Information Sciences 295 (2015) 247–257

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Constrained sequence analysis algorithms in computational biology

Effat Farhana [a], M. Sohel Rahman [a,b,*,1]

[a] AℓEDA Group, Department of Computer Science and Engineering (CSE), Bangladesh University of Engineering and Technology (BUET), Dhaka-1000, Bangladesh
[b] Department of Informatics, King's College London, UK

ARTICLE INFO

Article history:
Received 25 July 2013
Received in revised form 4 October 2014
Accepted 11 October 2014
Available online 22 October 2014

Keywords:
Sequence analysis
Algorithm
Finite automaton

ABSTRACT

The knowledge of the similarity of two or more sequences is crucial in computational molecular biology. The longest common subsequence (LCS) is a well-known and widely used measure for sequence similarity. Constrained variants of the LCS problem have been studied in the literature where the knowledge of the functionalities or structures of the input sequences are provided in the form of inclusion/exclusion constraint patterns. In this paper we focus on different variants of the LCS problem involving multiple input sequences and constraint patterns. Given L input sequences and ℓ constraint patterns, the goal here is to construct an LCS of the given sequences such that each of the constraint patterns occurs/does not occur in the LCS as a subsequence/substring.

We devise finite automata based efficient algorithms for all the variants of the problem that run in $O(|\Sigma|(\mathcal{R}+L)+nL+|\Sigma|\mathcal{R}n^{\ell})$ time, where $\mathcal{R}$ is the size of the resulting subsequence automaton, n is the length of each input sequence and Σ is the underlying alphabet. We also conduct an extensive experimental study to evaluate the practical performance of our algorithms. The experimental results suggest the superiority of our finite automata based algorithms. Therefore, we believe that our automata based algorithms will be useful in practical sequence analysis in computational biology and will replace the existing algorithms that are mostly based on memory intensive dynamic programming based methods.

1. Introduction

The knowledge of the similarity of two (or more) sequences is crucial in various applications. The sequence similarity can be defined in many ways and one commonly used metric is the length of the longest common subsequence (LCS) that has been studied extensively for decades (e.g., [40,4,6,24,29,41]). Informally speaking, an LCS of two given sequences is a subsequence having the highest length that is common to both input sequences (a subsequence is obtained from a sequence by deleting zero or more symbols). By using a simple dynamic programming technique, the LCS problem for two sequences of n elements each can be solved in $O(n^2)$ time. The only $O(n^2)$ results known so far are due to Masek and Paterson [29] and Crochemore et al. [13], that run in $O(n^2/\log n)$ and $O(hn^2/\log n)$ time respectively. For the latter work, $h \leqslant 1$ refers to the entropy of the input sequences. Hunt and Szymanski have proposed an $O(R\log n+n)$ time algorithm for the LCS problem in [22], where R is the number of ordered pairs of positions where the two input sequences match. Recently, Rahman and

* Corresponding author at: AℓEDA Group, Department of Computer Science and Engineering (CSE), Bangladesh University of Engineering and Technology (BUET), Dhaka-1000, Bangladesh.

E-mail addresses: effat34@gmail.com (E. Farhana), msrahman@cse.buet.ac.bd (M.S. Rahman).

[1] On a sabbatical leave from CSE, BUET.

http://dx.doi.org/10.1016/j.ins.2014.10.019
0020-0255/© 2014 Elsevier Inc. All rights reserved.

Iliopoulos have presented an improved algorithm that runs in $O(R \log\log n + n)$ time [24]. Although $R = O(n^2)$, there are large number of applications for which $R \sim n$ [22]. Apart from algorithmic results, the LCS problem has been investigated from combinatorial point of view as well (e.g., [32]). A number of attempts have also been made to obtain reductions from the LCS problem to other known problems (e.g., LCS to PSAT [16], LCS to SAT and 3SAT [17]).

For arbitrary number of sequences, the problem of computing an LCS is NP-hard [28]. As a result a number of heuristic and/or meta-heuristic techniques have also been applied in the literature to solve the LCS problem for arbitrary number of sequences (e.g., [7,31,34,35]). The related problem of global alignment, which is more relevant in computational biology, has also received much attention in the literature. For a very recent work on global alignment the readers are referred to [8].

It was noticed that direct application of the LCS measure to the field of computational and molecular biology can lead to biologically insignificant results. This is because it treats the sequences as strings of letters only and users (biologist) can not incorporate their knowledge of the functionalities or structures of the input sequences while using this as a similarity measure. This naturally triggered the study of variants of the LCS problem that allow users to provide additional (including/excluding) constraints. Tang et al. [36] and Tsai [38] have investigated the variant where an additional sequence called theconstraint pattern is given with the input. This problem, namely, the Constrained Longest Common Subsequence (CLCS) problem, imposes the constraint that the computed LCS of the two input sequences must contain the constraint pattern (i.e., the third input sequence) as a subsequence. Below we formally define this problem and some interesting variants thereof under a general framework.

Problem "GC-LCS(L,ℓ)": We are given L input sequences $S_1, S_2, \ldots, S_L$ and ℓ constraint patterns $P_1, P_2, \ldots, P_\ell$ over alphabet Σ where $L \geqslant 2$, $\ell \geqslant 1$, $|\Sigma| \geqslant 2$ and for all $i \in \{1,2,\ldots,L\}$ and all $j \in \{1,\ldots,\ell\}$, $n = \max\{|S_i|\}$, $|P_j| \leqslant S_i$. We have the following problem variants.

Problem 1. (*STR-IC-LCS*): *Compute an LCS of the input sequences* $S_i, i \in \{1,\ldots,L\}$ *including* P_j, $j \in \{1,\ldots,\ell\}$ *each as a substring.*

Problem 2. (*SEQ-IC-LCS*): *Compute an LCS of the input sequences* $S_i, i \in \{1,\ldots,L\}$ *including* $P_j, j \in \{1,\ldots,\ell\}$ *each as a subsequence.*

Problem 3. (*STR-EC-LCS*): *Compute an LCS of the input sequences* $S_i, i \in \{1,\ldots..,L\}$ *excluding* $P_j, j \in \{1,\ldots,\ell\}$ *each as a substring.*

Problem 4. (*SEQ-EC-LCS*): *Compute an LCS of the input sequences* $S_i, i \in \{1,\ldots..,L\}$ *excluding* $P_j, j \in \{1,\ldots,\ell\}$ *each as a subsequence.*

Clearly, under this framework, GC-LCS(2, 1)-SEQ-IC-LCS refers to the CLCS problem introduced and investigated in [36,38]. Notably, the GC-LCS(2, 1)-SEQ-IC-LCS problem (i.e., CLCS problem) has received significant attention in the literature from different perspectives (dynamic programming based approaches [5,9,11,23], bit parallel implementations [33], indeterminate/degenerate string versions [25,26], etc.). In what follows, for the sake of brevity, we will use SEQ-IC-LCS(L,ℓ) to refer to the problem GC-LCS(L,ℓ)-SEQ-IC-LCS. We will use a similar shorthand notation for the other three variants as well. Notably, the other three variants of GC-LCS(2, 1) (i.e., SEQ-EC-LCS(2, 1), STR-IC-LCS(2, 1) and STR-EC-LCS(2, 1)) were first studied by Chen and Chao [9]. In fact, they solved all four variants (for $L = 2$ and $\ell = 1$) in $O(|S_1||S_2||P_1|)$ time. Very recently, two subcubic time algorithms for the SEQ-EC-LCS(2, 1) problem have been proposed by Deorowicz et al. [14]. Gotthilf et al. [19] have given a polynomial-time algorithm that approximates SEQ-IC-LCS$(k,1)$ within a factor of $\sqrt{m|\Sigma|}$, where $m = \min_{1 \leqslant i \leqslant L}(|S_i|)$. Gotthilf et al. [18] also have studied the SEQ-EC-LCS(L,ℓ) problem. Very recently, Wu et al. [43] have proposed a dynamic programming solution for the STR-EC-LCS$(2,\ell)$ problem. A variant of the STR-IC-LCS$(2,\ell)$ problem has recently been proposed by Tseng et al. [39]. In this variant, referred to as the Sequential Substring Constrained Longest Common Subsequence Problem, the goal is to find an LCS of the two given strings such that the computed LCS contains the ordered sequence of constraint patterns $P_1, P_2, \ldots, P_\ell, P_1$ as substrings with the order of the sequence retained. Notably, the recent work of Deorowicz et al. [14] and Wu et al. [43] have come to our attention during the final stage of the review and publication process of this paper.

1.1. Our contributions

In this paper, finite automata based efficient algorithms are presented for all the variants of GC-LCS(L,ℓ) problem that run in $O(|\Sigma|(\mathcal{R} + L) + nL + |\Sigma|\mathcal{R}n^\ell)$ time, where $\mathcal{R}$ is the size of the resulting subsequence automaton. We also carry out extensive experiments to evaluate the practical performance of these algorithms. Additionally, a performance comparison between the automata based algorithm for SEQ-EC-LCS(L,ℓ) and the Dynamic Programming (DP) based algorithm of Gotthilf et al. [18] for the same problem has also been presented.

To the best of our knowledge, this is the first attempt to solve the other three variants namely, SEQ-IC-LCS(L,ℓ), STR-EC-LCS(L,ℓ) and STR-IC-LCS(L,ℓ). Furthermore, we experimentally compare our algorithm for SEQ-IC-LCS$(2,\ell)$ with the algorithm of Chen and Chao [9] for the same problem. On the other hand, while conducting the experiments, we came to know about a very recent work of Ann et al. [3], where a flaw in Chen and Chao's algorithm for STR-EC-LCS$(2,\ell)$ was pointed out.