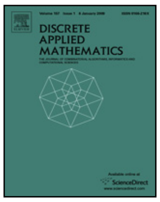
Contents lists available at ScienceDirect

Discrete Applied Mathematics

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



# Repetition-free longest common subsequence

Said S. Adi[a], Marília D.V. Braga[b], Cristina G. Fernandes[c], Carlos E. Ferreira[c], Fábio Viduani Martinez[a], Marie-France Sagot[b], Marco A. Stefanes[a], Christian Tjandraatmadja[c], Yoshiko Wakabayashi[c,*]

[a] *Universidade Federal do Mato Grosso do Sul, Brazil*
[b] *Université Claude Bernard, Lyon I, France*
[c] *Universidade de São Paulo, Brazil*

ARTICLE INFO

*Article history:*
Received 29 March 2008
Received in revised form 30 September 2008
Accepted 20 April 2009
Available online 3 June 2009

*Keywords:*
Longest common subsequence
APX-hardness
Approximation algorithms

ABSTRACT

We study the following problem. Given two sequences $x$ and $y$ over a finite alphabet, find a repetition-free longest common subsequence of $x$ and $y$. We show several algorithmic results, a computational complexity result, and we describe a preliminary experimental study based on the proposed algorithms. We also show that this problem is APX-hard.

## 1. Introduction

In the genome rearrangement domain, gene duplication is rarely considered as it usually makes the problem at hand harder. Sankoff [11] proposed the so-called exemplar model, which consists in searching, for each family of duplicated genes, an exemplar representative in each genome. In biological terms, the exemplar gene may correspond to the original copy of the gene, which later originated all other copies. Following the parsimony principle, the choices of exemplars should be made so as to minimize the reversal distance between the two simpler versions of both genomes, composed only by the exemplar genes. An alternative to the exemplar model is the multigene family model, which consists in maximizing the number of paired genes among a family. Again, the gene pairs should be chosen so as to minimize the reversal distance between the genomes. Both exemplar and multigene models were proven to lead to NP-hard problems [4,6].

To compare two sequences, we propose a similarity measure that takes into account the concept of exemplar genes. The measure we propose is the length of a repetition-free *longest common subsequence* (LCS) between the two sequences. The concept behind the exemplar model is captured by the repetition-free requirement in the sense that at most one representative of each family of duplicated genes is taken into account. The length of an LCS is a measure of similarity between sequences, so the length of a repetition-free LCS can be seen as the edit distance between two sequences where only deletions are allowed and, furthermore, for each family with $k$ duplicated genes, at least $k - 1$ of them must be deleted.

An *alphabet* is a finite set and we refer to each of its elements as a *symbol*. All sequences considered in this paper are finite and over some alphabet usually implicit, as it may be considered to be the set of all symbols appearing in the involved sequences. For a sequence $w$, we use $|w|$ to denote its length. The problem we are interested, denoted by RFLCS, consists

* Corresponding address: Universidade de Sao Paulo, Rua do Matao, 1010 Cidade Universitaria, 05508-090 Sao Paulo, Brazil. Tel.: +55 1130916135 ; fax: +55 1130916134.
*E-mail address:* YWakabayashi@gmail.com (Y. Wakabayashi).

0166-218X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2009.04.023

of the following: given two sequences $x$ and $y$, find a repetition-free LCS of $x$ and $y$. We write RFLCS $(x, y)$ when we refer to RFLCS for a generic instance consisting of a pair $(x, y)$. We denote by $\text{opt}(\text{RFLCS}(x, y))$ the length of an optimal solution of RFLCS $(x, y)$.

Bonizzoni et al. [5] considered some variants of the RFLCS. Among others, they considered the case where some symbols are required to appear in the sought LCS, and possibly more than once. They showed that these variants are APX-hard and that, in some cases, it is NP-complete just to decide whether an instance of the variants is feasible. This second complexity result makes these variants less tractable.

We present some algorithmic and some hardness results for the RFLCS. We also report on some computational experiments with the algorithms proposed in this paper. We start by showing, in Section 2, some polynomial cases and three approximation algorithms for RFLCS. We describe $c$-approximations for the case where each symbol appears at most $c$ times in at least one of the sequences. In Section 3, we prove that RFLCS is APX-hard even when each symbol appears at most twice in both sequences. Section 4 presents an integer linear programming formulation (IP) for RFLCS. Finally, in Section 5, we show some computational results we obtained for RFLCS, considering the three approximation algorithms and the use of an IP solver for the formulation presented in Section 4 for finding optimal solutions of the instances.

An extended abstract of this paper was presented at LAGOS 2007 (IV Latin-American Algorithms, Graphs, and Optimization Symposium) [1].

## 2. Algorithmic results

We first mention some polynomially solvable cases of RFLCS $(x, y)$. If each symbol appears at most once either in $x$ or in $y$ then the problem is easy: it is enough to find an LCS of $x$ and $y$. In this case, any LCS has no repetition and is therefore a solution of RFLCS $(x, y)$. There are polynomial algorithms for LCS, so this case is polynomially solvable (see [7]).

For each symbol $a$ and a sequence $w$, let $n(w, a)$ be the number of appearances of $a$ in $w$. Let $m_a(x, y) = \min\{n(x, a), n(y, a)\}$. The case above is the one in which $m_a(x, y) \leq 1$ for all $a$. Consider the slightly more general case in which there is a constant bound $k$ on the number of symbols $a$ for which $m_a(x, y) > 1$. This case is also polynomially solvable. Indeed, let $A_x$ be the set of symbols for which $m_a(x, y) = n(x, a)$, and $A_y$ be the remaining symbols. Try each subsequence $x'$ of $x$ and each subsequence $y'$ of $y$ obtained in the following way. For each symbol $a$ in $A_x$ and each of the $m_a(x, y)$ occurrences of $a$ in $x$, keep that occurrence and delete all the others from $x$, obtaining one $x'$. Do the same for $y$, obtaining one $y'$. For each $x'$ and $y'$, find an LCS of $x'$ and $y'$. Return a longest one among all obtained LCS s. This method needs to solve $O(n^k)$ different LCS instances and therefore is polynomial.

Now we describe three simple approximation algorithms for the problem: A1, A2, and A3. Algorithm A1 consists of the following: given $x$ and $y$, compute an LCS of $x$ and $y$ and remove all repeated symbols but one, in the obtained LCS. Return the resulting sequence. Let $m$ be the maximum value of $m_a(x, y)$ taken over all $a$. It is not hard to see that Algorithm A1 is an $m$-approximation for RFLCS $(x, y)$.

Algorithm A2 is probabilistic. It consists of the following: given $x$ and $y$, for each symbol $a$, if $m_a(x, y) = n(x, a)$, pick uniformly at random one of the $m_a(x, y)$ occurrences of $a$ in $x$, and delete all the others from $x$; if $m_a(x, y) \neq n(x, a)$, pick uniformly at random one of the $m_a(x, y)$ occurrences of $a$ in $y$, and delete all the others from $y$. Let $x'$ and $y'$ be the resulting sequences after this clean-up. Compute an LCS $w'$ of $x'$ and $y'$ and return $w'$.

Algorithm A3 is a variant of Algorithm A2 that uses less random bits. It works basically as Algorithm A2 in the sense that, for each repeated symbol $a$, the sequence (either $x$ or $y$) that is chosen to keep an occurrence of $a$ is the one with the least number of repetitions of $a$. The difference now is that A3 picks uniformly at random only one number, say $r$, in the interval $[0, 1]$ and uses $r$ to select which occurrence of each symbol will remain. For each repeated symbol, the selection is done using the same number $r$. (If a symbol $a$ occurs $k$ times in the chosen sequence, we partition the interval $[0, 1]$ into $k$ subintervals of the same size, each one corresponding to an occurrence of $a$, and then we keep the occurrence of $a$ that corresponds to the subinterval that contains $r$.)

### 2.1. Analysis of the probabilistic algorithms

Algorithms A2 and A3 are obviously polynomial, so we concentrate on their approximation ratio.

Let $A$ denote an arbitrary probabilistic algorithm for RFLCS $(x, y)$ and let $A(x, y)$ represent the (probabilistic) output of $A$ when given $x$ and $y$ as input. For a positive number $\alpha$, we say $A$ is an $\alpha$-approximation for RFLCS $(x, y)$ if $\mathbf{E}[|A(x, y)|] \geq \text{opt}(\text{RFLCS}(x, y))/\alpha$, for all $x$ and $y$. The number $\alpha$ may depend on $x$ and $y$.

**Theorem 1.** *Algorithm* A2 *is an $m$-approximation for* RFLCS $(x, y)$, *where $m$ is the maximum of $m_a(x, y)$, over all symbols $a$.*

**Proof.** For a subsequence $z'$ of a sequence $z$, denote by $I(z', z)$ a set of indices of $z$ that corresponds to an occurrence of $z'$ in $z$ (an arbitrary one, if there is more than one). For instance, if $z' = abc$ and $z = cacbbac$, the set $\{2, 4, 7\}$ corresponds to an occurrence of $z'$ in $z$. On the other hand, the set $\{2, 3, 5\}$ does not correspond to an occurrence of $z'$ in $z$.

Fix $x$, $y$, and a repetition-free LCS $w$ of $x$ and $y$. Recall that $x'$ and $y'$ are, respectively, the sequences $x$ and $y$ after the random clean-up in Algorithm A2, and $w' = \text{A2}(x, y)$. Next we define a random variable $Z$ that is the length of a common subsequence of $x'$ and $y'$ and therefore is a lower bound on $|w'|$.