Finding Diverse Strings and Longest Common Sequences in a Graph

Seminar for the course of Bioinformatics

Luca Lombardo

Based on the work of: Yuto Shida, Giulia Punzi, Yasuaki Kobayashi, Takeaki Uno, Hiroki Arimura

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.

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:

$$D^{\text{sum}}_{d_H}(\mathcal{X}) = \sum_{1 \le i < j \le K} d_H(X_i, X_j) \qquad \textsc{Max-Sum Diversity} \tag{1}$$

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

Notation

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

Where for $\tau \in \{sum, min\}$, $D^{\tau}_{d_H}$ 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**: $\mathrm{maxLen}(L) = \max_{X \in L} |X|$
- **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**: $\mathrm{size}(G)$, 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 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 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 $\ell :=$ maxlen(S) such that $L(G) =$ LCS(S) and can be computed in polynomial time l

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 $\ell =$ maxlen(S)

Exact Algorithms for Bounded Number of Diverse Strings

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 $\mathrm{Pattern}(P) = (\mathbf{w}, \mathbf{Z})$, 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 $Z_{i,j} = \min\{\Delta, d_H(\mathrm{str}(P_i), \mathrm{str}(P_j))\}$.

DP Table of Weights

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

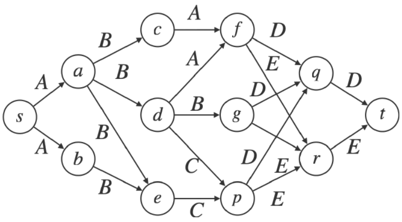
Boolean matrix where $\mathrm{Weights}(\mathbf{w}, \mathbf{Z}) = 1$ iff $(\mathbf{w}, \mathbf{Z})$ matches pattern $\mathrm{Pattern}(P)$ for some K-tuple of paths of length d from s to $\mathbf{w}$.

Algorithm 1

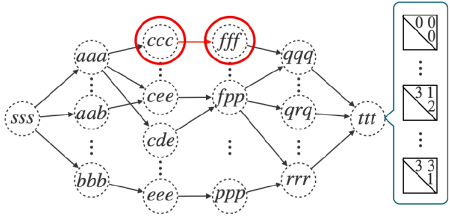
Set $\texttt{Weights}(\mathbf{s}, Z) = 0$ for all $Z \in (\Delta \cup \{0\})^{K \times K}$ and $\texttt{Weights}(\mathbf{s}, \mathbf{0}) \leftarrow 1$
for $d \leftarrow 1, \dots, r$ **do**
  for $\mathbf{v} \leftarrow (v_1, \dots v_K) \in (V_d)^K$ **do**
    for $(v_1, c_1, w_1) \in E^+(v_1), \dots, (v_K, c_K, w_K) \in E^+(v_K)$ **do**
      Set $\mathbf{w} = (w_1, \dots, w_K)$
      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 $\texttt{Yes}$ if $\texttt{Weights}(\mathbf{t}, Z) = 1$ and $D^{\min}_{d_H}(Z) \geq \Delta$ for some Z, else $\texttt{No}$

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 $\mathrm{size}(G) = M$

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

Modify the Max-Sum Diverse String Set problem

Instead of the entire $K \times K$ weight matrix Z, only the sum $z = \sum_{i<j} d_H(\mathrm{str}(P_i), \mathrm{str}(P_j))$ is needed for computing Max-Sum diversity.

New DP Table Weights

For $\mathbf{w} = (w_1, \ldots, w_K)$ of depth $0 \leq d \leq r$ and integer $0 \leq z \leq rK$, define:

$$\texttt{Weights}(\mathbf{w}, z) = 1$$

if and only if there exists a K-tuple of length-d prefix paths $(P_1, \ldots, P_K)$ from s to $w_1, \ldots, w_K$ with sum of pairwise Hamming distances z.

Algorithm for Max-Sum Diverse String Set problem

Algorithm 2

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 $u \leftarrow (0, \ldots, rK)$ such that $\texttt{Weights}(\mathbf{v}, U) = 1$ **do**
                Set $Z = (Z_{i,j})_{i<j}$ with $Z_{i,j} \leftarrow \min\{\Delta, u + \sum_{i<j} \mathbb{1}\{c_i \neq c_j\}\}$
                Set $\texttt{Weights}(\mathbf{w}, Z) \leftarrow 1$ ▷ Update
            end for
        end for
    end for
end for
Answer $\texttt{Yes}$ if $\texttt{Weights}(\mathbf{t}, Z) = 1$ and $D^{\min}_{d_H}(Z) \geq \Delta$ for some Z, else $\texttt{No}$

For any $K \geq 1$, 14 solves the Max-Sum Diverse String Set under Hamming Distance in $O(\Delta K^2 M^K (\log |V| + \log \Delta))$ time and space, where M is the size of the input Σ-DAG G

Approximation Algorithms for Unbounded Number of Diverse Strings

Max-Sum Diverse String Set problem

Use a local search algorithm for computing approximate solutions $\mathcal{X} \subseteq \mathcal{L}$ with $|\mathcal{X}| = K$ on a finite metric space $(\mathcal{L}, d)$, where $d : \mathcal{L} \times \mathcal{L} \to \mathbb{R}_{\geq 0}$.

Algorithm 3 $\texttt{LocalSearch}(\mathcal{L}, K, d)$

$\mathcal{X} \leftarrow$ arbitrary set of K solutions in $\mathcal{L}$
for $i \leftarrow 1, \ldots, \lceil \frac{K(K-1)}{K+1} \log \frac{(K+2)(K-1)^2}{4} \rceil$ **do**
  for $X \in \mathcal{X}$ s.t $\mathcal{L} \setminus \{X\} \neq \emptyset$ **do**
    $Y \leftarrow \mathrm{argmax}_{Y \in \mathcal{L} \setminus \{X\}} \sum_{X' \in \mathcal{X} \setminus \{X\}} d(X', Y)$
    $\mathcal{X} \leftarrow \mathcal{X} \setminus \{X\} \cup \{Y\}$
  end for
end for
return $\mathcal{X}$

Theorem

When the distance d is a semi-metric of negative type over $\mathcal{X}$, then LocalSearch *has improved approximation ratio $(1 - \frac{2}{K})$ for any $K \geq 2$. The Hamming distance d_H over the set of r-strings is a semi-metric of negative type.*

Max Sum Farthest r-String problem

How do we solve efficiently the following problem?

$$Y \leftarrow \text{argmax}_{Y \in \mathcal{L} \setminus \{X\}} \sum_{X' \in \mathcal{X} \setminus \{X\}} d(X', Y)$$

Algorithm 4 Decisional Max-Sum Farthest r-String

1: Set $\mathit{Weights}(s, z) := 0$ for all $z \in [\Delta]_+$, and $\mathit{Weights}(s, 0) := 1$
2: **for** $d := 1, \ldots, r$ **do**
3: **for** $0 \leq u \leq \Delta$ such that $\mathit{Weights}(v, u) := 1$ **do**
4: Set $\mathit{Weights}(w, z) := 1$ for $z := u + \sum_{i \in [K]} \mathbb{1}\{c \neq X_i[d]\}$ ▷ Update
5: **end for**
6: **end for**
7: Answer YES if $\mathit{Weights}(t, \Delta) = 1$, and NO otherwise ▷ Decide

Theorem (Polynomial Time Approximation Scheme for unbounded K)

When K is part of an input, Max-Sum Diverse String Set *problem on a Σ-DAG admits a PTAS*

Fixed-Parameter Tractable (FPT) Algorithms for Bounded Number and Length of Diverse Strings

FPT Algorithms for Max-Min and Max-Sum Diverse String Set problems

Definition (Fixed-Parameter Tractable (FPT) Algorithm)

A problem parametrized with κ is said to be *fixed-parameter tractable* (FPT) if there exists an algorithm for the problem running on an input x in time $f(\kappa(x) \cdot |x|^c)$, where f is a computable function and $c > 0$ is a constant.

Proposed FPT Algorithm

Color-coding technique with dynamic programming to solve these problems efficiently. Assign a random color to the edges of the Σ-DAG G, creating a colored graph called C-DAG, then

Computation of Reduced C-DAG

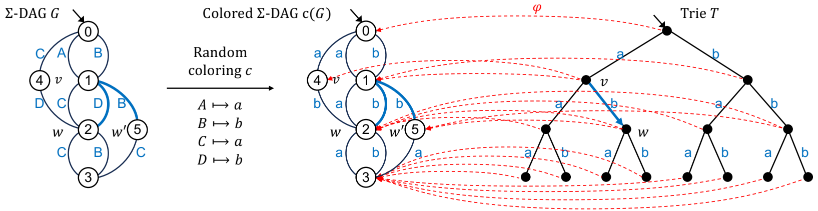


Figure: Computation of reduced C-DAG H from an input Σ-DAG G over alphabet $\Sigma = \{A, B, C, D\}$, which shows G (left), a random coloring c on $C = \{a, b\}$, a colored C-DAG $c(G)$ (middle), and a reduced C-DAG H in the form of trie T (right)

An analogous FPT algorithm can be designed for the Max-Sum Diverse String Set problem with a small modification in the research phase, that, in this case, is $O(\Delta K^2 M^K)$

Complexity Results for Diverse String Problems

Negative Results

- NP-hard for unbounded K ($\textsc{Max-Min}$, $\textsc{Max-Sum}$) in Σ-graphs for r-strings, $r \geq 3$.
- W[1]-hard parameterized with K for $\textsc{Max-Min}$ and $\textsc{Max-Sum}$ in Σ-DAGs.

Reduction to Diverse LCSs

$\textsc{Max-Min}$ and $\textsc{Max-Sum}$ problems are FPT-reducible to $\textsc{Diverse LCSs}$ for $m = 2$ strings.

Corollaries

NP-hard and W[1]-hard results extend to $\textsc{Diverse LCSs}$ for two r-strings.

Conclusion

- **Polynomial-Time Solutions**: When K is bounded, both the Max-Sum and Max-Min versions of Diverse String Set and Diverse LCSs can be solved in polynomial time using dynamic programming (DP).
- **PTAS for Input-Based** K: For input-dependent K, the Max-Sum versions of both Diverse String Set and Diverse LCSs admit a PTAS using local search due to the Hamming distance being a metric of negative type.
- **Fixed-Parameter Tractability (FPT)**: Both versions are FPT when parameterized by K and r, combining the color coding technique and DP.
- **NP-Hardness for Constant** $r \geq 3$: When K is part of the input, both the Max-Sum and Max-Min versions are NP-hard for any constant $r \geq 3$.
- **W[1]-Hard for Parameterized** K: Parameterized by K, both versions are W[1]-hard.