder. txt
Sequence File : C:\temp \c_temp\Subseq. txt	Sequence File : C:\temp \c_temp\Subseq. txt
Sequence Length: 2000	Sequence Length: 2000
Number of Subsequence Found: 1024	Number of Subsequence Found: 1024
Method Used: MLCS	Method Used: CMLCS
Time taken: 305 ms	Time taken: 228 ms

Fig. 7: Execution details of breast cancer

Report Time: 04-01-2012-:: 12:20:03	Report Time: 04-01-2012:: 12:20:12
Source File : C:\temp \c_temp\Colon. txt	Source File : C:\temp \c_temp\Colon. txt
Sequence File : C:\temp \c_temp\Subseq. txt	Sequence File : C:\temp \c_temp\Subseq. txt
Sequence Length: 2000	Sequence Length: 2000
Number of Subsequence Found: 4	Number of Subsequence Found: 4
Method Used: MLCS	Method Used: CMLCS
Time taken: 265 ms	Time taken: 210 ms

Fig. 8: Execution details of colon cancer

Report Time: 04-01-2012-:: 12:20:29	Report Time: 04-01-2012:: 12:20:38
Source File : C:\temp \c_temp\Endomatiral. txt	Source File : C:\temp \c_temp\Endomatiral. txt
Sequence File : C:\temp \c_temp\Subseq. txt	Sequence File : C:\temp \c_temp\Subseq. txt
Sequence Length: 2000	Sequence Length: 2000
Number of Subsequence Found: 5	Number of Subsequence Found: 5
Method Used: MLCS	Method Used: CMLCS
Time taken: 298 ms	Time taken: 211 ms

Fig. 9: Execution details of endomatrial cancer

Report Time: 04-01-2012-:: 12:20:55	Report Time: 04-01-2012:: 12:20:58
Source File : C:\temp \c_temp\Kidney. txt	Source File : C:\temp \c_temp\Kidney. txt
Sequence File : C:\temp \c_temp\Subseq. txt	Sequence File : C:\temp \c_temp\Subseq. txt
Sequence Length: 2000	Sequence Length: 2000
Number of Subsequence Found: 4	Number of Subsequence Found: 4
Method Used: MLCS	Method Used: CMLCS
Time taken: 279 ms	Time taken: 235 ms

Fig. 10: Execution details of kidney cancer

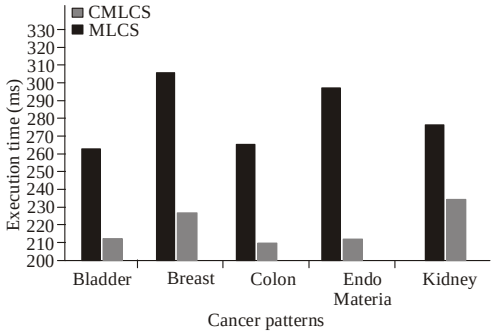

Fig. 11: Performance analysis of MLCS and CMLCS

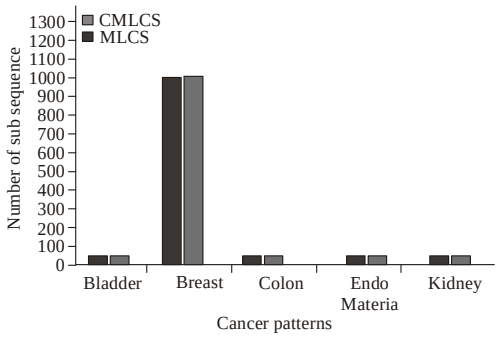

Fig. 12: Cancer pattern prediction

used different types of gene sequences, that have been used to identify the cancer pattern and it proves our proposed study performs better as compared with the existing system in terms of Execution Time and Memory Usage.

For this experiment, this work has taken different types of gene sequences (Cancer patterns) such as Bladder, Breast, Colon, Endomatiral, Kidney, Lukemia, Lung, Lymphoma, Melanoma, Pancreatic and Thyroid, as its data set each containing between 1000 and 2000 characters in length.

Figure 6-10 shows that the different execution details of Cancer Patterns. From these Figures, we identified that this proposed tool, search the given string of Cancer Pattern Sequences with all the available Cancer Patterns. This tool founds the given string as Breast Cancer pattern and it is also measured its sequence length is 1024.

Table 1 shows the Execution time of Multiple Longest Common Subsequence and Cache based MLCS (CMLCS) for different type of cancer patterns.

From the Fig. 11, it is observed that the execution time of this proposed system is considerably less as compared with existing approach MLCS. Figure 12 shows the Cancer Pattern Prediction, it identifies the type of cancer based on the maximum sequences matched.

The experimental results show that our proposed work performs better as compared with the existing system in terms of Execution Time and Memory Usage.

Table 1: Execution time of MLCS and CMLCS

	Execution time (ms)	
Cancer patterns	MLCS	CMLCS
Bladder	262	211
Breast	305	228
Colon	262	210
Endomatiral	298	211
Kidney	279	235

From our system settings, we have revealed that our work effectively utilizing both the Memory and Cache as well, which improves the System Performance.

CONCLUSION

This research study is implemented our proposed technique called Cache based MLCS (CMLCS) and studied thoroughly with the existing model. From the results, it is observed that the proposed work performs better than the existing method in terms of execution time and memory usage. It is also noted that in the MLCS, as the size of the Data set is increasing, its performance decreases in terms of execution time and memory usage. This work also classifies all the Cancer Patters effectively with higher system performance, which will improve the Data Center Server.

REFERENCES

Babu, K.N. and S. Saxena, 1997. Parallel algorithms for the longest common subsequence problem. Proceedings of the Fourth International Conference High Performance Computing, (HPC' 1997), pp: 120-125, 1997.

Chen, Y., A. Wan and W. Liu, 2006. A fast parallel algorithm for finding the longest common sequence of multiple biosequences. BMC Bioinforma., 7(4).

Sankoff, D., 1972. Matching sequences under deletion/insertion constraints. Proc. Natl. Acad. Sci. USA, 69(1):4-6.

Dmitry, K., Q. Wang and Y. Shang, 2008. An efficient parallel algorithm for the multiple longest common subsequence (mlcs) problem. Proceedings of the 37th International Conference on Parallel Processing IEEE Computer Society. Washington, DC, pp: 354-363.

Hunt, J.W. and T.G. Szymanski, 1977. A fast algorithm for computing longest common subsequences. Comm. ACM, 20(5): 350-353.

Korkin, D., 2001. A New Dominant Point-Based Parallel Algorithm for Multiple Longest Common Subsequence Problem. Technical Report TR01-148, University of New Brunswick.

Qingguo, W., K. Dmitry and S. Yi, 2011. A fast Multiple Longest Common Subsequence (MLCS) Algorithm. IEEE Transact. Knowledge Data Eng., 23(3).

Chowdhury, R.A., L. Hai-Son and R. Vijaya, 2010. Cache-oblivious dynamic programming for bioinformatics. IEEE/ACM Transact. Computat. Biol. Bioinformat., 7(3).

Temple, F.S. and M.S. Waterman, 1981. Identification of common molecular subsequences. J. Molecul. Biol., 147(1):195-197.

Xu, X., L. Chen, Y. Pan and P. He, 2005. Fast Parallel Algorithms for the Longest Common Subsequence Problem Using an Optical Bus. Lecture Notes in Computer Science, Springer, pp: 338-348.

Yap, T.K., O. Frieder and R.L. Martino, 1998. Parallel computation in biological sequence analysis. IEEE Trans. Parallel Distribut. Syst., 9(3): 283-294.