The Fine-grained Complexity of Median and Center String Problems under Edit Distance

Gary Hoppenworth
Department of Computer Science, University of Central Florida, USA
gary.hoppenworth@gmail.com

Jason W. Bentley
Department of Mathematics, University of Central Florida, USA
jason.bentley@ucf.edu

Daniel Gibney
Department of Computer Science, University of Central Florida, USA
https://www.cs.ucf.edu/~dgibney/
daniel.j.gibney@gmail.com

Sharma V. Thankachan
Department of Computer Science, University of Central Florida, USA
http://www.cs.ucf.edu/~sharma/
sharma.thankachan@ucf.edu

Abstract

We present the first fine-grained complexity results on two classic problems on strings. The first one is the k-Median-Edit-Distance problem, where the input is a collection of k strings, each of length at most n, and the task is to find a new string that minimizes the sum of the edit distances from itself to all other strings in the input. Arising frequently in computational biology, this problem provides an important generalization of edit distance to multiple strings and is similar to the multiple sequence alignment problem in bioinformatics. We demonstrate that for any $\varepsilon > 0$ and $k \geq 2$, an $O(n^{k-\varepsilon})$ time solution for the k-Median-Edit-Distance problem over an alphabet of size $O(k)$ refutes the Strong Exponential Time Hypothesis (SETH). This provides the first matching conditional lower bound for the $O(n^k)$ time algorithm established in 1975 by Sankoff.

The second problem we study is the k-Center-Edit-Distance problem. Here also, the input is a collection of k strings, each of length at most n. The task is to find a new string that minimizes the maximum edit distance from itself to any other string in the input. We prove that the same conditional lower bound as before holds. Our results also imply new conditional lower bounds for the k-Tree-Alignment and the k-Bottleneck-Tree-Alignment problems studied in phylogenetics.

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

Keywords and phrases Edit Distance, Median String, Center String, SETH.

Digital Object Identifier 10.4230/LIPIcs.ESA.2020.78

Funding Supported in part by the U.S. National Science Foundation (NSF) under CCF-1703489.

28th Annual European Symposium on Algorithms (ESA 2020).
Editors: Fabrizio Grandoni, Peter Sanders, and Grzegorz Herman; Article No. 78; pp. 78:1–78:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Recent years have seen a remarkable increase in our understanding of the hardness of problems in the complexity class P. By establishing conditional lower bounds based on popular conjectures, researchers have been able to identify which problems are unlikely to yield algorithms significantly faster than what is known, at least not without solving other long-standing open questions. We contribute to this growing body of research here by establishing tight conditional hardness results for the k-Median-Edit-Distance problem. This generalizes the seminal work by Backurs and Indyk in STOC 2015, which showed that conditioned on the Strong Exponential Time Hypothesis (SETH), there does not exist a strongly subquadratic algorithm for computing the edit distance between two strings [10].

▶ **Problem 1** (k-Median-Edit-Distance). *Given a set $\mathcal{S}$ of k strings, each of length at most n, find a string s^* (called a median string) that minimizes the sum of edit distances from the strings in $\mathcal{S}$ to s^*. This sum is called the median edit distance.*

When $k = 2$ this problem is equivalent to the well-known edit distance problem, whose famous dynamic programming solution was first given in 1965 by Vintsyuk [44]. An algorithm for solving this problem on k strings in time $O(n^k)$ was then given by Sankoff in 1975 [41] in the more general context of tree alignment (mutation trees). Since Sankoff's solution, no algorithms with significantly better time complexity have been developed. This is despite the problem being of practical importance as well as the subject of extensive study [29, 30, 33, 38]. Compelling reasons for this were finally given 25 years later by Higuera and Casacuberta in 2000 who showed the NP-completeness of the problem over unbounded alphabets [20]. This result was later strengthened to finite alphabets in [42] and then even to binary alphabets in [39]. In [39] it was also shown that the problem is W[1]-hard in k. This last result implies it is highly unlikely to find an algorithm with time complexity of the form $f(k) \cdot N^{O(1)}$, where N is the sum of the lengths of the k strings. None of these hardness results, however, rule out the possibility of algorithms where the time complexity is of the form $O(n^{k-\varepsilon})$. Nearly five decades after its creation, this paper gives a convincing argument as to why a significant improvement over Sankoff's algorithm is unlikely. Specifically, we show that an $O(n^{k-\varepsilon})$ time algorithm for any $\varepsilon > 0$ would refute SETH. We also prove that the same lower bound holds for a related problem known as the k-Center-Edit-Distance.

▶ **Problem 2** (k-Center-Edit-Distance). *Given a set $\mathcal{S}$ of k strings, each of length at most n, find a string s^* (called a center string) that minimizes the maximum of edit distances from the strings in $\mathcal{S}$ to s^*. The maximum edit distance from s^* to any string in $\mathcal{S}$ is called the center edit distance.*

Like k-Median-Edit-Distance, the k-Center-Edit-Distance problem is known to be NP-complete and W[1]-hard in k [39]. Additionally, k-Center-Edit-Distance has been shown to have an $O(n^{2k})$ time solution [39]. However, ours are the first fine-grained complexity results for both these problems. Finally, we note that our results imply similar conditional lower bounds for two classic tree alignment problems from phylogenetics called k-Tree-Alignment and k-Bottleneck-Tree-Alignment [18, 28, 43, 45]. The k-Tree-Alignment (resp. k-Bottleneck-Tree-Alignment) problem is defined as follows: given a tree $\mathcal{T}$ with k leaves where each leaf is labeled with a string of length n, find an assignment of strings to all internal vertices of $\mathcal{T}$ such that the sum (resp. max) of edit distances between adjacent strings/vertices over all edges is minimal. Note that the median (resp. center) edit distance problem on k strings is a special case of the k-Tree-Alignment (resp. k-Bottleneck-Tree-Alignment) problem, specifically when the tree has only one internal vertex.

1.1 Related Work

Recent progress in the field of fine-grained complexity has given us conditional hardness results for many popular problems. The list of problems includes those related to graphs, computational geometry, and strings [1, 3, 4, 6, 7, 8, 10, 15, 17, 19, 21, 24, 23, 31, 32]. Reductions based on SETH, such as the one considered here, tend to have a very similar structure. The Orthogonal Vectors problem [46] is typically used as an intermediate step in the reduction. The proof we provide here works off a generalized variant of the Orthogonal Vectors problem as used in [2]. Our work contributes to a growing list of conditional lower bounds for string problems which we describe in more detail below.

Along with the SETH-based lower bound for edit distance by Backurs and Indyk in [10], there has been a number of newly appearing conditional lower bounds for string related problems [9, 12, 14, 16]. Bringmann and Künnemann created a framework by which any string problem which allowed for a particular gadget construction has similar SETH-based lower bounds proven for it [13]. This framework includes the problems of the longest common subsequence, dynamic time warping, and edit distance under a binary alphabet (less than the four symbols used in the original reduction by Backurs and Indyk). Further work to extend these types of lower bounds to more than two strings was undertaken in [2], where it was shown that an algorithm which could find the longest common subsequence on k strings in time $O(n^{k-\varepsilon})$ for any $\varepsilon > 0$ would refute SETH. The study of conditional hardness of problems on k strings also includes [22], where the longest common increasing subsequence on k strings, k-LCIS, was studied. Likewise in [7] the local alignment problem on k strings under sum of pairs was considered. In both of the last two works mentioned, it was shown that an $O(n^{k-\varepsilon})$ algorithm would refute SETH.

Another notable achievement in this direction is in [5], where it was shown that it is possible to weaken the assumptions used to achieve many of these results. They showed that under much weaker conjectures than SETH regarding circuit complexity, many of the same hardness results still hold. In fact, for any problem where the gadgetry of Bringmann and Künnemann can be applied, having a strongly sub-quadratic time algorithm would have drastic implications for our ability to solve satisfiability problems on Boolean circuits much more complex than those required for 3-SAT. Furthermore, their work also demonstrated that if one could shave off arbitrarily large logarithmic factors, it would have drastic implications in the field of circuit complexity. In this same work, they showed that their reduction from branching programs to string problems can be adapted to k-LCS, implying circuit-based hardness results apply for LCS on k strings.

There exists a close relationship between LCS and edit distance on two strings. Namely, on two strings of lengths n and m, the edit distance with only the insertion and deletion operations is equal to $n + m - 2\ell$, where ℓ is the length of the strings' longest common subsequence. For more than two strings, such a clear relationship (in terms of just lengths and number of edits) seems unlikely. In fact, there exist collections of k strings where the lengths of the longest common subsequences are equal, but the median edit distances are not, e.g., with $k = 3$ and $n \geq 1$, the sets $\{a^n, a^n, b^n\}$ and $\{a^n, b^n, c^n\}$ both have a longest common subsequence of length zero, while the first has median edit distance n and the second has median edit distance $2n$. Because of this, it seems difficult to parlay the hardness results proven for k-LCS into hardness results for k-Median-Edit-Distance, even under only insertions and deletions. Hence, the hardness of k-Median-Edit-Distance was left open. On the other hand, a 2-approximation for k-Median-Edit-Distance can be easily obtained in $O(k^2n^2)$ time: simply choose the string within the collection that minimizes the sum of edit distances from itself to the other strings.

The problem of finding the center string of a set of k strings, the string which minimizes the maximum distance from itself to any string in the set, has more often been studied under the Hamming distance metric than the edit distance metric. In this context the problem is typically called the closest string problem [25, 27, 35, 36]. The problem under the Hamming distance metric is NP-complete [34], whereas the median version under Hamming distance can be easily solved in polynomial time. In the cases where this problem has been studied under the edit distance metric, it has made use of a parameter d, the maximum distance any solution is allowed to have from an input string. The problem is fixed parameter tractable in d, which is the basis of many solutions [11, 26, 37].

2 Hardness for k-Median-Edit-Distance

Our reduction will be from the k-Most-Orthogonal-Vectors problem, which was first introduced in [2]. It was shown that if it could be solved in $\mathcal{O}(n^{k-\varepsilon})$ time for some constant $\varepsilon > 0$, it would imply new upper bounds for MAX-CNF-SAT that would violate SETH.

▶ **Problem 3** (k-Most-Orthogonal-Vectors)**.** *Given $k \geq 2$ sets $S_1, S_2, \ldots, S_k$ each containing n binary vectors $v \in \{0,1\}^d$, and an integer $r < d$, are there k vectors $v_1, v_2, \ldots, v_k$ with $v_i \in S_i$ such that their inner product, defined as $\sum_{h=1}^{d} \prod_{t\in[1,k]} v_t[h]$, is at most r? A collection of vectors that satisfies this property will be called r-far, and otherwise called r-close.*

Modifying the Vectors: In our reduction we apply a modification to the vectors in our input sets $S_1, S_2, \ldots, S_k$. We prepend $(r+1)$ 0's to each vector $v \in S_1$ and $(r+1)$ 1's to each vector $v \in S_i$ where $i > 1$. Every vector is now of dimension $d + r + 1 \leq 2d$ and the k-Most-Orthogonal-Vectors problem is identical on the original and modified sets.

2.1 Technical Overview

Given sets $S_1, S_2, \ldots, S_k$ of binary vectors, we will design strings $T_1, T_2, \ldots, T_k$ such that if there exists a collection of r-far vectors in the input, then their median edit distance will be at most a constant E^-. Otherwise, if there does not exist any collection of r-far vectors in the input, their median edit distance will be equal to E^+, where $E^- < E^+$. Our strings will be constructed in three levels of increasing scope: coordinate level, vector level, and set level. We use $\mathrm{EDIT}(x_1, x_2, \ldots, x_k)$ to denote the *median edit distance* of k strings $x_1, x_2, \ldots, x_k$.

- **Coordinate Level:** Given k bits $b_1, b_2, \ldots, b_k$, we construct *coordinate gadget* strings $\mathrm{CG}_i(b_i)$ that can distinguish between the case when $b_1 b_2 \cdots b_k = 0$ and $b_1 b_2 \cdots b_k = 1$. Specifically, we will show that there exist constants C^- and C^+ with $C^- < C^+$ such that if $b_1 b_2 \cdots b_k = 0$, then $\mathrm{EDIT}(\mathrm{CG}_1(b_1), \mathrm{CG}_2(b_2), \ldots, \mathrm{CG}_k(b_k)) = C^-$, and else if $b_1 b_2 \cdots b_k = 1$, then $\mathrm{EDIT}(\mathrm{CG}_1(b_1), \mathrm{CG}_2(b_2), \ldots, \mathrm{CG}_k(b_k)) = C^+$.
- **Vector Level:** Given vectors $v_1, v_2, \ldots, v_k \in \{0,1\}^{d+r+1}$, we construct *vector gadget* strings $\mathrm{VG}_i(v_i)$ for $i \in [2, k]$ and a slightly more complicated *decision gadget* string $\mathrm{DG}_1(v_1)$ out of our coordinate gadgets. Together these gadgets can determine if the k vectors are r-far or not. Specifically, we will show that if $v_1, v_2, \ldots, v_k$ are r-far, then $\mathrm{EDIT}(\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \ldots, \mathrm{VG}_k(v_k)) \leq D^-$ and else if $v_1, v_2, \ldots, v_k$ are r-close, then $\mathrm{EDIT}(\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \ldots, \mathrm{VG}_k(v_k)) = D^+$, where D^+ and $D^- < D^+$ are constants. Our construction here is a generalization of the work in [10] to k strings.
- **Set Level:** In the set level step of the reduction, we will build our final strings $T_1, T_2, \ldots, T_k$ by concatenating our vector level gadgets and adding special $\$_i$ symbols. Our final strings will be designed so that if there is an r-far collection of vectors $v_1, v_2, \ldots, v_k$

with $v_i \in S_i$, then the corresponding gadgets $\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \mathrm{VG}_3(v_3), \ldots, \mathrm{VG}_k(v_k)$ will align in an optimal edit sequence of our strings. These vector gadgets will have a lower median edit distance, resulting in $\mathrm{EDIT}(T_1, T_2, \ldots, T_k) \leq E^-$. Otherwise, $\mathrm{EDIT}(T_1, T_2, \ldots, T_k) = E^+$, where $E^- < E^+$.

We now present a definition and an associated fact.

▶ **Definition 1** (Alignment). *Given a particular edit sequence (a sequence of insertions, substitutions, and deletions) on strings $x_1, x_2, \ldots, x_k$, we say symbol α in x_i is aligned with symbol β in another string x_j if neither α nor β is deleted but are instead preserved or substituted to correspond to the same symbol. We say a substring s of x_i is aligned with substring t of x_j, if there exists a pair of aligned characters in s and t.*

The following observation will be used implicitly throughout.

▶ **Fact 1** (No criss-crossed alignments). *Consider an edit sequence on a set of strings containing strings x and y. Let $i_1 < j_1$ and $i_2 < j_2$ be indices on these strings. If $x[i_1]$ is aligned with $y[j_2]$, then $x[i_2]$ cannot be aligned with $y[j_1]$.*

2.2 Coordinate level reduction

For $i \in [1, k]$, we define coordinate gadget strings CG_i over the alphabet $\Sigma = \{2_1, 2_2, \ldots, 2_k, 3, 4\}$. Let $\ell_1 = 10k^2$. For bits $b_1, b_2, \ldots, b_k \in \{0, 1\}$, we define

$$\mathrm{CG}_i(b_i) \coloneqq f_i(b_i) \circ 4^{\ell_1} \circ g_i(b_i) \circ 4^{\ell_1} \circ h_i(b_i) \quad \text{for } i \in [1, k], \text{ where}$$

$$f_i(b_i) = \begin{cases} 2_{i+1}^{k-1} & \text{if } b_i = 1, i < k \\ 2_1^{k-1} & \text{if } b_i = 1, i = k \\ 2_i^{k-1} & \text{if } b_i = 0 \end{cases} \quad g_i(b_i) = \begin{cases} 3^{k-1} & \text{if } b_i = 1 \\ 2_i^{k-1} & \text{if } b_i = 0 \end{cases} \quad h_i(b_i) = \begin{cases} 2_i^k & \text{if } b_i = 1 \\ \bigcirc_{j=1}^k 2_j & \text{if } b_i = 0 \end{cases}$$

We present the following examples on $k = 3$ to aid in the understanding of our $\mathrm{CG}_i(b_i)$.

b_1, b_2, b_3	$f_1(b_1), f_2(b_2), f_3(b_3)$	$g_1(b_1), g_2(b_2), g_3(b_3)$	$h_1(b_1), h_2(b_2), h_3(b_3)$	$\mathrm{EDIT}(\mathrm{CG}_1(b_1), \cdot, \cdot)$
$1, 1, 1$	$2_2 2_2, 2_3 2_3, 2_1 2_1$	$33, 33, 33$	$2_1 2_1 2_1, 2_2 2_2 2_2, 2_3 2_3 2_3$	$4 + 0 + 6 = 10$
$0, 1, 1$	$2_1 2_1, 2_3 2_3, 2_1 2_1$	$2_1 2_1, 33, 33$	$2_1 2_2 2_3, 2_2 2_2 2_2, 2_3 2_3 2_3$	$2 + 2 + 4 = 8$
$0, 0, 0$	$2_1 2_1, 2_2 2_2, 2_3 2_3$	$2_1 2_1, 2_2 2_2, 2_3 2_3$	$2_1 2_2 2_3, 2_1 2_2 2_3, 2_1 2_2 2_3$	$4 + 4 + 0 = 8$

▶ **Lemma 2.** *Let $C^- = 2(k-1)^2$ and let $C^+ = C^- + (k-1) = (2k-1)(k-1)$. Then*

$$\mathrm{EDIT}(\mathrm{CG}_1(b_1), \mathrm{CG}_2(b_2), \ldots, \mathrm{CG}_k(b_k)) = \begin{cases} C^+ & \textit{if } b_1 b_2 \cdots b_k = 1 \\ C^- & \textit{otherwise} \end{cases}$$

Proof. For the remainder of this proof, let $\pi = b_1 + b_2 + \cdots + b_k \in [0, k]$.

▷ Claim 3. The median edit distance of our f_i gadgets is

$$\mathrm{EDIT}(f_1(b_1), \ldots, f_k(b_k)) = \begin{cases} (k-1)^2 & \text{if } \pi = 0 \text{ or } k \\ (k-1)(k-2) & \text{otherwise} \end{cases}$$

▷ Claim 4. The median edit distance of our g_i gadgets is

$$\mathrm{EDIT}(g_1(b_1), \ldots, g_k(b_k)) = \begin{cases} (k-1)^2 & \text{if } \pi = 0 \\ (k-1)(k-\pi) & \text{otherwise} \end{cases}$$

▷ Claim 5. The median edit distance of our h_i gadgets is $\mathrm{EDIT}(h_1(b_1), \ldots, h_k(b_k)) = (k-1)\pi$.

We have chosen ℓ_1 to be sufficiently large that all f_i, g_i, and h_i gadgets align only with gadgets of their own type. Therefore,

$$\mathrm{EDIT}(\mathrm{CG}_1(b_1), \ldots, \mathrm{CG}_k(b_k)) = \begin{cases} (k-1)^2 + (k-1)^2 + 0 & \pi = 0 \\ (k-1)(k-2) + (k-1)(k-\pi) + (k-1)\pi & 0 < \pi < k \\ (k-1)^2 + 0 + (k-1)k & \pi = k \end{cases}$$

A simple calculation will show that $\mathrm{EDIT}(\mathrm{CG}_1(b_1), \ldots, \mathrm{CG}_k(b_k))$ is C^- when $\pi < k$ (and hence $b_1 b_2 \cdots b_k = 0$) and is C^+ when $\pi = k$ (and hence $b_1 b_2 \cdots b_k = 1$). ◀

2.3 Vector level reduction

At this step of the reduction we are given binary vectors $v_1, v_2, \ldots, v_k \in \{0,1\}^{d+r+1}$ and we want to determine whether or not they are r-far. We accomplish this by constructing vector level gadgets that will have a 'lower' median edit distance if the vectors are r-far. Let integer parameters $\ell_2 = 10d\ell_1$ and $\ell_3 = (10\ell_2)^2$. For vectors $v_1, v_2, \ldots, v_k$, we define

$$\mathrm{VG}_i(v_i) := 6^{\ell_3} \circ M_i(v_i) \circ 6^{\ell_3} \qquad \text{where} \qquad M_i(v_i) \quad := \bigcirc_{j\in[1,d+r+1]}(5^{\ell_2} \circ \mathrm{CG}_i(v_i[j]) \circ 5^{\ell_2})$$

Observe that the vector gadget of a vector v_i is just the concatenation of the coordinate gadgets corresponding to each coordinate in v_i, along with some additional padding symbols. It follows that the median edit distance of $\mathrm{VG}_1(v_1), \mathrm{VG}_2(v_2), \ldots, \mathrm{VG}_k(v_k)$ will be proportional to the inner product of $v_1, v_2, \ldots, v_k$. This is promising because we can now argue about whether or not $v_1, v_2, \ldots, v_k$ are r-far based on the median edit distance of the $\mathrm{VG}_i(v_i)$'s (a 'lower' distance implies the vectors are r-far and a 'higher' distance implies the vectors are r-close). Unfortunately, vectors with a very large inner product will result in a large median edit distance, which could interfere with our ability to detect r-far vectors in the next step of our reduction. What is desired here is to have vector level gadgets with a fixed 'higher' median edit distance when the vectors are r-close. We achieve this by replacing $\mathrm{VG}_1(v_1)$ with a decision gadget $\mathrm{DG}_1(v_1)$ that will ensure that no matter how large the inner product of a collection of r-close vectors, the median edit distance of their corresponding gadgets will be a constant D^+. For vector v_1, we define

$$\mathrm{DG}_1(v_1) := 7^{\ell_3} \circ M_1(v_1) \circ 6^{\ell_3} \circ M_1(\theta) \circ 7^{\ell_3}, \qquad \theta \in \{0,1\}^{d+r+1} \text{ and } \theta[i] = \begin{cases} 1 & i \le r+1 \\ 0 & \text{else} \end{cases}$$

The key properties of our vector level gadgets are captured in Lemma 6 and Lemma 7. In both proofs we let $m = |M_i| = (d+r+1)(2\ell_2 + 2\ell_1 + 3k - 2)$, and we define $D^- = 2\ell_3 + m + (d+1)C^- + rC^+$ and $D^+ = D^- + (k-1)$.

▶ **Lemma 6.** *For any given r-far vectors $v_1, v_2, \ldots, v_k \in \{0,1\}^{d+r+1}$, $\mathrm{EDIT}(\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \mathrm{VG}_3(v_3), \ldots, \mathrm{VG}_k(v_k)) \le D^-$.*

Proof. To upper bound the median edit distance of our k strings by D^-, we must give a complete edit sequence of our strings that requires D^- or fewer edits. Let $v_1, v_2, \ldots, v_k$ be r-far vectors. We decide to align $\mathrm{VG}_2(v_2), \mathrm{VG}_3(v_3), \ldots, \mathrm{VG}_k(v_k)$ with the $7^{\ell_3} \circ M_1(v_1) \circ 6^{\ell_3}$ substring of $\mathrm{DG}_1(v_1)$ as in Figure 1.

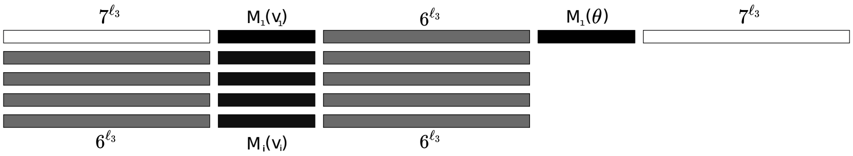

Figure 1 An optimal alignment of $\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \ldots, \mathrm{VG}_k(v_k)$ when $v_1, v_2, \ldots, v_k$ are r-far.

First we delete $M_1(\theta) \circ 7^{\ell_3}$ from $\mathrm{DG}_1(v_1)$ in $m + \ell_3$ edits. Then we substitute all the 7 symbols in the 7^{ℓ_3} prefix of $\mathrm{DG}_1(v_1)$ to 6 symbols in ℓ_3 edits. Finally, we must edit substrings $M_1(v_1), M_2(v_2), \ldots, M_k(v_k)$ to be the same. Each $M_i(v_i)$ contains $d + r + 1$ coordinate gadgets, and for $j \in [1, d + r + 1]$, we choose to align the jth leftmost coordinate gadgets of all $M_i(v_i)$ for $i \in [1, k]$. Note that the inner product of $v_1, v_2, \ldots, v_k$ is less than or equal to r because the vectors are r-far. It follows that we will have no more than r alignments of coordinate gadgets with cost C^+ and at least $d+1$ alignments with cost C^- (recall Lemma 2). Then $\mathrm{EDIT}(M_1(v_1), M_2(v_2), \ldots, M_k(v_k)) \leq (d + 1)C^- + rC^+$. The total number of edits performed in this edit sequence is at most $2\ell_3 + m + (d + 1)C^- + rC^+ = D^-$. ◀

We note that if $v_1, v_2, \ldots, v_k$ are r-close and as a result have an inner product greater than r, the optimal edit sequence of $\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \ldots, \mathrm{VG}_k(v_k)$ will align strings $\mathrm{VG}_2(v_2), \mathrm{VG}_3(v_3), \ldots, \mathrm{VG}_k(v_k)$ with the $6^{\ell_3} \circ M_1(\theta) \circ 7^{\ell_3}$ substring of $\mathrm{DG}_1(v_1)$ as in Fig. 2.

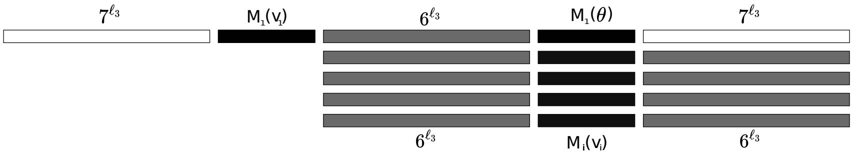

Figure 2 An optimal alignment of $\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \ldots, \mathrm{VG}_k(v_k)$ when $v_1, v_2, \ldots, v_k$ are r-close.

▶ **Lemma 7.** *For any given r-close vectors $v_1, v_2, \ldots, v_k \in \{0, 1\}^{d+r+1}$, $\mathrm{EDIT}(\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \mathrm{VG}_3(v_3), \ldots, \mathrm{VG}_k(v_k)) = D^+$.*

Proof. The proof of Lemma 7 is a straightforward generalization of the vector gadget proof in [10] to k strings. In the course of this proof we will make use of the fact that for any subset $x_{i_1}, x_{i_2}, \ldots, x_{i_j}$ of strings $x_1, x_2, \ldots, x_k$, $\mathrm{EDIT}(x_{i_1}, x_{i_2}, \ldots, x_{i_j}) \leq \mathrm{EDIT}(x_1, x_2, \ldots, x_k)$.

▷ Claim 8. $\mathrm{EDIT}(\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \mathrm{VG}_3(v_3) \ldots, \mathrm{VG}_k(v_k)) \leq D^+$

Subproof. Note that the inner product of $\theta, v_2, v_3, \ldots, v_k$ is equal to $r+1$ by the definition of θ and our modifications to the input vectors. Then we can align $\mathrm{VG}_2(v_2), \mathrm{VG}_3(v_3), \ldots, \mathrm{VG}_k(v_k)$ with the $6^{\ell_3} \circ M_1(\theta) \circ 7^{\ell_3}$ substring of $\mathrm{DG}_1(v_1)$ in a manner analogous to our edit sequence in Lemma 6. ◀

Now we "just" need to prove that $\mathrm{EDIT}(\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \ldots, \mathrm{VG}_k(v_k)) \geq D^+$. We proceed by cases on the alignments of the $M_i(v_i)$ substrings.

▷ Claim 9. $\mathrm{EDIT}(\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \mathrm{VG}_3(v_3) \ldots, \mathrm{VG}_k(v_k)) \geq D^+$

Subproof. We have the following cases to consider.

- **Case 1:** The $M_i(v_i)$ substring of some $\mathrm{VG}_i(v_i)$ gadget with $i > 1$ has alignments with both substrings $7^{\ell_3} \circ M_1(v_1)$ and $M_1(\theta) \circ 7^{\ell_3}$ of $\mathrm{DG}_1(v_1)$. In this case, the cost induced by the symbols in the 7^{ℓ_3} prefix and suffix of $\mathrm{DG}_1(v_1)$ and the 6^{ℓ_3} substring of $\mathrm{DG}_1(v_1)$ is ℓ_3 each, so $\mathrm{EDIT}(\mathrm{VG}_i(v_i), \mathrm{DG}_1(v_1)) \geq 3\ell_3 > D^+$. Our lower bound is satisfied. Note that since the inequality is strict, this case will not occur in an optimal edit sequence.

- **Case 2:** The $M_i(v_i)$ substring of some $\mathrm{VG}_i(v_i)$ gadget with $i > 1$ does not have any alignments with the $7^{\ell_3} \circ M_1(v_1)$ substring of $\mathrm{DG}_1(v_1)$.
- **Case 2.1:** The $M_j(v_j)$ substring of some $\mathrm{VG}_j(v_j)$ gadget with $j > 1$ does not have any alignments with substring $M_1(\theta) \circ 7^{\ell_3}$ of $\mathrm{DG}_1(v_1)$. We will consider $\mathrm{EDIT}(\mathrm{VG}_i(v_i), \mathrm{VG}_j(v_j), \mathrm{DG}_1(v_1))$, which is the same as $\mathrm{EDIT}(\mathrm{VG}_i(v_i), \mathrm{DG}_1(v_1))$ when $i = j$. The $M_i(v_i)$ substring of $\mathrm{VG}_i(v_i)$ has no alignments with the $7^{\ell_3} \circ M_1(v_1)$ substring of $\mathrm{DG}_1(v_1)$. Therefore at least $D_1 = \ell_3 + m$ edits need to be performed between the 6^{ℓ_3} prefix of $\mathrm{VG}_i(v_i)$ and the $7^{\ell_3} \circ M_1(v_1)$ prefix of $\mathrm{VG}_1(v_1)$. Likewise, the $M_j(v_j)$ substring of $\mathrm{VG}_j(v_j)$ has no alignments with the $M_1(\theta) \circ 7^{\ell_3}$ substring of $\mathrm{DG}_1(v_1)$, and so at least D_1 edits need to be performed between the 6^{ℓ_3} suffix of $\mathrm{VG}_j(v_j)$ and the $M_1(\theta) \circ 7^{\ell_3}$ suffix of $\mathrm{DG}_1(v_1)$. The above edit costs are disjoint, and it follows that $\mathrm{EDIT}(\mathrm{VG}_i(v_i), \mathrm{VG}_j(v_j), \mathrm{DG}_1(v_1)) \geq 2D_1 > D^+$. Our lower bound is satisfied.
- **Case 2.2:** We consider the complement of Case 2.1: the $M_i(v_i)$ substrings of all $\mathrm{VG}_i(v_i)$ gadgets with $i > 1$ have alignments with the substring $M_1(\theta) \circ 7^{\ell_3}$ of $\mathrm{DG}_1(v_1)$. By our analysis in Case 1, we may now assume that the $M_i(v_i)$ substrings of all $\mathrm{VG}_i(v_i)$ gadgets with $i > 1$ do not have alignments with the $7^{\ell_3} \circ M_1(v_1)$ substring of $\mathrm{DG}_1(v_1)$. Then by our argument in Case 2.1, at least D_1 edits must be performed on the 6^{ℓ_3} prefix of $\mathrm{VG}_i(v_i)$ and the $7^{\ell_3} \circ M_1(v_1)$ prefix of $\mathrm{VG}_1(v_1)$. Additionally, note that all $\mathrm{VG}_i(v_i)$ share the suffix 6^{ℓ_3}, whereas $\mathrm{DG}_1(v_1)$ has suffix 7^{ℓ_3}. It follows that at least $D_2 = \ell_3$ edits are needed to edit $\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \ldots, \mathrm{VG}_k(v_k)$ to have the same suffix. Furthermore, these edits are disjoint from the D_1 edits performed on the prefixes of $\mathrm{DG}_1(v_1)$ and the $\mathrm{VG}_i(v_i)$. We have shown that at least $D_1 + D_2 = 2\ell_3 + m$ edits are required to align $\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \ldots, \mathrm{VG}_k(v_k)$. Now all we must do is lower bound the edits internal to our $M_i(v_i)$ substrings. Recall that our $M_i(v_i)$ substrings are composed of $d + r + 1$ coordinate gadgets $\mathrm{CG}_i(v_i[j])$.
  - **Case 2.2.1:** There is some $\mathrm{VG}_i(v_i)$ gadget with $i > 1$ such that there are some $j, \ell \in [1, d + r + 1]$ with $j \neq \ell$ such that the jth leftmost coordinate gadget of $M_i(v_i)$ is aligned with the ℓth leftmost coordinate gadget of the $M_1(\theta)$ in $\mathrm{VG}_1(v_1)$. Then we incur an edit cost of at least $2\ell_2$ from the 5 symbols between the coordinate gadgets. It follows that $\mathrm{EDIT}(\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \ldots, \mathrm{VG}_k(v_k)) \geq D_1 + D_2 + 2\ell_2 > D^+$. Our lower bound is satisfied.
  - **Case 2.2.2:** We now consider the complement of Case 2.2.1. For all $i \in [1, d + r + 1]$, the ith leftmost coordinate gadget of $M_j(v_j)$ for all $j > 1$ is either aligned with the ith leftmost coordinate gadget of $M_1(\theta)$ or it's not aligned with any coordinate gadget of $M_1(\theta)$.
    * For all $i \in [1, d + r + 1]$ we analyze the edit costs of the ith leftmost coordinate gadgets in $M_1(\theta), M_2(v_2), \ldots, M_k(v_k)$. If the ith leftmost coordinate gadgets of all $M_j(v_j)$ for $j > 1$ are aligned with the ith leftmost coordinate gadget of $M_1(\theta)$. Then by the transitivity of the alignment relation, we have that the ith coordinate gadgets of $M_1(\theta), M_2(v_2), \ldots, M_k(v_k)$ are aligned. By our analysis of the coordinate gadgets in Lemma 2, this alignment of coordinate gadgets will incur cost at least C^- if $\theta[i]v_2[i]v_3[i] \ldots v_k[i] = 0$, and else incur cost at least C^+ if $\theta[i]v_2[i]v_3[i] \ldots v_k[i] = 1$.
    * Else for some $M_j(v_j)$ with $j > 1$, the ith leftmost coordinate gadget $\mathrm{CG}_j(v_j[i])$ is not aligned with any coordinate gadget of $M_1(\theta)$, then it incurs cost $|\,\mathrm{CG}_j(v_j[i])| \geq C^+$.

  Combining our case analysis for all $d + r + 1$ coordinate gadgets, we see that they collectively incur a cost of at least $D_3 = (r + 1)C^+ + dC^-$, since the inner product of vectors $\theta, v_2, v_3, \ldots, v_k$ is $r + 1$ (follows from our modification of the input vectors and our definition of θ). Then $D_1 + D_2 + D_3 = D^+$, and since the edits from D_1, D_2, and D_3

are all necessarily disjoint, we have that $\mathrm{EDIT}(\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \dots, \mathrm{VG}_k(v_k)) \geq D^+$.

- **Case 3:** The $M_i(v_i)$ substring of some $\mathrm{VG}_i(v_i)$ with $i > 1$ does not have alignments with the $M_1(\theta) \circ 7^{\ell_3}$ substring of $\mathrm{DG}_1(v_1)$. This case is symmetric to Case 2, with the only difference being that we have substring $M_1(v_1)$ as opposed to $M_1(\theta)$. Since we assumed that $v_1, v_2, \dots, v_k$ are r-close and hence have an inner product greater than or equal to $r+1$, it must be the case that $\mathrm{EDIT}(\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \dots, \mathrm{VG}_k(v_k)) \geq D^+$.

We have shown in every case that $\mathrm{EDIT}(\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \dots, \mathrm{VG}_k(v_k)) \geq D^+$, so we conclude that $\mathrm{EDIT}(\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \dots, \mathrm{VG}_k(v_k)) = D^+$. ◀

This completes the proof of Lemma 7. ◀

2.4 Set level reduction

In this step of the reduction we will construct our final strings $T_1, T_2, \dots, T_k$ that can detect r-far vectors in our input sets $S_1, S_2, \dots, S_k$. We will accomplish this by embedding in string T_i the vector level gadgets of the vectors belonging to set S_i for $i \in [1, k]$. Then if an r-far collection of vectors exists, we can align their corresponding vector gadgets and give our strings $T_1, T_2, \dots, T_k$ a ‘lower’ median edit distance.

We will construct our final strings in several steps. We start by padding our vector level gadgets to discourage them from aligning with more than one vector level gadget in any given string. We define integer parameter $\ell_4 = 10000k^4 d\ell_3$, and we add a new padding symbol 8 to our alphabet. For all $v \in \{0,1\}^{d+r+1}$, let

$$\mathrm{DG}'_1(v) \coloneqq 8^{\ell_4} \circ \mathrm{DG}_1(v) \circ 8^{\ell_4} \quad \text{and} \quad \mathrm{VG}'_i(v) \coloneqq 8^{\ell_4} \circ \mathrm{VG}_i(v) \circ 8^{\ell_4} \quad \text{for } i \in [1, k]$$

We now concatenate our vector level gadgets DG'_1 and VG'_i. Define

$$P_1 \coloneqq \bigcirc_{v \in S_1} \mathrm{DG}'_1(v) \quad \text{and} \quad P_i \coloneqq \bigcirc_{v \in S_i} \mathrm{VG}'_i(v) \quad \text{for } i \in [2, k]$$

Strings $P_1, P_2, \dots, P_k$ now contain all the vectors from our input sets. However, they are not sufficient to complete the reduction. To solve k-Most-Orthogonal-Vectors we must be able to check all n^k collections of vectors in $S_1 \times S_2 \times \cdots \times S_k$ for r-far-ness. Likewise, we must be able to align all n^k corresponding vector level gadgets in our final strings. In $P_1, P_2, \dots, P_k$ this is not always possible without incurring a large additional edit cost. For example, there is no optimal edit sequence of $P_1, P_2, \dots, P_k$ that aligns the leftmost vector level gadget of a string P_i with the rightmost vector level gadget of another string P_j – the number of insertions or deletions necessary would be too high.

Our strings $P_1, P_2, \dots, P_k$ are rigid, but we can give them the freedom to slide around by making the lengths of all strings distinct. Specifically, we will add a varying number of vector level gadgets to each string so that P_{i+1} will have more vector level gadgets than P_i for all $i \in [1, k-1]$. We define the *dummy vector* ϕ to be a vector of all ones of length $d+r+1$. Let

$$\begin{aligned} L_1 &\coloneqq \mathrm{VG}'_1(\phi)^{(50k+1)n} \circ \mathrm{DG}'_1(\phi)^{50kn} &\quad \text{and} \quad R_1 &\coloneqq \mathrm{DG}'_1(\phi)^{50kn} \circ \mathrm{VG}'_1(\phi)^{(50k+1)n} \\ L_i &\coloneqq \mathrm{VG}'_i(\phi)^{(100k+i)n} &\quad \text{and} \quad R_i &\coloneqq \mathrm{VG}'_i(\phi)^{(100k+i)n} \quad \text{for } i \in [2, k] \end{aligned}$$

Strings L_i and R_i will pad the left side and the right side of our P_i.

$$P'_i \coloneqq L_i \circ P_i \circ R_i \text{ for } i \in [1, k]$$

Observe that string P'_{i+1} has $2n$ more (dummy) vector level gadgets than P'_i for $i \in [1, k-1]$. This gives $P'_1, P'_2, \dots, P'_k$ a pyramid-like shape as in Figure 3. We will see that this allows the sort of sliding between strings necessary to complete our reduction.

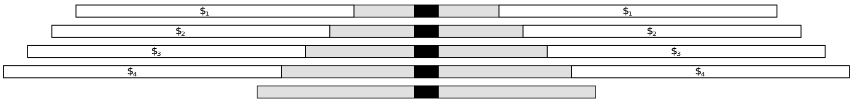

Figure 3 Final strings $T_1, T_2, \ldots, T_k$ when $k = 5$ shown from top to bottom. The vector gadgets corresponding to vectors from our input sets are shown in black, whereas the vector gadgets corresponding to dummy vectors ϕ are shown in gray. The special $\$_i$ symbols are shown in white.

However, because our strings $P'_1, P'_2, \ldots, P'_k$ are of different lengths, any complete edit sequence will require inserting or deleting vector level gadgets. This is problematic because it is difficult to reason about the edit costs of our vector level gadgets if they are inserted or deleted in the optimal edit sequence. To solve this problem we add special $\$_i$ symbols to our strings. We will see that the $\$_i$ symbols 'absorb' all the edits needed to make the lengths of the final strings equal and that no vector level gadgets will be inserted or deleted in the optimal edit sequence. We add $\$_1, \$_2, \ldots, \$_{k-1}$ to our alphabet, and we let $\ell_5 = 1000kn\ell_4$. Define

$$T_i := \$_i^{\ell_5} \circ P'_i \circ \$_i^{\ell_5} \quad \text{for } i \in [1, k-1] \qquad \text{and} \qquad T_k := P'_k$$

This completes the construction of our final strings $T_1, T_2, \ldots, T_k$. The length of each string as well as the time for their construction is $\mathcal{O}(nd^{O(1)})$. Their properties are summarized in Lemma 10 and Lemma 11 (proofs are deferred to Section 2.5 and Section 2.6, respectively).

▶ **Lemma 10.** *For any given sets $S_1, \ldots, S_k$ such that there is some collection $v_1, v_2, \ldots, v_k$ of r-far vectors with $v_i \in S_i$ for $i \in [1, k]$, $\mathrm{EDIT}(T_1, T_2, \ldots, T_k) \leq E^-$, where $E^- = D^- + (100kn + n - 1)D^+ + 101k(k-1)(2k-1)(d + r + 1)n + 2(k-1)\ell_5$.*

▶ **Lemma 11.** *For any given sets $S_1, S_2, \ldots, S_k$ such that there is no collection $v_1, v_2, \ldots, v_k$ of r-far vectors with $v_i \in S_i$ for $i \in [1, k]$, $\mathrm{EDIT}(T_1, T_2, \ldots, T_k) = E^+$, where $E^+ = E^- + (k-1)$.*

▶ **Theorem 12.** *If there is an $\varepsilon > 0$, an integer $k \geq 2$, and an algorithm that can solve k-Median-Edit-Distance on strings, each of length at most n, over an alphabet of size $\mathcal{O}(k)$ in $O(n^{k-\varepsilon})$ time, then* SETH *is false.*

Proof. Follows from Lemma 10 and Lemma 11. ◀

2.5 Proof of Lemma 10

Statement: *For any given sets $S_1, S_2, \ldots, S_k$ such that there is some collection $v_1, v_2, \ldots, v_k$ of r-far vectors with $v_i \in S_i$ for $i \in [1, k]$, $\mathrm{EDIT}(T_1, T_2, \ldots, T_k) \leq E^-$, where $E^- = D^- + (100kn + n - 1)D^+ + 101k(k-1)(2k-1)(d + r + 1)n + 2(k-1)\ell_5$.*

To upper bound the median edit distance of $T_1, T_2, \ldots, T_k$ by E^-, we must give an edit sequence of at most E^- edits. Initially, we will only edit the substrings $P'_1, P'_2, \ldots, P'_k$ and thus exclude the $\$_i$ symbols from consideration. We start by aligning the vector level gadgets.

Vector Level Gadget Alignment: We have assumed vectors $v_1, v_2, \ldots, v_k$ are r-far, and we choose to align their corresponding vector level gadgets $\mathrm{DG}_1(v_1), \mathrm{VG}_2(v_2), \ldots, \mathrm{VG}_k(v_k)$. We then align the rest of our vector level gadgets using the following rules:

1. Each vector level gadget in T_i aligns to exactly one vector level gadget in T_j for $j > i$.
2. If two vector level gadgets are adjacent in T_i, then they will be aligned to adjacent vector level gadgets in T_j for $j > i$.

Feasibility: We must demonstrate that this alignment is always achievable no matter how the vector level gadgets of $v_1, v_2, \ldots, v_k$ are embedded in strings $T_1, T_2, \ldots, T_k$. Recall that the vector level gadgets corresponding to vectors from our input sets are located in substrings P_i of T_i for all $i \in [1, k]$. Our construction gives paddings L_{i+1} and R_{i+1} exactly n more dummy vector level gadgets than L_i and R_i respectively for $i \in [1, k-1]$. It follows that even if the leftmost (resp. rightmost) vector level gadget in P_i is aligned with the rightmost (resp. leftmost) vector level gadget in P_{i+1}, the rules above remain satisfied.

Edit Cost for Vector Level Gadgets: There are $100kn + n$ decision gadgets DG_1 in T_1, so our edit sequence will yield $100kn + n$ alignments of $\mathrm{DG}_1, \mathrm{VG}_2, \ldots, \mathrm{VG}_k$, of which at least one such alignment will have cost D^- and the rest at most D^+. This gives an edit cost of at most $E_1^- = D^- + (100kn + n - 1)D^+$. At this point, all vector level gadgets in $P_1, P_2, \ldots, P_k$ have been edited (refer to Figure 4).

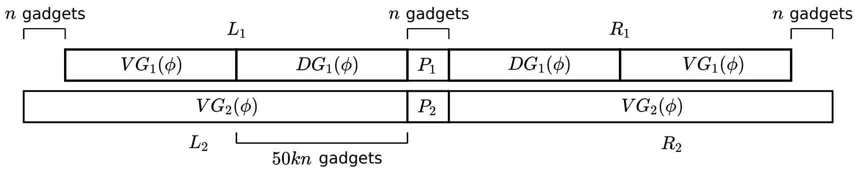

Figure 4 Strings P_1' and P_2'. All vector gadgets in P_2 align with decision gadgets DG_1 in P_1'.

Then there are exactly $2(50k + 1)n$ alignments of $\mathrm{VG}_1(\phi), \mathrm{VG}_2(\phi), \ldots, \mathrm{VG}_k(\phi)$ gadgets, and for all $i \in [2, k]$ there are exactly $2n$ alignments containing precisely the gadgets $\mathrm{VG}_i(\phi), \mathrm{VG}_{i+1}(\phi), \ldots, \mathrm{VG}_k(\phi)$. We will count the minimal number of edits needed to make these dummy vector gadgets identical. Let $F_i = (d + r + 1)(2k - 1)(k - i)$.

▷ Claim 13. For all $i \in [1, k]$, $\mathrm{EDIT}(\mathrm{VG}_i(\phi), \mathrm{VG}_{i+1}(\phi), \ldots, \mathrm{VG}_k(\phi)) = F_i$.

Proof. Each dummy vector gadget $\mathrm{VG}_j(\phi)$ is composed of $d+r+1$ coordinate gadgets. Each alignment of the coordinate gadgets $\mathrm{CG}_i(1), \mathrm{CG}_{i+1}(1), \ldots, \mathrm{CG}_k(1)$ will incur $(2k - 1)(k - i)$ total edits, with $(k - 1)(k - i)$ edits from f gadgets and $k(k - i)$ edits from h gadgets. ◀

Denote the sum of the internal edit costs of all alignments of $\mathrm{VG}_i, \mathrm{VG}_{i+1}, \ldots, \mathrm{VG}_k$ gadgets for $i \in [1, k]$ by

$$E_2^- = 2(50k + 1)n \cdot F_1 + \sum_{i \in [2,k]} 2n \cdot F_i = 101k(k - 1)(2k - 1)(d + r + 1)n$$

This completes our edits on all vector level gadgets.

Total Edit Cost: All substrings $P_1', P_2', \ldots, P_k'$ have been edited to $P_1^*, P_2^*, \ldots, P_k^*$, respectively, so that P_i^* is a substring of P_j^* for all $i < j$. We will now edit the $\$_i$ symbols in order to complete the edit sequence of $T_1, T_2, \ldots, T_k$. In particular, we will edit all k strings to be equal to P_k^* by substituting and deleting $\$_i$ symbols. For the ith string, we will perform substitutions on $|P_k^*| - |P_i^*|$ of the $\$_i$ symbols in T_i and delete the remaining $\$_i$ symbols. Since we substitute or delete every $\$_i$ symbol, this will incur an edit cost of $E_3^- = 2(k - 1)\ell_5$. The total number of edits performed in our edit sequence is no more than $E_1^- + E_2^- + E_3^- = E^-$. This completes the proof.

2.6 Proof of Lemma 11

Statement: *For any given sets $S_1, S_2, \ldots, S_k$ such that there is no collection $v_1, v_2, \ldots, v_k$ of r-far vectors with $v_i \in S_i$ for $i \in [1,k]$, $\mathrm{EDIT}(T_1, T_2, \ldots, T_k) = E^+ = E^- + (k-1)$.*

▷ Claim 14. $\mathrm{EDIT}(T_1, T_2, \ldots, T_k) \leq E^+$

Proof. We can achieve this upper bound by giving an edit sequence identical to the edit sequence in Lemma 10. Note that the only difference now is that there is no longer an r-far collection of vectors, so the edit cost of D^- in Lemma 10 is now D^+. This yields a complete edit sequence with $E^- + (D^+ - D^-) = E^+$ edits, so our inequality holds. ◀

We must now prove that $\mathrm{EDIT}(T_1, T_2, \ldots, T_k) \geq E^+$. Our lower bound on the number of edits comes from two disjoint sources: the edits incurred by the $\$_i$ symbols and the edits incurred by alignments between vector level gadgets.

▷ Claim 15. The $\$_i$ symbols for $i \in [1, k-1]$ incur a cost of at least $E_1^+ = 2(k-1)\ell_5$ edits in a complete edit sequence of $T_1, T_2, \ldots, T_k$.

Proof. First note that symbols $\$_i$ for $i \in [1, k-1]$ have $E_1^+ = 2(k-1)\ell_5$ occurrences in $T_1, T_2, \ldots, T_k$. We will show that each of these $\$_i$ symbols incurs one edit and that this edit is disjoint from the edits of any other $\$_j$ symbol. If a $\$_i$ symbol is deleted or substituted, then it certainly incurs one edit. Furthermore, these deletions and substitutions are necessarily disjoint. Otherwise, suppose that a $\$_i$ symbol is not substituted or deleted, but remains unmodified in the edit sequence. Then because there are no $\$_i$ symbols in string T_k, we must incur at least one edit in T_k. This edit must be disjoint from any other edits incurred by other $\$_i$ symbols. ◀

Now we will reason about the lower bound on the edits incurred by vector level gadgets by considering every possible configuration of alignments between vector level gadgets. In order to do this, we define a graph G whose vertices correspond to vector level gadgets. More specifically, for the jth leftmost vector level gadget in T_i, we add a vertex x_i^j to G for $i \in [1,k]$. Thus vertices $x_i^1, x_i^2, \ldots, x_i^{(200k+2i+1)n}$ correspond to the $2(100k+i)n + n$ vector level gadgets in T_i from left to right. Now for a particular edit sequence, we define G to have an unordered edge $(x_{i_1}^{j_1}, x_{i_2}^{j_2})$ if the j_1th vector level gadget of T_{i_1} is aligned with the j_2th vector level gadget of T_{i_2} in the edit sequence. Also, we say that $x_{i_1}^{j_1}$ and $x_{i_2}^{j_2}$ are from the same row if $i_1 = i_2$.

Every edit sequence now corresponds to a graph G. This graph can be decomposed into a set of connected components $\mathcal{C}$. For a component $c \in \mathcal{C}$, we define $\#(c,i)$ as the number of vertices belonging to string T_i in c. We say that $\mathrm{width}(c)$ of a component c is $\max_{i \in [1,k]} \#(c,i)$. We let $|c|$ denote the number of vertices in a component c. We now partition $\mathcal{C}$ into the following sets:

- $\mathcal{C}_1$ is the set of all components c with $\mathrm{width}(c) > 1$
- $\mathcal{C}_2$ is the set of all components c with $\mathrm{width}(c) = 1$ and $\#(c,k) = 0$
- $\mathcal{C}_3$ is the set of all components c with $\mathrm{width}(c) = 1$ and $\#(c,k) = 1$

We now lower bound the edit costs of components in $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$. Let $Q = 10kd\ell_3$.

▶ **Lemma 16.** *Every component c in $\mathcal{C}_1$ incurs at least $Q \cdot \mathrm{width}(c)$ edits.*

Proof. Because our component c is connected, the case illustrated in Figure 5 must occur at least $\mathrm{width}(c) - 1$ times. Then at least $2\ell_4(\mathrm{width}(c) - 1)$ edits must be performed on the padding 8 symbols between the vector level gadgets of c. Observe that because $\ell_4 > Q$, this cost is greater than $Q \cdot \mathrm{width}(c)$. These edits are disjoint from the edits of the $\$_i$ symbols. ◀

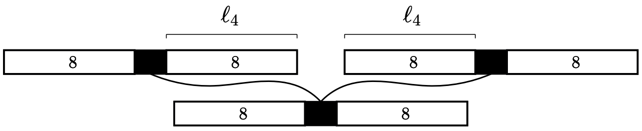

Figure 5 Case: one vector gadget in a string T_i is aligned with two vector gadgets in a string T_j. This alignment requires $2\ell_4$ edits of 8 symbols.

▶ **Lemma 17.** *Every component c in $\mathcal{C}_2$ incurs at least Q edits.*

Proof. By definition, the vector level gadgets in component c have no alignments with any vector level gadget VG_k in T_k. It follows that we incur a cost of at least $|\mathrm{VG}_k| > Q$. Furthermore, this edit cost is disjoint from the E_1^+ edit cost of our $\$_i$ symbols because there are no $\$_i$ symbols in T_k. ◀

We have given lower bounds for the edit costs of every component in $\mathcal{C}_1$ and $\mathcal{C}_2$, and these edit costs are disjoint by nature. Now we bound the costs of every component in $\mathcal{C}_3$. It will be useful to partition the components in $\mathcal{C}_3$ into the following sets:

- $\mathcal{C}_{3.1}$ is the set of all components c containing a vertex corresponding to a DG_1 gadget
- $\mathcal{C}_{3.2}$ is the remaining components in $\mathcal{C}_3$.

▶ **Lemma 18.** *All components c in $\mathcal{C}_{3.1}$ incur an edit cost of D^+.*

Proof. Our proof makes use of the following claim.

▷ Claim 19. No optimal edit sequence aligns a decision gadget DG_1 with any $\$_i$ symbol.

Subproof. Suppose some decision gadget DG_1 is aligned with a $\$_i$ symbol in string T_i for some $i \in [2, k-1]$. We will show that this incurs an edit cost greater than our upper bound E^+ established in Section 2.6, implying this cannot occur in an optimal edit sequence. We may assume w.l.o.g. that DG_1 is aligned with a $\$_i$ symbol on the left side of T_i. It follows that the substring $\mathrm{VG}'_1(\phi)^{(50k+1)n}$ of T_1 must occur to the left of the alignment, and the substring P'_i of T_i must occur to the right of the alignment (see Figure 4). Then this alignment of T_1 and T_i has a combined length greater than or equal to $|\mathrm{VG}'_1(\phi)^{(50k+1)n}| + |P'_i|$. We observe that $|\mathrm{VG}'_1(\phi)^{(50k+1)n}| > 100kn\ell_4$ and $|P'_i| > 400kn\ell_4$, so our alignment of T_1 and T_i has a combined length greater than $500kn\ell_4$. On the other hand, $|T_k| = (202k+1)n|\mathrm{VG}'_k| < 203kn(3\ell_3 + 2\ell_4)$. Our alignment of T_1 and T_i must be edited to have the same length as T_k in every complete edit sequence, so it follows that $\mathrm{EDIT}(T_1, T_i, T_k) > 500kn\ell_4 - 203kn(3\ell_3 + 2\ell_4) = kn(94\ell_4 - 609\ell_3) > 1000k^4dn\ell_3$. Then our edit sequence requires $1000k^4dn\ell_3 + E_1^+ > E^+$ edits, so this alignment cannot occur in an optimal edit sequence. ◀

Let c be a component in $\mathcal{C}_{3.1}$. Suppose $\#(c, i) = 0$ for some $i \in [2, k-1]$. Then by definition, our gadgets in c have no alignments with any vector level gadget in T_i. It follows that we must perform at least $|\mathrm{VG}_i| > D^+$ edits among the vector gadgets in c. This is because the vector gadgets in c are either aligned with no symbols in T_i and therefore require at least $|\mathrm{VG}_i|$ insertions or deletions in c to make all strings the same length, or the vector gadgets in c are aligned exclusively with 8 symbols in T_i and therefore require at least $|\mathrm{VG}_i|$ substitutions to make them the same. Note that the vector gadgets in c cannot be aligned with any $\$_i$ symbols in T_i by Claim 19. This is key for proving that these edits are disjoint from the E_1^- cost of editing the $\$_i$ symbols.

Now consider the case that $\#(c,i) \neq 0$ for all $i \in [2, k-1]$. Then we have that $\#(c,i) = 1$ for all $i \in [1,k]$, and by our analysis in Lemma 10, the edit cost of aligning the k vector level gadgets is at least D^+. ◀

▶ **Lemma 20.** *Let c be a component in $\mathcal{C}_{3.2}$ and let $\lambda = |c|$, then the edit cost incurred by the vector gadgets in c is $(d+r+1)(2k-1)(\lambda-1)$.*

Proof. Here we make use of the following claim, which has proof similar to Claim 19.

▷ Claim 21. Let $v_i \in S_i$ for some $i \in [2,k]$, then no optimal edit sequence aligns the vector gadget $\mathrm{VG}_i(v_i)$ in T_i with a $\$_1$ symbol in T_1, nor a dummy vector gadget $\mathrm{VG}_1(\phi)$ in T_1.

Subproof. Suppose some vector gadget $\mathrm{VG}_i(v_i)$ in string T_i with $i \in [2,k]$ and $v_i \in S_i$ is aligned with a dummy vector gadget $\mathrm{VG}_1(\theta)$ in string T_1. We will show that this incurs an edit cost greater than our upper bound E^+, implying this cannot occur in an optimal edit sequence. We may assume w.l.o.g. that $\mathrm{VG}_i(v_i)$ is aligned with a $\mathrm{VG}_1(\theta)$ gadget on the left side of T_1. It follows that the substring L_i of T_i must occur to the left of the alignment and the substring $\mathrm{DG}'_1(\phi)^{50kn} \circ P_1 \circ R_1$ of T_1 must occur to the right of the alignment. Then we can consider this alignment of T_i and T_1 to have a combined length greater than or equal to $|L_i| + |\mathrm{DG}'_1(\phi)^{50kn} \circ P_1 \circ R_1|$.

We observe that $|L_i| > 200kn\ell_4$ and $|\mathrm{DG}'_1(\phi)^{50kn} \circ P_1 \circ R_1| > 400kn\ell_4$, so our alignment of T_i and T_1 has a combined length greater than $600kn\ell_4$. On the other hand, $|T_k| = (202k+1)n|\mathrm{VG}'_k| < 203kn(3\ell_3 + 2\ell_4)$.

Our alignment of T_i and T_1 must be edited to have the same length as T_k in every complete edit sequence, so it follows that $\mathrm{EDIT}(T_1, T_i, T_k) > 600kn\ell_4 - 203kn(3\ell_3 + 2\ell_4) = kn(194\ell_4 - 609\ell_3) > 1000k^4dn\ell_3$. Then our edit sequence requires $1000k^4dn\ell_3 + E_1^+ > E^+$ edits, so this alignment cannot occur in an optimal edit sequence. It follows that $\mathrm{VG}_i(v_i)$ in T_i cannot align with a $\mathrm{VG}_1(\theta)$ gadget (and by extension a $\$_1$ symbol) in T_1. ◀

Let c be in $\mathcal{C}_{3.2}$. Suppose there is some $v_i \in S_i$ for $i \in [2,k]$ such that vector gadget $\mathrm{VG}_i(v_i)$ corresponds to a vertex in component c. Then the gadgets in our component cannot align with any decision gadgets DG_1, vector gadgets $\mathrm{VG}_1(\phi)$, or $\$_1$ symbols in T_1. It follows that we must perform at least $|\mathrm{VG}_i| > (d+r+1)(2k-1)(\lambda-1)$ insertions in T_i. Else, all vertices in component c correspond only to vector gadgets $\mathrm{VG}_i(\phi)$ for $i \in [1,k]$. By a similar argument as in Claim 13, the edit cost of component c is $(d+r+1)(2k-1)(\lambda-1)$. ◀

We have lower bounded the edit cost of all components in $\mathcal{C}_1, \mathcal{C}_2$, and $\mathcal{C}_3$. Now we must combine our component level arguments to obtain an overall lower bound on the edit cost. Let $W = \sum_{c \in \mathcal{C}_1 \cup \mathcal{C}_2} \mathrm{width}(c)$. Then we know that the components in $\mathcal{C}_1 \cup \mathcal{C}_2$ incur a cost of at least $E_2^+ = WQ$ edits by Lemma 16 and Lemma 17.

We now lower bound the total number of edits from components in $\mathcal{C}_3$. Note that components in $\mathcal{C}_{3.1}$ incur a much higher cost than components in $\mathcal{C}_{3.2}$. Then to lower bound the edits in $\mathcal{C}_3$, we must assume the least possible number of components in $\mathcal{C}_{3.1}$. There are $(100k+1)n$ decision gadgets DG_1 in our final strings and at most W decision gadgets in components in $\mathcal{C}_1 \cup \mathcal{C}_2$, so there must be at least $Z_1 = (100k+1)n - W$ components in $\mathcal{C}_{3.1}$. Note that if $W \geq (100k+1)n$, then $E_1^+ + E_2^+ \geq E^+$, so we may assume Z_1 is positive. Then components from $\mathcal{C}_{3.1}$ incur a cost of at least $E_3^+ = Z_1 D^+$ by Lemma 18.

There are at most $V_0 = kW$ vertices in components in $\mathcal{C}_1 \cup \mathcal{C}_2$, and there are at most $V_1 = kZ_1$ vertices in $\mathcal{C}_{3.1}$. Furthermore, there are $k(201k+2)n$ vertices in our graph G. It follows that there must be at least $V_2 = k(201k+2)n - V_1 - V_0 = k(101k+1)n$ vertices in all components in $\mathcal{C}_{3.2}$.

Because our edit cost lower bound for every component in $\mathcal{C}_{3.2}$ is linear in the component size, we have the following.

▷ Claim 22. Suppose there are Z components in $\mathcal{C}_{3.2}$ and a total of V vertices in all components in $\mathcal{C}_{3.2}$. Then the components in $\mathcal{C}_{3.2}$ incur $(d+r+1)(2k-1)(V-Z)$ edits.

Proof. By Lemma 20, each component of size λ in $\mathcal{C}_{3.2}$ incurs cost $(d+r+1)(2k-1)(\lambda-1)$. Let z_i denote the size of the ith component in $\mathcal{C}_{3.2}$ for $i \in [1, Z]$. Then we may sum the edit costs of all components in $\mathcal{C}_{3.2}$:

$$\sum_{i\in[1,Z]} (d+r+1)(2k-1)(z_i-1) = (d+r+1)(2k-1)(V-Z)$$

where $z_i > 0$ for $i \in [1, Z]$ and $z_1 + z_2 + \cdots + z_Z = V$. ◀

Claim 22 proves that the edit cost of all the components in $\mathcal{C}_{3.2}$ decreases with the number of components Z. Then to achieve our lower bound we must upper bound the number of components in $\mathcal{C}_{3.2}$. There are exactly $(202k+1)n$ vector level gadgets in T_k, so there can be at most $Z_2 = (202k+1)n - Z_1$ components in $\mathcal{C}_{3.2}$. It follows that the total edit cost contributed by the components of $\mathcal{C}_{3.2}$ is at least $E_4^+ = (d+r+1)(2k-1)(V_2 - Z_2)$.

Then since the edit costs contributed by E_1^+, E_2^+, E_3^+, and E_4^+ are disjoint, we achieve a lower bound $\text{EDIT}(T_1, T_2, \ldots, T_k) \geq E_1^+ + E_2^+ + E_3^+ + E_4^+$. Straightforward calculation will show that $E_1^+ + E_2^+ + E_3^+ + E_4^+ \geq E^+$ for all $W > 0$. It follows that $\text{EDIT}(T_1, \ldots, T_k) = E^+$.

3 Reduction from k-Median-Edit-Distance to k-Center-Edit-Distance

We now provide a simple, yet previously unknown reduction from k-Median-Edit-Distance to k-Center-Edit-Distance. Given a set of strings $X = \{x_1, x_2, \ldots, x_k\}$, each of length at most n over an alphabet Σ, we define another set of strings $Y = \{y_1, y_2, \ldots, y_k\}$ over an alphabet $\Sigma' = \Sigma \cup \{\$\}$ (where $\$ \notin \Sigma$) as follows (fix $\ell = k^2 n$):

$$\begin{aligned}
y_1 &= x_1 \circ \$^\ell \circ x_2 \circ \$^\ell \circ \cdots \circ \$^\ell \circ x_{k-1} \circ \$^\ell \circ x_k\\
y_2 &= x_2 \circ \$^\ell \circ x_3 \circ \$^\ell \circ \cdots \circ \$^\ell \circ x_k \circ \$^\ell \circ x_1\\
&\vdots\\
y_k &= x_k \circ \$^\ell \circ x_1 \circ \$^\ell \circ \cdots \circ \$^\ell \circ x_{k-2} \circ \$^\ell \circ x_{k-1}
\end{aligned}$$

Let $\text{CENTER-EDIT}(y_1, y_2, \ldots, y_k)$ denote the center edit distance of strings $y_1, y_2, \ldots, y_k$. We will prove the following, which will complete the reduction.

▶ **Lemma 23.** $\text{EDIT}(x_1, x_2, \ldots, x_k) = \text{CENTER-EDIT}(y_1, y_2, \ldots, y_k)$

Proof. Suppose that $\text{EDIT}(x_1, x_2, \ldots, x_k) = E$, and there is an optimal edit sequence on $x_1, x_2, \ldots, x_k$ that performs E_i edits on x_i for $i \in [1, k]$. It follows that $E_1 + E_2 + \cdots + E_k = E$.

▷ Claim 24. $\text{EDIT}(y_1, y_2, \ldots, y_k) = kE$

Subproof. It can be seen that $\text{EDIT}(y_1, y_2, \ldots, y_k) \leq kE$ since we may align all \$ symbols in the y_i in zero edits, and then we have k alignments of $x_1, x_2, \ldots, x_k$ substrings, each incurring E edits, for a total of kE edits.

Now note that no optimal edit sequence of $y_1, y_2, \ldots, y_k$ will delete an entire series of \$ symbols because it would incur cost ℓ greater than kE, our upper bound. It follows that for all $i \neq j$ the hth leftmost series of \$ symbols in y_i is aligned with the hth leftmost series of \$ symbols in y_j for $h \in \{1, \ldots, k-1\}$. Then the \$ alignments 'lock' the x_i substrings into place so that we have k alignments of $x_1, x_2, \ldots, x_k$ substrings, and because no x_i contains the \$ symbol, it follows that each alignment of the x_i incurs cost greater than or equal to E. Then $\text{EDIT}(y_1, y_2, \ldots, y_k) \geq kE$. ◀

We now have that $\text{EDIT}(y_1, y_2, \ldots, y_k) = kE$. Furthermore, there is an optimal edit sequence that performs exactly E edits on every string in $y_1, y_2, \ldots, y_k$. This can be seen because in every alignment of substrings $x_1, x_2, \ldots, x_k$ in our edit sequence of $y_1, y_2, \ldots, y_k$, we may choose to perform E_i edits on each x_i. Then there exists an optimal edit sequence where for every string y_i with $i \in [1, k]$, we perform $E_i + E_{i+1} + \cdots + E_k + E_1 + E_2 + \cdots + E_{i-1} = E$ edits on y_i.

It follows that $\text{CENTER-EDIT}(y_1, y_2, \ldots, y_k) \leq E$. Furthermore, suppose that $\text{CENTER-EDIT}(y_1, y_2, \ldots, y_k) < E$. Then $\text{EDIT}(y_1, y_2, \ldots, y_k) < kE$, a contradiction. We conclude that $\text{CENTER-EDIT}(y_1, y_2, \ldots, y_k) = E$ and our reduction is complete. Note that for all $i \in [1, k]$, $|y_i| = (k-1)k^2n + kn = \mathcal{O}(n)$. ◀

Lemma 23 directly implies the following result.

▶ **Theorem 25.** *If there is an $\varepsilon > 0$, a constant $k \geq 2$, and an algorithm that can solve k-Center-Edit-Distance on strings, each of length at most n, over an alphabet of size $\mathcal{O}(k)$ in $\mathcal{O}(n^{k-\varepsilon})$ time, then* SETH *is false.*

4 Discussion

Based on SETH, we have shown conditional hardness results for median string, center string, tree alignment, and bottleneck tree alignment problems, all under edit distance. These results suggest that the algorithms for the median string and tree-alignment problems are optimal (up to logarithmic factors). However, for the center string and bottleneck tree alignment problem, they leave an intriguing gap between the best known upper bounds. For center string (or the star instance of the bottleneck tree alignment) the best known dynamic programming algorithm works in time $O(n^{2k})$ [40], and as far as the authors know, no such algorithm exists for bottleneck tree alignment on more general trees. We conclude by asking: is an $O(n^k)$ algorithm waiting to be found for these problems, or does there exist a more efficient reduction which can prove that an $O(n^{2k-\varepsilon})$ algorithm is highly improbable?

References

1 Amir Abboud, Arturs Backurs, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Or Zamir. Subtree isomorphism revisited. *ACM Trans. Algorithms*, 14(3):27:1–27:23, 2018. doi:10.1145/3093239.

2 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78, 2015. doi:10.1109/FOCS.2015.14.

3 Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. Seth-based lower bounds for subset sum and bicriteria path. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 41–57, 2019. doi:10.1137/1.9781611975482.3.

4 Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1681–1697. SIAM, 2015. doi:10.1137/1.9781611973730.112.

5 Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 375–388, 2016. doi:10.1145/2897518.2897653.

6 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014. doi:10.1109/FOCS.2014.53.

7 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 39–51. Springer, 2014. doi:10.1007/978-3-662-43948-7_4.

8 Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. *SIAM J. Comput.*, 47(3):1098–1122, 2018. doi:10.1137/15M1050987.

9 Arturs Backurs and Piotr Indyk. Which regular expression patterns are hard to match? In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 457–466, 2016. doi:10.1109/FOCS.2016.56.

10 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J. Comput.*, 47(3):1087–1097, 2018. doi:10.1137/15M1053128.

11 Christina Boucher and Mohamed Omar. On the hardness of counting and sampling center strings. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 9(6):1843–1846, 2012. doi:10.1109/TCBB.2012.84.

12 Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.

13 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 79–97, 2015. doi:10.1109/FOCS.2015.15.

14 Karl Bringmann and Marvin Künnemann. Multivariate fine-grained complexity of longest common subsequence. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1216–1235, 2018. doi:10.1137/1.9781611975031.79.

15 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3sum via additive combinatorics. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 31–40. ACM, 2015. doi:10.1145/2746539.2746568.

16 Yi-Jun Chang. Hardness of RNA folding problem with four symbols. *Theor. Comput. Sci.*, 757:11–26, 2019. doi:10.1016/j.tcs.2018.07.010.

17 Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 21–40, 2019. doi:10.1137/1.9781611975482.2.

18 Yen Hung Chen and Chuan Yi Tang. On the bottleneck tree alignment problems. *Inf. Sci.*, 180(11):2134–2141, 2010. doi:10.1016/j.ins.2010.02.008.

19 Raphaël Clifford, Allan Grønlund, Kasper Green Larsen, and Tatiana Starikovskaya. Upper and lower bounds for dynamic data structures on strings. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 22:1–22:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.STACS.2018.22.

20 Colin de la Higuera and Francisco Casacuberta. Topology of strings: Median string is np-complete. *Theor. Comput. Sci.*, 230(1-2):39–48, 2000. doi:10.1016/S0304-3975(97)00240-5.

21 Erik D. Demaine, Andrea Lincoln, Quanquan C. Liu, Jayson Lynch, and Virginia Vassilevska Williams. Fine-grained I/O complexity via reductions: New lower bounds, faster algorithms, and a time hierarchy. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 34:1–34:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.34.

22 Lech Duraj, Marvin Künnemann, and Adam Polak. Tight conditional lower bounds for longest common increasing subsequence. *Algorithmica*, 81(10):3968–3992, 2019. doi:10.1007/s00453-018-0485-7.

23 Massimo Equi, Roberto Grossi, Veli Mäkinen, and Alexandru I. Tomescu. On the complexity of string matching for graphs. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 55:1–55:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.55.

24 Isaac Goldstein, Moshe Lewenstein, and Ely Porat. On the hardness of set disjointness and set intersection with bounded universe. In Pinyan Lu and Guochuan Zhang, editors, *30th International Symposium on Algorithms and Computation, ISAAC 2019, December 8-11, 2019, Shanghai University of Finance and Economics, Shanghai, China*, volume 149 of *LIPIcs*, pages 7:1–7:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ISAAC.2019.7.

25 Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Fixed-parameter algorithms for CLOSEST STRING and related problems. *Algorithmica*, 37(1):25–42, 2003. doi:10.1007/s00453-003-1028-3.

26 Franziska Hufsky, Léon Kuchenbecker, Katharina Jahn, Jens Stoye, and Sebastian Böcker. Swiftly computing center strings. In *Algorithms in Bioinformatics, 10th International Workshop, WABI 2010, Liverpool, UK, September 6-8, 2010. Proceedings*, pages 325–336, 2010. doi:10.1007/978-3-642-15294-8_27.

27 Franziska Hufsky, Léon Kuchenbecker, Katharina Jahn, Jens Stoye, and Sebastian Böcker. Swiftly computing center strings. *BMC Bioinformatics*, 12:106, 2011. doi:10.1186/1471-2105-12-106.

28 Tao Jiang, Lusheng Wang, and Kaizhong Zhang. Alignment of trees - an alternative to tree edit. In Maxime Crochemore and Dan Gusfield, editors, *Combinatorial Pattern Matching, 5th Annual Symposium, CPM 94, Asilomar, California, USA, June 5-8, 1994, Proceedings*, volume 807 of *Lecture Notes in Computer Science*, pages 75–86. Springer, 1994. doi:10.1007/3-540-58094-8_7.

29 Xiaoyi Jiang, Horst Bunke, and Janos Csirik. Median strings: A review. In *Data Mining in Time Series Databases*, pages 173–192. World Scientific, 2004.

30 Teuvo Kohonen. Median strings. *Pattern Recognition Letters*, 3(5):309–313, 1985. doi:10.1016/0167-8655(85)90061-3.

31 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1272–1287, 2016. doi:10.1137/1.9781611974331.ch89.

32 Robert Krauthgamer and Ohad Trabelsi. Conditional lower bounds for all-pairs max-flow. *ACM Trans. Algorithms*, 14(4):42:1–42:15, 2018. doi:10.1145/3212510.

33 Ferenc Kruzslicz. Improved greedy algorithm for computing approximate median strings. *Acta Cybern.*, 14(2):331–339, 1999. URL: http://www.inf.u-szeged.hu/actacybernetica/edb/vol14n2/Kruzslicz_1999_ActaCybernetica.xml.

34 J. Kevin Lanctôt, Ming Li, Bin Ma, Shaojiu Wang, and Louxin Zhang. Distinguishing string selection problems. *Inf. Comput.*, 185(1):41–55, 2003. doi:10.1016/S0890-5401(03)00057-9.

35 Ming Li, Bin Ma, and Lusheng Wang. On the closest string and substring problems. *J. ACM*, 49(2):157–171, 2002. doi:10.1145/506147.506150.

36 Bin Ma and Xiaoming Sun. More efficient algorithms for closest string and substring problems. *SIAM J. Comput.*, 39(4):1432–1443, 2009. doi:10.1137/080739069.

37 Hiromitsu Maji and Taisuke Izumi. Listing center strings under the edit distance metric. In *Combinatorial Optimization and Applications - 9th International Conference, COCOA 2015, Houston, TX, USA, December 18-20, 2015, Proceedings*, pages 771–782, 2015. doi:10.1007/978-3-319-26626-8_57.

38 Carlos D. Martínez-Hinarejos, Alfons Juan, Francisco Casacuberta, and Ramón Alberto Mollineda. Reducing the computational cost of computing approximated median strings. In Terry Caelli, Adnan Amin, Robert P. W. Duin, Mohamed S. Kamel, and Dick de Ridder, editors, *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops SSPR 2002 and SPR 2002, Windsor, Ontario, Canada, August 6-9, 2002, Proceedings*, volume 2396 of *Lecture Notes in Computer Science*, pages 47–55. Springer, 2002. doi:10.1007/3-540-70659-3_4.

39 François Nicolas and Eric Rivals. Hardness results for the center and median string problems under the weighted and unweighted edit distances. *J. Discrete Algorithms*, 3(2-4):390–415, 2005. doi:10.1016/j.jda.2004.08.015.

40 R. Ravi and John D. Kececioglu. Approximation algorithms for multiple sequence alignment under a fixed evolutionary tree. *Discrete Applied Mathematics*, 88(1-3):355–366, 1998. doi:10.1016/S0166-218X(98)00079-1.

41 David Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28(1):35–42, 1975.

42 Jeong Seop Sim and Kunsoo Park. The consensus string problem for a metric is np-complete. *J. Discrete Algorithms*, 1(1):111–117, 2003. doi:10.1016/S1570-8667(03)00011-X.

43 Andrés Varón and Ward C. Wheeler. The tree alignment problem. *BMC Bioinformatics*, 13:293, 2012. doi:10.1186/1471-2105-13-293.

44 T. K. Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57, 1968. Russian Kibernetika 4(1):81-88 (1968).

45 Lusheng Wang and Dan Gusfield. Improved approximation algorithms for tree alignment. *J. Algorithms*, 25(2):255–273, 1997. doi:10.1006/jagm.1997.0882.

46 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.