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 $s^*$ that minimizes the sum of the edit distances from $s^*$ to all other strings in the input. Arising frequently in computational biology, this problem provides an important generalization of edit distance to multiple strings. We demonstrate that for any $\epsilon > 0$ and $k \geq 2$, an $O(n^{k-\epsilon})$ 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 in phylogenetics.

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

Keywords and phrases Edit Distance, Median String, Center String, SETH.
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 $S$ 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 $S$ 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 [45]. An algorithm for solving this problem on $k$ strings in time $O(n^k)$ was then given by Sankoff in 1975 [42] 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 [30, 31, 34, 39]. 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 [21]. This result was later strengthened to finite alphabets in [43] and then even to binary alphabets in [40]. In [40] 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 on Sankoff’s algorithm is unlikely. Specifically, we show that an $O(n^{k-\varepsilon})$ algorithm for any $\varepsilon > 0$ would refute SETH. We also prove that the same lower bounds hold for a related problem known as the $k$-Center-Edit-Distance.

**Problem 2 ($k$-Center-Edit-Distance).** Given a set $S$ 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 $S$ to $s^*$. The maximum edit distance from $s^*$ to any string in $S$ 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 $k$ [40]. Additionally, $k$-Center-Edit-Distance has been shown to have an $O(n^{2k})$ solution [40]. 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 [19, 29, 44, 46]. The $k$-Tree-Alignment (resp. $k$-Bottleneck-Tree-Alignment) problem is defined as follows: given a tree $T$ with $k$ leaves where each leaf is labelled with a string of length $n$, find an assignment of strings to all internal vertices of $T$ 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-grain 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, 11, 16, 18, 20, 22, 25, 24, 32, 33]. Reductions based on SETH, such as the one considered here, tend to have a very similar structure. For example, the Orthogonal Vectors problem is often used as an intermediate problem. Relating this problem to SETH and using this for conditional lower bounds has been shown that a strongly subquadratic algorithm for Orthogonal Vectors would violate SETH [47]. It has since been extensively used in this field. The proof we provide here works off of a similar pattern as this, but with a generalized variant of the Orthogonal Vectors as used in [2]. Using these techniques, 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, 13, 15, 17], Bringmann and Künnemann created a framework by which any string problem which allowed for a particular gadget construction could have similar SETH based lower bounds proven for it [14]. This framework includes the problems of longest common subsequence, dynamic time warping, and edit distance under 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 than 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 [23], where the longest increasing subsequence on \( k \) strings \( k\text{-LCS} \) was considered. More results on \( k \) strings were provided in [7], where the local alignment problem on \( k \) strings under sum of pairs was considered. In both of the last two works mentioned, it was showed 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 for \( k\text{-LCS} \), implying circuit based hardness results apply for LCS on \( k \) strings. However, their work left open the question hardness for median string and other problems related to edit distance on \( k \) 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 [26, 28, 36, 37]. It has been shown that this problem under Hamming distance metric is NP-complete [35], 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 reason for this is that the problem is fixed parameter tractable in \( d \), a fact which has been the basis of many algorithmic solutions [12, 27, 38].
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 $O(n^{k-\varepsilon})$ 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 $\sum_{h=1}^{d} \prod_{i \in [1,k]} v_h[i]$, 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 $v \in S_i$ 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 $\text{EDIT}(x_1, x_2, \ldots, x_k)$ 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 $\text{CG}_i(b_i)$ that can distinguish between the case when $b_1b_2\cdots b_k = 0$ and $b_1b_2\cdots b_k = 1$. Specifically, we will show that there exist constants $C^-$ and $C^+$ such that if $b_1b_2\cdots b_k = 0$, then $\text{EDIT}(\text{CG}_1(b_1), \text{CG}_2(b_2), \ldots, \text{CG}_k(b_k)) = C^-$, and else if $b_1b_2\cdots b_k = 1$, then $\text{EDIT}(\text{CG}_1(b_1), \text{CG}_2(b_2), \ldots, \text{CG}_k(b_k)) = C^+$.

Vector Level: Given vectors $v_1, v_2, \ldots, v_k \in \{0,1\}^{d+r+1}$, we construct vector gadget strings $\text{VG}_i(v_i)$ for $i \in [2, k]$ and a slightly more complicated decision gadget string $\text{DG}_1(v_1)$ 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 $\text{EDIT}(\text{DG}_1(v_1), \text{VG}_2(v_2), \ldots, \text{VG}_k(v_k)) \leq D^-$ and else if $v_1, v_2, \ldots, v_k$ are $r$-close, then $\text{EDIT}(\text{DG}_1(v_1), \text{VG}_2(v_2), \ldots, \text{VG}_k(v_k)) = D^+$, where $D^+$ and $D^- < D^+$ are constants.

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

We now present a definition and an associated fact.

Definition 1 (Alignment). Given a particular edit sequence on strings $x_1, x_2, \ldots, x_k$, we say symbol $\alpha$ in $x_i$ is aligned with symbol $\beta$ in another string $x_j$ if neither $\alpha$ nor $\beta$ 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 $t$. 

2.2 Coordinate level reduction

For \( i \in [1, k] \), we define coordinate gadget strings \( \text{CG}_i \) 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

\[
\text{CG}_i(b_i) := f_i(b_i) \circ 4^{f_i} \circ g_i(b_i) \circ 4^{g_i} \circ h_i(b_i)
\]

for \( i \in [1, k] \), where

\[
f_i(b_i) = \begin{cases} 
2^{k-1} & \text{if } b_i = 1, i < k \\
2^{k-1} & \text{if } b_i = 1, i = k \\
2^{k-1} & \text{if } b_i = 0 
\end{cases}
\]

\[
g_i(b_i) = \begin{cases} 
3^{k-1} & \text{if } b_i = 1 \\
2^{k-1} & \text{if } b_i = 0 
\end{cases}
\]

\[
h_i(b_i) = \begin{cases} 
2_k & \text{if } b_i = 1 \\
\bigcap_{j=1}^{k} 2_j & \text{if } b_i = 0 
\end{cases}
\]

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

<table>
<thead>
<tr>
<th>( b_1, b_2, b_3 )</th>
<th>( f_1(b_1), f_2(b_2), f_3(b_3) )</th>
<th>( g_1(b_1), g_2(b_2), g_3(b_3) )</th>
<th>( h_1(b_1), h_2(b_2), h_3(b_3) )</th>
<th>EDIT(( \text{CG}_1(b_1), \ldots, \text{CG}_k(b_k) ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 1, 1</td>
<td>2_2, 2_3, 2_1, 2_3, 2_1, 2_2</td>
<td>3_3, 3_3, 3_3</td>
<td>2_1, 2_2, 2_3, 2_3, 2_1, 2_3</td>
<td>4 + 0 + 6 = 10</td>
</tr>
<tr>
<td>0, 1, 1</td>
<td>2_1, 2_2, 2_1, 2_3, 2_2, 2_3</td>
<td>2_2, 3_3, 3_3</td>
<td>2_1, 2_2, 2_3, 2_3, 2_1, 2_3</td>
<td>2 + 2 + 4 = 8</td>
</tr>
<tr>
<td>0, 0, 0</td>
<td>2_1, 2_2, 2_1, 2_2, 2_3, 2_3</td>
<td>2_1, 2_1, 2_2, 2_3, 2_3, 2_1</td>
<td>2_1, 2_2, 2_3, 2_3, 2_1, 2_3</td>
<td>4 + 4 + 0 = 8</td>
</tr>
</tbody>
</table>

\( \blacktriangleright \) Lemma 2. Let \( C^- = 2(k-1)^2 \) and let \( C^+ = C^- + (k-1) = (2k-1)(k-1) \). Then,

\[
\text{EDIT}(\text{CG}_1(b_1), \text{CG}_2(b_2), \ldots, \text{CG}_k(b_k)) = \begin{cases} 
C^+ & \text{if } b_1b_2\ldots b_k = 1 \\
C^- & \text{otherwise}
\end{cases}
\]

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

\( \blacktriangleright \) Claim 3. The median edit distance of our \( f_i \) gadgets is

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

\( \blacktriangleright \) Claim 4. The median edit distance of our \( g_i \) gadgets is

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

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

We have chosen \( \ell_1 \) to be sufficiently large that all \( f_i, g_i, \) and \( h_i \) gadgets align only with gadgets of their own type. Therefore,

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

A simple calculation will show that \( \text{EDIT}(\text{CG}_1(b_1), \ldots, \text{CG}_k(b_k)) \) is \( C^- \) when \( \pi < k \) (and hence \( b_1b_2\ldots b_k = 0 \)) and is \( C^+ \) when \( \pi = k \) (and hence \( b_1b_2\ldots b_k = 1 \)).
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 \( \ell_3 = 10d\ell_1 \) and \( \ell_3 = (10\ell_2)^2 \). For vectors \( v_1, v_2, \ldots, v_k \), we define

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

Observe that the vector gadget of a vector \( v_i \) is just the concatenation of the coordinate gadgets corresponding to each coordinate in \( v_i \). It follows that the median edit distance of \( \text{VG}_1(v_1), \text{VG}_2(v_2), \ldots, \text{VG}_k(v_k) \) 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 \( \text{VG}_i(v_i) \)’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 \( \text{VG}_1(v_1) \) with a decision gadget \( \text{DG}_1(v_1) \) 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

\[
\text{DG}_1(v_1) := 7^{\ell_3} \circ M_1(v_1) \circ 6^{\ell_3} \circ M_1(\theta) \circ 7^{\ell_3}, \text{ where } \theta \in \{0, 1\}^{d+r+1} \text{ such that } \theta[i] = 1 \text{ if } i \leq r + 1 \text{ and } 0 \text{ otherwise.}
\]

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 \( D^+ = D^- + (k - 1) \).

\[
\text{Lemma 6. For any given } r \text{-far vectors } v_1, v_2, \ldots, v_k \in \{0, 1\}^{d+r+1}, \\
\text{EDIT}(\text{DG}_1(v_1), \text{VG}_2(v_2), \text{VG}_3(v_3), \ldots, \text{VG}_k(v_k)) \leq D^-.
\]

**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 \( \text{VG}_2(v_2), \text{VG}_3(v_3), \ldots, \text{VG}_k(v_k) \) with the \( 7^{\ell_3} \circ M_1(v_1) \circ 6^{\ell_3} \) substring of \( \text{DG}_1(v_1) \) as in Figure 1.

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

First we delete \( M_1(\theta) \circ 7^{\ell_3} \) from \( \text{DG}_1(v_1) \) in \( m + \ell_3 \) edits. Then we substitute all the 7 symbols in the \( 7^{\ell_3} \) prefix of \( \text{DG}_1(v_1) \) to 6 symbols in \( \ell_3 \) 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 $j$th 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 $\text{EDIT}(M_1(v_1), M_2(v_2), \ldots, M_k(v_k)) \leq (d + 1)C^- + rC^+$. 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 $DG_1(v_1), VG_2(v_2), \ldots, VG_k(v_k)$ will align strings $VG_2(v_2), VG_3(v_3), \ldots, VG_k(v_k)$ with the $6^k \circ M_1(\theta) \circ 7^k$ substring of $DG_1(v_1)$ as in Fig. 2.

![Figure 2](image-url) An optimal alignment of $DG_1(v_1), VG_2(v_2), \ldots, VG_k(v_k)$ when $v_1, v_2, \ldots, v_k$ are $r$-close.

**Lemma 7.** For any given $r$-close vectors $v_1, v_2, \ldots, v_k \in \{0, 1\}^{d + r + 1},$

$\text{EDIT}(DG_1(v_1), VG_2(v_2), VG_3(v_3), \ldots, VG_k(v_k)) = D^+$. Proof. Deferred to Appendix A. The proof is a generalization of the vector gadget proof in [10] to $k$ strings and consists primarily of exhaustive case analysis. ◀

### 2.4 Set level reduction

![Figure 3](image-url) 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 $\phi$ are shown in gray. The special $\$i$ symbols are shown in white.

In this step of the reduction we will construct our final strings $T_1, T_2, \ldots, T_k$ that can detect $r$-far vectors in our input sets $S_1, S_2, \ldots, S_k$. 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 $T_1, T_2, \ldots, T_k$ 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 per string.

We define integer parameter $\ell_4 = 10000k^3d\ell_3$, and we add a new padding symbol $\$8$ to our alphabet. For all $v \in \{0, 1\}^{d + r + 1}$, let

$$DG'_i(v) := 8^i \circ DG_i(v) \circ 8^{i+1}$$

$$VG'_i(v) := 8^i \circ VG_i(v) \circ 8^{i+1} \quad \text{for} \quad i \in [1, k]$$
We now concatenate our vector level gadgets $DG'_1$ and $VG'_i$. Define

$$P_1 := \bigcup_{v \in S_i} DG'_1(v)$$
$$P_i := \bigcup_{v \in S_i} VG'_i(v) \quad \text{for } i \in [2, k]$$

Strings $P_1, P_2, \ldots, P_k$ 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 $P_1, P_2, \ldots, P_k$, this is not always possible without incurring a large additional edit cost. For example, there is no optimal edit sequence of $P_1, P_2, \ldots, P_k$ 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 $P_1, P_2, \ldots, P_k$ are rigid, but we can give them the freedom to slide around by making each string a different length. 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 $\phi$ to be a vector of all ones of length $d + r + 1$.

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

Our strings $L_i$ and $R_i$ will pad the left side and the right side of our $P_i$.

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

Observe that string $P'_{i+1}$ has $2n$ more (dummy) vector level gadgets than $P'_i$ for $i \in [1, k-1]$. This gives $P'_1, P'_2, \ldots, P'_k$ 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.

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 must be inserted or deleted in the optimal edit sequence. To solve this problem we add special $S_i$ symbols to our strings. We will see that the $S_i$ symbols ‘absorb’ all the edits needed to make our final strings the same length, and no vector level gadgets will be inserted or deleted in the optimal edit sequence. We add $S_1, S_2, \ldots, S_{k-1}$ to our alphabet, and we let $\ell_5 = 1000kn\ell_4$. Define

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

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 $O(ndO(1))$. Their properties are summarized in Lemma 8 and Lemma 9 (proofs are deferred to Section 2.5 and Section 2.6, respectively).

\textbf{Lemma 8.} 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 $i \in [1, k]$, EDIT($T_1, T_2, \ldots, T_k$) $\leq E^-$, where $E^- = D^- + (100kn + n - 1)D^+ + 101k(k-1)(d+r+1)n + 2(k-1)\ell_5$.

\textbf{Lemma 9.} 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 $i \in [1, k]$, EDIT($T_1, T_2, \ldots, T_k$) $= E^+$, where $E^+ = E^- + (k-1)$.

\textbf{Theorem 10.} 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 $O(k)$ in $O(n^{k-\varepsilon})$ time, then SETH is false.

\textbf{Proof.} Follows from Lemma 8 and Lemma 9.
2.5 Proof of Lemma 8

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 $i \in [1, k]$, 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 a complete edit sequence of our strings that requires $E^-$ or fewer edits. 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 $DG_1(v_1), DG_2(v_2), \ldots, DG_k(v_k)$. 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 $DG_1, VG_2, \ldots, VG_k$, 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 $E^- = D^- + (100kn + n - 1)D^+$. 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-t1-and-t2-all-vector-gadgets-in-p2-align-with-decision-gadgets-dg1-in-t1.png)

Then there are exactly $2(50k + 1)n$ alignments of $VG_1(\phi), VG_2(\phi), \ldots, VG_k(\phi)$, and for all $i \in [2, k]$ there are exactly $2n$ alignments containing precisely the gadgets $VG_i(\phi), VG_{i+1}(\phi), \ldots, VG_k(\phi)$. 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 11. For all $i \in [1, k]$, EDIT($VG_i(\phi), VG_{i+1}(\phi), \ldots, VG_k(\phi)$) = $F_i$.

Proof. Each vector gadget $VG_j(\phi)$ is composed of $d + r + 1$ coordinate gadgets. Each alignment of the coordinate gadgets $CG_i(1), CG_{i+1}(1), \ldots, CG_k(1)$ 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 \( VG_i, VG_{i+1}, \ldots, VG_k \) gadgets for \( i \in [1, k] \) by
\[
E^-_a = 2(50k + 1)nF_1 + \sum_{i \in [2, k]} 2nF_i = 101k(k - 1)(2k - 1)(d + r + 1)n
\]
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 \( i < j \). To finish our edit sequence and make all strings equal, we extend all \( P_i^* \) for \( i \in [1, k - 1] \) to match \( P_k^* \). We achieve this for a given \( P_i^* \) by substituting \( |P_i^*| - |P_k^*| \) of the \( S_i \) symbols in \( T_i \) and deleting the remaining \( S_i \) symbols in \( T_i \). Since we substitute or delete every \( S_i \) symbol, this will incur an edit cost of 
\[
\text{Total Number of Edits} = 2(k - 1)\ell_5. 
\]
The total number of edits performed in our edit sequence is no more than
\[
E^- + E^- + E^- = E^- + E^- + (k - 1)
\]
This completes the proof.

### 2.6 Proof of Lemma 9

**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 \( i \in [1, k] \), \( \text{EDIT}(T_1, T_2, \ldots, T_k) = E^+ = E^- + (k - 1) \).

\( \triangleright \) Claim 12. \( \text{EDIT}(T_1, T_2, \ldots, T_k) \leq E^+ \)

**Proof.** We can achieve this upper bound by giving an edit sequence identical to the edit sequence in Lemma 8. 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 8 is now \( D^+ \). This yields a complete edit sequence with \( E^- + (D^+ - D^-) = E^+ \) edits, so our inequality holds.

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

\( \triangleright \) Claim 13. Every \( S_i \) symbol in \( T_i \) for \( i \in [1, k - 1] \) incurs at least one edit in our edit sequence.

**Proof.** Observe that each \( S_i \) symbol occurs only in \( T_i \) for \( i \in [1, k - 1] \). Then each \( S_i \) symbol is deleted or is aligned with other symbols not equal to \( S_i \) and incurs one edit.

There are 2\((k - 1)\ell_5\) of the \( S_i \) symbols in \( T_1, T_2, \ldots, T_k \), so they incur at least \( E_1^+ = 2(k - 1)\ell_5 \) edits.

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 \( j \)th leftmost vector level gadget in \( T_i \), we add a vertex \( x_i^j \) to \( G \) for \( i \in [1, k] \). Thus vertices \( x_1^1, x_1^2, \ldots, x_1^{200k + 2i + 1}n \) 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 \( (x_i^{j_1}, x_i^{j_2}) \) if the \( j_1 \)th vector level gadget of \( T_i \) is aligned with the \( j_2 \)th vector level gadget of \( T_i \) in the edit sequence. Also, we say that \( x_i^{j_1} \) and \( x_i^{j_2} \) 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 \( C \). For a component \( c \in C \), we define \( \#(c, i) \) as the
number of vertices belonging to string $T_i$ in $c$. We say that 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 $C$ into the following sets:

- $C_1$ is the set of all components $c$ with width$(c) > 1$
- $C_2$ is the set of all components $c$ with width$(c) = 1$ and $\#(c,k) = 0$
- $C_3$ is the set of all components $c$ with width$(c) = 1$ and $\#(c,k) = 1$

We now lower bound the edit costs of components in $C_1$, $C_2$, and $C_3$. Let $Q = 10kd\ell_3$.

**Lemma 14.** Every component $c$ in $C_1$ incurs at least $Q \cdot \text{width}(c)$ edits.

**Proof.** Because our component $c$ is connected, the case illustrated in Figure 5 must occur at least width$(c) - 1$ times. Then at least $2\ell_4(\text{width}(c) - 1)$ 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 $Q \cdot \text{width}(c)$. These edits are disjoint from the edits of the $\$i$ symbols.

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

**Lemma 15.** Every component $c$ in $C_2$ incurs at least $Q \ell_4$ edits.

**Proof.** By definition, the vector level gadgets in component $c$ have no alignments with any vector level gadget $\text{VG}_k$ in $T_k$. It follows that we incur a cost of at least $|\text{VG}_k| > Q$. Furthermore, this edit cost is disjoint from the $E_i^+$ 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 $C_1$ and $C_2$, and these edit costs are disjoint by nature. Now we bound the costs of every component in $C_3$. It will be useful to partition the components in $C_3$ into the following sets:

- $C_{3,1}$ is the set of all components $c$ containing a vertex corresponding to a $\text{DG}_1$ gadget
- $C_{3,2}$ is the remaining components in $C_3$.

**Lemma 16.** All components $c$ in $C_{3,1}$ incur an edit cost of $D^+$.

**Proof.** We find the following claim useful in our proof.

**Claim 17.** No optimal edit sequence aligns a decision gadget $\text{DG}_1$ with any $\$i$ symbol.

\[ \text{Proof.} \] Suppose some decision gadget $\text{DG}_1$ 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 $\text{DG}_1$ is aligned with a $\$i$ symbol on the left side of $T_i$. It follows that the substring $\text{VG}_1'(\phi)^{(50k+1)n}$ 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 $|\text{VG}_1'(\phi)^{(50k+1)n}| + |P'_i|$. We observe that $|\text{VG}_1'(\phi)^{(50k+1)n}| > 100kn\ell_4$ and $|P'_i| > 400kn\ell_4$, so our alignment of $T_1$
and $T_i$ has a combined length greater than $500k\ell_4$. On the other hand, $|T_k| = (202k + 1)n|VG_k| < 203kn(3\ell_3 + 2\ell_4)$. Our alignment of $T_i$ and $T_k$ must be edited to have the same length as $T_k$ in every complete edit sequence, so it follows that EDIT$(T_1, T_i, T_k) > 500kn\ell_4 - 203kn(3\ell_3 + 2\ell_4) = kn(94\ell_4 - 609\ell_3) > 100k^4dn\ell_3$. Then our edit sequence requires $100k^4dn\ell_3 + E^+_1 > E^+$ edits, so this alignment cannot occur in an optimal edit sequence.

Let $c$ be a component in $C_{3,1}$. 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 $|VG_i| > D^+$ insertions in $T_i$. Furthermore, these edits are disjoint from the $E^+_1$ cost of editing the $S_i$ symbols by Claim 17. Else, we have that $\#(c, i) = 1$ for all $i \in [1, k]$, and by our analysis in Lemma 8, the edit cost of aligning the $k$ vector level gadgets is at least $D^+$.

▶ Lemma 18. Let $c$ be a component in $C_{3,2}$ and let $\lambda = |c|$, then the edit cost incurred by the vector gadgets in $c$ is $(d + r + 1)(2k - 1)(\lambda - 1)$.

Proof. We begin with a claim (proof is similar to Claim 17 and is deferred to Appendix B).

▷ Claim 19. Let $v_i \in S_i$ for some $i \in [2, k]$, then no optimal edit sequence aligns the vector gadget $VG_i(v_i)$ in $T_i$ with a $S_i$ symbol in $T_1$, nor a dummy vector gadget $VG_1(\phi)$ in $T_1$.

Let $c$ be in $C_{3,2}$. Suppose there is some $v_i \in S_i$ for $i \in [2, k]$ such that vector gadget $VG_i(v_i)$ corresponds to a vertex in component $c$. Then the gadgets in our component cannot align with any decision gadgets $DG_1$, vector gadgets $VG_1(\phi)$, or $S_1$ symbols in $T_1$. It follows that we must perform at least $|VG_i| > (d + r + 1)(2k - 1)(\lambda - 1)$ insertions in $T_i$. Else, all vertices in component $c$ correspond only to vector gadgets $VG_i(\phi)$ for $i \in [1, k]$. By a similar argument as in Claim 11, 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 $C_1, C_2$, and $C_3$. Now we must combine our component level arguments to obtain an overall lower bound on the edit cost. Let $W = \sum_{c \in C_1 \cup C_2} \text{width}(c)$. Then we know that the components in $C_1 \cup C_2$ incur a cost of at least $E^+_2 = WQ$ edits by Lemma 14 and Lemma 15.

We now lower bound the total number of edits from components in $C_3$. Note that components in $C_{3,1}$ incur a much higher cost than components in $C_{3,2}$. Then to lower bound the edits in $C_3$, we must assume the least possible number of components in $C_{3,1}$. There are $(100k + 1)n$ decision gadgets $DG_1$ in our final strings and at most $W$ decision gadgets in components in $C_1 \cup C_2$, so there must be at least $Z_1 = (100k + 1)n - W$ components in $C_{3,1}$. Note that if $W \geq (100k + 1)n$, then $E^+_1 + E^+_3 \geq E^+$, so we may assume $Z_1$ is positive. Then components from $C_{3,1}$ incur a cost of at least $E^+_3 = Z_1D^+$ by Lemma 16.

There are at most $V_0 = kW$ vertices in components in $C_1 \cup C_2$, and there are at most $V_1 = kZ_1$ vertices in $C_{3,1}$. 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 $C_{3,2}$.

Because our edit cost lower bound for every component in $C_{3,2}$ is linear in the component size, we have the following.

▷ Claim 20. Suppose there are $Z$ components in $C_{3,2}$ and a total of $V$ vertices in all components in $C_{3,2}$. Then the components in $C_{3,2}$ incur $(d + r + 1)(2k - 1)(V - Z)$ edits.
Proof. By Lemma 18, each component of size $\lambda \in C_{3,2}$ incurs cost $(d + r + 1)(2k - 1)(\lambda - 1)$. Let $z_i$ denote the size of the $i$th component in $C_{3,2}$ for $i \in [1, Z]$. Then we may sum the edit costs of all components in $C_{3,2}$:

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

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

Claim 20 proves that the edit cost of all the components in $C_{3,2}$ decreases with the number of components $Z$. Then to achieve our lower bound we must upper bound the number of components in $C_{3,2}$. 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 $C_{3,2}$. It follows that the total edit cost contributed by the components of $C_{3,2}$ 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. Hardness for $k$-Center-Edit-Distance

We now provide a simple, yet previously unknown reduction from the $k$-Median-Edit-Distance to $k$-Center-Edit-Distance. Given a set of strings $X = \{x_1, x_2, \ldots, x_k\}$, each of length $n$ over an alphabet $\Sigma$, we define another set of strings $Y = \{y_1, y_2, \ldots, y_k\}$ over an alphabet $\Sigma' = \Sigma \cup \{\$\}$ (where $\$ \not\in \Sigma$) as follows (fix $\ell = k^2n$):

$$
y_1 = x_1 \circ \$^\ell \circ x_2 \circ \$^\ell \cdots \circ \$^\ell \circ x_k \circ \$^\ell \circ x_k
$$

$$
y_2 = x_2 \circ \$^\ell \circ x_3 \circ \$^\ell \cdots \circ \$^\ell \circ x_k \circ \$^\ell \circ x_1
$$

$$
\vdots
$$

$$
y_k = x_k \circ \$^\ell \circ x_1 \circ \$^\ell \cdots \circ \$^\ell \circ x_{k-2} \circ \$^\ell \circ x_{k-1}
$$

It can be easily verified that the $k$-Center-Edit-Distance of the strings in $Y$ is the same as the $k$-Median-Edit-Distance of the strings in $X$. The length of each string in $Y$ is $(k-1)k^2n + k\ell = O(n)$. Therefore, an $O(n^{k-\varepsilon})$ time algorithm for the $k$-Center-Edit-Distance would give an $O(n^{k-\varepsilon})$ time algorithm for the $k$-Median-Edit-Distance and contradict SETH.

Theorem 21. If there is an $\varepsilon > 0$, an integer $k \geq 2$, and an algorithm that can solve $k$-Center-Edit-Distance on strings, each of length at most $n$, over an alphabet of size $O(k)$ in $O(n^{k-\varepsilon})$ time, then SETH is false.

4. Discussion

Based on SETH, we have shown tight conditional hardness results for median string, center string, tree-alignment, and bottleneck-tree alignment problems, all under edit distance. These results show optimality (at least up to logarithmic factors) of algorithms for median string and tree-alignment problems established many decades ago. 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 known dynamic programming algorithm works in time $O(n^{2k})$ [41], and as far as the authors no such algorithm for bottleneck-tree alignment on more general trees. We conclude by asking: is an $O(n^k)$ algorithm is waiting to be found for these problems, or does there exists a more efficient reduction which can prove that an $O(n^{2k-\varepsilon})$ algorithm highly improbable?
References


14 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *IEEE 56th Annual Symposium on Foundations of


Fine-grained Complexity of the Median and Center String Problems under Edits


A  Proof of Lemma 7

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

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 subset
\( v_1, v_2, \ldots, v_k \) of strings \( v_1, v_2, \ldots, v_k \), \( \text{EDIT}(v_1, v_2, \ldots, v_k) \leq \text{EDIT}(v_1, v_2, \ldots, v_k) \).

\[ \blacktriangledown \text{Claim 22.} \quad \text{EDIT}(\text{DG}_1(v_1), \text{VG}_2(v_2), \text{VG}_3(v_3), \ldots, \text{VG}_k(v_k)) \leq D^+ \]

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

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

Case 1: The \( M_i(v_i) \) substring of some \( \text{VG}_i(v_i) \) gadget with \( i > 1 \) has alignments with both
substrings \( 7^s \circ M_1(v_1) \) and \( M_i(\theta) \circ 7^s \) of \( \text{DG}_1(v_1) \). In this case, the cost induced by the
symbols in the \( 7^s \) prefix and suffix of \( \text{DG}_1(v_1) \) and the \( 6^s \) substring of \( \text{DG}_1(v_1) \) is \( \ell_3 \) each,
so \( \text{EDIT}(\text{DG}_1(v_1), \text{VG}_1(v_1)) \geq 3\ell_3 > D^+ \). Our lower bound is satisfied.

Case 2: The \( M_i(v_i) \) substring of some \( \text{VG}_i(v_i) \) gadget with \( i > 1 \) does not have any align-
ments with the \( 7^s \circ M_1(v_1) \) substring of \( \text{DG}_1(v_1) \).

Case 2.1: The \( M_j(v_j) \) substring of some \( \text{VG}_j(v_j) \) gadget with \( j > 1 \) does not have any align-
ments with substring \( M_1(\theta) \circ 7^s \) of \( \text{DG}_1(v_1) \). We will consider \( \text{EDIT}(\text{VG}_1(v_1), \text{VG}_j(v_j), \text{DG}_1(v_1)) \)
or \( \text{EDIT}(\text{VG}_i(v_i), \text{DG}_1(v_1)) \) if \( i = j \). The \( M_i(v_i) \) substring of \( \text{VG}_i(v_i) \) has no alignments
with the \( 7^s \circ M_1(v_1) \) substring of \( \text{DG}_1(v_1) \). Therefore at least \( D_1 = \ell_3 + m \) edits need
to be performed between the \( 6^s \) prefix of \( \text{VG}_1(v_1) \) and the \( 7^s \circ M_1(v_1) \) prefix of \( \text{VG}_1(v_1) \).

Likewise, the \( M_j(v_j) \) substring of \( \text{VG}_j(v_j) \) has no alignments with the \( M_1(\theta) \circ 7^s \) substring
of \( \text{DG}_1(v_1) \), and so at least \( D_1 \) edits need to be performed between the \( 6^s \) suffix of \( \text{VG}_j(v_j) \)
and the \( M_1(\theta) \circ 7^s \) suffix of \( \text{DG}_1(v_1) \). The above edit costs are disjoint, and it follows that
\( \text{EDIT}(\text{VG}_1(v_1), \text{VG}_j(v_j), \text{DG}_1(v_1)) \geq 2D_1 > D^+ \). Our lower bound is satisfied.

Case 2.2: We consider the complement of Case 2.1: the \( M_i(v_i) \) substrings of all \( \text{VG}_i(v_i) \)
gadgets with \( i > 1 \) have alignments with the substring \( M_1(\theta) \circ 7^s \) of \( \text{DG}_1(v_1) \). By our
analysis in Case 1, we may now assume that the \( M_i(v_i) \) substrings of all \( \text{VG}_i(v_i) \) gadgets
with \( i > 1 \) do not have alignments with the \( 7^s \circ M_1(v_1) \) substring of \( \text{DG}_1(v_1) \). Then by our
argument in Case 2.1, at least \( D_1 \) edits must be performed on the \( 6^s \) prefix of \( \text{VG}_1(v_1) \) and
the \( 7^s \circ M_1(v_1) \) prefix of \( \text{VG}_1(v_1) \). Additionally, note that all \( \text{VG}_i(v_i) \) share the suffix \( 6^s \),
whereas \( \text{DG}_1(v_1) \) has suffix \( 7^s \). It follows that at least \( D_2 = \ell_3 \) edits are needed to edit
\( \text{DG}_1(v_1), \text{VG}_2(v_2), \ldots, \text{VG}_k(v_k) \) to have the same suffix. Furthermore, these edits are disjoint
from the \( D_1 \) edits performed on the prefixes of \( \text{DG}_1(v_1) \) and the \( \text{VG}_i(v_i) \). We have shown
that at least \( D_1 + D_2 = 2\ell_3 + m \) edits are required to align \( \text{DG}_1(v_1), \text{VG}_2(v_2), \ldots, \text{VG}_k(v_k) \).
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 \( \text{CG}_i(v_i[j]) \).

Case 2.2.1: There is some \( \text{VG}_i(v_i) \) gadget with \( i > 1 \) such that there are some \( j, \ell \in [1, d + r + 1] \) with \( j \neq \ell \) such that the \( j \)th leftmost coordinate gadget of \( M_i(v_i) \) is aligned
with the \( \ell \)th leftmost coordinate gadget of the \( M_1(\theta) \) in \( \text{VG}_1(v_1) \). Then we incur an edit
cost of at least \( 2\ell_2 \) from the 5 symbols between the coordinate gadgets. It follows that
EDIT(DG_1(v_1), VG_2(v_2), ..., 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 $i$th leftmost coordinate gadget of $M_1(v_j)$ for all $j > 1$ is either aligned with the $i$th 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 $i$th leftmost coordinate gadgets in $M_1(\theta), M_2(v_2), ..., M_k(v_k)$. If, for some $M_j(v_j)$ with $j > 1$, the $i$th leftmost coordinate gadget $CG_j(v_j[i])$ is not aligned with any coordinate gadget of $M_1(\theta)$, then it incurs cost $|CG_j(v_j[i])| \geq C^+$. Else the $i$th leftmost coordinate gadgets of all $M_j(v_j)$ for $j > 1$ are aligned with the $i$th leftmost coordinate gadget of $M_1(\theta)$. Then by the transitivity of the alignment relation, we have that the $i$th leftmost coordinate gadgets of $M_1(\theta), M_2(v_2), ..., M_k(v_k)$ are aligned. By our analysis of the coordinate gadgets in Lemma 2, this alignment of coordinate gadgets will incur cost at least $C^-$ if $a_\theta[i]v_2[i]v_3[i]...v_k[i] = 0$, and else incur cost at least $C^+$ if $a_\theta[i]v_2[i]v_3[i]...v_k[i] = 1$.

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 $\theta, v_2, v_3, ..., v_k$ is $r + 1$ (this follows from our modification of the input vectors and our definition of $\theta$). 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 $EDIT(DG_1(v_1), VG_2(v_2), ..., VG_k(v_k)) \geq D^+$.

**Case 3:** The $M_i(v_i)$ substring of some $VG_i(v_i)$ with $i > 1$ does not have alignments with the $M_1(\theta) \circ T_{\theta^3}$ substring of $DG_1(v_1)$. 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 $v_1, v_2, ..., v_k$ are $r$-close and hence have an inner product greater than or equal to $r + 1$, it must be the case that $EDIT(DG_1(v_1), VG_2(v_2), ..., VG_k(v_k)) \geq D^+$.

We have shown in every case that $EDIT(DG_1(v_1), VG_2(v_2), ..., VG_k(v_k)) \geq D^+$, so we conclude that $EDIT(DG_1(v_1), VG_2(v_2), ..., VG_k(v_k)) = D^+$.

**B Proof of Claim 19**

$\triangleright$ Claim 19. Let $v_i \in S_i$ for some $i \in [2, k]$, then no optimal edit sequence aligns the vector gadget $VG_i(v_i)$ in $T_i$ with a $S_1$ symbol in $T_1$, nor a dummy vector gadget $VG_1(\phi)$ in $T_1$.

Suppose some vector gadget $VG_i(v_i)$ in string $T_i$ with $i \in [2, k]$ and $v_i \in S_i$ is aligned with a dummy vector gadget $VG_1(\theta)$ 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 $VG_i(v_i)$ is aligned with a $VG_1(\theta)$ 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 $DG_1(\phi)^{50kn} \circ P_1 \circ R_1$ 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 $|L_i| + |DG_1(\phi)^{50kn} \circ P_1 \circ R_1|$.

We observe that $|L_i| > 200kn \ell_4$ and $|DG_1(\phi)^{50kn} \circ P_1 \circ R_1| > 400kn \ell_4$, so our alignment of $T_i$ and $T_1$ has a combined length greater than $600kn \ell_4$. On the other hand, $|T_k| = (202k + 1)n|VG_1^k| < 203kn(3\ell_3 + 2\ell_4)$.

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 $EDIT(T_i, T_1, T_k) > 600kn \ell_4 - 203kn(3\ell_3 + 2\ell_4) = kn(194\ell_4 - 609\ell_3) > 1000k^4dn \ell_3$. Then our edit sequence requires $1000k^4dn \ell_3 + E^+ > E^+$ edits, so this alignment cannot occur in an optimal edit sequence. It follows that $VG_i(v_i)$ in $T_i$ cannot align with a $VG_1(\theta)$ gadget (and by extension a $S_1$ symbol) in $T_1$. 
