LCS Algorithm with Vector-markers

Levon Aslanyan
Institute for Informatics and Automation Problems
of the National Academy of Science of Armenia
Yerevan, Armenia
lasl@sci.am

***Abstract*—The Multiple Longest Common Subsequence (MLCS) problem is aimed at constructing a maximum length subsequence, common to a given set of sequences, defined on some finite alphabet of symbols. The paper considers the particular case of two input sequences (LCS), which is simply extendable to the general MLCS problem. We consider the problem in an online manner, where symbols arrive one-by-one and the next acquired symbol is appending any one of the two input sequences. The sought-for LCS algorithm acts by recursive handling of parts of sequences arrived so far, constructing and updating specific supportive structures of markers representing the interrelations of the longest common subsequences of the two input sequences. In paper we discuss a perfect online parallelization framework of the algorithm for the "simple" memory model, so that the parallel complexity becomes $O(mn/t)$ for t parallel threads. The general outcome of paper is the use of vector markers instead of matrix markers or graphs, which helps in minimization of the memory, used by the algorithm.**

***Keywords*— LCS, online algorithm, parallelization, iteration, data structure**

I. Introduction

This introductory section presents the required definitions and preliminaries, and provides a short survey on LCS problem area. The dynamic programming as well as the Dominant match based models are considered.

A. Preliminries

Information of various application areas often can be modelled as a set of finite alphanumeric sequences. Examples are DNA or protein sequences in biology. Consider a finite alphanumeric alphabet Σ. A *subsequence* A° of a sequence A defined on alphabet Σ is a sequence that can be derived from A by removing some of its items. Obviously, in the general case the same subsequence can be obtained from the same sequence A by removing its different sets of items. The empty sequence is said to be obtained from any sequence by removing all its items, so it is a subsequence of each sequence. A sequence C is said to be a *common subsequence* of sequences A and B, if C, in separate, can be obtained as subsequences of both sequences. Note that the empty sequence is a zero length common subsequence of A and B, thus, formally, the set of all common subsequences of A and B is not empty. Some subsequences in the pair A and B are deadlock (not extendable). Some of them are of maximal length, and each of these sequences is called the *longest common subsequence* or *LCS*. The *LCS problem* is to construct algorithmically one or the entire longest common subsequences to the given pair (or a larger set) of sequences.

Among the *LCS* of A and B it is to distinct those with minimal maximal elements in A, and/or in B. This kind of structures are identified in considered algorithm below for all lengths less than the length of entire LCS. These collections of subsequences define the *specific structures of markers* used in online mining of LCS. These novel data structures and the algorithm developed by the use of these structures will help in solving new online applications of traditional LCS problems.

The well-recognized theoretical and applied value of the LCS model may be introduced in terms of bioinformatics ([1]-[4]), where sequences are the most basic mathematical model of genomics, which can describe the primary structure of the nucleic acid and protein molecules. Searching LCS in genomic sequencing and alignment issues an important approach of identifying the sequence similarities that can be further utilized in gene identification, in construction of optimal haplotype mechanisms, in mutation determination, in genotype-phenotype similarity searches, classifications, etc. With the successful implementation of the Human Genome Project, the available number of, lengths and sizes of biological sequences are growing explosively and exponentially. Mining the LCS from these sequences is becoming more hardly computable and important in theory and in applications.

B. Survey on the LCS problem

Consider two sequences $A = a_1 \cdots a_i \cdots a_m$ and $B = b_1 \cdots b_j \cdots b_n$ defined on the same *alphabet* Σ. If $a_i = b_j$ for some i, $1 \le i \le m$, and for some j, $1 \le j \le n$, then (i,j) is called a *match* between A and B. A match (i,j) is said to be *preceding* another match (i',j'), if concurrently $i < i'$ and $j < j'$. The cross intersecting matches are complementary to the set of matches of A and B and they are useless in constructing of LCS. Note that a *longest common subsequence* (LCS) of A and B is some sequence $C = c_1 \cdots c_k \cdots c_l$ of matches $(i_k, j_k)_{k=1}^{l}$, $l \ge 0$, such that matches $(i_k, j_k)_{k=1}^{l}$ between A and B, with increase of k, proceed each other, and l is maximum length among such sequences.

978-1-5386-2830-0/17/$31.00 ©2017 IEEE

For each i, $0 \le i \le m$, we denote by A_i the i-th prefix of A, $A_i \coloneqq a_1, \cdots, a_i$, and for each j, $0 \le j \le n$ B_j is the j-th prefix of B, $B_j \coloneqq b_1, \cdots, b_j$. In particular, A_0 and B_0 are the empty sequences.

In 1970, S. Needleman and C. Wunsch, being the first, proposed a heuristic homology algorithm using the *match-mismatch,* and *insertion-deletion* operations for sequence alignment [5]. This is a global alignment algorithm that requires $O(m, n)$ calculation steps (m and n are the lengths of the two sequences being aligned). The algorithm uses the iterative calculation of a matrix for the purpose of modelling the global alignment. In the following, D. Sankoff [6], A. Reichert et al. [7], W. Beyer et al. [8] and others formulated alternative heuristic algorithms for analyzing gene sequence similarities. P. Sellers introduced a system for measuring sequence distances [9]. In 1981, Smith and Waterman published a new local alignment calculation algorithm. The Smith–Waterman algorithm is to align two sequences of lengths m and n, and it is rather time-consuming requiring $O(m^2 n)$ steps. O. Gotoh [10] and S. Altschul [11] optimized this algorithm to $O(mn)$ steps. The space complexity was optimized by W. Myers and E. Miller [12] from $O(mn)$ to $O(n)$ (linear), where n is the length of the shorter sequence.

In Big Data era, the lengths and sizes of alphanumeric sequences of experiments are growing explosively, leading to grand challenges for the classical NP-hard problem of searching for the Longest Common Subsequences of the two or more input sequences. The state-of-the-art LCS algorithms are hardly applied to long and large-scale sequences alignments. To overcome their drawbacks and tackle the longer or even big sequences alignments, various strategies, e.g., parallel hierarchical sorting, optimal labeling, reuse of intermediate results, subsection calculation and overall integration into the hybrid analytic systems is required ([13]-[17]). The target is to achieve the real linear time and space complexity algorithm for aligned sequences. This is very similar to the case of solving the sparse linear algebraic systems. Initially being of complexity $O(n^3)$, current research tends to complete the development of the large size linear-complexity sparse linear solvers.

The widely known algorithm (D. Hirschberg [18]), and its consecutive modifications solve the LCS problem by the dynamic programming approach. We refer to this algorithm as a "classical" algorithm. It is an incremental algorithm based on a notion that the pair of last elements of sequences help to shorten the considered portions of the sequences. Let

$$l_{i,j} = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ \max(l_{i-1,j}, l_{i,j-1}) & \text{if } i > 0 \text{ and } j > 0 \text{ and } a_i \neq b_j \\ l_{i-1,j-1} + 1 & \text{if } i > 0 \text{ and } j > 0 \text{ and } a_i = b_j \end{cases} \quad (1)$$

where $l_{i,j}$ denotes the length of the longest common subsequence of A_i and B_j for $0 \le i \le m$ and $0 \le j \le n$. Based on this equation, for the given sequences A and B the "classical" algorithm obtains an $(m+1) \times (n+1)$ matrix $(l_{i,j})_{i,j=0}^{m,n}$ of scores, and based on that matrix in a second stage it obtains one or more longest common subsequences of A and B. It runs $\Theta(mn)$ steps, though the Method of Four Russians (the one used in Boolean matrix multiplication algorithms [19]) can be applied to that algorithm [20], reducing the complexity to $\Theta(mn/\log m)$ (assuming $m \le n$). The "classical" algorithm can be implemented in an online manner, but it can't when the Method of Four Russians is applied.

[21] examined the lower bound of LCS problems in a decision tree model of computation, where the tree vertices represent "equal – unequal" comparisons. It is shown that in such model the lower bound is $\Omega(ms)$ (assuming $m \le n$, and s is the size of Σ). Not all algorithms solving the LCS problem necessarily fit in the "equal – unequal" comparison model, but in particular the one using the Method of Four Russians doesn't. The complexity $O(mn/\log m)$ is asymptotically the best among all known upper bounds when complexity is expressed in terms of lengths of the input sequences and the size of the input alphabet only [19]. For those, with the "equal – unequal" comparisons model, $O(mn)$ is asymptotically the best complexity. In either case there is a huge gap between the best known upper and lower bounds. Due to this situation for an algorithm solving LCS problem we see that the dependency of its complexity only from m, n and s poorly describes the algorithm. For many known algorithms solving the LCS problem their complexities essentially depend on other nontraditional parameters describing the input pair of sequences. The most common parameter of this kind is the LCS length itself, denoted by l; the number of matches between the input sequences, denoted by r (this is related to the sparsity measure); and the number of some special kind of matches called *dominant matches,* denoted by d [22]. Different parameters are studied in a way they appear in this and other discrete models [23]-[30].

A match (i, j) is called *dominant*, if $l_{i-1,j-1} = l_{i-1,j} = l_{i,j-1} = l_{i,j} - 1$ (recall that $l_{i,j}$ is the LCS length of i-th prefix of A and j-th prefix of B), and $l_{i,j}$ is called the *rank* of the dominant match (i, j). Note that in the worst case $l = \Theta(m)$, $r = \Theta(mn)$ and it can be shown that $d = \Theta(mn)$. A more detail survey on the complexities of the algorithms solving the LCS problem is provided in [31]. Depending on interrelations between values m, n, s, l, r, d, as well as depending on other issues of particular applications, some of the LCS algorithms may become preferable than the other ones.

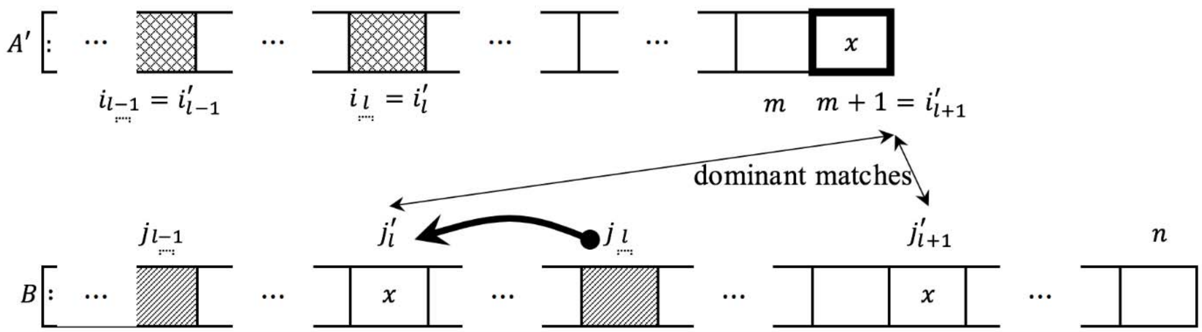

Fig. 1. Online update of sequence A, dominant matches, and markers.

II. THE ALGORITHM

As it is mentioned above, our target is to investigate the LCS problem in an online performance manner, where the next symbol arrival is an action that appends the element to one of the two input sequences. The algorithm iteratively processes that arrivals, updating the maintained structures, representing the LCS of sequences arrived so far. Let A and B be the already registered sequences and let a new symbol $x \in \Sigma$ is appending to A, thus resulting a new sequence $A' \coloneqq Aa_{m+1}$ with $a_{m+1} = x$. Starting from the next point we will describe the current iteration step in an online LCS algorithm which is based on analysis of the new data arrival, constructing the special algorithmic work-time vector data, corresponding to sequences A and B. These structures provide an LCS of A' and B, and algorithm may simply update these data to correspond them to the sequences A' and B.

A. Markers and their update

As before, let $A = a_1 \cdots a_i \cdots a_m$, $m \geq 1$, and $B = b_1 \cdots b_j \cdots b_n$, $n \geq 1$, be sequences defined on some symbol alphabet Σ, and let l be the LCS length of A and B. Denote by i_k, $1 \leq k \leq l$, the minimum among all A-indices of the last elements of k-length common subsequences of A and B, and denote by j_k, $1 \leq k \leq l$, the minimum among all B-indices of the last elements of k-length common subsequences of A and B, so that

$$i_k \coloneqq \min\{p_k \mid \exists\, (p_r, q_r)_{r=1}^{k} : a_{p_r} = b_{q_r}, 1 \leq r \leq k; 1 \leq p_1 < \cdots < p_k \leq m; 1 \leq q_1 < \cdots < q_k \leq n\}, \quad (2)$$

$$j_k \coloneqq \min\{q_k \mid \exists\, (p_r, q_r)_{r=1}^{k} : a_{p_r} = b_{q_r}, 1 \leq r \leq k; 1 \leq p_1 < \cdots < p_k \leq m; 1 \leq q_1 < \cdots < q_k \leq n\}. \quad (3)$$

We call i_k the k-th *mark* of B in A and we call j_k the k-th *mark* of A in B.

Lemma 1: Marker sequences $(i_k)_{k=1}^{l}$ and $(j_k)_{k=1}^{l}$ are strictly increasing.

Indeed, let for some i_k, $k \geq 2$, C be a k-length common subsequences of A and B ending at i_k in A. Removing the last element from C we will get a $(k-1)$-length common subsequence of A and B ending in A at an index not greater than $i_k - 1$, and as i_{k-1} is the minimum among such indices, then we will get that $i_{k-1} < i_k$. Similarly it can be checked that $(j_k)_{k=1}^{l}$ is also strictly increasing. Note, that the subsequences of A induced by i_k-s and the subsequences of B induced by j_k-s are not necessarily common subsequences of A and B. Also the (i_k, j_k)-s are not necessarily matches between A and B.

Now recall that $A' = Aa_{m+1}$, where $a_{m+1} = x$, and denote by l' the LCS length of A' and B. Obviously l' equals either l or $l + 1$. Then let i'_k, $1 \leq k \leq l'$, denote the k-th mark of B in A' and j'_k, $1 \leq k \leq l'$, denote the k-th mark of A' in B. Next we show how to obtain $(i'_k)_{k=1}^{l'}$ and $(j'_k)_{k=1}^{l'}$ based on $(i_k)_{k=1}^{l}$ and $(j_k)_{k=1}^{l}$.

Lemma 2: For $\boldsymbol{k}$, $\boldsymbol{1 \leq k \leq l}$, it holds $\boldsymbol{i'_k = i_k}$.

Lemma 3: It holds $l' = l + 1$ if and only if B has symbol x after index j_l, and in that case it holds $i'_{l+1} = m + 1$.

Corollary 1: If B has symbol x after index j_l, then it holds $l' = l + 1$ and j'_{l+1} is the index of first x after j_l in B.

Thus Lemma 2 and Lemma 3 show how to obtain the marks of B in A'. Next we show how to obtain the marks of A' in B.

Lemma 4: For k, $1 \leq k \leq l$, it holds $j_{k-1} < j'_k \leq j_k$.

Lemma 5: For k, $1 \leq k \leq l$, if there is x between indexes j_{k-1} and j_k in B, then j'_k is the index of the first of them, otherwise $j'_k = j_k$.

Thus the Lemma 4 and Lemma 5 show how to obtain the marks of A' in B, and previously we have shown how to obtain the marks of B in A'. Thus we have shown how to update the marks of A and B to marks of A' and B. For the usability purposes we combine this claim into the final postulations (see Fig. 1).

Theorem 1: For k, $1 \leq k \leq l$, it holds $i'_k = i_k$; if there is x between indexes j_{k-1} and j_k in B, then j'_k is the index of the first of them, and otherwise it holds $j'_k = j_k$; it holds $l' = l + 1$ if and only if B has symbol x after index j_l, and in that case it holds $i'_{l+1} = m + 1$ and j'_{l+1} is the index of first x after j_l in B.

Theorem 2: For some j, $1 \leq j \leq n$, the match $(m + 1, j)$ is dominant if and only if for some k, $1 \leq k \leq l$, it holds $j = j'_k < j_k$ or $l' = l + 1$ and $j = j'_{l+1}$.

Thus the Theorem 1 shows how to update the marks of A and B to the marks of A' and B, and Theorem 2 shows how to enumerate all dominant matches of A' and B with index $m + 1$ in A' during that update. Recall, that in order to provide an online algorithm solving the LCS problem it is sufficient to provide an online algorithm which enumerates the dominant matches of the input sequences.

III. Conclusion

The LCS (Longest Common Subsequence) problem is intensively applied and broadly investigated. A very basic role plays the dynamic programming style algorithm of its solution that have today many interpretations. Besides the classical postulation of the problem it is attractive to consider its online version due to network based applications. And in both cases static and online it is required to split the task into the parallel computational threads. The online parallel algorithm introduced in this paper presents another interpretation of the mentioned de-facto standard algorithm of the domain that provides additional structures that is able to accompany the algorithmic iterations, providing it the same way perfect parallelization for arbitrary number of processors.

The designed online parallel algorithm is given for the "simple" case of the basic algorithm when ordinary sequential data structure to store and update are used. The specific case when tree like structures are used to reduce the complexity is still waiting for its elaboration.

In general comparison dynamic programming [1], dominant match based [15], and the current vector marker algorithms differ in creation and use of the run time data of the algorithms. Dynamic programming algorithm is the most interpretable one because of it creates the complete matrix of the complementary score data. For short sequences this is the preferable way of LCS computation. More economical is the algorithm of dominant matches, which constructs the part of scores matrices corresponding to these matches in an optimized manner. The technique is based on creation and analysis of networks and/or directed origin-sink graphs, but there is no theoretical estimate about the size of those graphs. The vector marker algorithm of this paper is based on construction and analysis of two vectors of indexes with a total length not exceeding $2LCS$.

It is to mention that the other known parallel algorithms in the domain are developed on base of the classical algorithm so that they can't be online. They also depend critically on the lengths of input sequences and the number of processors.

References

[1] M. S Waterman, T. F Smith, and W. A Beyer, "Some biological sequence metrics", *Advances in Mathematics*, vol. 20, pp. 367–387, 1976.

[2] R. Durbin, S. Eddy, A. Krogh, G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, eleventh edition, Cambridge University Press, 2006.

[3] Yu. I. Zhuravlev, L. A. Aslanyan, and V. V. Ryazanov, "Analysis of a training sample and classification in one recognition model", *Pattern recognition and image analysis*, vol. 24, no. 3, pp. 347-352, 2014.

[4] A. Arakelyan, L. Aslanyan, A. Boyajyan, "On Knowledge-based gene expression data analysis", *Selected Revised Papers of 9th Computer Science and Information Technologies conference, IEEE Xplore*, pp. 105-109, 2013.

[5] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins", *Journal of Molecular Biology*, vol. 48, pp. 443–453, 1970.

[6] D. Sankoff, "Matching sequences under deletion/insertion constraints", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 69, no. 1, pp. 4-6, 1972.

[7] T. A. Reichert, D. N. Cohen and A. K. C. Wong, "An application of information theory to genetic mutations and the matching of polypeptide sequences", *Journal of Theoretical Biology,* vol. 42, pp. 245–261, 1973.

[8] W. A. Beyer, M. L. Stein, T. F. Smith, and S. M. Ulam, A molecular sequence metric and evolutionary trees, Mathematical Biosciences. 19: 9–25, 1974.

[9] P. H. Sellers, "On the Theory and computation of evolutionary distances", *Journal of Applied Mathematics*, vol. 26, pp. 787–793, 1974.

[10] O. Gotoh, "An improved algorithm for matching biological sequences, Journal of molecular biology", vol. 162, pp. 705–708, 1982.

[11] S. F. Altschul and B. W. Erickson, "Optimal sequence alignment using affine gap costs", *Bulletin of Mathematical Biology*, vol. 48, pp. 603–616, 1986.

[12] W. Miller and E. Myers, "Optimal alignments in linear space", *Computer applications in the biosciences*, vol. 4, pp. 11–17, 1988. doi:10.1093/bioinformatics/4.1.11.

[13] J. Yang, Y. Xu, and Y. Shang, *An Efficient Parallel Algorithm for Longest* Common Subsequence Problem on GPUs, Proceedings of the World Congress on Engineering 2010 Vol I WCE 2010, June 30 - July 2, 2010, London, U.K, 6p.

[14] W. Liu, L. Chen, and L. Zou, A Parallel LCS Algorithm for Biosequences Alignment, Conference: Proceedings of the 2nf International Conference on Scalable Information Systems, Infoscale 2007, Suzhou, China, June 6-8, 2007.

[15] E. David and G.Zvi, "Parallel algorithmic techniques for combinatorial computation", *Annual Review of Computer Science*, vol. 3, pp. 233–283, 1988.

[16] L. Aslanyan, J. Castellanos, F. Mingo, H. Sahakyan, and V. Ryazanov, "Algorithms for data flows", *International Journal Information Theories and Applications*, vol. 10, no. 3, pp. 279-282, 2003.

[17] L. Aslanyan and V. Minasyan, "LCS Algorithm for Big Data Flows", *CSIT Conference 2017*, Yerevan, Armenia, Sept. 25-29, 2017, pp. 183-186.

[18] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequences', *Communications of the ACM*, vol. 18, no. 6, pp. 341–343, 1975.

[19] A. Aho, J. Hopcroft, J. Ullman, D*ata Structures and Algorithms*, Addison-Wesley, 1983.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Clifford Stein, Introduction to Algorithms*, 3rd edition, MIT Press, 2009.

[21] A. Aho, D. Hirschberg, J. Ullman, “Bounds on the complexity of the Longest common subsequence problem”, *Journal of the Association for Computing Machinery*, vol 23, no. l, pp 1-12, January 1976.

[22] Y. Li , H. Li , T. Duan , S. Wang , Z. Wang, Y. Cheng, “A real linear and parallel multiple longest common subsequences (MLCS) algorithm”, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 13-17, San Francisco, California, USA 2016, 10 p.

[23] V. Minasyan, “On the structure of maximum independent sets in bipartite graphs”, *International Journal Information Theories and Applications (IJITA)*, vol. 17, no. 2, pp. 177-188, 2010.

[24] L. Aslanyan and H. Sahakyan, “Numerical characterization of n-cube subset partitioning”, *Electronic Notes in Discrete Mathematics*, *Elsevier B. V., ODSA 2006 - Conference on Optimal Discrete Structures and algorithms*, vol. 27, pp.3-4, October 2006.

[25] H. Sahakyan, “Numerical characterization of n-cube subset partitioning, discrete applied mathematics”, vol. 157, no. 9, pp. 2191-2197, 2009.

[26] H. Sahakyan, “(0,1)-matrices with different rows”, *CSIT 2013 - 9th International Conference on Computer Science and Information Technologies, Revised Selected Papers*, 7 pages, 2013.

[27] L. Aslanyan, H.-D. Gronau, H. Sahakyan, and P. Wagner, “Constraint satisfaction problems on specific subsets of the n-dimensional unit cube,” *CSIT 2015 – 10th International Conference on Computer Science and Information Technologies, Revised Selected Papers,*” 2015, pp. 47-52.

[28] L. Aslanyan and H. Danoyan, “On the optimality of the Hash-Coding type nearest neighbour search algorithm,” *CSIT 2013 - 9th International Conference on Computer Science and Information Technologies, Revised Selected Papers*, 7 pages, 2013.

[29] L. Aslanyan, H. Danoyan, “Complexity of hash-coding type search algorithms with perfect codes”, *J. Next Gener. Inf. Technol., vol.* 5 (4) pp. 26–35, 2014.

[30] H. Sahakyan, “Essential points of n-cube subsets partitioning characterization,” *Discrete Applied Mathematics*, vol. 163, no. 2, pp. 205-213, 2014.

[31] L. Bergroth, H. Hakonen, and T. Raita, “A survey of longest common subsequence algorithms”, Technical report, Dept. of Computer Science, University of Turku, Finland, 2000.