Information Processing Letters 130 (2018) 46–51

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Two-string consensus problem under non-overlapping inversion and transposition distance

Toan Thang Ta, Cheng-Yao Lin, Chin Lung Lu *

Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan

ARTICLE INFO

Article history:
Received 7 November 2016
Accepted 22 October 2017
Communicated by R. Uehara

Keywords:
Algorithms
Computational biology
Two-string consensus problem
Inversion
Transposition

ABSTRACT

For biological sequences that can be represented as strings over a finite alphabet, inversion and transposition are commonly observed mutation operations. The non-overlapping inversion and transposition distance (also simply called mutation distance) between two strings is defined as the minimum number of non-overlapping inversion and transposition operations used to transform one string into the other. Given two strings of the same length n and a constant $c \geq 0$, the two-string consensus problem under mutation distance is to determine whether or not there exists a string s^* such that the mutation distance from s^* to each input string does not exceed c. In this study, we present an $O(n^5)$ time and $O(n^4)$ space algorithm to solve this problem.

1. Introduction

Given a set of strings $S = \{s_1, s_2, \ldots, s_k\}$ and a constant c, the *consensus* (*string*) *problem* is to find, if it exists, a string s^* such that the distance of s^* from each string of S does not exceed c, where s^* is then called the *consensus string* of S. This problem is a fundamental problem in computer science and has also many applications in web searching, computational geometry and computational biology [1,5,8]. In fact, finding the consensus of a given set of strings is a hard and challenging problem. Amir et al. [1] have showed that the consensus problem is NP-hard for the swap metric and APX-hard for the reversal metric. Under the Hamming distance, Frances and Litman [4] have proved that the consensus problem is NP-hard even when the input strings are binary.

In biology, DNA sequences are subject to be changed by genome rearrangements (i.e., genetic mutations at the level of sequence fragments) during their evolutionary process, such as inversion and transposition [6]. An *inversion* operation substitutes a segment of the DNA sequence by its reverse complement sequence. A *transposition* (also called *translocation* in some literature, e.g., [3]) operation moves a fragment of the DNA sequence from one location to another or, equivalently, exchanges two adjacent and non-overlapping fragments on the DNA sequence. The DNA sequences changed by the genome rearrangements can be inherited to subsequent generations, resulting in new DNA sequences descended from an ancestral sequence. Basically, the DNA sequences descended from a common ancestor can be different because they may evolve with different genetic mutations. Recently, Cho et al. [2] proposed the following problem to study whether or not two DNA sequences are mutated from a common ancestral sequence: given two strings (or sequences), find a *common ancestor* (described as an *alignment* in [2]) for two input strings using non-overlapping inversions. In fact, this problem can be considered as a relaxed version of the consensus problem on two strings under non-overlapping inversion distance, where the constant c is not present as a parameter or, equivalently, $c = \infty$. For this problem, Cho et al. [2] also provided an $O(n^3)$ time algorithm using $O(n^2)$ space, where n is the size of the two input strings. Later, Zohora

* Corresponding author.
E-mail addresses: toanthanghy@gmail.com (T.T. Ta), begoingto0830@gmail.com (C.-Y. Lin), cllu@cs.nthu.edu.tw (C.L. Lu).

https://doi.org/10.1016/j.ipl.2017.10.006
0020-0190/

and Rahman [9] gave a counterexample to show that the algorithm proposed by Cho et al. [2] is incorrect for some instances. They also designed an algorithm, whose worst-case and average-case time complexities are claimed to be $O(n^4)$ and $O(n^3)$, respectively, to determine the existence of the consensus string under the non-overlapping inversion distance. Very recently, Cho et al. [3] extended their algorithm in [2] to deal with reversals and transpositions, where a *reversal* operation changes a substring by reversing the order of its characters. Note that the transposition used in [3] is restricted to exchange two adjacent and non-overlapping substrings of the same length. The algorithm in [3] has the same time and space complexities as the one from [2]. However, we found that the algorithm proposed by Cho et al. [3] is still incorrect with the counterexample given by Zohora and Rahman in [9].

To address the shortcoming and limitations in the algorithm proposed by Cho et al. [3], we study the consensus problem on two strings (also called *two-string consensus problem*) using non-overlapping inversion and transposition distance in this work. As a result, we devise an algorithm to determine the existence of a consensus string for two input strings in $O(n^5)$ time and $O(n^4)$ space in the worst case. It should be noted that in our study, the threshold c is required as a part of the problem's input, inversion is the reverse and complement of a string, and the lengths of two adjacent and non-overlapping substrings exchanged by a transposition operation can be different (not necessarily equivalent). The rest of the paper is organized as follows. In Section 2, we give the formal definition of the problem. Subsequently, we discuss the main algorithm and its complexity analysis in Section 3. Finally, we have a brief conclusion in Section 4.

2. Preliminaries

Below, we introduce some notations and terminology from [7]. Let z be a string of length n over a finite alphabet Σ. A character at position i of z is represented with z_i (or $z[i]$), where $1 \le i \le n$. A substring of z from position i to j is indicated as $z_{i,j}$, i.e., $z_{i,j} = z_i z_{i+1} \dots z_j$, for $1 \le i \le j \le n$. For biological sequences, $\Sigma = \{A, C, G, T\}$, in which A-T and C-G are considered as *complementary base pairs*. We use $\theta(z)$ to denote an *inversion operation* acting on z, resulting in a reverse and complement of z. For example, $\theta(A) = T$, $\theta(T) = A$, $\theta(G) = C$, $\theta(C) = G$ and $\theta(CGA) = TCG$. In addition, we utilize $\tau(uv) = vu$ to represent a *transposition operation* to exchange two non-empty strings u and v. Note that the lengths of u and v are may be different in this study. For convenience, we call θ and τ as *mutation operations*. We also let $\theta_{i,j}(z) = \theta(z_{i,j})$ for $1 \le i \le j \le n$ and $\tau_{i,j,k}(z) = z_{k,j} z_{i,k-1}$ for $1 \le i < k \le j \le n$, where $[i, j]$ is called a *mutation range* for $\theta_{i,j}$ and $\tau_{i,j,k}$. For an integer t, where $1 \le t \le n$, we say that a mutation operation $\theta_{i,j}$ or $\tau_{i,j,k}$ *covers* t if $i \le t \le j$. Given two mutation operations, they are *non-overlapping* if the intersection of their mutation ranges is empty. Given a set Θ of non-overlapping mutation operations and a string z, let $\Theta(z)$ be the resulting string after consecutively applying mutation operations in Θ on z. For example, if $\Theta = \{\tau_{1,3,2}, \theta_{5,5}\}$ and $z = TAGAC$, then $\Theta(z) = AGTAG$.

Definition 1 *(Non-overlapping inversion and transposition distance).* Given two strings x and y of the same length, the non-overlapping inversion and transposition distance (simply called *mutation distance*) between x and y, denoted by $md(x, y)$, is defined as the minimum number of non-overlapping inversion and transposition operations used to transform x into y. If there does not exist any set of non-overlapping mutations that converts x into y, then $md(x, y)$ is infinite. Formally,

$$md(x, y) = \begin{cases} \min\{|\Theta| : \Theta(x) = y\} & \text{if } \exists\Theta \text{ such that } \Theta(x) = y \\ \infty & \text{otherwise} \end{cases}$$

Definition 2 *(Two-string consensus problem under mutation distance).*
Input: Two strings x and y of the same length n and a constant $c \ge 0$.
Question: Does there exist a string s^* such that $md(s^*, x) \le c$ and $md(s^*, y) \le c$?

For example, suppose that $x = TAGAC$ and $y = TAACG$. When $c = 1$, the answer is positive because there exists $s^* = y$ such that $md(s^*, x) = 1$ and $md(s^*, y) = 0$. However, when $c = 0$, the answer is negative, since $x \ne y$ and hence there does not exist any s^* satisfying both $md(s^*, x) = 0$ and $md(s^*, y) = 0$.

3. Our algorithm

In fact, a mutation acting on a string z comprises several sub-operations, called *mutation fragments*, each of which is denoted either by a tuple (i, j, z_j) or $(i, j, \theta(z_j))$, where (i, j, z_j) (respectively, $(i, j, \theta(z_j))$) means that z_j (respectively, complement of z_j) is moved into the position i in the resulting string obtained when applying the mutation operation on z. For convenience, we arrange all possible mutation fragments for z in two *mutation tables* M_1^z and M_2^z, where $M_1^z[i, j] = (i, j, \theta(z_j))$ and $M_2^z[i, j] = (i, j, z_j)$ for all $1 \le i, j \le n$. For example, if $z = TAGAC$, then its mutation tables are shown in Fig. 1.

Given a string z and its mutation tables M_1^z and M_2^z, the result of an inversion $\theta_{i,j}(z)$ can be obtained by concatenating $(j - i + 1)$ mutation fragments starting at $M_1^z[i, j]$ and continuing to move in the anti-diagonal direction to $M_1^z[j, i]$ (see the shaded entries of M_1^z in Fig. 1 for an example). That is, we can decompose $\theta_{i,j}(z)$ into $(j - i + 1)$ mutation fragments $\mathcal{F}(\theta_{i,j}, z, t)$ for $i \le t \le j$, where $\mathcal{F}(\theta_{i,j}, z, t) = (t, i + j - t, \theta(z_{i+j-t}))$. In addition, the result of a transposition $\tau_{i,j,k}(z)$ can be obtained by concatenating the mutation fragments on the following two paths, one starting at $M_2^z[i, k]$ and continuing to move in the diagonal direction to $M_2^z[i + j - k, j]$ and the other beginning at $M_2^z[i + j - k + 1, i]$ and also continuing to move in the diagonal direction to $M_2^z[j, k - 1]$ (see the shaded entries of M_2^z in Fig. 1 for an example). More clearly, $\tau_{i,j,k}(z)$ can be decomposed into $(j - i + 1)$ mutation fragments $\mathcal{F}(\tau_{i,j,k}, z, t)$ for $i \le t \le j$, where if $i \le t \le i + j - k$, then $\mathcal{F}(\tau_{i,j,k}, z, t) = (t, k + t - i, z_{k+t-i})$; otherwise, $\mathcal{F}(\tau_{i,j,k}, z, t) = (t, t - j + k - 1, z_{t-j+k-1})$. For the purpose of brevity, we further let $\tau_{i,j,k}(z, 1) = \{(t, k + t - i,$