Finding Diverse Strings and Longest Common Subsequences in a Graph

Luca Lombardo

Seminar for the course of Bioinformatics

Structure of the Presentation

Longest Common Subsequence (LCS)

Definition

Given a set of m strings $S = \{S_1, S_2, \ldots, S_m\}$, a **common subsequence** (CS) is a sequence that appears in all m strings. A **longest common subsequence** LCS is a common subsequence of maximum length. We denote the set of all LCSs of S as $LCS(S)$.

Goal

The goal is to find a diverse set of solutions to the LCS problem under the Hamming distance.

Definition

Given two strings $X, Y \in \Sigma^r$, the **Hamming distance** between X and Y, denoted with $d_H(X, Y)$, is the number of positions at which the corresponding symbols differ.

A simple example

Add here the example of the paper

Efficient methods for finding a diverse set of solutions

More formally, let's consider the following two diversity measures for a multiset $\mathcal{X} = \{X_1, X_2, \ldots, X_K\} \subseteq \Sigma^r$ of solutions, allowing repetitions:

$$D_{d_H}^{\text{sum}}(\mathcal{X}) = \sum_{1 \leq i < j \leq K} d_H(X_i, X_j) \qquad \text{Max-Sum Diversity} \qquad (1)$$

$$D_{d_H}^{\text{min}}(\mathcal{X}) = \min_{i<j} d_H(X_i, X_j) \qquad \text{Max-Min Diversity} \qquad (2)$$

Notation

A subset $\mathcal{X} \subseteq \Sigma^r$ is Δ-diverse w.r.t. $D_{d_H}^{\tau}$ if $D_{d_H}^{\tau}(\mathcal{X}) \geq \Delta$ for some $\Delta \geq 0$.

Where for $\tau \in \{sum, min\}$, $D_{d_H}^{\tau}$ denotes one of the two diversity measures.

Two problems

Problem 1: Diverse LCSs with Diversity measure $D^{\tau}_{d_H}$

Input: A set $S = \{S_1, S_2, \ldots, S_m\}$ of $m \geq 2$ strings over Σ, an integer $K \geq 1$ and $\Delta \geq 0$.
Question: Is there some set $\mathcal{X} \subseteq LCS(S)$ such that $\mathcal{X} = K$ and $D^{\tau}_{d_H}(\mathcal{X}) \geq \Delta$?

Problem 2: Diverse String Set

Input: $K, r, \Delta \in \mathbb{Z}$ and a Σ-DAG G for a set $L(G) \subseteq \Sigma^r$ of strings.
Question: Decide if there exists some subset $\mathcal{X} \subseteq L(G)$ such that $|\mathcal{X}| = K$ and $D^{\tau}_{d_H}(\mathcal{X}) \geq \Delta$.

Σ-Labeled Directed Acyclic Graphs (Σ-DAGs)

Definitions

- **Alphabet (Σ)**: Set of symbols.
- **String Set (Language)**: $L = \{X_1, X_2, \ldots, X_n\} \subseteq \Sigma^*$, with:
  - **Total Length**: $||L|| = \sum_{X \in L} |X|$
  - **Max Length**: $\text{maxLen}(L) = \max_{X \in L} |X|$
- **r-String**: Any string X where $|X| = r$

Σ-DAG Structure

A graph $G = (V, E, s, t)$ with:

- V: vertices, E: labeled edges (v, c, w) with $c \in \Sigma$
- **Source** s and **Sink** t such that paths exist from s to all vertices
- **Size**: $\text{size}(G)$, the number of its labeled edges

Σ-Labeled Directed Acyclic Graphs (Σ-DAGs)

Any path $P = (e_1, e_2, \ldots, e_n)$ of outgoing edges *spells out* a string $\text{str}(P) = c_1 c_2 \ldots c_n \in \Sigma^n$ where c_i is the label of edge e_i.

Language Representation

A Σ-DAG represents $L(G) \subset \Sigma^*$: all strings spelled from paths $s \to t$. Equivalent to an NFA over Σ with initial s, final t, and no ϵ-edges.

Remarks

- For any set L of strings, a Σ-DAG G exists s.t. $L(G) = L$ and $\text{size}(G) \leq ||L||$. Construction time: $O(||L|| \log |\Sigma|)$.
- If G represents a set L of r-strings ($L \subseteq \Sigma^r, r \geq 0$), all paths $s \to v$ have the same length $d \leq r$.

Σ-DAGs for LCSs and Diverse String Sets

Lemma (Σ-DAG for LCSs)

For any constant $m \geq 1$ and set $S = \{S_1, \ldots, S_m\} \subseteq \Sigma^*$ of m strings, there exists a Σ-DAG G of polynomial size in $\ell := \text{maxlen}(S)$ such that $L(G) = \text{LCS}(S)$ and can be computed in polynomial time I

Consequence of the Lemma

- **If** Max-Min (or Max-Sum) Diverse String Set solvable in $f(M, K, r, \Delta)$,
- **Then** Max-Min (or Max-Sum) Diverse LCSs on $S \subseteq \Sigma^r$ solvable in $O(|\Sigma| \cdot \ell^m + f(\ell^m, K, r, \Delta))$ time.

where $\ell = \text{maxlen}(S)$

Algorithm for Max-Min Diverse String Set problem

Dynamic Programming Approach

Pattern of Path Tuple: For each $d \leq r$ and K-tuple of length-d paths $P = (P_1, \ldots, P_K)$, define pattern $\text{Pattern}(P) = (\mathbf{w}, \mathbf{Z})$, where:

- $\mathbf{w} = (w_1, \ldots, w_K)$: K-tuple of vertices representing endpoints of P_i paths.
- $\mathbf{Z} = (Z_{i,j})$: Upper triangular matrix of Hamming distances, where $Z_{i,j} = \min\{\Delta, d_H(\text{str}(P_i), \text{str}(P_j))\}$.

DP Table of Weights

$$\text{Weights} : V^K \times (\Delta \cup \{0\})^{K \times K} \to \{0, 1\} \tag{3}$$

Boolean matrix where $\text{Weights}(\mathbf{w}, \mathbf{Z}) = 1$ iff $(\mathbf{w}, \mathbf{Z})$ matches pattern $\text{Pattern}(P)$ for some K-tuple of paths of length d from s to $\mathbf{w}$.

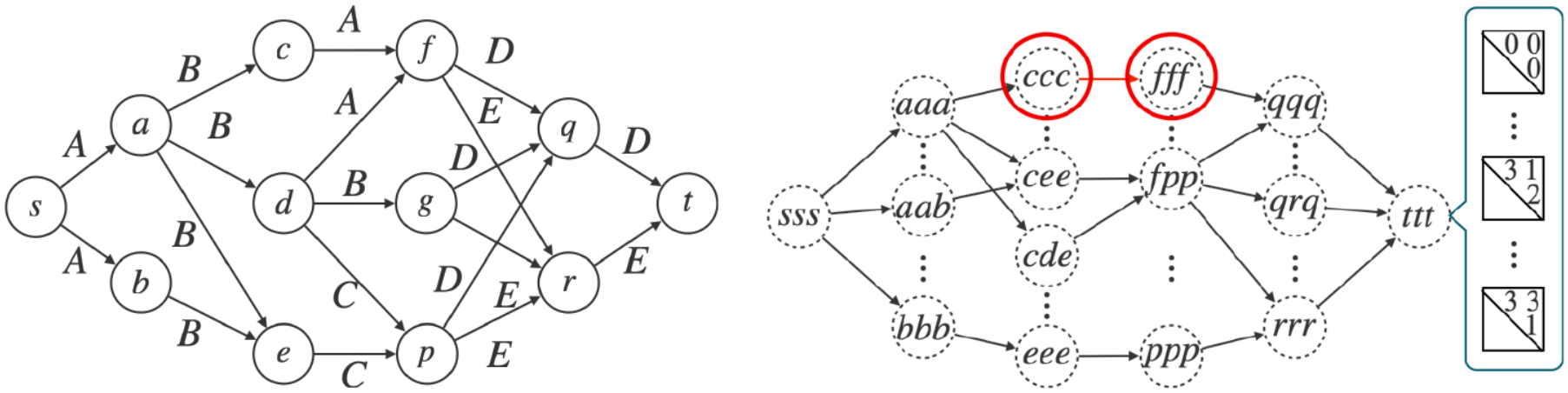
Algorithm for Max-Min Diverse String Set problem

Algorithm 1

Set $\texttt{Weights}(\mathbf{s}, Z) = 0$ for all $Z \in (\Delta \cup \{0\})^{K \times K}$ and $\texttt{Weights}(\mathbf{s}, \mathbf{0}) \leftarrow 1$
for $d \leftarrow 1, \ldots, r$ **do**
  for $\mathbf{v} \leftarrow (v_1, \ldots v_K) \in (V_d)^K$ **do**
    for $(v_1, c_1, w_1) \in E^+(v_1), \ldots, (v_K, c_K, w_K) \in E^+(v_K)$ **do**
      Set $\mathbf{w} = (w_1, \ldots, w_K)$
      for $U \in (\Delta \cup \{0\})^{K \times K}$ such that $\texttt{Weights}(\mathbf{v}, U) = 1$ **do**
        Set $Z = (Z_{i,j})_{i<j}$ with $Z_{i,j} \leftarrow \min\{\Delta, U_{i,j} + \mathbb{1}\{c_i \neq c_j\}\} \quad \forall\, i, j \in K$
        Set $\texttt{Weights}(\mathbf{w}, Z) \leftarrow 1$ ▷ Update
      end for
    end for
  end for
end for
Answer `Yes` if $\texttt{Weights}(\mathbf{t}, Z) = 1$ and $D_{d_H}^{\min}(Z) \geq \Delta$ for some Z, else `No`

For any $K \geq 1$ and $\Delta \geq 0$, it solves the Max-Min Diverse String Set problem in $O(\Delta^{K^2} K^2 M^K (\log |V| + \log \Delta))$ time and space when an input string set L is represented by a Σ-DAG G with size$(G) = M$

Example Run of the Algorithm



(a) An input Σ-DAG G_1 for $LCS(X_1, Y_1)$ **(b)** Example run of Algorithm 1 on G_1

Figure: (a) An input Σ-DAG G_1 over $\Sigma = \{A, B, C, D, E\}$ for the set of all longest common subsequences of two strings $X_1 =$ ABABCDDEE and $Y_1 =$ ABCBAEEDD and (b) an example run of Algorithm 1 based on dynamic programming with $K = 3$ on an input G_1

Algorithm for Max-Sum Diverse String Set problem

Algorithm for Max-Sum Diverse String Set problem

Max-Min Diverse String Set problem

Max-Sum Farthest r-String problem

FPT Algorithms for Bounded Number and Length of Diverse Strings

FPT Algorithms for Bounded Number and Length of Diverse Strings

Hardness of Diverse String Set for Unbounded K

Hardness of Diverse String LCSs for Unbounded K