Finding Diverse Strings and Longest Common Subsequences in a Graph

Yuto Shida
Hokkaido University, Japan

Giulia Punzi
National Institute of Informatics, Tokyo, Japan

Yasuaki Kobayashi
Hokkaido University, Japan

Takeaki Uno
National Institute of Informatics, Tokyo, Japan

Hiroki Arimura
Hokkaido University, Japan

Abstract

In this paper, we study for the first time the *Diverse Longest Common Subsequences* (LCSs) problem under Hamming distance. Given a set of a constant number of input strings, the problem asks to decide if there exists some subset $\mathcal{X}$ of K longest common subsequences whose *diversity* is no less than a specified threshold Δ, where we consider two types of diversities of a set $\mathcal{X}$ of strings of equal length: the *Sum diversity* and the *Min diversity* defined as the sum and the minimum of the pairwise Hamming distance between any two strings in $\mathcal{X}$, respectively. We analyze the computational complexity of the respective problems with Sum- and Min-diversity measures, called the *Max-Sum* and *Max-Min Diverse LCSs*, respectively, considering both approximation algorithms and parameterized complexity. Our results are summarized as follows. When K is bounded, both problems are polynomial time solvable. In contrast, when K is unbounded, both problems become NP-hard, while Max-Sum Diverse LCSs problem admits a PTAS. Furthermore, we analyze the parameterized complexity of both problems with combinations of parameters K and r, where r is the length of the candidate strings to be selected. Importantly, all *positive results* above are proven in a more general setting, where an input is an edge-labeled directed acyclic graph (DAG) that succinctly represents a set of strings of the same length. *Negative results* are proven in the setting where an input is explicitly given as a set of strings. The latter results are equipped with an encoding such a set as the longest common subsequences of a specific input string set.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Sequence analysis, longest common subsequence, Hamming distance, dispersion, approximation algorithms, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.CPM.2024.27

Related Version *Technical Report*: `https://arxiv.org/abs/2405.00131`

Funding *Giulia Punzi*: Work supported by JSPS KAKENHI Grant Number JP20H05962.
Yasuaki Kobayashi: Supported by JSPS KAKENHI Grant Numbers JP20H00595, JP23K28034, JP24H00686, and JP24H00697.
Takeaki Uno: Supported by JSPS KAKENHI Grant Number JP20H05962.
Hiroki Arimura: Supported by JSPS KAKENHI Grant Numbers JP20H00595, JP20H05963, and JST CREST Grant Number JPMJCR18K2.

Acknowledgements The authors express sincere thanks to anonymous reviewers for their valuable comments, which significantly improved the presentation and quality of this paper. The last author would like to thank Norihito Yasuda, Tesshu Hanaka, Kazuhiro Kurita, Hirotaka Ono of AFSA project, and Shinji Ito for fruitful discussions and helpful comments.

1 Introduction

The problem of finding a *longest common subsequence* (LCS) of a set of m strings, called the LCS problem, is a fundamental problem in computer science, extensively studied in theory and applications for over fifty years [8, 31, 33, 38, 41]. In application areas such as computational biology, pattern recognition, and data compression, longest common subsequences are used for consensus pattern discovery and multiple sequence alignment [25, 41]. It is also common to use the length of longest common subsequence as a similarity measure between two strings. For example, Table 1 shows longest common subsequences (underlined) of the input strings $X_1 = ABABCDDEE$ and $Y_1 = ABCBAEEDD$.

Table 1 Longest common subsequences of two input strings X_1 and Y_1 over $\Sigma = \{A, B, C, D, E\}$.

ϵ, A, B, C, D, E, AA, AB, AC, AD, AE, BA, $\ldots$, CD, CE, DD, EE,
ABA, ABB, ABC, ABD, $\ldots$, CEE, $ABAD$, $ABAE$, $ABBD$, $\ldots$, $BCEE$,
$\underline{ABADD}$, $\underline{ABAEE}$, $\underline{ABBDD}$, $\underline{ABBEE}$, $\underline{ABCDD}$, $\underline{ABCEE}$

The LCS problem can be solved in polynomial time for constant $m \geqslant 2$ using dynamic programming by Irving and Fraser [33] requiring $O(n^m)$ time, where n is the maximum length of m strings. When m is unrestricted, LCS is NP-complete [38]. From the view of parameterized complexity, Bodlaender, Downey, Fellows, and Wareham [8] showed that the problem is W[t]-hard parameterized with m for all t, is W[2]-hard parameterized with the length ℓ of a longest common subsequence, and is W[1]-complete parameterized with ℓ and m. Bulteau, Jones, Niedermeier, and Tantau [9] presented a *fixed-paraemter tractable* (FPT) algorithm with different parameterization.

Recent years have seen increasing interest in efficient methods for finding a *diverse set of solutions* [5, 20, 27, 39]. Formally, let $(\mathcal{F}, d)$ be a distance space with a set $\mathcal{F}$ of feasible solutions and a distance $d : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geqslant 0}$, where $d(X, Y)$ denotes the distance between two solutions $X, Y \in \mathcal{F}$. We consider two diversity measures for a subset $\mathcal{X} = \{X_1, \ldots, X_K\} \subseteq \mathcal{F}$ of solutions:

$$D_d^{\mathrm{sum}}(\mathcal{X}) := \textstyle\sum_{i<j} d(X_i, X_j), \qquad (\textsc{Sum diversity}), \qquad (1)$$

$$D_d^{\mathrm{min}}(\mathcal{X}) := \min_{i<j} d(X_i, X_j), \qquad (\textsc{Min diversity}). \qquad (2)$$

For $\tau \in \{\mathrm{sum}, \mathrm{min}\}$, a subset $\mathcal{X} \subseteq \mathcal{F}$ of feasible solutions is said to be Δ-*diverse* w.r.t. D_d^τ (or simply, *diverse*) if $D_d^\tau(\mathcal{X}) \geqslant \Delta$ for a given $\Delta \geqslant 0$. Generally, the Max-Sum (resp. Max-Min) Diverse Solutions problem related to a combinatorial optimization problem Π is the problem of, given an input I to Π and a nonnegative number $\Delta \geqslant 0$, deciding if there exists a subset $\mathcal{X} \subseteq \mathrm{Sol}_\Pi(I)$ of K solutions on I such that $D_d^{\mathrm{sum}}(\mathcal{X}) \geqslant \Delta$ (resp. $D_d^{\mathrm{min}}(\mathcal{X}) \geqslant \Delta$), where $\mathrm{Sol}_\Pi(I) \subseteq \mathcal{F}$ is the set of solutions on I. For many distance spaces related to combinatorial optimization problems, both problems are known to be computationally hard with unbounded K [5, 6, 11, 18, 20, 27–29, 34, 45].

In this paper, we consider the problem of finding a diverse set of solutions for *longest common subsequences* of a set $\mathcal{S}$ of input strings under Hamming distance. The task is to select K longest common subsequences, maximizing the minimum pairwise Hamming distance among them. In general, a set of m strings of length n may have exponentially many longest common subsequences in n. Hence, efficiently finding such a diverse subset of solutions for longest common subsequences is challenging.

Let $d_H(X, Y)$ denote the Hamming distance between two strings $X, Y \in \Sigma^r$ of the equal length $r \geqslant 0$, called *r-strings*. Throughout this paper, we consider two diversity measures over sets of equi-length strings, the Sum-diversity $D^{\rm sum}_{d_H}$ and the Min-diversity $D^{\rm min}_{d_H}$ under the Hamming distance d_H. Let $LCS(\mathcal{S})$ denotes the *set of all longest common subsequences* of a set $\mathcal{S}$ of strings. Now, we state our first problem.

▶ **Problem 1** (Diverse LCSs with Diversity Measure $D^{\tau}_{d_H}$).
Input: Integers $K, r \geqslant 1$, and $\Delta \geqslant 0$, and a set $\mathcal{S} = \{S_1, \dots, S_m\}$ of $m \geqslant 2$ strings over Σ of length at most r ;
Question: Is there some set $\mathcal{X} \subseteq LCS(\mathcal{S})$ of longest common subsequences of $\mathcal{S}$ such that $|\mathcal{X}| = K$ and $D^{\tau}_{d_H}(\mathcal{X}) \geqslant \Delta$?

Then, we analyze the computational complexity of Diverse LCSs from the viewpoints of approximation algorithms [43] and parameterized complexity [15, 22]. For proving positive results for the case that K is bounded, actually, we work with a more general setting in which a set of strings to select is implicitly represented by the language $L(G)$ accepted by an edge-labeled DAG G, called a Σ-DAG. This is motivated by the fact implicit within the well-known algorithm for K-LCSs by Irving and Fraser [33] that the set $LCS(\mathcal{S})$ can be succinctly represented by such a Σ-DAG (see Lemma 2). In contrast, *negative results* will be proven in the setting where an input is explicitly given as a set of strings.

Let $\tau \in \{\text{sum}, \text{min}\}$ be any diversity type. Below, we state the modified version of the problem, where an input string set is an arbitrary set of equi-length strings, no longer a set of LCSs, and it is implicitly represented by either a Σ-DAG G or the set L itself.

▶ **Problem 2** (Diverse String Set with Diversity Measure $D^{\tau}_{d_H}$).
Input: Integers K, r, and Δ, and a Σ-DAG G for a set $L(G) \subseteq \Sigma^r$ of r-strings.
Question: Decide if there exists some subset $\mathcal{X} \subseteq L(G)$ such that $|\mathcal{X}| = K$ and $D^{\tau}_{d_H}(\mathcal{X}) \geqslant \Delta$.

Main results. Let $K \geqslant 1$, $r > 0$, and $\Delta \geqslant 0$ be integers and Σ be an alphabet. The underlying distance is always Hamming distance d_H over r-strings. In Diverse String Set, we assume that an input string set $L \subseteq \Sigma^r$ of r-strings is represented by either a Σ-DAG G or the set L itself. In Diverse LCS, we assume that the number $m = |\mathcal{S}|$ of strings in an input set $\mathcal{S}$ is assumed to be constant throughout. Then, the main results of this paper are summarized as follows.

1. When K is bounded, both Max-Sum and Max-Min versions of Diverse String Set and Diverse LCSs can be solved in polynomial time using dynamic programming (DP). (see Theorem 6, Theorem 8)
2. When K is part of the input, the Max-Sum version of Diverse String Set and Diverse LCSs admit a PTAS by local search showing that the Hamming distance is a *metric of negative type*[1]. (see Theorem 13)
3. Both of the Max-Sum and Max-Min versions of Diverse String Set and Diverse LCSs are fixed-parameter tractable (FPT) when parameterized by K and r (see Theorem 15, Theorem 16). These results are shown by combining Alon, Yuster, and Zwick's *color coding technique* [1] and the DP method in Result 1 above.
4. When K is part of the input, the Max-Sum and Max-Min versions of Diverse String Set and Diverse LCSs are NP-hard for any constant $r \geqslant 3$ (Theorem 17, Corollary 20).
5. When parameterized by K, the Max-Sum and Max-Min versions of Diverse String Set and Diverse LCSs are W[1]-hard (see Theorem 18, Corollary 21).

[1] It is a finite metric satisfying a class of inequalities of negative type [16]. For definition, see Sec. 4.

Table 2 Summary of results on Diverse String Set and Diverse LCSs under Hamming distance, where K, r, and Δ stand for the *number, length*, and *diversity threshold* for a subset $\mathcal{X}$ of r-strings, and α: *const, param*, and *input* indicate that α is a constant, a parameter, and part of an input, respectively. A representation of an input set L is both of Σ-DAG and LCS otherwise stated.

Problem	Type	K: const	K: param	K: input
Max-Sum Diverse String Set & LCS	Exact	Poly-Time (Theorem 8)	W[1]-hard on Σ-DAG (Theorem 18))	NP-hard on Σ-DAG if $r \geqslant 3$:const (Theorem 17)
			W[1]-hard on LCS (Corollary 21))	NP-hard on LCS (Corollary 20)
	Approx. or FPT	—	FPT if r: param (Theorem 16)	PTAS (Theorem 13)
Max-Min Diverse String Set & LCS	Exact	Poly-Time (Theorem 6)	W[1]-hard on Σ-DAG (Theorem 18)	NP-hard on Σ-DAG if $r \geqslant 3$:const (Theorem 17)
			W[1]-hard on LCS (Corollary 21)	NP-hard on LCS (Corollary 20)
	Approx. or FPT	—	FPT if r: param (Theorem 15)	Open

A summary of these results is presented in Table 2. We remark that the Diverse String Set problem coincides the original LCS problem when $K = 1$. It is generally believed that a W[1]-hard problem is unlikely to be FPT [17,22]. Future work includes the approximability of the *Max-Min* version of both problems for unbounded K, and extending our results to other distances and metrics over strings, e.g., *edit distance* [35,44] and *normalized edit distance* [21].

1.1 Related work

Diversity maximization for point sets in metric space and graphs has been studied since 1970s under various names in the literature [7,10,11,18,29,34,40,42] (see Ravi, Rosenkrantz, and Tayi [40] and Chandra and Halldórsson [11] for overview). There are two major versions: Max-Min version is known as *remote-edge*, *p-Dispersion*, and *Max-Min Facility Dispersion* [18,42,45]; Max-Sum version is known as *remote-clique*, *Maxisum Dispersion*, and *Max-Average Facility Dispersion* [7,10,29,40]. Both problems are shown to be NP-hard with unbounded K for general distance and metrics (with triangle inequality) [18,29], while they are polynomial time solvable for 1- and 2-dimensional ℓ_2-spaces [45]. It is trivially solvable in $n^{O(k)}$ time for bounded K.

Diversity maximization in combinatorial problems. However, extending these results for finding diverse solutions to combinatorial problems is challenging [5,20]. While methods such as *random sampling*, *enumeration*, and *top-K optimization* are commonly used for increasing the diversity of solution sets in optimization, they lack theoretical guarantee of the diversity [5,6,20,27]. In this direction, Baste, Fellows, Jaffke, Masarík, de Oliveira Oliveira, Philip, and Rosamond [5,6] pioneered the study of finding diverse solutions in combinatorial problems, investigating the parameterized complexity of well-know graph problems such as *Vertex Cover* [6]. Subsequently, Hanaka, Kiyomi, Kobayashi, Kobayashi, Kurita, and Otachi [28] explored the fixed-parameter tractability of finding various *subgraphs*.

They further proposed a framework for *approximating* diverse solutions, leading to efficient approximation algorithms for diverse matchings, and diverse minimum cuts [27]. While previous work has focused on diverse solutions in graphs and set families, the complexity of finding diverse solutions in *string problems* remains unexplored. Arrighi, Fernau, de Oliveira Oliveira, and Wolf [2] conducted one of the first studies in this direction, investigating a problem of finding a diverse set of subsequence-minimal synchronizing words.

DAG-based representation for all LCSs have appeared from time to time in the literature. The LCS algorithm by Irving and Fraser [33] for more than two strings can be seen as DP on a grid DAG for LCSs. Lu and Lin's parallel algorithm [37] for LCS used a similar grid DAG. Hakata and Imai [26] presented a faster algorithm based on a DAG of *dominant matches*. Conte, Grossi, Punzi, and Uno [12] and Hirota and Sakai [30] independently proposed DAGs of maximal common subsequences of two strings for enumeration.

The relationship between Hamming distance and other metrics has been explored in string and geometric algorithms. Lipsky and Porat [36] presented linear-time reductions from String Matching problems under Hamming distance to equivalent problems under ℓ_1-metric. Gionis, Indyk, and Motwani [24] used an *isometry* (a distance preserving mapping) from an ℓ_1-metric to Hamming distance over binary strings with a polynomial increase in dimension. Cormode and Muthukrishnan [14] showed an efficient ℓ_1-embedding of edit distance allowing moves over strings into ℓ_1-metric with small distortion. Despite these advancements, existing techniques haven't been successfully applied to our problems.

2 Preliminaries

We denote by $\mathbb{Z}$, $\mathbb{N} = \{ x \in \mathbb{Z} \mid x \geqslant 0 \}$, $\mathbb{R}$, and $\mathbb{R}_{\geqslant 0} = \{ x \in \mathbb{R} \mid x \geqslant 0 \}$ the sets of *all integers*, *all non-negative integers*, *all real numbers*, and *all non-negative real numbers*, respectively. For any $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \dots, n\}$. Let A be any set. Then, $|A|$ denotes the *cardinality* of A. Throughout, our model of computation is the word RAM, where the space is measured in $\Theta(\log n)$-bit machine words.

Let Σ be an *alphabet* of σ symbols. For any $n \geqslant 0$, Σ^n and Σ^* denote the sets of all strings of length n and all finite strings over Σ, respectively. Let $X = a_1 \dots a_n \in \Sigma^n$ be any string. Then, the *length* of X is denoted by $|X| = n$. For any $1 \leqslant i, j \leqslant n$, $X[i..j]$ denotes the substring $a_i \dots a_j$ if $i \leqslant j$ and the *empty string* ε otherwise. A *string set* or a *language* is a set $L = \{X_1, \dots, X_n\} \subseteq \Sigma^*$ of $n \geqslant 0$ strings over Σ. The *total length* of a string set L is denoted by $||L|| = \sum_{X \in L} |X|$, while the length of the longest strings in L is denoted by $\mathrm{maxlen}(L) := \max_{S \in L} |S|$. For any $r \geqslant 0$, we call any string X an *r-string* if its length is r, i.e., $X \in \Sigma^r$. Any string set L is said to be of *equi-length* if $L \subseteq \Sigma^r$ for some $r \geqslant 0$.

2.1 Σ-DAGs

A Σ-*labeled directed acyclic graph* (Σ-*DAG*, for short) is an edge-labeled directed acyclic graph (DAG) $G = (V, E, s, t)$ satisfying: (i) V is a set of vertices; (ii) $E \subseteq V \times \Sigma \times V$ is a set of labeled directed edges, where each edge $e = (v, c, w)$ in E is labeled with a symbol $c = \mathrm{lab}(e)$ taken from Σ; (iii) G has the unique *source* s and *sink* t in V such that every vertex lies on a path from s to t. We define the *size* of G as the number $\mathrm{size}(G)$ of its labeled edges. From (iii) above, G contains no unreachable nodes. For any vertex v in V, we denote the *set of its outgoing edges* by $E^+(v) = \{ (v, c, w) \in E \mid w \in V \}$. Any path $P = (e_1, \dots, e_n) \in E^n$ of length n *spells out* a string $\mathrm{str}(P) = \mathrm{lab}(e_1) \cdots \mathrm{lab}(e_n) \in \Sigma^n$ of length n, where $n \geqslant 0$. A Σ-DAG G *represents* the string set, or language, denoted $L(G) \subseteq \Sigma^*$, as the collection of all strings spelled out by its (s, t)-paths. Essentially, G is equivalent to a non-deterministic finite automaton (NFA) [32] over Σ with initial and final states s and t, and without ε-edges.

Fig. 1a shows an example of Σ-DAG representing the set of six longest common subsequences of two strings in Table 1. Sometimes, a Σ-DAG can succinctly represent a string set by its language $L(G)$. Actually, the size of G can be logarithmic in $|L(G)|$ in the best case,[2] while size(G) can be arbitrary larger than $||L(G)||$ (see Lemma 14 in Sec. 5).

▶ Remark 1. For any set L of strings over Σ, the following properties hold: (1) there exists a Σ-DAG G such that $L(G) = L$ and $\mathrm{size}(G) \leqslant ||L||$. (2) Moreover, G can be constructed from L in $O(||L||f(\sigma))$ time, where $f(n)$ is the query time of search and insert operation on a dictionary with n elements. (3) Suppose that a Σ-DAG G represents a set of strings $L \subseteq \Sigma^*$. If $L \subseteq \Sigma^r$ for $r \geqslant 0$, then all paths from the source s to any vertex v spell out strings of the same length, say $d \leqslant r$.

Proof. (1) We can construct a *trie* T for a set L of strings over Σ, which is a deterministic finite automaton for recognizing L in the shape of a rooted trees and has at most $O(||L||)$ vertices and edges. By identifying all leaves of T to form the sink, we obtain a Σ-DAG with $||L||$ edges for L. (2) It is not hard to see that the trie T can be built in $O(||L|| \log \sigma)$ time from L. (3) In what follows, we denote the string spelled out by any path π in G by $str(\pi)$. Suppose by contradiction that G has some pair of paths π_1 and $\pi_2 \in E^*$ from s to a vertex v such that $|str(\pi_1)| - |str(\pi_2)| > 0$ (*). By assumption (iii) in the definition of a Σ-DAG, the vertex v is contained in some (s,t)-path in G. Therefore, we have some path θ that connects v to t. By concatenating π_k and θ, we have two (s,t)-paths $\tau_k = \pi_k \cdot \theta$ for all $k = 1, 2$. Then, we observe from claim (*) that $|str(\tau_1)| - |str(\tau_2)| = |str(\pi_1 \cdot \theta)| - |str(\pi_2 \cdot \theta)| = (|str(\pi_1)| + |str(\theta)|) - (|str(\pi_2)| + |str(\theta)|) = |str(\pi_1)| - |str(\pi_2)| > 0$. On the other hand, we have $L(G)$ contains both of $str(\pi_1)$ and $str(\pi_2)$ since τ_1 and τ_2 are (s,t)-paths. This means that $L(G)$ contains two strings of distinct lengths, and this contradicts that $L(G) \subseteq \Sigma^r$ for some $r \geqslant 1$. Hence, all paths from s to v have the same length. Hence, (3) is proved. ◀

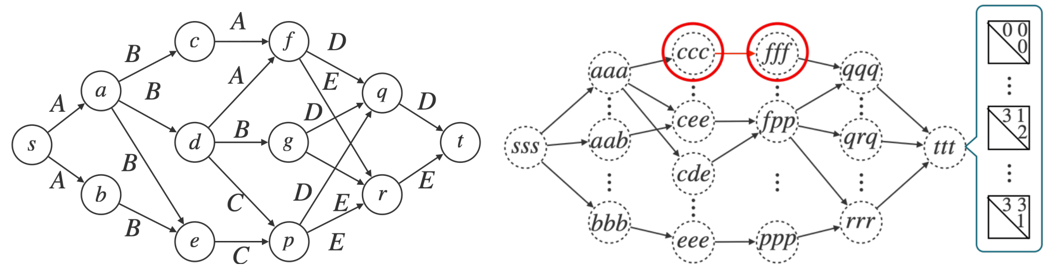
By Property (3) of Remark 1, we define the *depth* of a vertex v in G by the length depth(v) of any path P from the source s to v, called a *length-d prefix (path).* In other words, $\mathrm{depth}(v) = |\mathrm{str}(P)|$. Then, the vertex set V is partitioned into a collection of disjoint subsets $V_0 = \{s\} \cup \cdots \cup V_r = \{t\}$, where V_d is the subset of all vertices with depth d for all $d \in [r] \cup \{0\}$.

2.2 Longest common subsequences

A string X is a *subsequence* of another string Y if X is obtained from Y by removing some characters retaining the order. X is a *common subsequence* (CS) of any set $\mathcal{S} = \{S_1, \ldots, S_m\}$ of m strings if X is a subsequence of any member of $\mathcal{S}$. A CS of $\mathcal{S}$ is called a *longest common subsequence* (LCS) if it has the maximum length among all CSs of $\mathcal{S}$. We denote by $CS(\mathcal{S})$ and $LCS(\mathcal{S})$, respectively, the *sets of all CSs and all LCSs* of $\mathcal{S}$. Naturally, all LCSs in $LCS(\mathcal{S})$ have the same length, denoted by $0 \leqslant lcs(\mathcal{S}) \leqslant \min_{S \in \mathcal{S}} |S|$. While a string set $\mathcal{S}$ can contain exponentially many LCSs compared to the total length $||\mathcal{S}||$ of its strings, we can readily see the next lemma.

▶ **Lemma 2** (Σ-DAG for LCSs). *For any constant $m \geqslant 1$ and any set $\mathcal{S} = \{S_1, \ldots, S_m\} \subseteq \Sigma^*$ of m strings, there exists a Σ-DAG G of polynomial size in $\ell := \mathrm{maxlen}(\mathcal{S})$ such that $L(G) = LCS(\mathcal{S})$, and G can be computed in polynomial time in ℓ.*

[2] For example, for any $r \geqslant 1$, the language $L = \{a, b\}^r$ over an alphabet $\Sigma = \{a, b\}$ consists of $|L| = 2^r$ strings, while it can be represented by a Σ-DAG with $2r$ edges.



(a) An input Σ-DAG G_1. **(b)** An example run of Algorithm 1 for $K = 3$.

Figure 1 Illustration of Algorithm 1 based on dynamic programming. In (a) a Σ-DAG G_1 represents six LCSs in Table 1. In (b), circles and arrows indicate the states of the algorithm, which are K-tuples of vertices of G_1, and transition between them, respectively. All states are associated with a set of $K \times K$-weight matrices, which are shown only for the sink *ttt* in the figure.

Proof. By Irving and Fraser's algorithm [33], we can construct a m-dimensional grid graph N in $O(\ell^m)$ time and space, where (i) the source and sink are $s = (0, \dots, 0)$ and $t = (|S_1|, \dots, |S_m|)$, respectively; (ii) edge labels are symbols from $\Sigma \cup \{\varepsilon\}$; (iii) the number of edges is $\mathrm{size}(N) = \prod_{i=1}^{m} |S_i| \leqslant O(\ell^m)$; and (iv) all of (s,t)-paths spell out $LCS(\mathcal{S})$. Then, application of the ε-removal algorithm [32] yields a Σ-DAG G in $O(|\Sigma| \cdot \mathrm{size}(N))$ time and space, where G has $O(|\Sigma| \cdot \mathrm{size}(N)) = O(|\Sigma|\ell^m)$ edges. ◀

▶ Remark 3. As a direct consequence of Lemma 2, we observe that if Max-Min (resp. Max-Sum) Diverse String Set can be solved in $f(M, K, r, \Delta)$ time and $g(M, K, r, \Delta)$ space on a given input DAG G of size $M = \mathrm{size}(G)$, then Max-Min (resp. Max-Sum) Diverse LCSs on $\mathcal{S} \subseteq \Sigma^r$ can be solvable in $t = O(|\Sigma| \cdot \ell^m + f(\ell^m, K, r, \Delta))$ time and $s = O(\ell^m + g(\ell^m, K, r, \delta))$ space, where $\ell = \mathrm{maxlen}(\mathcal{S})$, since $\mathrm{size}(G) = O(\ell^m)$.

From Remark 3, for any constant $m \geqslant 2$, there exist a polynomial time reduction from Max-Min (resp. Max-Sum) Diverse LCSs for m strings to Max-Min (resp. Max-Sum) Diverse String Set on Σ-DAGs, where the distance measure is Hamming distance.

2.3 Computational complexity

A problem with parameter κ is said to be *fixed-parameter tractable* (FPT) if there is an algorithm that solves it, whose running time on an instance x is upperbounded by $f(\kappa(x)) \cdot |x|^c$ for a computable function $f(\kappa)$ and constant $c > 0$. A many-one reduction ϕ is called an *FPT-reduction* if it can be computed in FPT and the parameter $\kappa(\phi(x))$ is upper-bounded by a computable function of $\kappa(x)$. For notions not defined here, we refer to Ausiello *et al.* [3] for *approximability* and to Flum and Grohe [22] for *parameterized complexity.*

3 Exact Algorithms for Bounded Number of Diverse Strings

In this section, we show that both of Max-Min and Max-Sum versions of Diverse String Set problems can be solved by dynamic programming in polynomial time and space in the size an input Σ-DAG and integers r and Δ for any constant K. The corresponding results for Diverse LCSs problems will immediately follow from Remark 3.

3.1 Computing Max-Min Diverse Solutions

We describe our dynamic programming algorithm for the Max-Min Diverse String Set problem. Given an Σ-DAG $G = (V, E, s, t)$ with n vertices, representing a set $L(G) \subseteq \Sigma^r$ of r-strings, we consider integers $\Delta \geqslant 0, r \geqslant 0$, and constant $K \geqslant 1$. A brute-force approach could solve the problem in $O(|L(G)|^K)$ time by enumerating all combinations of K (s,t)-paths in G and selecting a Δ-diverse solution $\mathcal{X} \subseteq L(G)$. However, this is impractical even for constant K because $|L(G)|$ can be exponential in the number of edges.

The DP-table. A straightforward method to solve the problem is enumerating all combinations of K-tuples of paths from s to t to find the best K-tuple. However, the number of such K-tuples can be exponential in r. Instead, our DP-algorithm keeps track of only *all possible patterns* of their pairwise Hamming distances. Furthermore, it is sufficient to record only Hamming distance up to a given threshold Δ. In this way, we can efficiently computes the best combination of K paths provided that the number of patterns is manageable.

Formally, we let d $(0 \leqslant d \leqslant r)$ be any integer and $\boldsymbol{P} = (P_1, \dots, P_K) \in (E^d)^K$ be any K-tuple of length-d paths starting from the sink s and ending at some nodes. Then, we define the *pattern* of K-tuple $\boldsymbol{P}$ by the pair $\texttt{Pattern}(\boldsymbol{P}) = (\boldsymbol{w}, Z)$, where

- $\boldsymbol{w} = (w_1, \dots, w_K) \in V_d^K$ is the K-tuple of vertices in G, called a *state*, such that for all $i \in [K]$, the i-th path P_i starts from the source s and ends at the i-th vertex w_i of $\boldsymbol{w}$.
- $Z = (Z_{i,j})_{i<j} \in [\Delta \cup \{0\}]^{K \times K}$ is an upper triangular matrix, called the *weight matrix* for $\boldsymbol{P}$. For all $1 \leqslant i < j \leqslant K$, $Z_{i,j} = \min(\Delta, d_H(\mathrm{str}(P_i), \mathrm{str}(P_j))) \in [\Delta \cup \{0\}]$ is the Hamming distance between the string labels of P_i and P_j truncated by the threshold Δ.

Our algorithm constructs as the DP-table $\texttt{Weights} = (\texttt{Weights}(\boldsymbol{w}, Z))_{\boldsymbol{w},Z}$, which is a Boolean-valued table that associates a collection of weight matrices Z to each state $\boldsymbol{w}$ such that Z belongs to the collection if and only if $\texttt{Weights}(\boldsymbol{w}, Z) = 1$. See Fig. 1 for example. Formally, we define $\texttt{Weights}$ as follows.

▶ **Definition 4.** $\texttt{Weights} : V^K \times [\Delta \cup \{0\}]^{K \times K} \to \{0, 1\}$ *is a Boolean table such that for every K-tuple of vertices $\boldsymbol{w} \in V^K$ and weight matrix $Z \in [\Delta \cup \{0\}]^{K \times K}$, $\texttt{Weights}(\boldsymbol{w}, Z) = 1$ holds if and only if $(\boldsymbol{w}, Z) = \texttt{Pattern}(\boldsymbol{P})$ holds for some $0 \leqslant d \leqslant r$ and some K-tuple $\boldsymbol{P} \in (E^d)^K$ of length-d paths from the source s to $\boldsymbol{w}$ in G.*

We estimate the size of the table $\texttt{Weights}$. Since Z takes at most $\Gamma = O(\Delta^{K^2} K^2)$ distinct values, it can be encoded in $\log \Gamma = O(K^2 \log \Delta)$ bits. Therefore, $\texttt{Weights}$ has at most $|V|^K \times \Gamma = O(\Delta^{K^2} K^2 M^K)$ entries, where $M = \mathrm{size}(G)$. Consequently, for constant K, $\texttt{Weights}$ can be stored in a multi-dimensional table of polynomial size in M and Δ supporting random access in constant expected time or $O(\log\log(|V| \cdot \Delta))$ worst-case time [13, 46].

Computation of the DP-table. We denote the K-tuples of copies of the source s and sink t by $\boldsymbol{s} := (s, \dots, s)$ and $\boldsymbol{t} := (t, \dots, t) \in V^K$, respectively, as the initial and final states. The *zero matrix* $\texttt{Zero} = (\texttt{Zero}_{i,j})_{i<j}$ is a special matrix where $\texttt{Zero}_{i,j} = 0$ for all $i < j$. Now, we present the recurrence for the DP-table $\texttt{Weights}$.

▶ **Lemma 5** (recurrence for $\texttt{Weights}$). *For any $0 \leqslant d \leqslant r$, any $\boldsymbol{w} \in V^K$ and any $Z = (Z_{i,j})_{i<j} \in [\Delta \cup \{0\}]^{K \times K}$, the entry $\texttt{Weights}(\boldsymbol{w}, Z) \in \{0, 1\}$ satisfies the following:*

(1) *Base case: If $\boldsymbol{w} = \boldsymbol{s}$ and $Z = \texttt{Zero}$, then $\texttt{Weights}(\boldsymbol{w}, Z) = 1$.*

(2) *Induction case: If $\boldsymbol{w} \neq \boldsymbol{s}$ and all vertices in $\boldsymbol{w}$ have the same depth d $(1 \leqslant d \leqslant r)$, namely, $\boldsymbol{w} \in V_d^K$, then $\texttt{Weights}(\boldsymbol{w}, z) = 1$ if and only if there exist*

Algorithm 1 An exact algorithm for solving Max-Min Diverse r-String problem. Given a Σ-DAG $G = (V, E, s, t)$ representing a set $L(G)$ of r-strings and integers $K \geqslant 1, \Delta \geqslant 0$, decide if there exists some Δ-diverse set of K r-strings in $L(G)$.

Set $\texttt{Weights}(\boldsymbol{s}, Z) := 0$ for all $Z \in [\Delta \cup \{0\}]^{K\times K}$, and set $\texttt{Weights}(\boldsymbol{s}, \texttt{Zero}) \leftarrow 1$;
for $d \leftarrow 1, \ldots, r$ **do**
  for $\boldsymbol{v} \leftarrow (v_1, \ldots, v_K) \in (V_d)^K$ **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 $\boldsymbol{w} \leftarrow (w_1, \ldots, w_K)$;
      for $U \in [\Delta \cup \{0\}]^{K\times K}$ such that $\texttt{Weights}(\boldsymbol{v}, U) = 1$ **do**
        Set $Z = (Z_{i,j})_{i<j}$ with $Z_{i,j} \leftarrow \min(\Delta, U_{i,j} + \mathbb{1}\{c_i \neq c_j\}), \forall i, j \in [K]$;
        Set $\texttt{Weights}(\boldsymbol{w}, Z) \leftarrow 1$; ▷ *Update*
Answer YES if $\texttt{Weights}(\boldsymbol{t}, Z) := 1$ and $D^{\min}_{d_H}(Z) \geqslant \Delta$ for some Z, and NO otherwise;

- $\boldsymbol{v} = (v_i)_{i=1}^K \in V_{d-1}^K$ *such that each v_i is a parent of w_i, i.e., $(v_i, c_i, w_i) \in E$, and*
- $U = (U_{i,j})_{i<j} \in [\Delta \cup \{0\}]^{K\times K}$ *such that (i)* $\texttt{Weights}(\boldsymbol{v}, U) = 1$*, and (ii)* $Z_{i,j} = \min(\Delta, U_{i,j} + \mathbb{1}\{c_i \neq c_j\})$ *for all* $1 \leqslant i < j \leqslant K$*.*

(3) *Otherwise:* $\texttt{Weights}(\boldsymbol{w}, Z) = 0$.

Proof. The proof is straightforward by induction on $0 \leqslant d \leqslant r$. Thus, we omit the detail. ◀

Fig. 1b shows an example run of Algorithm 1 on a Σ-DAG G_1 in Fig. 1a representing the string set $L(G_1) = LCS(X_1, Y_1)$, where squares indicate weight matrices. From Lemma 5, we show Theorem 6 on the correctness and time complexity of Algorithm 1.

▶ **Theorem 6** (Polynomial time complexity of Max-Min Diverse String Set)**.** *For any $K \geqslant 1$ and $\Delta \geqslant 0$, Algorithm 1 solves Max-Min Diverse String Set 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, where $M = \mathrm{size}(G)$ is the number of edges in G.*

3.2 Computing Max-Sum Diverse Solutions

We can solve Max-Sum Diverse String Set by modifying Algorithm 1 as follows. For computing the Max-Sum diversity, we only need to maintain the sum $z = \sum_{i<j} d_H(\mathrm{str}(P_i), \mathrm{str}(P_j))$ of all pairwise Hamming distances instead of the entire $(K \times K)$-weight matrix Z.

The new table $\texttt{Weights}'$. For any $\boldsymbol{w} = (w_1, \ldots, w_K)$ of the same depth $0 \leqslant d \leqslant r$ and any integer $0 \leqslant z \leqslant rK$, we define: $\texttt{Weights}'(\boldsymbol{w}, z) = 1$ if and only if there exists a K-tuple of length-d prefix paths $(P_1, \ldots, P_K) \in (E^d)^K$ from s to $w_1, \ldots, w_K$, respectively, such that the sum of their pairwise Hamming distances is z, namely, $z = \sum_{i<j} d_H(\mathrm{str}(P_i), \mathrm{str}(P_j))$.

▶ **Lemma 7** (recurrence for $\texttt{Weights}'$)**.** *For any $\boldsymbol{w} = (w_1, \ldots, w_K) \in V^K$ and any integer $0 \leqslant z \leqslant rK$, the entry $\texttt{Weights}'(\boldsymbol{w}, z) \in \{0, 1\}$ satisfies the following:*

(1) *Base case: If $\boldsymbol{w} = \boldsymbol{s}$ and $z = 0$, then $\texttt{Weights}(\boldsymbol{w}, z) = 1$.*

(2) *Induction case: If $\boldsymbol{w} \neq \boldsymbol{s}$ and all vertices in $\boldsymbol{w}$ have the same depth d $(1 \leqslant d \leqslant r)$, namely, $\boldsymbol{w} \in V_d^K$, then $\texttt{Weights}(\boldsymbol{w}, z) = 1$ if and only if there exist*
- $\boldsymbol{v} = (v_i)_{i=1}^K \in V_{d-1}^K$ *such that each v_i is a parent of w_i, i.e., $(v_i, c_i, w_i) \in E$, and*
- $0 \leqslant u \leqslant rK$ *such that (i)* $\texttt{Weights}(\boldsymbol{v}, u) = 1$*, and (ii)* $z = \min(\Delta, u + \sum_{i<j} \mathbb{1}\{c_i \neq c_j\})$*.*

(3) *Otherwise:* $\texttt{Weights}(\boldsymbol{w}, z) = 0$.

Algorithm 2 A $(1 - 2/K)$-approximation algorithm for Max-Sum Diversification for a metric d of negative type on $\mathcal{X}$.

1 **procedure** LocalSearch($\mathcal{D}, K, d$);
2 $\mathcal{X} \leftarrow$ arbitrary K solutions in $\mathcal{D}$;
3 **for** $i \leftarrow 1, \ldots, \lceil \frac{K(K-1)}{(K+1)} \ln \frac{(K+2)(K-1)^2}{4} \rceil$ **do**
4 　**for** $X \in \mathcal{X}$ such that $\mathcal{D} \setminus \mathcal{X} \neq \emptyset$ **do**
5 　　$Y \leftarrow \operatorname{argmax}_{Y \in \mathcal{D} \setminus \mathcal{X}} \sum_{X' \in \mathcal{X} \setminus \{X\}} d(X', Y)$;
6 　　$\mathcal{X} \leftarrow (\mathcal{X} \setminus \{X\}) \cup \{Y\}$;
7 Output $\mathcal{X}$;

From the above modification of Algorithm 1 and Lemma 7, we have Theorem 8. From this theorem, we see that the Max-Sum version of Diverse String Set can be solved faster than the Max-Min version by factor of $O(\Delta^{K-1})$.

▶ **Theorem 8** (Polynomial time complexity of Max-Sum Diverse String Set). *For any constant $K \geqslant 1$, the modified version of Algorithm 1 solves Max-Sum Diverse String Set under Hamming Distance in $O(\Delta K^2 M^K (\log |V| + \log \Delta))$ time and space, where $M = \mathrm{size}(G)$ is the number of edges in G and the input set L is represented by a Σ-DAG.*

4 Approximation Algorithm for Unbounded Number of Diverse Strings

To solve Max-Sum Diverse String Set for unbounded K, we first introduce a local search procedure, proposed by Cevallos, Eisenbrand, and Zenklusen [10], for computing approximate diverse solutions in general finite metric spaces (see [16]). Then, we explain how to apply this algorithm to our problem in the space of equi-length strings equipped with Hamming distance.

Let $\mathcal{D} = \{x_1, \ldots, x_n\}$ be a finite set of $n \geqslant 1$ elements. A semi-metric is a function $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}_{\geqslant 0}$ satisfying the following conditions (i)–(iii): (i) $d(x, x) = 0, \forall x \in \mathcal{D}$; (ii) $d(x, y) = d(y, x), \forall x, y \in \mathcal{D}$; (iii) $d(x, z) \leqslant d(x, y) + d(y, z), \forall x, y, z \in \mathcal{D}$ (triangle inequalities). Consider an inequality condition, called a *negative inequality*:

$$\boldsymbol{b}^\top D\, \boldsymbol{b} := \textstyle\sum_{i<j} b_i b_j d(x_i, x_j) \leqslant 0, \qquad \forall \boldsymbol{b} = (b_1, \ldots, b_n) \in \mathbb{Z}^n, \tag{3}$$

where $\boldsymbol{b}$ is a column vector and $D = (d_{ij})$ with $d_{ij} = d(x_i, x_j)$. For the vector $\boldsymbol{b}$ above, we define $\sum \boldsymbol{b} := \sum_{i=1}^n b_i$. A semi-metric d is said to be *of negative type* if it satisfies the inequalities Eq. (3) for all $\boldsymbol{b} \in \mathbb{Z}^n$ such that $\sum \boldsymbol{b} = 0$. The class $\mathbb{NEG}$ of semi-metrics of negative type satisfies the following properties.

▶ **Lemma 9** (Deza and Laurent [16]). *For the class $\mathbb{NEG}$, the following properties hold: (1) All ℓ_1-metrics over $\mathbb{R}^r$ are semi-metrics of negative type in $\mathbb{NEG}$ for any $r \geqslant 1$. (2) The class $\mathbb{NEG}$ is closed under linear combination with nonnegative coefficients in $\mathbb{R}_{\geqslant 0}$.*

In Algorithm 2, we show a local search procedure LocalSearch, proposed by Cevallos *et al.* [10], for computing a diverse solution $\mathcal{X} \subseteq \mathcal{D}$ with $|\mathcal{X}| = K$ approximately maximizing its Max-Sum diversity under a given semi-metric $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}_{\geqslant 0}$ on a finite metric space $\mathcal{D}$ of n points. The Farthest Point problem refer to the subproblem for computing Y at Line 5. When the distance d is a semi-metric of *negative type*, they showed the following theorem.

Algorithm 3 An exact algorithm for solving the MAX-SUM FARTHEST r-STRING problem. Given a Σ-DAG G for a set $L(G) \subseteq \Sigma^r$, a set $\mathcal{X} = \{X_1, \dots, X_K\} \subseteq \Sigma^r$, and an integer $\Delta \geqslant 0$, it decides if there exists a $Y \in L(G)$ such that $D^{\mathrm{sum}}_{d_H}(\mathcal{X}, Y) \geqslant \Delta$.

```
Set Weights(s, z) := 0 for all z ∈ [Δ]_+, and Weights(s, 0) := 1;
for d := 1, ..., r do
    for v ∈ V_d and (v, c, w) ∈ E^+(v) do
        for 0 ⩽ u ⩽ Δ such that Weights(v, u) := 1 do
            Set Weights(w, z) := 1 for z := u + Σ_{i∈[K]} 1{c ≠ X_i[d]} ;        ▷ Update
Answer YES if Weights(t, Δ) = 1, and NO otherwise ;                                ▷ Decide
```

▶ **Theorem 10** (Cevallos *et al.* [10]). *Suppose that d is a metric of negative type over $\mathcal{X}$ in which the FARTHEST POINT problem can be solved in polynomial time. For any $K \geqslant 1$, the procedure LOCALSEARCH in Algorithm 2 has approximation ratio $(1 - \frac{2}{K})$.*

We show that the Hamming distance actually has the desired property.

▶ **Lemma 11.** *For any integer $r \geqslant 1$, the Hamming distance d_H over the set Σ^r of r-strings is a semi-metric of negative type over Σ^r.*

Proof. Let $\Sigma = [\sigma]$ be any alphabet. We give an isometry ϕ (see Sec. 1.1) from the Hamming distance (Σ^r, d_H) to the ℓ_1-metric (W, d_{ℓ_1}) over a subset W of $\mathbb{R}^m$ for $m = r\sigma$. For any symbol $i \in \Sigma$, we extend ϕ by $\phi_\Sigma(i) := 0^{i-1}(0.5)0^{\sigma-i} \in \{0, 0.5\}^\sigma$ be a bitvector with 0.5 at i-th position and 0 at other bit positions such that for each $c, c' \in \Sigma$, $||\phi_\Sigma(c) - \phi_\Sigma(c')||_1 = \mathbb{1}\{\, c \neq c' \,\}$. For any r-string $S = S[1] \dots S[r] \in \Sigma^r$, we let $\phi(S) := \phi_\Sigma(S[1]) \cdots \phi_\Sigma(S[r]) \in W$, where $W := \{0, 0.5\}^m$ and $m := r\sigma$. For any $S, S' \in \Sigma^r$, we can show $d_{\ell_1}(\phi(S), \phi(S')) = ||\phi(S)_j - \phi(S')_j||_1 = \sum_{i\in[r]} ||\phi_\Sigma(S[i]) - \phi_\Sigma(S'[i])||_1 = \sum_{i\in[r]} \mathbb{1}\{\, S[i] \neq S'[i] \,\} = d_H(S, S')$. Thus, $\phi : \Sigma^r \to W$ is an *isometry* [16] from (Σ^r, d_H) to $(\{0, 0.5\}^m, d_{\ell_1})$. By Lemma 9, ℓ_1-metric is a metric of negative type [10, 16], and thus, the lemma is proved. ◀

The remaining thing is efficiently solving the string version of the subproblem, called the FARTHEST STRING problem, that given a set $\mathcal{X}' \subseteq \Sigma^r$, asks to find the *farthest* Y from all elements in $\mathcal{X}'$ by maximizing the sum $D^{\mathrm{sum}}_{d_H}(\mathcal{X}', Y) := \sum_{X' \in \mathcal{X}'} d_H(X', Y)$ over all elements $Y \in L(G) \setminus \mathcal{X}'$. For the class of r-strings, we show the next lemma.

▶ **Lemma 12** (FARTHEST r-STRING). *For any $K \geqslant 1$ and $\Delta \geqslant 0$, given G and $\mathcal{X}' \subseteq L(G)$, Algorithm 3 computes the farthest r-string $Y \in L(G)$ that maximizes $D^{\mathrm{sum}}_{d_H}(\mathcal{X}', Y)$ over all r-strings in $L(G)$ in $O(K\Delta M)$ time and space, where M is the number of edges in G.*

Proof. The procedure in Algorithm 3 solves the decision version of MAX-SUM FARTHEST r-STRING. Since it is obtained from Algorithm 1 by fixing $K - 1$ paths and searching only a remaining path in G, its correctness and time complexity immediately follows from that of Theorem 6. It is easy to modify Algorithm 3 to compute such Y that maximizes $D^{\mathrm{sum}}_{d_H}(\mathcal{X}', Y)$ by recording the parent pair (v, y) of each (w, z) and then tracing back. ◀

Combining Theorem 10, Lemma 12, and Lemma 11, we obtain the following theorem on the existence of a *polynomial time approximation scheme* (PTAS) [3] for MAX-SUM DIVERSE STRING SET on Σ-DAGs. From Theorem 13 and Remark 3, the corresponding result for MAX-SUM DIVERSE LCSs will immediately follow.

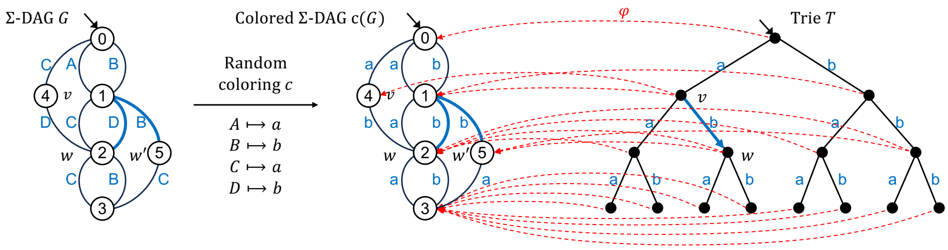


Figure 2 Illustration of the proof for Lemma 14, where dashed lines indicates a correspondence φ.

▶ **Theorem 13** (PTAS for unbounded K). *When K is part of an input, Max-Sum Diverse String Set problem on a Σ-DAG G admits PTAS.*

Proof. We show the theorem following the discussion of [10, 27]. Let $\varepsilon > 0$ be any constant. Suppose that $\varepsilon < 2/K$ holds. Then, $K < 2/\varepsilon$, and thus, K is a constant. In this case, by Theorem 6, we can exactly solve the problem in polynomial time using Algorithm 1. Otherwise, $2/K \leqslant \varepsilon$. Then, the $(1 - 2/K)$ approximation algorithm in Algorithm 2 equipped with Algorithm 3 achieves factor $1 - \varepsilon$ since d_H is a negative type metric by Lemma 11. Hence, Max-Sum Diverse String Set admits a PTAS. This completes the proof. ◀

5 FPT Algorithms for Bounded Number and Length of Diverse Strings

In this section, we present fixed-parameter tractable (FPT) algorithms for the Max-Min and Max-Sum Diverse String Set parameterized with combinations of K and r. Recall that a problem parameterized with κ is said to be *fixed-parameter tractable* if there exists an algorithm for the problem running on an input x in time $f(\kappa(x)) \cdot |x|^c$ for some computable function $f(\kappa)$ and constant $c > 0$ [22].

For our purpose, we combine the *color-coding technique* by Alon, Yuster, and Zwick [1] and the algorithms in Sec. 3. Consider a random C-coloring $c : \Sigma \to C$ from a set C of $k \geqslant 1$ colors, which assigns a color $c(a)$ chosen from C randomly and independently to each $a \in \Sigma$. By applying this C-coloring to all each edges of an input Σ-DAG G, we obtain the C-colored DAG, called *a C-DAG*, and denote it by $c(G)$. We show a lemma on reduction of $c(G)$.

▶ **Lemma 14** (computing a reduced C-DAG in FPT). *For any set C of k colors, there exists some C-DAG H obtained by reducing $c(G)$ such that $L(H) = L(c(G))$ and $||H|| \leqslant k^r$. Furthermore, such a C-DAG H can be computed from G and C in $t_{\mathrm{pre}} = O(k^r \cdot \mathrm{size}(G))$ time and space.*

Proof. We show a proof sketch. Since $L(G) \subseteq \Sigma^r$, we see that the C-DAG $c(G)$ represents $L(c(G)) \subseteq C^r$ of size at most $||L(c(G))|| \leqslant k^r$. By Remark 1, there exists a C-DAG H for $L(H) = L(c(G))$ with at most k^r edges. However, it is not straightforward how to compute such a succinct H directly from G and c in $O(k^r \cdot \mathrm{size}(G))$ time and space since $||L(G)||$ can be much larger than $k^r + \mathrm{size}(G)$. We build a trie T for $L(H)$ top-down using breadth-first search of G from the source s by maintaining a correspondence $\varphi \subseteq V \times U$ between vertices V in G and vertices U in T (Fig. 2). Then, we identify all leaves of T to make the sink t. This runs in $O(k^r \cdot \mathrm{size}(G))$ time and $O(k^r + \mathrm{size}(G))$ space. ◀

Fig. 2 illustrates computation of reduced C-DAG H from an input Σ-DAG G over alphabet $\Sigma = \{A, B, C, D\}$ in Lemma 14, 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). Combining Lemma 14, Theorem 6, and Alon *et al.* [1], we show the next theorem.

▶ **Theorem 15.** *When r and K are parameters, the* Max-Min Diverse String Set *on a Σ-DAG for r-strings is fixed-parameter tractable (FPT), where* size(G) *is an input.*

Proof. We show a sketch of the proof. We show a randomized algorithm using Alon *et al.*'s color-coding technique [1]. Let $L(G) \subseteq \Sigma^r, k = rK$, and $C = [rK]$. We assume without loss of generality that $\Delta \leqslant r$. We randomly color edges of G from C. Then, we perform two phases below.

- *Preprocessing phase*: Using the FPT-algorithm of Lemma 14, reduce the colored C-DAG $c(G)$ with size(G) into another C-DAG H with $L(H) = L(c(G)) \subseteq C^r$ and size bounded by $(rK)^r$. Lemma 14 shows that this requires $t_{\text{pre}} = O((rK)^r \cdot \text{size}(G))$ time and space.
- *Search phase*: Find a Δ-diverse subset $\mathcal{Y}$ in $L(H)$ of size $|\mathcal{Y}| = K$ from H using a modified version of Algorithm 1 in Sec. 3 (details in footnote[3]). If such $\mathcal{Y}$ exists and c is invertible, then $\mathcal{X} = c^{-1}(\mathcal{Y})$ is a Δ-diverse solution for the original problem. The search phase takes $t_{\text{search}} = O(K^2\Delta^{K^2}(rK)^{rK}) =: g(K, r)$ time, where $\Delta \leqslant r$ is used.

With the probability $p = (rK)!/(rK)^{rK} \geqslant 2^{-rK}$, for $C = [rK]$, the random C-coloring yields a colorful Δ-diverse subset $\mathcal{Y} = c(\mathcal{X}) \subseteq L(H)$. Repeating the above process 2^{rK} times and derandomizing it using Alon *et al.* [1] yields an FPT algorithm with total running time $t = 2^{rK} r \log(rK)(t_{\text{pre}} + t_{\text{search}}) = f(K, r, \Delta) \cdot \text{size}(G)$, where $f(K, r, \Delta) = O(2^{rK} r \log(rK) \cdot \{(rK)^r + g(K, r)\})$ depends only on parameters r and K. This completes the proof. ◀

Similarly, we obtain the following result for Max-Sum Diversity.

▶ **Theorem 16.** *When r and K are parameters, the Max-Sum Diverse String Set on Σ-graphs for r-strings is fixed-parameter tractable (FPT), where* size(G) *is part of an input.*

Proof. The proof proceeds by a similar discussion to the one in the proof of Theorem 15. The only difference is the time complexity of t_{search}. In the case of Max-Sum diversity, the search time of the modified algorithm in Theorem 8 is $t_{\text{search}} = O(\Delta K^2 M^K)$, where $M = \text{size}(G)$. By substituting $M \leqslant (rK)^k$ for t_{search}, we have $t_{\text{search}} = g'(K, r)\Delta$, where $g'(K, r) := O(K^2(rK)^{rK})$. Since $g'(K, r)$ depends only on parameters, the claim follows. ◀

6 Hardness results

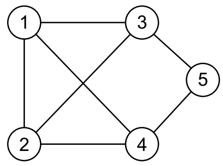
To complement the positive results in Sec. 3 and Sec. 4, we show some negative results in classic and parameterized complexity. In what follows, $\sigma = |\Sigma|$ is an alphabet size, K is the number of strings to select, r is the length of equi-length strings, and Δ is a diversity threshold. In all results below, we assume that σ are constants, and without loss of generality from Remark 1 that an input set L of r-strings is explicitly given as the set itself.

6.1 Hardness of Diverse String Set for Unbounded K

Firstly, we observe the NP-hardness of Max-Min and Max-Sum Diverse String Set holds for unbounded K even for constants $r \geqslant 3$.

▶ **Theorem 17** (NP-hardness for unbounded K). *When K is part of an input,* Max-Min *and* Max-Sum Diverse String Set *on Σ-graphs for r-strings are NP-hard even for any constant $r \geqslant 3$.*

[3] This modification of Algorithm 1 is easily done at Line 7 of Algorithm 1 by replacing the term $\mathbb{1}\{lab(e_i) \neq lab(e_j)\}$ with the term $\mathbb{1}\{\{c(lab(e_i)) \neq c(lab(e_j))\} \wedge \{lab(e_i) \neq lab(e_j)\}\}$.



	12	13	14	15	23	24	25	34	35	45
S_1	1	1	1	0	1	1	1	1	1	1
S_2	2	2	2	2	2	2	0	2	2	2
S_3	3	3	3	3	3	3	3	0	3	3
S_4	4	4	4	4	4	4	4	0	4	4
S_5	5	5	5	0	5	5	0	5	5	5

Figure 3 An example of reduction for the proof of Theorem 18 in the case of $n = 5$, consisting of an instance G of Clique, with a vertex set $V = \{1, \ldots, 5\}$ and a edge set $E \subseteq \mathcal{E} = \{12, 13, \ldots, 45\}$ (left), and the associated instance $F = \{S_1, \ldots, S_n\}$ of Diverse r-String Set, where F contains $n = 5$ r-strings with $r = |\mathcal{E}| = 10$ (right). Shadowed cells indicate the occurrences of symbol 0.

Proof. We reduce an NP-hard problem 3DM [23] to Max-Min Diverse String Set by a trivial reduction. Recall that given an instance consists of sets $A = B = C = [n]$ for some $n \geqslant 1$ and a set family $F \subseteq [n]^3$, and 3DM asks if there exists some subset $M \subseteq F$ that is a *matching*, that is, any two vectors $X, Y \in M$ have no position $i \in [3]$ at which the corresponding symbols agree, i.e., $X[i] = Y[i]$. Then, we construct an instance of Max-Min Diverse String Set with $r = 3$ with an alphabet $\Sigma = A \cup B \cup C$, a string set $L = F \subseteq \Sigma^3$, integers $K = n$ and $\Delta = r = 3$. Obviously, this transformation is polynomial time computable. Then, it is not hard to see that for any $M \subseteq F$, M is a matching if and only if $D^{\min}_{d_H}(M) \geqslant \Delta$ holds. On the other hand, for Max-Min Diverse String Set, if we let $\Delta' = \binom{K}{2}$ then for any $M \subseteq F$, M is a matching if and only if $D^{\mathrm{sum}}_{d_H}(M) \geqslant \Delta'$ holds. Combining the above arguments, the theorem is proved. ◀

We remark that 3DM is shown to be in FPT by Fellows, Knauer, Nishimura, Ragde, Rosamond, Stege, Thilikos, and Whitesides [19]. Besides, we showed in Sec. 5 that Diverse r-String Set is FPT when parameterized with $K + r$ (Max-Sum) or $K + r + \Delta$ (Max-Min), respectively. We show that the latter problem is W[1]-hard parameterized with K.

▶ **Theorem 18** (W[1]-hardness of the string set and Σ-DAG versions for unbounded K). *When parameterized with K, Max-Min and Max-Sum Diverse String Set for a set L of r-strings are W[1]-hard whether a string set L is represented by either a string set L or a Σ-DAG for L, where r and Δ are part of an input.*

Proof. We establish the W[1]-hardness of Max-Min Diverse String Set with parameter K by reduction from Clique with parameter K. This builds on the NP-hardness of r-Set packing in Ausiello *et al.* [4] with minor modifications (see also [19]). Given a graph $G = (V, E)$ with n vertices and a parameter $K \in \mathbb{N}$, where $V = [n]$ and $E \subseteq \mathcal{E}$, we let $\mathcal{E} := \{\, \{i, j\} \mid i, j \in V, i \neq j \,\}$. We define the transformation ϕ_1 from $\langle G, K \rangle$ to $\langle \Sigma, r, F, \Delta \rangle$ and $\kappa(K) = K$ as follows. We let $\Sigma = [n] \cup \{0\}$, $r = |\mathcal{E}| = \binom{n}{2}$, and $\Delta = r$. We view each r-string S as a mapping $S : \mathcal{E} \to \Sigma$ assigning symbol $S(e) \in \Sigma$ to each unordered pair $e \in \mathcal{E}$. We construct a family $F = \{\, S_i \mid i \in V \,\}$ of r-strings such that G has a clique of K elements if and only if there exists a subset $M \subseteq F$ with (a) size $|M| \geqslant \kappa(K) = K$, and (b) diversity $d_H(S, S') \geqslant r = \Delta$ for all distinct $S, S' \in M$ (*). Each r-string S_i is defined based on the existence of the edges in E: (i) $S_i(e) = 0$ if $(i \in e) \wedge (e \notin E)$, and (ii) $S_i(e) = i$ otherwise. By definition, $d_H(S_i, S_j) \leqslant r$. We show that for any $i, j \in \mathcal{E}$, S_i and S_j have conflicts at all positions, i.e. $d_H(S_i, S_j) = r$, if and only if $\{i, j\} \in E$. See Fig. 3 for example of F. To see the correctness, suppose that $e = \{i, j\} \notin E$. Then, it follows from (i) that $S_i(e) = S_j(e) = 0$ since $(i \in e) \wedge (j \in e) \wedge e \notin E$. Conversely, if $e' = \{i, j\} \in E$, the condition (i) $S_i(e) = S_j(e) = 0$ does not hold for any $e \in \mathcal{E}$ because if $e \neq e'$, one of $(i \in e)$ and $(j \in e)$ does not hold, and if $e = e'$, $e \notin E$ does not hold. This proves the claim (*). Since ϕ_1 and κ form an FPT-reduction. The theorem is proved. ◀

In this subsection, we show the hardness results of Diverse LCSs for unbounded K in classic and parameterized settings by reducing them to those of Diverse String Set in Sec. 6.1.

▶ **Theorem 19.** *Under Hamming distance, Max-Min (resp. Max-Sum) Diverse String Set for $m \geqslant 2$ strings parameterized with K is FPT-reducible to Max-Min (resp. Max-Sum) Diverse LCSs for two string ($m = 2$) parameterized with K, where m is part of an input. Moreover, the reduction is also a polynomial time reduction from Max-Min (resp. Max-Sum) Diverse String Set to Max-Min (resp. Max-Sum) Diverse LCSs.*

We defer the proof of Theorem 19 in Sec. 6.1.1. Combining Theorem 17, Theorem 18, and Theorem 19, we have the corollaries.

▶ **Corollary 20** (NP-hardness). *When K is part of an input, Max-Min and Max-Sum Diverse LCSs for two r-strings are NP-hard, where r and Δ are part of an input.*

▶ **Corollary 21** (W[1]-hardness). *When parameterized with K, Max-Min and Max-Sum Diverse LCSs for two r-strings are W[1]-hard, where r and Δ are part of an input.*

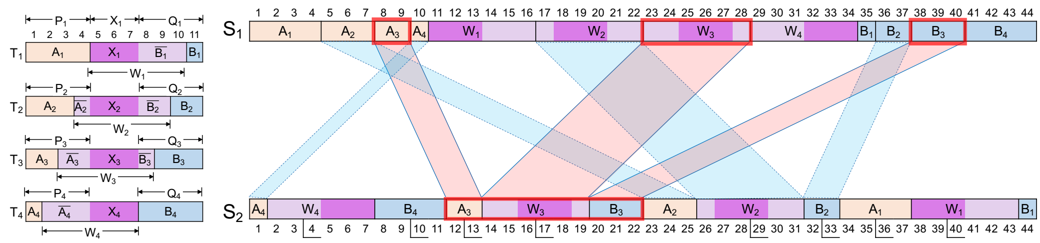
6.1.1 Proof for Theorem 19

In this subsection, we show the proof of Theorem 19, which is deferred in the previous section. Suppose that we are given an instance of Max-Sum Diverse String Set consisting of integers $K, r \geqslant 1$, $\Delta \geqslant 0$, and any set $L = \{X_i\}_{i=1}^s \subseteq \Sigma^r$ of r-strings, where $s = |L| \geqslant 2$. We let $\Xi = \{ a_{i,j}, b_{i,j} \mid i, j \in [s] \}$ be a set of mutually distinct symbols, and $\Gamma = \Sigma \cup \Xi$ be a new alphabet with $\Sigma \cap \Xi = \emptyset$. We let $\mathcal{T} = \{T_i := P_i X_i Q_i\}_{i=1}^s$ be the set of s strings of length $|T_i| = r + 2s$ over Γ, where $P_i := a_{i1} \dots a_{is} \in \Gamma^s, Q_i := b_{i1} \dots b_{is} \in \Gamma^s, \forall i \in [s]$.

Now, we construct two input strings S_1 and S_2 over Γ in an instance of Max-Min Diverse LCSs so that $LCS(\mathcal{S}_1, \mathcal{S}_2) = \mathcal{T}$. For all $i \in [s]$, we factorize each strings T_i of length $(r+2s)$ into three substrings $A_i, W_i, B_i \in \Gamma^+$, called *segments*, such that $T_i = A_i \cdot W_i \cdot B_i$ such that (i) We partition P_i into $P_i = A_i \cdot \overline{A_i}$, where $A_i := P_i[1..s-i+1]$ is the prefix with length $s-i+1$ and $\overline{A_i} = P_i[s-i+2..s]$ is the suffix with length $i-1$ of P_i. (ii) We partition Q_i into $Q_i = \overline{B_i} \cdot B_i$, where $\overline{B_i} = Q_i[1..s-i]$ is the prefix with length $s-i$ and $B_i := Q_i[s-i+1..s]$ is the suffix with length i of Q_i. (iii) We obtain a string $W_i := \overline{A_i} \cdot X_i \cdot \overline{B_i}$ with length $r+s-1$ from X_i by prepending and appending $\overline{A_i}$ and $\overline{B_i}$ to X_i. Let $\mathcal{A} = \{ A_i \}_{i=1}^s$, $\mathcal{B} = \{ B_i \}_{i=1}^s$, and $\mathcal{W} = \{ W_i \}_{i=1}^s$ be the *groups* of the segments of the *same types*. See Fig. 4a for examples of $\mathcal{A}, \mathcal{B}$, and $\mathcal{W}$. Then, we define the set $\mathcal{S} = \{S_1, S_2\}$ of two input strings S_1 and S_2 of the same length $|S_1| = |S_2| = s(r+2s)$ by:

$$\begin{aligned}
\mathcal{S}_1 &= \textstyle\prod_{i=1}^s A_i \cdot \prod_{i=1}^s W_i \cdot \prod_{i=1}^s B_i = (A_1 \cdots A_s) \cdot (W_1 \cdots W_s) \cdot (B_1 \cdots B_s), \\
\mathcal{S}_2 &= \textstyle\prod_{i=s}^1 T_i = \prod_{i=s}^1 (A_i \cdot W_i \cdot B_i) = (A_s \cdot W_s \cdot B_s) \cdots (A_1 \cdot W_1 \cdot B_1).
\end{aligned} \tag{4}$$

Fig. 4b shows an example of $\mathcal{S}$ for $s = 4$. We observe the following properties of $\mathcal{S}$: (P1) S_1 and S_2 are segment-wise permutations of each other; (P2) if all segments in any group $\mathcal{Z} = \{Z_i\}_{i=1}^s \in \{\mathcal{A}, \mathcal{B}, \mathcal{W}\}$ occur one of two input strings, say S_1, in the order $Z_1, \dots, Z_s$, then they occur in the other, say S_2, in the reverse order $Z_s, \dots, Z_1$; (P3) A_i's (resp. B_i's) appear in S_2 from left to right in the order $A_s, \dots, A_1$ (resp. $B_s, \dots, B_1$); (P4) $\mathcal{A}$ and $\mathcal{B}$ satisfy $|A_1| > \cdots > |A_s|$ and $|B_1| < \cdots < |B_s|$. We associate a bipartite graph $\mathcal{B}(\mathcal{S}) = (V = V_1 \cup V_2, E)$ to $\mathcal{S}$, where (i) V_k consists of all positions in S_k for $k = 1, 2$, and (ii) $E \subseteq V_1 \times V_2$ is an edge set such that $e = (i_1, i_2) \in E$ if and only if both ends of e have the same label $S_1[i_1] = S_2[i_2] \in \Sigma$. Any sequence $M = ((i_k, j_k))_{k=1}^\ell \in E^\ell$ of ℓ edges is an *(ordered) matching* if $i_1 \neq j_1$ and $i_2 \neq j_2$, and is *non-crossing* if $(i_1 < j_1)$ and $(i_2 < j_2)$.



(a) The set $\mathcal{T}$. **(b)** Input strings S_1 and S_2.

Figure 4 Construction of the FPT-reduction from Max-Min Diverse String Set to Max-Min Diverse LCS in the proof of Theorem 19, where $s = 4$. We show (a) the set $\mathcal{T}$ of s r-strings and (b) a pair of input strings S_1 and S_2. Red and blue parallelograms, respectively, indicate allowed and prohibited matchings between the copies of blocks $T_3 = A_3W_3B_3$ in S_1 and S_2.

▶ **Lemma 22.** *For any $M \subseteq V_1 \times V_2$ and any $\ell \geqslant 0$, $\mathcal{B}(\mathcal{S})$ has a non-crossing matching M of size ℓ if and only if S_1 and S_2 have a common subsequence C with length ℓ of S_1 and S_2. Moreover, the length $\ell = |M|$ is maximum if and only if $C \in LCS(S_1, S_2)$.*

Proof. If there exists a non-crossing matching $M = \{\, (i_k, j_k) \mid k \in [\ell] \,\} \subseteq E$ of size $\ell \geqslant 0$, we can order the edges in the increasing order such that $i_{\pi(1)} < \cdots < i_{\pi(\ell)}, j_{\pi(1)} < \cdots < j_{\pi(\ell)}$ for some permutation π on $[\ell]$. Then, the string $S_1(M) := S_1[i_{\pi(1)}] \cdots S_1[i_{\pi(\ell)}] \in \Sigma^\ell$ (equivalently, $S_2(M) := S_2[j_1] \cdots S_2[j_\ell]$) forms the common subsequence associated to M. ◀

In Lemma 22, we call a non-crossing ordered matching M associated with a common subsequence C a *matching labeled with* C. We show the next lemma.

▶ **Lemma 23.** *$LCS(S_1, S_2) = \{\, T_j \mid i \in [s] \,\}$, where $T_j = P_j \cdot X_j \cdot Q_j$ for all $j \in [s]$.*

Proof. We first observe that each segment $Z \in \Sigma^+$ in each group $\mathcal{Z}$ within $\{\mathcal{A}, \mathcal{B}, \mathcal{W}\}$ occurs exactly once in each of S_1 and S_2, respectively, as a consecutive substring. Consequently, For each Z in $\mathcal{Z}$, $\mathcal{B}(\mathcal{S})$ has exactly one non-crossing matching M_Z *labeled with Z connecting* the occurrences of Z in S_1 and S_2. From (P2), we show the next claim.

▷ Claim 24. If $\mathcal{B}(\mathcal{S})$ contains any inclusion-wise maximal non-crossing matching M_*, it connects exactly one segment Z from each of three groups $\mathcal{A}, \mathcal{B}$, and $\mathcal{W}$.

From Claim 24, we assume that a maximum (thus, inclusion-maximal) non-crossing matching M_* contains submatches labeled with segments A_i, W_j, B_k one from each group in any order, where $i, j, k \in [s]$. Then, M must contain A_i, W_j, B_k in this order, namely, $A_i \cdot W_j \cdot B_k \in CS(S_1, S_2)$ because some edges cross otherwise (see Fig. 4b). Therefore, we have that the concatenation $T_{j_*} := A_{j_*} \cdot W_{j_*} \cdot B_{j_*}$ belongs to $CS(S_1, S_2)$, and it always has a matching in $\mathcal{B}(S_1, S_2)$. From (P3) and (P4), we can show the next claim.

▷ Claim 25. If M_* is maximal and contains $A_i \cdot W_{j_*} \cdot B_k$, then $i = j_* = k$ holds.

From Claim 25, we conclude that $T_{j_*} = A_j \cdot W_j \cdot B_j$ is the all and only members of $LCS(S_1, S_2)$ for all $j \in [s]$. Since $A_j \cdot W_j \cdot B_j = P_j \cdot X_j \cdot B_j = T_j$, the lemma is proved. ◀

Using Lemma 23, we finish the proof for Theorem 19.

Proof for Theorem 19. Recall that integers $K, r \geqslant 1$, $\Delta \geqslant 0$, and a string set $L = \{X_1, \ldots, X_s\} \subseteq \Sigma^r$ of r-strings form an instance of Max-Min Diverse String Set. Let $\Delta' := \Delta + 2s$, $K' = \kappa(K) := K$, and $\mathcal{S} = \{S_1, S_2\} \subseteq \Gamma^*$ be the associated instance of

Max-Min Diverse LCS for two input strings. Since the parameter $\kappa(K) = K$ depends only on K, it is obvious that this transformation can be computed in FPT. We show that this forms an FPT-reduction from the former problem to the latter problem. By Lemma 23, we have the next claim.

▷ Claim 26. For any $i, j \in [s]$, $d_H(T_i, T_j) = d_H(X_i, X_j) + 2s$.

Proof of Claim 26. By Lemma 23, $LCS(S_1, S_2) = \{ T_j \mid i \in [K] \}$. By construction, $T_j = P_j \cdot X_j \cdot Q_j$ and $|P_j| = |Q_j| = s$, and $d_H(P_i, P_j) = d_H(Q_i, Q_j) = s$ for any $i, j \in [s]$, $i \neq j$. Therefore, we can decompose $d_H(T_i, T_j)$ by $d_H(T_i, T_j) = d_H(P_i, P_j) + d_H(X_i, X_j) + d_H(Q_i, Q_j) = d_H(X_i, X_j) + 2s$ ◁

Suppose that $\mathcal{Y} \subseteq LCS(S_1, S_2)$ is any feasible solution such that $|\mathcal{Y}| = K'$ for Max-Sum Diverse LCSs, where $K' = K$. From Lemma 23, we can put $\mathcal{Y} = \{T_{i_j}\}_{j \in J}$ for some $J \subseteq [s]$. From Claim 26, we can see that $D^{\min}_{d_H}(\mathcal{Y}) = D^{\min}_{d_H}(\mathcal{X}) + 2s$, where $\mathcal{X} = \{X_j\}_{j \in J}$ is a solution for Max-Min Diverse String Set. Thus, $D^{\min}_{d_H}(\mathcal{X}) \geqslant \Delta$ if and only if $D^{\min}_{d_H}(\mathcal{Y}) \geqslant \Delta + 2s = \Delta'$. This shows that the transformation properly forms NP- and FPT-reductions. To obtain NP- and FPT-reductions for the Max-Sum version, we keep K and $\mathcal{S} = \{S_1, S_2\}$ in the previous proof, and modify $\Delta' := \Delta + 2s\binom{K}{2}'$, where $\binom{K}{2}' := \{(i, j) \in \binom{K}{2} \mid i \neq j\}$. From Claim 26, we have that $D^{\text{sum}}_{d_H}(\mathcal{Y}) = D^{\text{sum}}_{d_H}(\mathcal{X}) + 2s\binom{K}{2}'$, and the correctness of the reduction immediately follows. Combining the above arguments, the theorem is proved. ◀

References

1 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

2 Emmanuel Arrighi, Henning Fernau, Mateus de Oliveira Oliveira, and Petra Wolf. Synchronization and diversity of solutions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37(10), pages 11516–11524, 2023.

3 Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 2012.

4 Giorgio Ausiello, Alessandro D'Atri, and Marco Protasi. Structure preserving reductions among convex optimization problems. *JCSS*, 21(1):136–153, 1980.

5 Julien Baste, Michael R. Fellows, Lars Jaffke, Tomáš Masařík, Mateus de Oliveira Oliveira, Geevarghese Philip, and Frances A Rosamond. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artificial Intelligence*, 303:103644, 2022.

6 Julien Baste, Lars Jaffke, Tomáš Masařík, Geevarghese Philip, and Günter Rote. FPT algorithms for diverse collections of hitting sets. *Algorithms*, 12(12):254, 2019.

7 Benjamin Birnbaum and Kenneth J. Goldman. An improved analysis for a greedy remote-clique algorithm using factor-revealing LPs. *Algorithmica*, 55(1):42–59, 2009.

8 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Harold T. Wareham. The parameterized complexity of sequence alignment and consensus. *Theoretical Computer Science*, 147(1-2):31–54, 1995.

9 Laurent Bulteau, Mark Jones, Rolf Niedermeier, and Till Tantau. An FPT-algorithm for longest common subsequence parameterized by the maximum number of deletions. In *33rd Ann. Symp. on Combinatorial Pattern Matching (CPM 2022), LIPIcs*, volume 223, pages 6:1–6:11, 2022.

10 Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. An improved analysis of local search for max-sum diversification. *Math. Oper. Research*, 44(4):1494–1509, 2019.

11 Barun Chandra and Magnús M Halldórsson. Approximation algorithms for dispersion problems. *J. of Algorithms*, 38(2):438–465, 2001.

12 Alessio Conte, Roberto Grossi, Giulia Punzi, and Takeaki Uno. A compact DAG for storing and searching maximal common subsequences. In *ISAAC 2023*, pages 21:1–21:15, 2023.

13 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, fourth edition*. MIT Press, 2022.

14 Graham Cormode and Shan Muthukrishnan. The string edit distance matching problem with moves. *ACM Transactions on Algorithms (TALG)*, 3(1):1–19, 2007.

15 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

16 Michel Deza and Monique Laurent. *Geometry of Cuts and Metrics*, volume 15 of *Algorithms and Combinatorics*. Springer, 1997.

17 Rodney G. Downey and Michael R. Fellows. *Parameterized complexity*. Springer, 2012.

18 Erhan Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46(1):48–60, 1990.

19 Michael R. Fellows, Christian Knauer, Naomi Nishimura, Prabhakar Ragde, Frances A. Rosamond, Ulrike Stege, Dimitrios M. Thilikos, and Sue Whitesides. Faster fixed-parameter tractable algorithms for matching and packing problems. *Algorithmica*, 52:167–176, 2008.

20 Michael R. Fellows and Frances A. Rosamond. The DIVERSE X Paradigm (Open problems). In Henning Fernau, Petr Golovach, Marie-France Sagot, et al., editors, *Algorithmic enumeration: Output-sensitive, input-sensitive, parameterized, approximative (Dagstuhl Seminar 18421), Dagstuhl Reports, 8(10)*, 2019.

21 Dana Fisman, Joshua Grogin, Oded Margalit, and Gera Weiss. The normalized edit distance with uniform operation costs is a metric. In *33rd Ann. Symp. on Combinatorial Pattern Matching (CPM 2022), LIPIcs*, volume 223, pages 17:1–17:17, 2022.

22 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer, 2006.

23 Michael R. Garey and David S. Johnson. Computers and intractability: A guide to the theory of NP-completeness, 1979.

24 Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB*, volume 99(6), pages 518–529, 1999.

25 Dan Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, 55(1):141–154, 1993.

26 Koji Hakata and Hiroshi Imai. The longest common subsequence problem for small alphabet size between many strings. In *ISAAC'92*, pages 469–478. Springer, 1992.

27 Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, Yusuke Kobayashi, Kazuhiro Kurita, and Yota Otachi. A framework to design approximation algorithms for finding diverse solutions in combinatorial problems. In *AAAI 2023*, pages 3968–3976, 2023.

28 Tesshu Hanaka, Yasuaki Kobayashi, Kazuhiro Kurita, and Yota Otachi. Finding diverse trees, paths, and more. In *AAAI 2021*, pages 3778–3786, 2021.

29 Pierre. Hansen and I.D. Moon. Dispersing facilities on a network. In *the TIMS/ORSA Joint National Meeting, Washington, D.C.* RUTCOR, Rutgers University., 1988. Presentation.

30 Miyuji Hirota and Yoshifumi Sakai. Efficient algorithms for enumerating maximal common subsequences of two strings. *CoRR*, abs/2307.10552, 2023. `arXiv:2307.10552`.

31 Daniel S Hirschberg. *Recent results on the complexity of common-subsequence problems, in Time warps, String edits, and Macromolecules*, pages 323–328. Addison-Wesley, 1983.

32 John E. Hopcroft and Jeff D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

33 Robert W. Irving and Campbell B. Fraser. Two algorithms for the longest common subsequence of three (or more) strings. In *CPM 1992*, pages 214–229. Springer, 1992.

34 Michael J. Kuby. Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems. *Geographical Analysis*, 19(4):315–329, 1987.

35 Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8):707–710, 1966.

36 Ohad Lipsky and Ely Porat. L1 pattern matching lower bound. *Information Processing Letters*, 105(4):141–143, 2008.

37 Mi Lu and Hua Lin. Parallel algorithms for the longest common subsequence problem. *IEEE Transactions on Parallel and Distributed Systems*, 5(8):835–848, 1994.

38 David Maier. The complexity of some problems on subsequences and supersequences. *J. ACM*, 25(2):322–336, 1978.

39 Tatsuya Matsuoka and Shinji Ito. Maximization of minimum weighted Hamming distance between set pairs. In *Asian Conference on Machine Learning*, pages 895–910. PMLR, 2024.

40 Sekharipuram S. Ravi, Daniel J. Rosenkrantz, and Giri Kumar Tayi. Heuristic and special case algorithms for dispersion problems. *Operations research*, 42(2):299–310, 1994.

41 David Sankoff. Matching sequences under deletion/insertion constraints. *Proceedings of the National Academy of Sciences*, 69(1):4–6, 1972.

42 Douglas R. Shier. A min-max theorem for p-center problems on a tree. *Transportation Science*, 11(3):243–252, 1977.

43 Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2010.

44 Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, 1974.

45 Da-Wei Wang and Yue-Sun Kuo. A study on two geometric location problems. *Information Processing Letters*, 28(6):281–286, 1988.

46 Dan E. Willard. Log-logarithmic worst-case range queries are possible in space $\theta(n)$. *Information Processing Letters*, 17(2):81–84, 1983.