Computing the Tandem Duplication Distance is NP-Hard

Manuel Lafond, Binhai Zhu, Peng Zou

COMPUTING THE TANDEM DUPLICATION DISTANCE IS NP-HARD*

MANUEL LAFOND†, BINHAI ZHU‡, AND PENG ZOU §

Abstract. In computational biology, tandem duplication is an important biological phenomenon which can occur either at the genome or at the DNA level. A tandem duplication takes a copy of a genome segment and inserts it right after the segment — this can be represented as the string operation $AXB \Rightarrow AXXB$. Tandem exon duplications have been found in many species such as human, fly or worm, and have been largely studied in computational biology.

The *Tandem Duplication* (TD) distance problem we investigate in this paper is defined as follows: given two strings S and T over the same alphabet Σ, compute the smallest sequence of tandem duplications required to convert S to T. The natural question of whether the TD distance can be computed in polynomial time was posed in 2004 by Leupold et al. and had remained open, despite the fact that tandem duplications have received much attention ever since. In this paper, we focus on the special case when all characters of S are distinct. This is known as the *exemplar* TD distance, which is of special relevance in bioinformatics. We first prove that this problem is NP-hard when the alphabet size is unbounded, settling the 16-year old open problem. We then show how to adapt the proof to $|\Sigma| = 4$, hence proving the NP-hardness of the TD problem for any $|\Sigma| \geq 4$. One of the tools we develop for the reduction is a new problem called *Cost-Effective Subgraph*, for which we obtain W[1]-hardness results that might be of independent interest. We finally show that computing the exemplar TD distance between S and T is fixed-parameter tractable. Our results open the door to many other questions, and we conclude with several open problems.

Key words. Tandem duplication, Text processing, Formal languages, Computational genomics, FPT algorithm

AMS subject classifications. 68Q25, 68R10, 68W32

1. Introduction.

Tandem duplication is a biological process that creates consecutive copies of a segment of a genome during DNA replication. Representing genomes as strings, this event transforms a string AXB into another string $AXXB$. This process is known to occur either at small scale at the nucleotide level, or at large scale at the genome level [5, 6, 7, 23, 30]. For instance, it is known that the Huntington disease is associated with the duplication of 3 nucleotides `CAG` [26], whereas at genome level, tandem duplications are known to involve multiple genes during cancer progression [27]. Furthermore, gene duplication is believed to be the main driving force behind evolution, and the majority of duplications affecting organisms are believed to be of the tandem type (see e.g. [31]). As a result, around 3% of the human genome are formed of tandem repeats.

For these reasons, tandem duplications have received significant attention in the last decades, both in practice and theory. The combinatorial aspects of tandem duplications have been studied extensively by computational biologists [2, 12, 14, 21, 32], one question of interest being to reconstruct the evolution of a cluster of tandem repeats by duplications that could have given rise to the observed sequences. In parallel, various formal language communities [9, 33, 25] have investigated the expressive

*Submitted to the editors DATE. An abstract of this paper was presented at STACS'20 [20], and was also presented as part of an invited talk at CSR'20 [34].

Funding: This work was funded by NSERC of Canada.

†Department of Computer Science, Universite de Sherbrooke, Sherbrooke, Quebec J1K 2R1, Canada (manuel.lafond@usherbrooke.ca).

‡Gianforte School of Computing, Montana State University, Bozeman, MT 59717-3880, USA (bhz@montana.edu).

§Gianforte School of Computing, Montana State University, Bozeman, MT 59717-3880, USA (peng.zou@student.montana.edu).

power of tandem duplications on strings.

From the latter perspective, a natural question arises: given a string S, what is the language that can be obtained starting from S and applying (any number of) tandem duplications, i.e. rules of the form $AXB \to AXXB$, where X can be any substring of S? This question was first asked in 1984 in the context of so-called *copying systems* [11]. Combined with results from [3], it was shown that this language is regular if S is on a binary alphabet, but not regular for larger alphabets. These results were rediscovered 15 years later in [9, 33]. In [25], it was shown that the membership, inclusion and regularity testing problems on the language defined by S can all be decided in linear time (still on binary alphabets). In [25, 24, 17], similar problems are also considered on non-binary alphabets, when the length $|X|$ of duplicated strings is bounded by a constant. More recently, Cho et al. [8] introduced a tandem duplication system where the *depth* of a character, i.e. the number of "generations" it took to generate it, is considered. In [16, 18], the authors study the *expressive power* of tandem duplications, a notion based on the subsequences that can be obtained from various types of copying mechanisms.

More directly related to our work, Alon et al. [1] recently investigated the minimum number of duplications required to transform a string S into another string T. We call this the *Tandem Duplication (TD) distance*. More specifically, the authors show that on binary strings, the maximum TD distance between a square-free string S and a string T of length n is $\Theta(n)$. They also mention the unsolved algorithmic problem of computing the TD distance between S and T. In fact, this question was posed earlier in [25] (pp. 306, Open Problem 3) by Leupold et al. and has remained open ever since. We settle this open problem in this paper for an unbounded alphabet. (We then extend the result when the alphabet size is 4.) As will be seen, our technique is different from that used in [1], which only works for binary strings.

On the other hand, the TD distance is one of the many ways of comparing two genomes represented as strings in computational biology — other notable examples include breakpoint [15] and transpositions distances, the latter having recently been shown NP-hard in a celebrated paper of Bulteau et al. [4]. The TD distance has itself received special attention recently, owing to its role in cancer evolution [28].

Our results. In this paper, we solve the problem posed by Leupold et al. in 2004 and show that computing the TD distance from a string S to a string T is NP-hard. We show that this result holds even if S is *exemplar*, i.e. if each character of S is distinct. Exemplar strings are commonly studied in computational biology [29], since they represent genomes that existed prior to duplication events. We note that simply deciding if S can be transformed into T by a sequence of TDs still has unknown complexity. In our case, we show that the hardness of minimizing TDs holds on instances in which such a sequence is guaranteed to exist.

As demonstrated by the transpositions distance in [4], obtaining NP-hardness results for string distances can sometimes be an involving task. Our hardness reduction is also quite technical, and one of the tools we develop for it is a new problem we call the *Cost-Effective Subgraph*. In this problem, we are given a graph G with a cost c, and we must choose a subset X of $V(G)$. Each edge with both endpoints in X has a cost of $|X|$, every other edge costs c, and the goal is to find a subset X of minimum cost. We show that this problem is W[1]-hard for parameter $p + c$, where p is a parameter that asks if one can achieve a cost of at most $c|E(G)| - p$ (here $c|E(G)|$

is an upper bound on the cost[1]). The problem enforces optimizing the tradeoff between covering many edges versus having a large subset of high cost, which might be applicable to other problems. In our case it captures the main difficulty in computing TD distances. We then obtain some positive results by showing that if S is exemplar, then one can decide if S can be transformed into T using at most k duplications in time $2^{O(k^2)} + poly(n)$, where n is the length of T. The result is obtained through an exponential size kernel. All of our results concern strings with unbounded alphabet sizes. Finally, we conclude with several open problems that might be of interest to the theoretical computer science community.

This paper is organized as follows. In Section 2, we give basic definitions. In Section 3, we show that computing the TD distance is NP-hard through the Cost-Effective Subgraph problem when $|\Sigma|$ is infinite. In Section 4, we show that the same problem is NP-hard for any constant $|\Sigma| \geq 4$. In Section 5, we show that computing the exemplar TD distance is FPT. In Section 6, we conclude the paper with several open problems.

2. Preliminary notions. We borrow the string terminology and notation from [13]. In particular, $[n]$ denotes the set of integers $\{1, 2, \ldots, n\}$. Unless stated otherwise, all the strings in the paper are on an alphabet denoted Σ. If S_1 and S_2 are two strings, we usually denote their concatenation by S_1S_2. For a string S, we write $\Sigma(S)$ for the subset of characters of Σ that have at least one occurrence in S. A string S is called *exemplar* if $|S| = |\Sigma(S)|$, i.e. each character present in S occurs only once. A *substring* of S is a contiguous sequence of characters within S. A *prefix* (resp. *suffix*) of S is a substring that occurs at the beginning (resp. end) of S, i.e. if $S = S_1S_2$ for some strings S_1 and S_2, then S_1 is a prefix of S and S_2 a suffix of S. A *subsequence* of S is a string that can be obtained by successively deleting characters from S.

A *tandem duplication* (TD) is an operation on a string S that copies a substring X of S and inserts the copy after the occurrence of X in S. In other words, a TD transforms $S = AXB$ into $AXXB$. Given another string T, we write $S \Rightarrow T$ if there exist strings A, B, X such that $S = AXB$ and $T = AXXB$. More generally, we write $S \Rightarrow_k T$ if there exist $S_1, \ldots, S_{k-1}$ such that $S \Rightarrow S_1 \Rightarrow \ldots \Rightarrow S_{k-1} \Rightarrow T$. We also write $S \Rightarrow_* T$ if there exists some k such that $S \Rightarrow_k T$.

DEFINITION 2.1. *The TD distance $dist_{TD}(S, T)$ between two strings S and T is the minimum value of k satisfying $S \Rightarrow_k T$. If $S \Rightarrow_* T$ does not hold, then $dist_{TD}(S, T) = \infty$.*

We use the term *distance* here to refer to the number of TD operations from a string S to another string T, but one may note that TD is not a metric in the formal sense. In particular, $dist_{TD}$ is not symmetric since duplications can only increase the length of a string.

A *square string* is a string of the form XX, i.e. a concatenation of two identical substrings. A string is *square-free* if none of its substring is a square. Given a string S, a *contraction* is the reverse of a tandem duplication. That is, it takes a square string XX contained in S and deletes one of the two copies of X. We write $T \rightarrowtail S$ if there exist strings A, B, X such that $T = AXXB$ and $S = AXB$. We also define $T \rightarrowtail_k S$ and $T \rightarrowtail_* S$ for contractions analogously as for TDs. Observe that by the symmetry of duplications and contractions, $T \rightarrowtail_k S$ if and only if $S \Rightarrow_k T$ and $T \rightarrowtail_* S$ if and

[1] In other words, if we were to state the maximization version of the Cost-Effective Subgraph problem, p would be the value to maximize. The minimization version, however, is more convenient to use for our needs.

only if $S \Rightarrow_* T$. When there is no possible confusion, we will sometimes write $T \rightarrowtail S$ instead of $T \rightarrowtail_* S$.

We have the following problem.

The Tandem Duplication (TD) problem:
Input: Two strings S and T over the same alphabet Σ and an integer k.
Question: Is $dist_{TD}(S,T) \leq k$?

In the Exemplar-TD variant of this problem, S is required to be exemplar. In either variant, we may call S the *source string* and T the *target string*. We will often use the fact that S and T form a YES instance if and only if T can be transformed into S by a sequence of at most k contractions. See Fig.1 for a simple example.

		Sequence	*Operations*
Sequence	$T = \langle a, c, \underline{g, g}, a, c, g\rangle$		*contraction on* $\langle g, g\rangle$
		$\langle \underline{a, c, g, a, c, g}\rangle$	*contraction on* $\langle a, c, g, a, c, g\rangle$
Sequence	$S = \langle a, c, g\rangle$		

FIG. 1. *An example for transforming sequence T to S by two contractions. The corresponding sequence of TDs from S to T would duplicate a, c, g, and then duplicate the first g.*

We recall that although we study the minimization problem here, it is unknown whether the question $S \Rightarrow_* T$ can be decided in polynomial time. Nonetheless, our NP-hardness reduction applies to 'promise' instances in which $S \Rightarrow_* T$ always holds.

3. NP-hardness of Exemplar-TD. To facilitate the presentation of our hardness proof, we first make an intermediate reduction using the Cost-Effective Subgraph problem, which we will then reduce to the promise version of the Exemplar-TD problem.

The Cost-Effective Subgraph problem. Suppose we are given a graph $G = (V, E)$ and an integer cost $c \in \mathbb{N}_{>0}$. For $X \subseteq V$, let $E(X) = \{uv \in E : u, v \in X\}$ denote the edges inside of X. The *cost* of X is defined as

$$cost(X) = c \cdot (|E(G)| - |E(X)|) + |X| \cdot |E(X)|.$$

The Cost-Effective Subgraph problem asks for a subset X of minimum cost. In the decision version of the problem, we are given an integer r and we want to know if there is a subset X whose cost is at most r. Observe that $X = \emptyset$ or $X = V$ are possible solutions.

The idea is that each edge "outside" of X costs c and each edge "inside" costs $|X|$. Therefore, we pay for each edge not included in X, but if X gets too large, we pay more for edges in X. We must therefore find a balance between the size of X and its number of edges. The connection with the TD problem can be roughly described as follows: in our reduction, we will have many substrings which need to be deleted through contractions. We will have to choose an initial set of contractions X and then, each substring will have two ways to be contracted: one way requires c contractions, and the other requires $|X|$.

An obvious solution for a Cost-Effective Subgraph is to take $X = \emptyset$, which is of cost $c|E(G)|$. Another formulation of the problem could be whether there is a subset X of cost at most $c|E(G)|-p$, where p can be seen as a "profit" to maximize. Treating c and p as parameters, we show the NP-hardness and W[1]-hardness in parameters $c+p$ of the Cost-Effective Subgraph problem (we do not study the parameter r). Our reduction to the TD problem does not preserve W[1]-hardness and we only use the NP-hardness in this paper, but the W[1]-hardness might be of independent interest.

Before proceeding, we briefly argue the relevance of parameter c in the W[1]-hardness. If c is a fixed constant, then we may assume that any solution X satisfies $|X| \leq c$. This is because if $|X| > c$, every edge included in X will cost more than c and putting $X = \emptyset$ yields a lower cost. Thus for fixed c, it suffices to brute-force every subset X of size at most c and we get a $n^{O(c)}$ time algorithm. Our W[1]-hardness shows that it is difficult to remove this exponential dependence between n and c.

THEOREM 3.1. *The Cost-Effective Subgraph problem is NP-hard and W[1]-hard for parameter $c+p$.*

Proof. We reduce from CLIQUE. In this classic problem, we are given a graph G and an integer k, and must decide whether G contains a clique of size at least k, where a clique is a set of vertices in which every pair shares an edge. This problem is NP-hard [19] and also W[1]-hard in parameter k [10]. We will assume that k is even (which does not alter either hardness results).

Let (G, k) be a CLIQUE instance, letting $n := |V(G)|$ and $m := |E(G)|$. The graph in our Cost-Effective Subgraph instance is also G. We set the cost $c = 3k/2$, which is an integer since k is even, and set

$$r := c\left(m - \binom{k}{2}\right) + k\binom{k}{2} = cm + \binom{k}{2}(k-c) = cm - \frac{k}{2}\binom{k}{2}.$$

We ask whether G admits a subgraph X satisfying $cost(X) \leq r$. We show that (G, k) is a YES instance to CLIQUE if and only if G contains a set $X \subseteq V(G)$ of cost at most r. This will prove both NP-hardness and W[1]-hardness in $c+p$ (noting that here $p = k/2\binom{k}{2}$).

The forward direction is easy to see. If G is a YES instance, it has a clique X of size exactly k. Since $|E(X)| = \binom{k}{2}$, the cost of X is precisely r.

Let us consider the converse direction. Assume that (G, k) is a NO instance of CLIQUE. Let $X \subseteq V(G)$ be any subset of vertices. We will show that $cost(X) > r$. There are 3 cases to consider depending on $|X|$.

Case 1: $|X| = k$. Since G is a NO instance, X is not a clique and thus $|E(X)| = \binom{k}{2} - h$, where $h > 0$. We have that $cost(X) = c(m - \binom{k}{2} + h) + k(\binom{k}{2} - h) = cm + \binom{k}{2}(k-c) + h(c-k) = r + h(c-k)$. Since $c > k$ and $h > 0$, the cost of X is strictly greater than r.

Case 2: $|X| = k + l$ for some $l > 0$. Denote $|E(X)| = \binom{k+l}{2} - h$, where $0 \leq h \leq \binom{k+l}{2}$ The cost of X is

$$\begin{aligned} cost(X) &= c\left(m - \binom{k+l}{2} + h\right) + (k+l)\left(\binom{k+l}{2} - h\right) \\ &= cm + \binom{k+l}{2}(k+l-c) + h(c-k-l) \\ &= cm + \binom{k+l}{2}\left(l - \frac{k}{2}\right) + h\left(\frac{k}{2} - l\right). \end{aligned}$$

Considering the difference

$$\begin{aligned} cost(X) - r &= \binom{k+l}{2}\left(l - \frac{k}{2}\right) + h\left(\frac{k}{2} - l\right) - \left(-\frac{k}{2}\right)\binom{k}{2} \\ &= \frac{3kl^2}{4} - \frac{kl}{4} + \frac{l^3}{2} - \frac{l^2}{2} + h\left(\frac{k}{2} - l\right), \end{aligned}$$

if $k/2 - l \geq 0$, then the difference is clearly above 0 regardless of h, and then $cost(X) > r$ as desired. Thus we may assume that $k/2 - l < 0$. In this case, we may further assume that $h = \binom{k+l}{2}$, as this minimizes the difference. But in this case,

$$cost(X) = cm + \binom{k+l}{2}\left(l - \frac{k}{2}\right) + \binom{k+l}{2}\left(\frac{k}{2} - l\right) = cm > r,$$

which concludes this case.

Case 3: $|X| = k - l$, with $l > 0$. If $k = l$, then $X = \emptyset$ and $cost(X) = cm > r$. So we assume $k > l$. Put $|E(X)| = \binom{k-l}{2} - h$, where $h \geq 0$. We have

$$\begin{aligned} cost(X) &= c\left(m - \binom{k-l}{2} + h\right) + (k-l)\left(\binom{k-l}{2} - h\right) \\ &= cm + \binom{k-l}{2}(k - l - c) + h(c - k + l) \\ &= cm + \binom{k-l}{2}\left(-\frac{k}{2} - l\right) + h\left(\frac{k}{2} + l\right). \end{aligned}$$

The difference with this cost and r is

$$\begin{aligned} cost(X) - r &= \binom{k-l}{2}\left(-\frac{k}{2} - l\right) + h\left(\frac{k}{2} + l\right) - \left(-\frac{k}{2}\right)\binom{k}{2} \\ &= \frac{3kl^2}{4} + \frac{kl}{4} - \frac{l^3}{2} - \frac{l^2}{2} + h\left(\frac{k}{2} + l\right) \\ &> \frac{1}{4}(3l^3 + l^2) - \frac{1}{2}(l^3 + l^2) \geq 0, \end{aligned}$$

the latter since $k > l \geq 1$. Again, it follows that $cost(X) > r$. ∎

Reduction to Exemplar-TD (promise version). Since the reduction is somewhat technical, we provide an overview of the techniques that we will use. Let (G, c, r) be a Cost-Effective Subgraph instance where c is the cost and r the optimization value, and with vertices $V(G) = \{v_1, \ldots, v_n\}$. Here we present a simplified idea of the reduction, and leave for later the technical modifications required to make this idea work. We will construct strings S and T and argue on the number of contractions to go from T to S. We would like our source string to be $S = x_1 x_2 \ldots x_n$, where each x_i is a distinct character that corresponds to vertex v_i. Then, the T string will handle the relationship between vertices and edges. First, T begins with a substring S' obtained by doubling every x_i, i.e., $S' = x_1 x_1 x_2 x_2 \ldots x_n x_n$. Second, each edge $e_i \in E(G)$ will have a corresponding substring E_i in T. We call E_i the *substring gadget of* e_i, and its exact content will be detailed later. Our goal is to put $T = S' E_1 E_2 \ldots E_m$. It follows that, to contract T into S, we must get rid of both the doubled x_i's and get rid of the E_i's. We make it so that we first want to contract some, but not necessarily all, of the doubled x_i's of S', resulting in another string S''. The choice of x_i's to contract in this

first step corresponds to vertices in a Cost-Effective Subgraph. Let t be the number of x_i's contracted from S' to S''. For instance, we could have $S'' = x_1x_1x_2x_3x_3x_4x_5x_5$, where only x_2 and x_4 were contracted, and thus $t = 2$. After T is transformed to $S''E_1 \dots E_m$, we then force each E_i to use S'' to contract it. For $m = 3$, a contraction sequence that we would like to enforce would take the form

$$\underline{S'}E_1E_2E_3 \rightarrowtail \underline{S''E_1}E_2E_3 \rightarrowtail \underline{S''E_2}E_3 \rightarrowtail \underline{S''E_3} \rightarrowtail \underline{S''} \rightarrowtail S,$$

where we underline the substring affected by contractions at each step. We make it so that when contracting $S''E_iE_{i+1} \dots E_m$ into $S''E_{i+1} \dots E_m$, we have two options. Suppose that v_j, v_k are the endpoints of edge e_i. If, in S'', we had chosen to contract x_j and x_k, we can contract E_i using a sequence of t moves. Otherwise, we must contract E_i using another more costly sequence of c moves. The total cost to eliminate the E_i gadgets will be $c(m - e) + te$, where e is the number of edges that can be contracted using the first choice, i.e. for which both endpoints were chosen in S''.

Unfortunately, constructing S' and the E_i's to implement the above idea is not straightforward. The main difficulty lies in forcing an optimal solution to behave as we describe, i.e. enforcing going from S' to S'' first, enforcing the E_i's to use S'', and enforcing the two options to contract E_i with the desired costs. In particular, we must replace the x_i's by carefully constructed substrings X_i. We must also insert in T many consecutive copies of the $E_1 \dots E_m$ substring, since this will multiply the impact of choosing a suboptimal set of contractions of S' in the first step. This in turn allows us to argue that, in an optimal sequence of contractions, "enough" $E_1 \dots E_m$ copies behave as we want in the above form, which is sufficient for our purposes.

Another important remark on the construction of the E_i gadget is that they should not be able to contract each other cheaply. That is, we want only the S'' from the above form to be able to remove them, unless we are willing to pay a high cost to avoid it. In order to achieve this, each E_i will have two parts $B_i \dots \Delta B_{2p} \dots \Delta$, each ending with a separation symbol Δ. These are designed so that each B_i is different, and the first part of E_{i-1} cannot affect the first part of E_i (because they start with distinct B_{i-1} and B_i, respectively). However, the second part of E_{i-1} will be able to remove all of E_i, but at a high cost. We can do this by making each E_i start with $B_iB_{i-1} \dots B_2B_1^*B_0$, and making the second part start with $B_{2p} \dots B_2B_1B_0$, where a high number of contractions are needed to turn B_1^* into B_1.

We now proceed with the technical details. To faciliate the readers, we will also present a detailed example after the proof of the following theorem.

THEOREM 3.2. *The Exemplar-TD problem is NP-complete, even if for the given string S and T, $S \Rightarrow_* T$ is guaranteed to hold.*

Proof. To see that the problem is in NP, note that $dist_{TD}(S, T) \leq |T|$ since each contraction from T to S removes at least character. Thus a sequence of contractions can serve as a certificate, has polynomial size and is easy to verify.

For hardness, we reduce from the Cost-Effective Subgraph problem, which has been shown NP-hard in Theorem 3.1. Let (G, c, r) be an instance of Cost-Effective Subgraph, letting $n := |V(G)|$ and $m := |E(G)|$. Here c is the "outsider edge" cost and we ask whether there is a subset $X \subseteq V(G)$ such that $c(m - |E(X)|) + |X||E(X)| \leq r$. We denote $V(G) = \{v_1, \dots, v_n\}$ and $E(G) = \{e_1, \dots, e_m\}$. The ordering of vertices and edges is arbitrary but remains fixed for the remainder of the proof. For convenience, we allow the edge indices to loop through 1 to m, and so we put $e_i = e_{i+lm}$ for any integer $l \geq 0$. Thus we may sometimes refer to an edge e_h with an index $h > m$, meaning that e_h is actually the edge $e_{((h-1) \mod m)+1}$.

The construction. Now we show how to construct S and T. The reader should bear in mind that all strings described in the construction can be generated in time polynomial with respect to n and m, and we will not reiterate this point. First let $d = m + 1$ and $p = m(n + m)^{10}$. The exact values of d and p are not crucial and will only refer to them when needed: for the most part, it is enough to think of d and p as simply "large enough", but with p much larger than d. It is however important to note that p is a multiple of m. Roughly speaking, d controls the cost for the number of contractions chosen in the first step discussed above, and p/m is the number of $E_1 \dots E_m$ copies in T. For later reference, the value of k that we will use in the reduction to the TD instance is $k = p/m \cdot d(r + nm) + 4cdn$.

Instead of having a single character x_i corresponding to each vertex v_i of G, we have a substring X_i of length d corresponding to v_i. It is then the characters of each X_i that we double. More precisely, for each $i \in [n]$, define an exemplar string $X_i = x_i^1 x_i^2 \dots x_i^d$ of length d. Moreover, create enough characters so that no two X_i strings contain a character in common. Let $X_i^d = x_i^1 x_i^1 x_i^2 x_i^2 \dots x_i^d x_i^d$ be obtained from X_i by doubling every character, and notice that $dist_{TD}(X_i, X_i^d) = d$.

Also define the strings

$$\mathcal{X} = X_1 X_2 \dots X_n, \qquad \mathcal{X}^d = X_1^d X_2^d \dots X_n^d,$$

Then for each $j \in \{0, 1, \dots, 2p\}$, define an exemplar string B_j. Ensure that no B_j contains a character from an X_i string, and no two B_j's contain a common character. The B_j strings can consist of a single character, with the exception of B_0 and B_1 which are special. We assume that for B_0 and B_1, we have strings B_0^* and B_1^* such that

$$dist_{TD}(B_0, B_0^*) = dc + 2d - 2,$$
$$dist_{TD}(B_1, B_1^*) = dn + 2d - 1.$$

The B_0 and B_0^* strings can be constructed in the same manner as the X_i and X_i^d strings, i.e. using a doubling trick. The same applies to B_1 and B_1^*. The B_j's are the building blocks of larger strings. For each $q \in [2p]$, define

$$\mathcal{B}_q = B_q B_{q-1} \dots B_2 B_1 B_0, \qquad \mathcal{B}_q^0 = B_q B_{q-1} \dots B_2 B_1 B_0^*,$$
$$\mathcal{B}_q^1 = B_q B_{q-1} \dots B_2 B_1^* B_0, \qquad \mathcal{B}_q^{01} = B_q B_{q-1} \dots B_2 B_1^* B_0^*.$$

These strings are used as "blockers" and prevent certain contractions from happening. Note that $\mathcal{B}_q^0$ and $\mathcal{B}_q^1$ can be turned into $\mathcal{B}_q$ using $dc+2d-2$ contractions and $dn+2d-1$ contractions, respectively. Moreover, $\mathcal{B}_q^{01}$ can be turned into $\mathcal{B}_q^0$ using $dn + 2d - 1$ contractions and into $\mathcal{B}_q^1$ using $dc + 2d - 2$ contractions.

Finally, define a new additional character Δ, which will be used to separate some of the components of our string. We can now define S by putting

$$S = \mathcal{B}_{2p} \mathcal{X} \Delta = B_{2p} B_{2p-1} \dots B_2 B_1 B_0 X_1 X_2 \dots X_n \Delta.$$

It follows from the definitions of $\mathcal{B}_{2p}, \mathcal{X}$ and Δ that S is exemplar.

Next, we define T. But first the E_i gadgets must be defined. For each edge $e_q = v_i v_j$ with $q \in [p]$ whose endpoints are v_i and v_j, define

$$\mathcal{X}_{e_q} = X_1^d \dots X_{i-1}^d X_i X_{i+1}^d \dots X_{j-1}^d X_j X_{j+1}^d \dots X_n^d.$$

Thus in $\mathcal{X}_{e_q}$, all X_h substrings are turned into X_h^d, except X_i and X_j. Now for $i \in [p]$, define

$$E_i := \mathcal{B}_i^{01}\mathcal{X}_{e_i}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta,$$

which we will call the *edge gadget*. Define T as

$$\begin{aligned} T &= \mathcal{B}_{2p}^0\mathcal{X}^d\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta E_1E_2\dots E_p \\ &= \mathcal{B}_{2p}^0\mathcal{X}^d\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\left[\mathcal{B}_1^{01}\mathcal{X}_{e_1}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right]\left[\mathcal{B}_2^{01}\mathcal{X}_{e_2}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right]\dots\left[\mathcal{B}_p^{01}\mathcal{X}_{e_p}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right]. \end{aligned}$$

We add brackets for clarity only — they indicate the distinct E_i substrings, but the brackets are not actual characters of T. The idea is that T starts with $S' = \mathcal{B}_{2p}^0\mathcal{X}^d\Delta$, a modified S in which $\mathcal{B}_{2p}$ becomes $\mathcal{B}_{2p}^0$ and the X_i substrings are turned into X_i^d. This $\mathcal{X}^d$ substring serves as a choice of vertices in our cost-effective subgraph. Each edge e_i has a "gadget substring" $E_i = \mathcal{B}_i^{01}\mathcal{X}_{e_i}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta$. Since p is a multiple of m, the sequence of edge gadgets $E_1E_2\dots E_m$ is repeated p/m times. Our goal to go from T to S is to get rid of all these edge gadgets by contractions. Note for later reference that each pair E_i and E_{i+1} start with different character, namely those of B_i and of B_{i+1}, respectively.

The hardness proof. We now show that G admits a subset of vertices W of cost at most r if and only if T can be contracted to S using at most $p/m \cdot d(r+nm) + 4cdn$ contraction operations. It can be deduced from the ($\Rightarrow$) direction that $T \rightarrowtail_* S$ holds.

($\Rightarrow$) Suppose that G admits a subset of vertices W of cost at most r. Thus $c(m - |E(W)|) + |W| \cdot |E(W)| \leq r$. To go from T to S, first consider an edge e_i that does not have both endpoints in W, with $i \in [p]$. We show how to get rid of the gadget substring E_i for e_i using $dn + dc$ contractions. Note that T contains the substring $\mathcal{B}_{2p}^1\mathcal{X}\Delta E_i = \mathcal{B}_{2p}^1\mathcal{X}\Delta[\mathcal{B}_i^{01}\mathcal{X}_{e_i}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta]$, where brackets surround the E_i occurrence that we want to remove (note that here for $i > 1$, the prefix $\mathcal{B}_{2p}^1\mathcal{X}\Delta$ is the suffix of the previous E_{i-1} gadget, and for $i = 1$, it is the suffix of the starting block of T). We can first contract the $\mathcal{B}_i^{01}$ of E_i to $\mathcal{B}_i^1$ using $dc + 2d - 2$ contractions, then contract $\mathcal{X}_{e_i}$ to $\mathcal{X}$ using $d(n-2)$ contractions. The result is the $\mathcal{B}_{2p}^1\mathcal{X}\Delta[\mathcal{B}_i^1\mathcal{X}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta]$ substring, which becomes $\mathcal{B}_{2p}^1\mathcal{X}\Delta$ using two contractions (see below). This sums to $dc + 2d - 2 + d(n-2) + 2 = dc + dn$ operations. More visually, the sequence of contractions works as follows (as before, brackets indicate the E_i substring and what remains of it, and the underlines are there to emphasize the substrings that participate in the contractions(s)):

$$\begin{aligned} &\mathcal{B}_{2p}^1\mathcal{X}\Delta\left[\underline{\mathcal{B}_i^{01}}\mathcal{X}_{e_i}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right] && (dc + 2d - 2 \text{ contractions}) \\ \rightarrowtail &\mathcal{B}_{2p}^1\mathcal{X}\Delta\left[\mathcal{B}_i^1\underline{\mathcal{X}_{e_i}}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right] && (d(n-2) = dn - 2d \text{ contractions}) \\ \rightarrowtail &\mathcal{B}_{2p}^1\mathcal{X}\Delta\left[\mathcal{B}_i^1\mathcal{X}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right] \\ = &B_{2p}B_{2p-1}\dots B_{i+1}\underline{\mathcal{B}_i^1\mathcal{X}\Delta}\left[\underline{\mathcal{B}_i^1\mathcal{X}\Delta}\mathcal{B}_{2p}^1\mathcal{X}\Delta\right] && (1 \text{ contraction}) \\ \rightarrowtail &B_{2p}B_{2p-1}\dots B_{i+1}\mathcal{B}_i^1\mathcal{X}\Delta\left[\mathcal{B}_{2p}^1\mathcal{X}\Delta\right] \\ = &\underline{\mathcal{B}_{2p}^1\mathcal{X}\Delta}\left[\underline{\mathcal{B}_{2p}^1\mathcal{X}\Delta}\right] && (1 \text{ contraction}) \\ \rightarrowtail &\mathcal{B}_{2p}^1\mathcal{X}\Delta. \end{aligned}$$

This sequence of $dn + dc$ contractions effectively removes the E_i substring gadget. Observe that after applying this sequence, it is still true that every remaining E_j

gadget substring is preceded by $\mathcal{B}_{2p}^1\mathcal{X}\Delta$. We may therefore repeatedly apply this contraction sequence to every e_i not contained in W (including those E_i gadgets for which $i > m$). This procedure is thus applied to $p/m \cdot (m - |E(W)|)$ gadgets. We assume that we have done so, and that every e_i for which the E_i gadget substring remains is in W. Call the resulting string T'. Thus

$$T' = \mathcal{B}_{2p}^0\mathcal{X}^d\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta E_{a_1}\dots E_{a_l}$$

for some indices $a_1, \dots, a_l$ corresponding to edges inside W.

Now, let $\mathcal{X}_W$ be the substring obtained from $\mathcal{X}^d$ by contracting, for each $v_i \in W$, the string X_i^d to X_i. We assume that we have contracted the $\mathcal{X}^d$ substring of T' to $\mathcal{X}_W$, which uses $d|W|$ contractions (note that there is only one occurrence of $\mathcal{X}^d$ in T', namely right before the first Δ). Call

$$T' = \mathcal{B}_{2p}^0\mathcal{X}_W\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta E_{a_1}\dots E_{a_l}$$

the resulting string. At this point, for every E_i substring gadget that remains, where E_i corresponds to edge $e_i = v_jv_k$, $\mathcal{X}_W$ contains the substrings X_j and X_k (instead of X_j^d and X_k^d).

Let i be the smallest integer for which the e_i substring gadget E_i is still in T'' (i.e., $i = a_1$ using the notation for T'' just above). This is the leftmost edge gadget still in T'', meaning that T'' has the prefix

$$\mathcal{B}_{2p}^0\mathcal{X}_W\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\left[\mathcal{B}_i^{01}\mathcal{X}_{e_i}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right]\dots,$$

where brackets indicate the E_i substring. To remove E_i, first contract $\mathcal{B}_i^{01}$ to $\mathcal{B}_i^0$, and contract $\mathcal{X}_{e_i}$ to X_W (this is possible since $e_i \subseteq W$). The result is

$$\mathcal{B}_{2p}^0\mathcal{X}_W\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\left[\mathcal{B}_i^0\mathcal{X}_W\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right]\dots$$

One more contraction gets rid of the substring in brackets. This process requires $dn + 2d - 1 + d(|W| - 2) + 1 = dn + d|W|$ contractions. To recap, the contraction sequence for E_i does as follows:

$$\begin{aligned}
&\mathcal{B}_{2p}^0\mathcal{X}_W\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\left[\underline{\mathcal{B}_i^{01}}\mathcal{X}_{e_i}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right] && (dn + 2d - 1 \text{ contractions})\\
\rightarrowtail&\mathcal{B}_{2p}^0\mathcal{X}_W\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\left[\mathcal{B}_i^0\underline{\mathcal{X}_{e_i}}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right] && (d(|W| - 2) \text{ contractions})\\
\rightarrowtail&\underline{\mathcal{B}_{2p}^0\mathcal{X}_W\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta}\left[\underline{\mathcal{B}_i^0\mathcal{X}_W\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta}\right] && (1 \text{ contraction})\\
\rightarrowtail&\mathcal{B}_{2p}^0\mathcal{X}_W\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta.
\end{aligned}$$

After we repeat this for every E_i, all that remains is the string $\mathcal{B}_{2p}^0\mathcal{X}_W\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta$. We contract $\mathcal{X}_W$ to $\mathcal{X}$ using $d(n-|W|)$ contractions (in total, going from $\mathcal{X}^d$ to $\mathcal{X}$ required dn contractions). Then contract $\mathcal{B}_{2p}^0$ and $\mathcal{B}_{2p}^1$ to $\mathcal{B}_{2p}$ using $dc+2d-2+dn+2d-1 = d(c+n+4)-3$ contractions. At this point, we have reached $\mathcal{B}_{2p}\mathcal{X}\Delta\mathcal{B}_{2p}\mathcal{X}\Delta$. One more contraction of the second half of the string yields S. The summary of the number of

contractions made is

$$\begin{aligned}
&\frac{p}{m} \cdot (m - |E(W)|) \cdot (dc + dn) + \frac{p}{m} \cdot |E(W)| \cdot (dn + d|W|) + dn + d(c + n + 4) - 3 \\
\leq &\frac{p}{m} \cdot (m - |E(W)|) \cdot (dc + dn) + \frac{p}{m} \cdot |E(W)| \cdot (dn + d|W|) + 4cdn \\
= &\frac{p}{m} \cdot d \cdot (c + n)(m - |E(W)|) + \frac{p}{m} \cdot d \cdot (n + |W|)|E(W)| + 4cdn \\
= &\frac{p}{m} \cdot d \cdot [c(m - |E(W)|) + |W||E(W)| + nm] + 4cdn \\
\leq &\frac{p}{m} \cdot d(r + nm) + 4cdn,
\end{aligned}$$

as desired.

($\Leftarrow$) Suppose that T can be turned into S using α contractions, where $\alpha \leq p/m \cdot d(r + nm) + 4cdn$. Let $C_1, \ldots, C_\alpha$ be a corresponding sequence of contractions. Here, each C_i contraction is given by a pair of positions ranging over both copies of the contracted substring. The idea is to show that, for some integer t, many of the E_i substrings are removed, but only after t of the X_i^d substrings from $\mathcal{X}^d$ have been contracted to X_i. This set of t X_i's corresponds to the vertices of a cost-effective subgraph. The main components of the proof are to show that 1) each E_i must be removed; 2) no two E_i's are completely removed by the same contraction; and 3) most (though perhaps not all) E_i require either $dn + dt$ or $dc + dn - 1$ contractions.

Denote $T(l)$ as the string obtained from T after applying the first l contractions $C_1, \ldots, C_l$ in the sequence, with $T(0) = T$ and $T(\alpha) = S$. A *block* of $T(l)$ is a substring P of $T(l)$ whose last character is Δ, that has only one occurrence of Δ and that is a maximal string with this property (hence in $T(l)$, the first character of P is either preceded by Δ or is the start of $T(l)$). For instance, each E_i substring of T is made of 2 blocks.

We need to distinguish between individual occurrences of each character in our strings. A *marker* is a specific occurrence of a character. That is, for string T, each character $T[i]$ is a distinct marker, even if other positions contain the same character. One may think of a marker as having a unique identifier in a string. For concreteness, let us assume that in T, each marker has, as its unique identifier, its position in T, so that each character is distinguishable. When contracting a substring DD of $T(l)$ to the substring D of $T(l + 1)$, we assume that the markers of the second half are deleted. That is, if $T(l) = LDDR$ and $T(l + 1) = LDR$, only the markers from the first, leftmost D substring remain. This is without loss of generality, since either choice would lead to the S string. It follows that when going from $T(l)$ to $T(l + 1)$, some markers might change position but they remain "the same" marker (i.e. their identifier remains the same). This lets us assert that each marker of $T(l)$ is a marker that was originally in T.

For a substring P of T, we say that P is *removed in* $T(l)$ if none of the markers of P are in $T(l)$. We say that P is removed if there exists some $T(l)$, with $1 \leq l \leq \alpha$, in which P is removed.

Claim 1. *Each E_i substring must be removed in $T(\alpha)$.*

Proof. Consider the first, leftmost block $\mathcal{B}_{2p}^0 \mathcal{X}^d \Delta$ of T. Notice that the Δ marker of this block is the first occurrence of Δ in T. Also note that for any $T(l)$, the Δ marker from this block must also be present in $T(l)$, since there is no way to remove it, by our way of always deleting the rightmost copy in contractions. Now, if we assume that some E_i is not removed from $T(\alpha)$, some marker x of E_i occurs in $T(\alpha) = S$.

Since x occurs to the right of the first Δ in T, x also occurs to the right of the first Δ in S, because contractions cannot change the relative ordering of two markers. This is a contradiction, since Δ is the last character of S. ∎

For what follows, notice that in T, $E_i = \mathcal{B}_i^{01}\mathcal{X}_{e_i}\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta$ has two blocks. We write $E_i' = \mathcal{B}_i^{01}\mathcal{X}_{e_i}\Delta$ to denote the first block of E_i. For the rest of the proof, we will mainly focus on how the sequence of contractions removes the first block E_i'. This gives us sufficient bounds on the number of contractions required to remove each E_i, which simplifies the analysis. We let $E_i'(l)$ be the substring of $T(l)$ formed by all the markers that belong to E_i', noting that $E_i'(l)$ is possibly the empty string or a subsequence of E_i'. For $a \in \{0, 1, 01\}$, a block $BX\Delta$ is called a $\mathcal{B}_i^a\mathcal{X}\Delta$-block if $BX\Delta$ is a subsequence of $\mathcal{B}_i^a\mathcal{X}^d\Delta$ and $\Sigma(BX\Delta) = \Sigma(\mathcal{B}_i^a\mathcal{X}^d\Delta)$. In other words, $BX\Delta$ has all the characters that occur in $\mathcal{B}_i^a\mathcal{X}^d\Delta$ in the same order, although the number of occurrences of a character might differ. A string obtained by concatenating an arbitrary number of $\mathcal{B}_{2p}^1\mathcal{X}\Delta$-blocks is called a $\mathcal{B}_{2p}^1\mathcal{X}\Delta$-*cluster*, which could possibly be the empty string. Using notation borrowed from regular languages, we write $(\mathcal{B}_{2p}^1\mathcal{X}\Delta)^*$ to denote a $\mathcal{B}_{2p}^1\mathcal{X}\Delta$-cluster.

The next claim shows that a specific block structure is preserved during the contraction process. An important feature of this structure is that each E_i' starts as a $\mathcal{B}_j^{01}\mathcal{X}\Delta$-block in T, and remains so until it is completely removed. This means that until E_i' is removed, all the characters that appear in E_i' remain present (though perhaps in different numbers of copies).

Claim 2. *For any l, $T(l)$ has the form*

$$BX\Delta(\mathcal{B}_{2p}^1\mathcal{X}\Delta)^*E_{i_1}'(l)(\mathcal{B}_{2p}^1\mathcal{X}\Delta)^*E_{i_2}'(l)(\mathcal{B}_{2p}^1\mathcal{X}\Delta)^*\dots E_{i_h}'(l)(\mathcal{B}_{2p}^1\mathcal{X}\Delta)^*,$$

where

- *$BX\Delta$ is a $\mathcal{B}_{2p}^0\mathcal{X}\Delta$-block,*
- *$1 \le i_1 < i_2 < \dots < i_h \le p$,*
- *each $(\mathcal{B}_{2p}^1\mathcal{X}\Delta)^*$ is a $\mathcal{B}_{2p}^1\mathcal{X}\Delta$-cluster, and*
- *for each $j \in \{i_1, \dots, i_h\}$, $E_j'(l)$ is a $\mathcal{B}_j^{01}\mathcal{X}\Delta$-block.*

Proof. Notice that the statement is true for $l = 0$, since T has the required form. Assume the claim is false and let l be the smallest integer for which $T(l)$ is a counter-example to the claim. Thus we may assume that $T(l-1)$ has the same form as in the claim statement. Let D be the string that was contracted from $T(l-1)$ to $T(l)$, so that $T(l-1)$ contained DD as a substring, and the second D substring gets removed from $T(l-1)$. We shall refer to the first D as the left D and the second as the right D. If D does not contain a Δ character, then DD is entirely contained in a single block. Contracting DD cannot remove all occurrences of a character nor change their order, and thus the above form must be preserved (every $\mathcal{B}_i^a\mathcal{X}\Delta$-block will remain a $\mathcal{B}_i^a\mathcal{X}\Delta$-block). Assume instead that the last character of D is Δ. Then $DD = D'\Delta D'\Delta$ for some string D', and removing the second $D'\Delta$ half only removes entire blocks of $T(l-1)$. As this block cannot be $BX\Delta$ and since each $E_{i_j}'(l-1)$ is itself a block, this preserves the form of the claim.

Therefore, we may assume that the last character of D is not Δ, but that D has at least one Δ character. Observe that no character from the $BX\Delta$-block can get removed by such a contraction, since the left half of DD is kept. It follows that the first condition of the claim is preserved after contracting DD. It is easy to see that the second condition is also preserved. For the other two conditions, we have four

cases to consider depending on where the right half of DD, i.e. the removed substring, is located in $T(l-1)$.

1. The leftmost character removed belongs to a $E'_j(l-1)$ substring of $T(l-1)$. In this case, because D contains a Δ, the right half of DD must contain the Δ of $E'_j(l-1)$. Let b be the first marker of $E'_j(l-1)$. If no confusion is possible, we treat b as both a character and a marker. Note that b is the first character of $\mathcal{B}_j^{01}$ since $E'_j(l-1)$ is a $\mathcal{B}_j^{01}\mathcal{X}\Delta$-block, by assumption. Also note that the b marker is preceded by Δ in $T(l-1)$. There are two subcases: either this b is the leftmost removed character or not. In the first case, $D = bD'$ for some D', which we illustrate as follows (we add brackets around the two copies of D, and underline the removed half):

$$T(l-1) = T'[bD'][\underline{bD'}]T'',$$

for some strings T' and T''. Here the second b is the one from $E'_j(l-1)$. Since it is preceded by Δ in $T(l-1)$, this implies that D' (and thus D) ends with a Δ. But we are assuming that D does not end with Δ. Therefore we know that b is not the leftmost character removed from $T(l-1)$.
It follows that the b marker belongs to the left half of DD. This case can be illustrated as follows:

$$T(l-1) = T'[D'bD''][\underline{D'bD''}]T'',$$

where $D = D'bD''$ for some strings D' and D''. Here, the first b is the first character from $E'_j(l-1)$. We can argue that D' is not empty: if D' is empty, then D'' contains Δ because we are assuming that D contains Δ. But the first b is the start of $E'_j(l-1)$, and thus the first Δ encountered after this b is the end of the $E'_j(l-1)$ block. In other words, the left $D'bD''$ entirely contains $E'_j(l-1)$, contradicting that the leftmost deleted marker is inside $E'_j(l-1)$.
Since D' is not empty, it follows that D must contain Δb as a substring. But there is only one occurrence of Δb in $T(l-1)$, as $E'_j(l-1)$ is the only block that starts with b ($BX\Delta$ and all the blocks in $\mathcal{B}_{2p}^1\mathcal{X}\Delta$-clusters start with B_{2p}, and each E'_{i_h} block starts with B_{i_h}). Therefore, $T(l-1)$ cannot contain DD as a substring since it would contain Δb twice, a contradiction.
2. The rightmost character removed is in some $E'_j(l-1)$ substring. That is, the last marker of DD is in $E'_j(l-1)$. Since the right D contains Δ, it must contain the Δ that precedes b, the first marker of $E'_j(l-1)$. This means that the D contains Δb as a substring, which has only one occurrence. We get the same contradiction.
3. The leftmost and rightmost characters that get removed belong to distinct $(\mathcal{B}_{2p}^1\mathcal{X}\Delta)$-clusters of $T(l-1)$. This implies the existence of at least one $E'_j(l-1)$ substring in between. The same type of Δb substring argument applies, since the removed D contains the first character of $E'_j(l-1)$ and its preceding Δ.
4. The leftmost and rightmost characters that get removed belong to the same $(\mathcal{B}_{2p}^1\mathcal{X}\Delta)$-cluster. In this case, it is not hard to verify that the result is yet another $(\mathcal{B}_{2p}^1\mathcal{X}\Delta)$-cluster, which preserves the desired form.

The cases above cover every possibility: we have covered the cases where the removed substring begins or ends in a $E'_j(l-1)$, and the two cases where both its extremities end in a cluster. This proves the claim. ∎

We will say that a contraction C_l *affects* $E'_i(l)$ if at least one marker of $E'_i(l)$ is in the substring corresponding to C_l. Recall that C_l spans over both copies of the contracted substring, and so $E'_i(l)$ could be affected by C_l even if none of its characters gets removed.

CLAIM 3. *For any $l \in [\alpha]$, the contraction C_l from $T(l)$ to $T(l+1)$ does not affect two distinct $E'_i(l)$ and $E'_j(l)$ substrings of $T(l)$.*

Proof. Suppose the claim is false, and let $T(l)[a_1..a_2]$ be the substring of $T(l)$ affected by the contraction, where $T(l)[a_1..a_2] = DD$ for some string D. Assume that $T(l)[a_1..a_2]$ contains characters from both $E'_i(l)$ and $E'_j(l)$, where $i < j$. Let b_i, b_j be the first characters of $E'_i(l)$ and $E'_j(l)$, respectively, which are the first characters of $\mathcal{B}_i^{01}$ and $\mathcal{B}_j^{01}$, respectively, by Claim 2. Then $T(l)[a_1..a_2]$ must contain the substring Δb_j, since $E'_j(l)$ occurs later than $E'_i(l)$ in $T(l)$. Since Δb_j occurs only once in $T(l)$ as argued in the previous claim, Δb_j cannot be a substring of D. This is only possible if D starts with b_j (and consequently ends with Δ). Now, since $E'_i(l)$ does not contain b_j, $T(l)[a_1..a_2]$ cannot start with a suffix of $E'_i(l)$. Yet some characters of $E'_i(l)$ are in $T(l)[a_1..a_2]$, implying that $E'_i(l)$ is entirely in $T(l)[a_1..a_2]$, and furthermore that the substring Δb_i is in $T(l)[a_1..a_2]$. Again, this substring occurs only once in $T(l)$, and thus D must start with b_i. But this is impossible since now, D start with b_i and by b_j, and $b_i \neq b_j$. ∎

Notice that T has one occurrence of the $\mathcal{X}^d = X_1^d \dots X_n^d$ substring. We will therefore refer to the $\mathcal{X}^d$ substring of T without ambiguity. For $i \in [n]$, we let $X_i(l)$ denote the substring of $T(l)$ formed by all the markers that belong to the X_i^d substring of $\mathcal{X}^d$. We will say that X_i is *activated* in $T(l)$ if $X_i(l) = X_i$. Intuitively speaking, X_i is activated in $T(l)$ if it has undergone d contractions to turn it from X_i^d into X_i.

CLAIM 4. *Let $i \in [p]$, and suppose that E'_i is not removed in $T(l-1)$ but is removed in $T(l)$. Let t be the number of X_i's that were activated in $T(l-1)$. Suppose that v_{i_1} and v_{i_2} are the two endpoints of edge e_i.*

Then the number of contractions that have affected E'_i is at least $dc + dn - 1$ if X_{i_1} or X_{i_2} is not activated in $T(l-1)$, or at least $\min\{dt + dn, dc + dn - 1\}$ if X_{i_1} and X_{i_2} are both activated in $T(l-1)$.

Proof. By Claim 2, in $T(l-1)$, $E'_i(l-1)$ belongs to a $\mathcal{B}_i^{01}\mathcal{X}\Delta$-block. As $E'_i(l-1)$ gets removed completely after the l-th contraction of some substring DD, it follows that D must contain a substring that is equal to $E'_i(l-1)$. The right D of the DD string certainly contains the $E'_i(l-1)$ substring that gets removed, but consider the copy of $E'_i(l-1)$ in the first D of the DD square. That is, we can represent the contraction as

$$T'[D_1\hat{E}'_i(l-1)D_2][\underline{D_1E'_i(l-1)D_2}]T'',$$

where $D = D_1E'_i(l-1)D_2$ and $\hat{E}'_i(l-1)$ is a substring equal to $E'_i(l-1)$. Since $E'_i(l-1)$ is itself a block, this $\hat{E}'_i(l-1)$ copy is a substring of another block (though $\hat{E}'_i(l-1)$ might not be a block by itself). By Claim 3, the contraction does not affect another $E'_j(l-1)$, and so there are only two possible blocks that contain $\hat{E}'_i(l-1)$: either it is $BX\Delta$, which is the $\mathcal{B}_{2p}^0\mathcal{X}\Delta$-block at the start of $T(l-1)$, or it is a $\mathcal{B}_{2p}^1\mathcal{X}\Delta$-block from a cluster preceding $E'_i(l-1)$. We analyze these two cases.

Suppose that $\hat{E}'_i(l-1)$ is located in the first block $BX\Delta$ of $T(l-1)$. Note that since $\mathcal{X}_{e_i}$ contains X_{i_1} and X_{i_2} (as opposed to $X_{i_1}^d$ or $X_{i_2}^d$), X_{i_1} and X_{i_2} must be activated in $T(l-1)$ for the DD contraction to be possible. Moreover for $E'_i(l-1)$ to

be equal to a substring of $BX\Delta$, every other X_j with $j \neq i_1, i_2$ that is activated must be contracted in $E'_i(l-1)$ (i.e. E'_i contains X_j^d, but must contain X_j in $E'_i(l-1)$). This requires at least $d(t-2)$ contractions. Moreover, B contains the B_1 substring, whereas E'_i contains B_1^*. There must have been at least $dn + 2d - 1$ contractions affecting the $\mathcal{B}_i^{01}$ substring of E'_i. We note that the character doubling strategy forces all contractions enumerated so far to be distinct. Counting the contraction removing $E'_i(l-1)$, this implies the existence of $d(t-2)+dn+2d-1+1 = dn+dt$ contractions affecting E'_i.

If instead $\hat{E}'_i(l-1)$ was located in a $\mathcal{B}_{2p}^1\mathcal{X}\Delta$-block, call this block P, then it suffices to note that P contains B_0 as a substring whereas E'_i contains B_0^* note that B_0 cannot be modified since it contains only distinct characters, forcing us to transform B_0^* into B_0). Counting the contraction that removes $E'_i(l-1)$, it follows that at least $dc+2d-1$ contractions must have affected E'_i. □

The above shows that there are two types of contractions that can remove E'_i from $T(l)$. Either it uses the $BX\Delta$ block at the start of $T(l-1)$, or it uses a block from a preceding $\mathcal{B}_{2p}^1\mathcal{X}\Delta$-cluster. We will call the E'_i's that get removed in the first manner Type 1, and those that get removed in the second manner Type 2.

We would like to show that every Type 1 E'_j gets removed with the same set of activated X_i's, but it might not be the case. Rather, our next goal is to show that "many" E'_j's of Type 1 use the same activated X_i's. This is precisely why we created p/m copies of $E_1 \dots E_m$. For $k \in [p]$, denote by $act(E'_k)$ the set of activated X_i's when E'_k gets removed (i.e. when E'_k is not removed from $T(l-1)$ but is removed from $T(l)$). Recall that the E_i indices range from 1 to p. Let us partition $[p]$ into intervals of integers $P_a = [am+1..am+m]$, where $a \in \{0, \dots, p/m-1\}$. We say that interval P_a is *homogeneous* if, for each $i, j \in P_a$ such that E'_i and E'_j are of Type 1, $act(E'_i) = act(E'_j)$. In other words, P_a is homogeneous if all the Type 1 E'_i substrings corresponding to those in P_a are removed with the same set of activated X_i's.

Claim 5. *There are at least $p/m - 2n$ homogeneous intervals.*

Proof. Observe that once an X_i is activated, it remains so for the rest of the contraction sequence. Since there are n of the X_i's, there are only $n+1$ possible values for $act(E'_k)$ (counting the case when none of them are activated). There are p/m intervals, and it follows that at most $n+1 \leq 2n$ of them are not homogeneous. □

We can now go on with the final elements of the proof. Define $cost(E'_i)$ as the number of contractions that affect E'_i. Let $P_{a_1}, \dots, P_{a_h}$ be the set of homogeneous intervals, $h \geq p/m - 2n$. Choose the P_a interval among those whose sum of corresponding E'_i costs is minimized — in other words choose P_a such that

$$\sum_{i \in P_a} cost(E'_i) = \min_{j \in [h]} \sum_{i \in P_{a_j}} cost(E'_i).$$

By Claim 3, no two E'_i's share their cost, and by the minimality of P_a the total number of contractions is at least

$$\left(\frac{p}{m} - 2n\right) \sum_{i \in P_a} cost(E'_i)$$

We will only bother with these contractions and we make no assumption on the non-homogeneous intervals. Assume that there is at least one $i \in P_a$ such that E'_i is of Type 1. Then by Claim 4, $cost(E'_i)$ is either at least $\min\{dc+dn-1, dt+dn\}$

where $t = |act(E_i')|$, or $cost(E_i')$ is at least $dc + dn - 1$. If $dt + dn \geq dc + dn - 1$, we may assume that E_i' is of Type 2 since removing E_i' using Type 2 contractions will not increase its cost. We will therefore assume that if there is at least one E_i' of Type 1 in P_a, then $dt + dn < dc + dn - 1$ and thus $cost(E_i') \geq dt + dn$.

Now, choose any i in P_a such that E_i' is of Type 1, and let W be the set of vertices of G corresponding to those in $act(E_i')$. That is, $v_j \in W$ if and only if X_j is activated when E_i' gets removed. If there does not exist an E_i' of Type 1 to choose, then define $W = \emptyset$. Denote $t := |W|$ and $s := |E(W)|$. We claim that W is a subgraph of G satisfying $c(m-s) + ts \leq r$.

Assume $c(m-s)+ts > r$ (otherwise, we are done). As we are dealing with integers, this means $c(m-s)+ts \geq r+1$. We will derive a contradiction on the assumed number of contractions. For any E_i' where $i \in P_a$, by Claim 4, either an endpoint of e_i is not in W and $cost(E_i') \geq dc+dn-1$, or e_i is in W and $cost(E_i') \geq dt+dn$. Note that we needed to choose P_a to be homogeneous to guarantee that every Type 1 E_i' uses the same value of t in the cost $dt + dn$. It follows that the total number of contractions is at least

$$
\begin{aligned}
&\left(\frac{p}{m} - 2n\right) \sum_{i \in P_a} cost(E_i') \\
\geq &\left(\frac{p}{m} - 2n\right) [(m-s)(dc+dn-1) + s(dt+dn)] \\
= &\left(\frac{p}{m} - 2n\right) [d((c+n)(m-s) + s(t+n)) - m + s] \\
= &\left(\frac{p}{m} - 2n\right) [d(c(m-s) + st + nm - ns + ns) - m + s] \\
= &\left(\frac{p}{m} - 2n\right) \cdot d \cdot [c(m-s) + st + nm] + \left(\frac{p}{m} - 2n\right)(s-m) \\
\geq &\left(\frac{p}{m} - 2n\right) \cdot d \cdot [r + 1 + nm] + \left(\frac{p}{m} - 2n\right)(s-m) \\
= &\left(\frac{p}{m} - 2n\right) \cdot d \cdot [r + nm] + \left(\frac{p}{m} - 2n\right)(d+s-m) \\
= &\frac{p}{m} \cdot d(r+nm) - 2dn(r+nm) + \left(\frac{p}{m} - 2n\right)(d+s-m)
\end{aligned}
$$

Now if d and p are large enough, the above is strictly greater $p/m \cdot d(r+nm)+4cdn$, leading to a contradiction. Our chosen values $d = m+1$ and $p = (n+m)^{10}$ easily verify this. We have therefore shown that W has the desired cost. This concludes the proof. ∎

An example. In Fig.2, we show a graph G with $W = \{v_4, v_5, v_6\}$. The sequence T containing E_1 and E_2 is as follows. We set $d = 4$ (certainly it can be larger, e.g., 9), $\mathcal{X} = X_1X_2X_3X_4X_5X_6$, where $X_i = x_{i1}x_{i2}x_{i3}x_{i4}$. And we set $X_i^4 = x_{i1}x_{i1}x_{i2}x_{i2}x_{i3}x_{i3}x_{i4}x_{i4}$ and $\mathcal{X}^4 = X_1^4X_2^4X_3^4X_4^4X_5^4X_6^4$. For $e_i = (v_j, v_k)$, $\mathcal{X}_{e_i} = X_1^4 \cdots X_{j-1}^4 X_j X_{j+1}^4 \cdots X_{k-1}^4 X_k X_{k+1}^4 \cdots X_n^4$.

$$
\begin{aligned}
T = B_{2p}B_{2p-1}\cdots B_2B_1B_0^*X_1^4X_2^4X_3^4X_4^4X_5^4X_6^4\Delta \quad & //\mathcal{B}_{2p}^0\mathcal{X}^d\Delta \\
B_{2p}B_{2p-1}\cdots B_2B_1^*B_0X_1X_2X_3X_4X_5X_6\Delta \quad & //\mathcal{B}_{2p}^1\mathcal{X}\Delta \\
B_1^*B_0^*X_1X_2^4X_3X_4^4X_5^4X_6^4\Delta \quad & //\mathcal{B}_1^{01}\mathcal{X}_{e_1}\Delta \\
B_{2p}B_{2p-1}\cdots B_2B_1^*B_0X_1X_2X_3X_4X_5X_6\Delta \quad & //\mathcal{B}_{2p}^1\mathcal{X}\Delta \\
B_2B_1^*B_0^*X_1^4X_2^4X_3^4X_4X_5X_6^4\Delta \quad & //\mathcal{B}_2^{01}\mathcal{X}_{e_2}\Delta \\
B_{2p}B_{2p-1}\cdots B_2B_1^*B_0X_1X_2X_3X_4X_5X_6\Delta \quad & //\mathcal{B}_{2p}^1\mathcal{X}\Delta \\
\cdots\cdots \quad & \textit{omitted}
\end{aligned}
$$

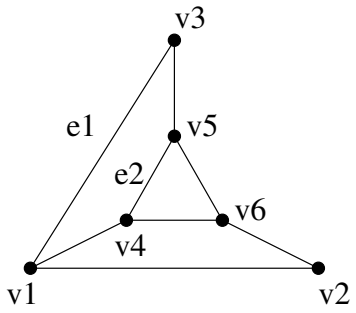

FIG. 2. *A graph G with $W = \{v_4, v_5, v_6\}$. In the constructed gadgets, we only show those for $e_1 = (v_1, v_3)$ and $e_2 = (v_4, v_5)$. For clarity, we assume $d = 4$.*

In the forward direction, as $e_1 \notin E(W)$, E_1 is contracted as follows:

$$
\begin{aligned}
& B_{2p}B_{2p-1}\cdots B_2B_1^*B_0X_1X_2X_3X_4X_5X_6\Delta\left[B_1^*B_0^*X_1X_2^4X_3X_4^4X_5^4X_6^4\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right] \\
\Rightarrow & B_{2p}B_{2p-1}\cdots B_2B_1^*B_0X_1X_2X_3X_4X_5X_6\Delta\left[B_1^*B_0X_1X_2^4X_3X_4^4X_5^4X_6^4\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right] \\
\Rightarrow & B_{2p}B_{2p-1}\cdots B_2B_1^*B_0X_1X_2X_3X_4X_5X_6\Delta\left[B_1^*B_0X_1X_2X_3X_4X_5X_6\Delta\mathcal{B}_{2p}^1\mathcal{X}\Delta\right] \\
\Rightarrow & B_{2p}B_{2p-1}\cdots B_2B_1^*B_0X_1X_2X_3X_4X_5X_6\Delta\left[\mathcal{B}_{2p}^1\mathcal{X}\Delta\right] \\
\Rightarrow & B_{2p}B_{2p-1}\cdots B_2B_1^*B_0X_1X_2X_3X_4X_5X_6\Delta \\
= & \mathcal{B}_{2p}^1\mathcal{X}\Delta.
\end{aligned}
$$

Similarly, as $e_2 \in E(W)$, E_2 is contracted as follows.

$$
\begin{aligned}
&\mathcal{B}^0_{2p}X^4_1X^4_2X^4_3X^4_4X^4_5X^4_6\Delta B_{2p}B_{2p-1}\cdots B_2B_1^*B_0\mathcal{X}\Delta\left[B_2B_1^*B_0^*X^4_1X^4_2X^4_3X_4X_5X^4_6\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
=&\mathcal{B}^0_{2p}\mathcal{X}^4\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\left[\mathcal{B}^{01}_2\mathcal{X}_{e_2}\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
\to&\mathcal{B}^0_{2p}X^4_1X^4_2X^4_3X_4X_5X_6\Delta B_{2p}B_{2p-1}\cdots B_2B_1^*B_0\mathcal{X}\Delta\left[B_2B_1^*B_0^*X^4_1X^4_2X^4_3X_4X_5X^4_6\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
=&\mathcal{B}^0_{2p}\mathcal{X}_W\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\left[\mathcal{B}^{01}_2\mathcal{X}_{e_2}\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
\to&\mathcal{B}^0_{2p}X^4_1X^4_2X^4_3X_4X_5X_6\Delta B_{2p}B_{2p-1}\cdots B_2B_1^*B_0\mathcal{X}\Delta\left[B_2B_1B_0^*X^4_1X^4_2X^4_3X_4X_5X^4_6\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
=&\mathcal{B}^0_{2p}\mathcal{X}_W\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\left[\mathcal{B}^0_2\mathcal{X}_{e_2}\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
\to&\mathcal{B}^0_{2p}X^4_1X^4_2X^4_3X_4X_5X_6\Delta B_{2p}B_{2p-1}\cdots B_2B_1^*B_0\mathcal{X}\Delta\left[B_2B_1B_0^*X^4_1X^4_2X^4_3X_4X_5X_6\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
=&\mathcal{B}^0_{2p}\mathcal{X}_W\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\left[\mathcal{B}^0_2\mathcal{X}_W\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
\to&B_{2p}B_{2p-1}\cdots B_2B_1B_0^*X^4_1X^4_2X^4_3X_4X_5X_6\Delta\cdot B_{2p}B_{2p-1}\cdots B_2B_1^*B_0X_1X_2X_3X_4X_5X_6\Delta\\
=&\mathcal{B}^0_{2p}\mathcal{X}_W\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\\
\to&\cdots \text{ contract other } E_i \text{ in } E(W)\\
\to&B_{2p}B_{2p-1}\cdots B_2B_1B_0^*X_1X_2X_3X_4X_5X_6\Delta\cdot B_{2p}B_{2p-1}\cdots B_2B_1^*B_0X_1X_2X_3X_4X_5X_6\Delta\\
=&\mathcal{B}^0_{2p}\mathcal{X}\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\\
=&S.
\end{aligned}
$$

Note that in the reverse direction, we do not necessarily know the optimal way to contract E_k (e.g., when E_k is not in a homogeneous interval). For instance, the optimal solution could contract E_1 as follows

$$
\begin{aligned}
&\mathcal{B}^0_{2p}X^4_1X^4_2X^4_3X^4_4X^4_5X^4_6\Delta B_{2p}B_{2p-1}\cdots B_2B_1^*B_0\mathcal{X}\Delta\left[B_1^*B_0^*X_1X^4_2X_3X^4_4X^4_5X^4_6\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
=&\mathcal{B}^0_{2p}\mathcal{X}^4\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\left[\mathcal{B}^{01}_1\mathcal{X}_{e_1}\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
\to&\mathcal{B}^0_{2p}X^4_1X^4_2X^4_3X^4_4X^4_5X^4_6\Delta B_{2p}B_{2p-1}\cdots B_2B_1^*B_0\mathcal{X}\Delta\left[B_1B_0^*X_1X^4_2X_3X^4_4X^4_5X^4_6\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
=&\mathcal{B}^0_{2p}\mathcal{X}^4\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\left[\mathcal{B}^0_1\mathcal{X}_{e_1}\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
\to&\mathcal{B}^0_{2p}X_1X^4_2X_3X^4_4X^4_5X^4_6\Delta B_{2p}B_{2p-1}\cdots B_2B_1^*B_0\mathcal{X}\Delta\left[B_1B_0^*X_1X^4_2X_3X^4_4X^4_5X^4_6\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
=&\mathcal{B}^0_{2p}\mathcal{X}_{e_1}\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\left[\mathcal{B}^0_1\mathcal{X}_{e_1}\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta\right]\\
\to&\mathcal{B}^0_{2p}X_1X^4_2X_3X^4_4X^4_5X^4_6\Delta B_{2p}B_{2p-1}\cdots B_2B_1^*B_0X_1X_2X_3X_4X_5X_6\Delta\\
=&\mathcal{B}^0_{2p}\mathcal{X}_{e_1}\Delta\mathcal{B}^1_{2p}\mathcal{X}\Delta.
\end{aligned}
$$

Of course, in this way the $\mathcal{X}^d$ after $\mathcal{B}^0_{2p}$ will lose contents (i.e., some X^d_i will be contracted into X_i). However, the above contractions can only last in at most n homogeneous intervals.

In the next section, we show that the hardness result holds even when the alphabet is of size four (fitting that the DNA and sequences are over the set of nucleotides $\{A, C, G, T\}$ and $\{A, C, G, U\}$ respectively).

4. NP-hardness with constant alphabet size. We reduce the TD problem on arbitrary alphabet to the TD problem on alphabets of size four. Note that this implies that the TD problem is NP-hard on input strings on alphabet of size c for any constant $c \geq 4$.

We show that we can encode each individual character $x \in \Sigma$ with a substring X on alphabet of size four and preserve the hardness. The main difficulty is to show

that no contraction occurs inside the encoded substrings X, which requires a specific form of square-free strings.

LEMMA 4.1. *Let m be an integer. Then there exists an algorithm that takes time polynomial in m that outputs pairs of square-free strings $(A_1, B_1), \ldots, (A_m, B_m)$ over alphabet $\{a, b, c\}$ such that for each $i \in \{1, \ldots, m\}$:*

1. *$100i - 3 \leq |A_i| \leq 100i$ and $|B_i| = 1000m - |A_i|$;*
2. *$A_i[1] = a$*
3. *$B_i[1] = b$ and $B_i[|B_i|] = b$.*

Proof. In [22], Leech describes the following process to generate an infinite ternary square-free string. Define $w_0 = a$, and for $j > 0$, recursively obtain w_j from w_{j-1} by replacing each character with a specified substring of length 13 (the replacement does not matter here, see [22] for details). Each w_j is square-free, and for our purposes we may apply this process and stop when the condition $|w_j| > 2000m$ is met for the first time (note that this takes polynomial time).

Assume thus that we generated a square-free string w_j of length above $2000m$. Further assume that the first a occurrence of w_j is at position p, and for $k \geq 1$, define $A(k) = w_j[p..p + k - 1]$. Thus each $A(k)$ starts with a and has length k (and we will choose a subset of these to prove the lemma).

Similarly, assume that the first b of w_j is at position q. Consider $k \geq 1$. Let $r \geq q + k - 1$ be the first position at or after $q + k - 1$ such that $w_j[r] = b$. Define $B(k) = w_j[q..r]$. Note that $r \leq q + k + 2$, as otherwise $w_j[q + k - 1..q + k + 2]$ would be a binary string of length 4 and would therefore contain a square. It follows that $k \leq |B(k)| \leq k + 3$. Moreover, each $B(k)$ starts and ends with a b.

To conclude the lemma, for each $i \in \{1, \ldots, m\}$, put $B_i = B(1000m - 100i)$ and $A_i = A(1000m - |B_i|)$. We have $1000m - 100i \leq B_i \leq 1000m - 100i + 3$, and, since $|A_i| = 1000m - |B_i|$, we have $100i - 3 \leq |A_i| \leq 100i$. The other conditions of the lemma follow from our construction. ∎

We can now proceed to our reduction.

THEOREM 4.2. *The TD problem is NP-hard for any $|\Sigma| \geq 4$, where Σ is the alphabet of the input strings.*

Proof. Our reduction is from the TD problem with unbounded alphabet size, which is NP-hard owing to Theorem 3.2. We show that the TD problem is NP-hard when $|\Sigma| = 4$, which clearly implies hardness for any $|\Sigma| \geq 4$ since one can add new dummy character at the end of S and T is $|\Sigma|$ is smaller than the desired value. Let (S, T, k) be an instance of the TD problem, let $\Sigma = \{s_1, \ldots, s_m\}$ be the alphabet of S and T, with $m = |\Sigma|$.

Let $(A_1, B_1), \ldots, (A_m, B_m)$ be pairs of ternary square-free strings that satisfy the conditions of Lemma 4.1, i.e. each $|A_i|$ is between $100i - 3$ and $100i$, $|B_i| = 1000m - |A_i|$, with A_i starting with a, and B_i starting and ending with b. By the lemma, these can be obtained in polynomial time. Assume that the alphabet of the A_i and B_i strings is $\{a, b, c\}$, and let x be another character.

We encode each character $s_i \in \Sigma$ on alphabet $\{a, b, c, x\}$ by defining

$$h(s_i) = xcxA_ixB_i$$

Note that each $h(s_i)$ string has the same length $1000m + 4$. Also note that each A_i has a distinct length, and thus that each B_i also has a distinct length. This implies that each A_i is matched with a distinct B_i determined by its length (and vice-versa).

Moreover, since A_i and B_i are square-free, we can argue that $h(s_i)$ is also square-free. Indeed, no square can start at the first x of $h(s_i)$ since xc only occurs once in $h(s_i)$; the square could not be $cxcx$ since the first cx would have to start at the second character c of $h(s_i)$, and the second cx could only be at the end of A_i, but the latter has length at least 97, making $cxcx$ impossible; no longer square can start at the second character c since such a square would need to start with cxa, of which there is only one occurrence; no square can start at the second x since xa occurs only once; and no square can be entirely in A_i or entirely in B_i because they are square-free. Therefore, a square would need to start in A_i and contain the x after it, which is not possible since there is only one such x. This lets us assert that no tandem duplication can occur inside an $h(s_i)$.

We then encode S and T into S' and T', respectively, by replacing each character with their encoding. That is, put

$$S' = h(S[1])h(S[2])\dots h(S[|S|]) \qquad T' = h(T[1])h(T[2])\dots h(T[|T|])$$

Observe that in both strings, all occurrences of $xcxa$ mark the start of an encoded character (since all A_i strings start with a), and occurrences of xb mark a separation point between some A_i and B_i. In what follows, we say that a string Y *encodes* X if $Y = h(X[1])\dots h(X[|X|])$.

We show that T can be transformed into S using k contractions if and only if T' can be turned into S' using k contractions.

($\Rightarrow$) Assume that T can be transformed into S using k contractions. Let $T = T_0, T_1, \dots, T_k = S$ be the sequence of strings encountered during these contractions. We define a sequence of contractions from T' to S' in which $T' = T'_0, T'_1, \dots, T'_k = S'$ are the strings encountered during this sequence, such that each T'_i encodes T_i. This is true when $i = 0$. Assume inductively that T'_{i-1} encodes T_{i-1}, and that the i-th contraction affects some substring RR of T_{i-1}, i.e. $T_{i-1} = ARRB \rightarrowtail ARB$. Then $T'_{i-1} = A'R'R'B'$, where A' encodes A, R' encodes R and B' encodes B. We can apply $A'R'R'B' \rightarrowtail A'R'B'$ and obtain T'_i that encodes T_i. It follows that $T'_k = S'$, as desired.

($\Leftarrow$) Assume that there is a sequence of k contractions $D'_1, \dots, D'_k$ transforming T' into S' (assume that the D'_i's contain the start and end index of the contraction). Let T'_i be the string obtained after applying the i-th contraction in this sequence, with $T'_0 = T'$. We construct a sequence of contractions $D_1, \dots, D_k$ transforming T into S. To achieve this, we construct the D_i contractions by maintaining the following invariant: by defining T_i as the string obtained after applying $D_1, \dots, D_i$ on T, then T'_i encodes T_i.

This is true for $i = 0$ since $T'_0 = T'$ encodes $T_0 = T$. Now assume inductively that T'_{i-1} is obtained by encoding each character of T_{i-1}.

Assume that D'_i contracts a string RR of T'_{i-1}. Since T'_{i-1} is a concatenation of $h(s_j)$ strings, the first character of the RR substring must be in some $h(s_j)$ substring of T'_{i-1}. We consider four possible cases.

Case 1. The first character of RR is in the xcx substring at the start of $h(s_j) = xcxA_jxB_j$. Thus we may write $xcx = C_1C_2$ such that C_2 is a prefix of RR (note that C_1 is possibly empty, but not C_2). It is easy to check that R must contain at least a prefix of A_j. This implies that R contains xa as a subtring, i.e. $R = C_2aY$ for some Y.

The contraction $T'_{i-1} \rightarrowtail T'_i$ has the form

$$T'_{i-1} = PC_1\underline{C_2aY}\ \underline{C_2aY}Q \rightarrowtail PC_1\underline{C_2aY}Q = T'_i$$

for some strings P, Q. Because C_2a contains xa, it must mark the start of an encoding, since xa can only occur there. Looking at the second copy of C_2a above, we deduce that Y has C_1 as a suffix. Writing $Y = Y'C_1$, we get

$$T'_{i-1} = PC_1\underline{C_2aY'C_1}\ \underline{C_2aY'C_1}Q \rightarrowtail PC_1\underline{C_2aY'C_1}Q = T'_i$$

We can replace D'_i with the following contraction:

$$T'_{i-1} = P\underline{C_1C_2aY'}\ \underline{C_1C_2aY'}C_1Q \rightarrowtail P\underline{C_1C_2aY'}C_1Q = T'_i$$

Notice that P encodes some prefix $\hat{P}$ of T_{i-1} (P is possibly empty). The C_1C_2aY' substring that follows also encodes some substring $\hat{Y}$ of T_{i-1} and, since all encodings have the same length, the second C_1C_2aY' copy that follows encodes the same substring $\hat{Y}$. This implies that C_1Q, if not empty, starts with xcx and encodes a suffix $\hat{Q}$ of T_{i-1}. Therefore, $T_{i-1} = \hat{P}\hat{Y}\hat{Y}\hat{Q}$, and it becomes easy to see that if D_i contracts $\hat{Y}\hat{Y}$, T'_i encodes T_i.

Case 2. The first character of RR is in the A_j substring of $h(s_j)$. Write $A_j = A_j^1A_j^2$ such that A_j^2 is a prefix of RR. Because A_j is square-free, RR cannot be entirely contained in A_j. Thus R contains the x after A_j. This x cannot be the last character of R, as otherwise RR would be fully contained in $h(s_j)$ since $|B_j| > |A_j|$, which is again not possible since $h(s_j)$ is square-free. Thus the first occurrence of x in R is followed by b, implying that the second R copy must start in the A_l portion of some other $h(s_l)$ encoding (since this is the only way to have xb as a subtring for the first x occurrence).

Hence, we may assume that $R = A_j^2xB_jxY$ for some non-empty Y. The contraction has the form

$$PxcxA_j^1\underline{A_j^2xB_jxY}\ \underline{A_j^2xB_jxY}Q \rightarrowtail PxcxA_j^1\underline{A_j^2xB_jxY}Q$$

for some strings P and Q. Since each B_j is associated with a unique A_j, Y must have $xcxA_j^1$ as a suffix. We may write $Y = Y'xcxA_j^1$ and

$$PxcxA_j^1\underline{A_j^2xB_jxY'xcxA_j^1}\ \underline{A_j^2xB_jxY'xcxA_j^1}Q \rightarrowtail PxcxA_i^1\underline{A_j^2xB_jxY'xcxA_j^1}Q$$

It becomes easy to see that D'_i can be replaced by the contraction

$$P\underline{xcxA_j^1A_j^2xB_jxY'}\ \underline{xcxA_j^1A_j^2xB_jxY'}xcxA_j^1Q \rightarrowtail P\underline{xcxA_i^1A_j^2xB_jxY'}xcxA_j^1Q$$

and we can apply Case 1.

Case 3. The first character of RR is the x occurrence in $h(s_j)$ between A_j and B_j. Thus R starts with xb, implying that the second R copy also starts with the x occurrence between some A_l and B_l. The contraction $T'_{i-1} \rightarrowtail T'_i$ has the form

$$PxcxA_j\underline{xB_jxY}\ \underline{xB_jxY}Q \rightarrowtail PxcxA_j\underline{xB_jxY}Q$$

for some strings P, Y, Q. Because B_j is associated only with A_j in the encodings, we may write $Y = Y'xcxA_j$. We get

$$PxcxA_j\underline{xB_jxY'xcxA_j}\ \underline{xB_jxY'xcxA_j}Q \rightarrowtail PxcxA_j\underline{xB_jxY'xcxA_j}Q$$

We can replace D'_i with an alternate contraction as follows:

$$P\underline{xcxA_jxB_jxY'}\ \underline{xcxA_jxB_jxY'}xcxA_jQ \rightarrowtail P\underline{xcxA_jxB_jxY'}xcxA_jQ$$

and, once again, we can apply Case 1.

Case 4. The first character of RR is in the B_j substring of $h(s_j)$. Write $B_j = B_j^1 B_j^2$ such that B_j^2 is a prefix of R. We argue that the first R copy cannot be entirely in B_j^2. Otherwise, the second R copy would contain characters outside of B_j^2 since B_j is square-free. That is, the second R copy would contain an x but not the first, which is impossible.

Thus the first R copy contains the x after B_j^2. We cannot have $R = B_j^2 x$. Indeed, $B_j^2 x$ is followed by cx, and because $R = B_j^2 x$, the only possible contraction is $RR = B_j^2 xcx = cxcx$. This is not possible, since B_j^2 ends with a b.

Therefore, R contains the x after B_j^2, and also the c after it. The only occurrences of xc are at the start of encodings, so the contraction has the form

$$P\underline{B_j^2 xcY}\ \underline{B_j^2 xcY}Q \rightarrowtail P\underline{B_j^2 xcY}Q$$

Now, the second R copy starts with a suffix of some B_l, and it happens that B_j and B_l have the same suffix B_j^2. We can write $B_l = B_l^1 B_j^2$ such that Y has $xA_l xB_l^1$ as a suffix. Write $Y = Y' xA_l xB_l^1$ so that

$$P\underline{B_j^2 xcY' xA_l xB_l^1}\ \underline{B_j^2 xcY' xA_l xB_l^1}Q \rightarrowtail P\underline{B_j^2 xcY' xA_l xB_l^1}Q = T_i'$$

Recall that A_l is uniquely matched with $B_l = B_l^1 B_j^2$ in the encodings, so Q must have B_j^2 as a prefix. Write $Q = B_j^2 Q'$ so that

$$T_{i-1}' = P\underline{B_j^2 xcY' xA_l xB_l^1}\ \underline{B_j^2 xcY' xA_l xB_l^1}B_j^2 Q' \rightarrowtail P\underline{B_j^2 xcY' xA_l xB_l^1}B_j^2 Q' = T_i'$$

We can replace this contraction with

$$T_{i-1}' = PB_j^2\underline{xcY' xA_l xB_l^1 B_j^2}\ \underline{xcY' xA_l xB_l^1 B_j^2}Q' \rightarrowtail PB_j^2\underline{xcY' xA_l xB_l^1 B_j^2}Q' = T_i'$$

We can once again apply Case 1.

We have thus shown how to construct a sequence of contractions that transforms T into a string T_k such that T_k' encodes T_k. Since T_k' encodes S', it follows that $T_k = S$, which concludes the proof. ∎

5. An FPT algorithm for the exemplar problem. Let us recall that in the Exemplar-k-TD problem, we receive a string S in which every character is different, a string T, and an integer k. We want to know whether S can be transformed into T using at most k tandem duplications. In this section, we show that Exemplar-k-TD can be solved in time $2^{O(k^2)} + poly(n)$ by obtaining a kernel of size $O(k2^k)$, where n is the length of T.

We first note that there is a very simple, brute-force algorithm to solve the k-TD problem, which is the variant of the TD problem with parameter k, the number of TDs to turn S into T (including Exemplar-k-TD as a particular case). This only establishes membership in the XP class, but it will be useful to evaluate the complexity of our kernelization later on.

PROPOSITION 5.1. *The k-TD problem can be solved in time $O(n^{2k})$, where n is the size of the target string.*

Proof. Let (S, T) be a given instance of k-TD. Consider the branching algorithm that, starting from T, tries to contract every substring of the form XX in T and

recurses on each resulting string, decrementing k by 1 each time (the branching stops when S is obtained or when k reaches 0 without attaining S). We obtain a search tree of depth at most k and degree at most n^2, and thus it has $O(n^{2k})$ nodes. Visiting the internal nodes of this search tree only requires enumerating $O(n^2)$ substrings, which form the set of children of the node. Hence, there is no added computation cost to consider when visiting a node. ∎

From now on, we assume that we have an Exemplar-k-TD instance (S, T), and so that S is exemplar.

Let x and y be two consecutive characters in S (i.e. xy is a substring of S). We say that xy is an (S, T)-*stable* pair if in T, every occurrence of x in T is followed by y and every occurrence of y is preceded by x. That is, the direct successor of every x character is y, and the direct predecessor of every y character is x. Note that in this definition, S and T could be any strings. However, for the rest of the section, we assume that S and T are the input strings.

We will show that every (S, T)-stable pair can be replaced by a single character, and that if T can be obtained from S using at most k tandem duplications, then this leaves strings of bounded size.

We first show that, roughly speaking, stability is maintained by all tandem duplications when going from S to T.

LEMMA 5.2. *Suppose that $dist_{TD}(S, T) = k$ and let xy be any (S, T)-stable pair. Let $S = S_0, S_1, \ldots, S_k = T$ be any minimum sequence of strings transforming S to T by tandem duplications. Then xy is (S, S_i)-stable for every $i \in [k]$.*

Proof. Note that xy is trivially (S, S_0)-stable. Assume the lemma is false, and let S_i be the first of $S_1, \ldots, S_k$ that does not satisfy the statement. Then xy is (S, T)-stable, but xy is not (S, S_i)-stable.

We claim that, with our assumption, xy is not (S, S_j)-stable for any $j \in \{i, \ldots, k\}$. As this includes $S_k = T$, this will contradict that xy is (S, T)-stable. We do this by induction — as a base case, xy is not (S, S_i)-stable so this is true for $j = i$. Assume that xy is not (S, S_{j-1})-stable, where $i < j \leq k$. Let D be the duplication transforming S_{j-1} to S_j (here $D = (a, b)$ contains the start and end positions of the substring of S_{j-1} to duplicate).

Suppose first that xy is not (S, S_{j-1})-stable because S_{j-1} has an occurrence of x that is not followed by y. Thus S_{j-1} has an occurrence of xz, with x at some position p_x, where $z \neq y$. If we assume that xy is (S, S_j)-stable, then a y character must have appeared after this x from S_{j-1} to S_j. If D contains the xz occurrence, then xz will still be an occurrence of S_j. If D does not contain the x character, then again xz will still occur in S_j. It follows that changing the character next to this x is only possible if the last character duplicated by D is the x at position p_x and the first character of D is a y. In other words, denoting $S_{j-1} = A_1 y A_2 x z A_3$ for appropriate A_1, A_2, A_3 substrings, the D duplication must do the following

$$A_1 \underline{y A_2 x} z A_3 \Rightarrow A_1 \underline{y A_2 x y A_2 x} z A_3,$$

But then, there is still an occurrence of x followed by z, and it follows that xy cannot be (S, S_j)-stable.

So suppose instead that xy is not (S, S_{j-1})-stable because S_{j-1} has an occurrence of y preceded by $z \neq x$. Assume the character preceding this y has changed in S_j and has become x. But one can verify that this is impossible. For completeness, we present each possible case: either D includes both z and y, includes one of them or

none. These cases are represented below, and each one of them leads to an occurrence of y still preceded by z (the left-hand side represents S_{j-1} and the right-hand side represents S_j, and the A_i's are the relevant substrings in each case):

$$\text{Include both: } A_1\underline{A_2zyA_3}A_4 \Rightarrow A_1\underline{A_2zyA_3A_2zyA_3}A_4,$$

$$\text{Include } z \text{ only: } A_1\underline{A_2zyA_3} \Rightarrow A_1\underline{A_2zA_2zyA_3},$$

$$\text{Include } y \text{ only: } A_1zy\underline{A_2A_3} \Rightarrow A_1zy\underline{A_2yA_2A_3},$$

$$\text{Include none: } A_1\underline{A_2zyA_3} \Rightarrow A_1\underline{A_2A_2zyA_3} \text{ or } A_1zy\underline{A_2A_3} \Rightarrow A_1zy\underline{A_2A_2A_3}.$$

We have therefore shown that xy cannot be (S, S_j)-stable, and therefore not (S, T)-stable, which concludes the proof. ∎

Our next step is to show that no tandem duplication starts or ends between two characters of a given stable pair xy. This might not be the case in general, but we show that any duplication that does so can be modified to an equivalent duplication that does not "cut" any occurrence of xy.

Let xy be an (S, T)-stable pair. Let S' be a substring obtained from S by tandem duplications, and assume that $S'[p_x..p_x + 1] = xy$. Suppose that we then apply a duplication $D = (c, d)$ on S', which copies the substring $S'[c..d]$ and pastes it after position d. Then we say that D *cuts* xy if one of the following holds:

- $c = p_x + 1$, in which case we say that D cuts xy *to the right*;
- $d = p_x$, in which case we say that D cuts X *to the left*;

In other words, cutting to the right takes the form $Axy\underline{BC} \Rightarrow Axy\underline{ByBC}$. Cutting to the left takes the form $A\underline{Bx}yC \Rightarrow A\underline{BxBx}yC$. Observe that if D does not cut any occurrence of xy, and S'' is obtained by applying D on S', then xy is (S, S'')-stable.

LEMMA 5.3. *Suppose that* $dist_{TD}(S, T) = k$ *and let* xy *be an* (S, T)*-stable pair. Then there exists a sequence of tandem duplications* $D_1, \ldots, D_k$ *transforming* S *into* T *such that no occurrence of* xy *gets cut by a* D_i.

In other words, for all $i \in [k]$*, the tandem duplication* D_i *does not cut any occurrence of* xy *in the string obtained by applying* $D_1, \ldots, D_{i-1}$ *to* S.

Proof. Let $D_1^*, \ldots, D_k^*$ be a sequence of tandem duplications transforming S into T, and for $i \in [k]$, let S_i be the string obtained by applying the first i duplications. Put $S_0 := S$. We show that any S_i, $i \in [k]$, can be obtained from S_{i-1} by a duplication D_i that does not cut any occurrence of xy. This proves the lemma, since $D_1, \ldots, D_k$ will form the desired sequence of duplications.

Fix $i \in [k]$, and assume that D_i^* cuts at least one xy pair. Notice that D_i^* can cut at most two occurrences of xy, at most one to the left and at most one to the right. Also, by Lemma 5.2, we know that every (S, T)-stable xy pair is (S, S_i)-stable. We have four cases to consider:

- D_i^* cuts some xy to the right, but no other occurrence to the left. Then we may write S_{i-1} as $S_{i-1} = AxyBC$ such that D copies the substring yB. This gives

$$S_{i-1} = Axy\underline{BC} \Rightarrow Axy\underline{ByBC} = S_i.$$

  But by Lemma 5.2, xy is (S, S_i)-stable. Since S_i has By as a substring, this must mean that $B = \hat{B}x$ for some substring $\hat{B}$. Therefore $S_{i-1} = Axy\hat{B}xC$. Since xy is also (S, S_{i-1})-stable, this in turn implies that $C = y\hat{C}$ for some substring $\hat{C}$, and in fact we get

$$S_{i-1} = Axy\underline{\hat{B}xy}\hat{C} \Rightarrow Axy\underline{\hat{B}xy\hat{B}xy}\hat{C} = S_i.$$

We can replace D_i^* by a duplication that copies $xy\hat{B}$, i.e.

$$S_{i-1} = Axy\underline{\hat{B}xy}\hat{C} \Rightarrow Axy\underline{\hat{B}xy\hat{B}xy}\hat{C} = S_i.$$

Recall that x and y are distinct characters because S is exemplar. Since this duplication starts with xy and copies itself right before another occurrence of xy, it is clear that it does not cut any occurrence of xy, as desired.

- D_i^* cuts some xy to the left, but cuts no string to the right. Then we may write

$$S_{i-1} = A\underline{Bxy}C \Rightarrow A\underline{BxBxy}C = S_i.$$

Similarly as in the previous case, since xy is (S, S_i)-stable, we must have $B = y\hat{B}$. We are led to deduce that $A = \hat{A}x$. Therefore we have

$$S_{i-1} = \hat{A}xy\underline{\hat{B}xy}C \Rightarrow \hat{A}xy\underline{\hat{B}xy\hat{B}xy}C = S_i.$$

As before, we could instead duplicate the substring $xy\hat{B}$ occuring right after $\hat{A}$.

- D_i^* cuts some xy to the left and some xy to the right. We get

$$S_{i-1} = Axy\underline{Bxy}C \Rightarrow Axy\underline{BxyBxy}C = S_i.$$

We could instead duplicate the copy of xyB in S_{i-1} that occurs right after the A.

We have thus shown that if D_i^* cuts some occurrence of one or more of the xy's, then D_i^* can be replaced by a duplication D_i that yields the same string S_i as D_i^*. The only case remaining is when D_i^* does not cut any X_j. In that case, we put $D_i = D_i^*$. This shows that we can find the claimed sequence $D_1, \ldots, D_k$ in which no xy ever gets cut. ∎

The above implies that we may replace any (S, T)-stable pair xy by a single character, since we may assume that characters of xy are always duplicated together (assuming, of course, that S is exemplar).

LEMMA 5.4. *Let xy be an (S, T)-stable pair, let z be a character not occurring in S or T, and let S', T' be obtained by replacing each occurrence of xy in S and T, respectively, by z. Then $dist_{TD}(S, T) = dist_{TD}(S', T')$.*

Proof. Clearly if (S', T') can be solved using at most k duplications, then the same applies to (S, T), since we can modify each duplication that affects z to affect xy instead. By Lemma 5.3, the converse also holds: if (S, T) can be solved with at most k duplications, we may assume that these duplications never cut an xy occurrence, and so these duplications can be applied on (S', T'). ∎

It only remains to show that the resulting strings are small enough. This can be achieved by repeated application of Lemma 5.4. That is, as long as there is an (S,T)-stable pair xy, we can replace xy by a new character z. When this procedure terminates, the resulting strings S' and T' have the same TD distance as S and T, but contain no (S,T)-stable pair. We can then rely on the following intuition: a tandem duplication can only "break" two stable pairs, so if S' and T' are too long, k duplications won't suffice to break them all.

LEMMA 5.5. *Assume that there is no (S,T)-stable pair. If $dist_{TD}(S,T) \leq k$, then S has length at most $2k+1$.*

Proof. Let $S = S_0, S_1, \ldots, S_k = T$ be any minimum sequence of strings transforming S to T by tandem duplications. Let n be the length of S. We show by induction that, for each $i \in \{0,1,\ldots,k\}$, the number of (S,S_i)-stable pairs is at least $n-2i-1$. For $i=0$, there are $n-1$ (S,S)-stable pairs, so our assertion holds. Now assume that there are at least $n-2(i-1)-1 = n-2i+1$ (S,S_{i-1})-stable pairs, with $i>0$. Assume that the duplication transforming S_{i-1} to S_i takes the form

$$S_{i-1} = Axy\underline{Buv}C \Rightarrow Axy\underline{BuyBuv}C = S_i$$

for some characters x,y,u,v and substrings A,B,C. The only (S,S_{i-1})-stable pairs that might not be (S,S_i)-stable pairs are xy and uv (since no other substring of length two was modified). Note that A,x,v or C might not exist if the duplication contains the start or end of S_{i-1}, but in this case the number of stable pairs still cannot decrease by more than 2. It follows that the number of (S,S_i)-stable pairs is at least $n-2i+1-2 = n-2i-1$, as desired.

By putting $i=k$ and recalling that $S_k = T$, the number of (S,T)-stable pairs is at least $n-2k-1$. Since this number is 0 by assumption, we get $0 \geq n-2k-1$, implying $n \leq 2k+1$. ∎

We have thus shown how to transform an instance (S,T) of Exemplar-k-TD to a kernel, an equivalent instance (S',T') of size depending only on k.

THEOREM 5.6. *An instance (S,T) of Exemplar-k-TD admits a kernel (S',T') in which $|S'| \leq 2k+1$ and $|T'| \leq (2k+1)2^k$.*

Proof. As shown in Lemma 5.5, after repeatedly replacing each (S,T)-stable substring by a single character, we obtain equivalent strings S' and T' with $|S'| \leq 2k+1$. Under the assumption that $dist_{TD}(S',T') \leq k$, then each duplication can at most double the size of the previous string. Therefore, T' must have size at most $(2k+1)2^k$. ∎

The kernelization can be performed in polynomial time, as one only needs to identify (S,T)-stable pairs and contract them. Running the brute-force algorithm from Proposition 5.1 yields the following.

COROLLARY 5.7. *The exemplar k-tandem duplication problem can be solved in time $O(((2k+1)2^k)^{2k} + poly(n)) = 2^{O(k^2)} + poly(n)$, where n is the size of the input.*

6. Open problems. Although this work answers some open questions, many of them still deserve investigation. We conclude with some of these questions along with future research perspectives.

- Is the k-TD problem FPT in parameter k? As we observe in our Exemplar-k-TD kernelization, if T and S are large compared to k, they must share many long common substrings which could be exploited for an FPT algorithm. It is also an interesting question whether Exemplar-k-TD admits a polynomial size kernel.

- If $|\Sigma| = 2$ or 3, is the TD problem still NP-hard?
- Can one decide in polynomial time whether $S \Rightarrow_* T$? The only known result on this topic is that it can be done if $|\Sigma| = 2$, as one can construct a finite automaton accepting all strings generated by S (though this automaton does not give the minimum number of duplications required).
- Does the TD problem admit a constant factor approximation algorithm? The answer might depend on the hardness of deciding whether $S \Rightarrow_* T$, but one might still consider the promise version of the problem.
- If the length of each duplicated string is bounded by d, is k-TD in P (with d treated as a constant)? We believe that it is FPT in $k + d$, but is it FPT in d?
- Another parameter of interest is the maximum number of occurrences of a character in S or T. That is, if each $a \in \Sigma$ occurs at most c times in both S and T, is the problem FPT in parameter c?

REFERENCES

[1] N. Alon, J. Bruck, F. F. Hassanzadeh, and S. Jain, *Duplication distance to the root for binary sequences*, IEEE Transactions on Information Theory, 63 (2017), pp. 7793–7803, https://doi.org/10.1109/TIT.2017.2744608.

[2] G. Benson and L. Dong, *Reconstructing the duplication history of a tandem repeat*, in Proceedings of the 17th International Conference on Intelligent Systems for Molecular Biology, ISMB'99, AAAI, 1999, pp. 44–53.

[3] D. P. Bovet and S. Varricchio, *On the regularity of languages on a binary alphabet generated by copying systems*, Information Processing Letters, 44 (1992), pp. 119–123, https://doi.org/10.1016/0020-0190(92)90050-6.

[4] L. Bulteau, G. Fertin, and I. Rusu, *Sorting by transpositions is difficult*, SIAM Journal on Discrete Mathematics, 26 (2012), pp. 1148–1180, https://doi.org/10.1137/110851390.

[5] B. Charlesworth, P. Sniegowski, and W. Stephan, *The evolutionary dynamics of repetitive dna in eukaryotes*, Nature, 371 (1994), pp. 215–220, https://doi.org/10.1038/371215a0.

[6] K. Chaudhuri, K. Chen, R. Mihaescu, and S. Rao, *On the tandem duplication-random loss model of genome rearrangement*, in Proceedings of 17th ACM-SIAM Symposium on Discrete Algorithms, SODA'06, SIAM, 2006, pp. 564–570.

[7] Z.-Z. Chen, L. Wang, and Z. Wang, *Approximation algorithms for reconstructing the duplication history of tandem repeats*, Algorithmica, 54 (2009), pp. 501–529, https://doi.org/10.1007/s00453-008-9209-8.

[8] D.-J. Cho, Y.-S. Han, and H. Kim, *Bound-decreasing duplication system*, Theoretical Computer Science, 793 (2019), pp. 152–168, https://doi.org/10.1016/j.tcs.2019.06.018.

[9] J. Dassow, V. Mitrana, and G. Paun, *On the regularity of the duplication closure*, Bulletin of the EATCS, 69 (1999), pp. 133–136.

[10] R. G. Downey and M. R. Fellows, *Fixed-parameter tractability and completeness ii: On completeness for w[1]*, Theoretical Computer Science, 141 (1995), pp. 109–131, https://doi.org/10.1016/0304-3975(94)00097-3.

[11] A. Ehrenfeucht and G. Rozenberg, *On regularity of languages generated by copying systems*, Discrete Applied Mathematics, 8 (1984), pp. 313–317, https://doi.org/10.1016/0166-218X(84)90129-X.

[12] O. Gascuel, M. D. Hendy, A. Jean-Marie, and R. McLachlan, *The combinatorics of tandem duplication trees*, Systematic Biology, 52 (2003), pp. 110–118, https://doi.org/10.1080/10635150390132821.

[13] D. Gusfield, *Algorithms on strings, trees and sequences: computer science and computational biology*, Cambridge University Press, 1997.

[14] D. Gusfield and J. Stoye, *Linear time algorithms for finding and representing all the tandem repeats in a string*, Journal of Computer and System Sciences, 69 (2004), pp. 525–546, https://doi.org/10.1016/j.jcss.2004.03.004.

[15] S. Hannenhalli and P. A. Pevzner, *Transforming men into mice (polynomial algorithm for genomic distance problem)*, in Proceedings of IEEE 36th Symposium on Foundations of Computer Science, FOCS'95, IEEE, 1995, pp. 581–592, https://doi.org/10.1109/SFCS.1995.492588.

[16] F. F. Hassanzadeh, M. Schwartz, and J. Bruck, *The capacity of string-duplication systems*, IEEE Transactions on Information Theory, 62 (2016), pp. 811–824, https://doi.org/10.1109/TIT.2015.2505735.

[17] M. Ito, P. Leupold, and K. Shikishima-Tsuji, *Closure of languages under bounded duplications*, in Proceedings of the 10th International Conference on Developments in Language Theory, vol. 4036 of Lecture Notes in Computer Science, Spring-Verlag, 2006, pp. 238–247, https://doi.org/10.1007/11779148_22.

[18] S. Jain, F. F. Hassanzadeh, and J. Bruck, *Capacity and expressiveness of genomic tandem duplication*, IEEE Transactions on Information Theory, 63 (2017), pp. 6129–6138, https://doi.org/10.1109/TIT.2017.2728079.

[19] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, eds., The IBM Research Symposia Series, Springer, Boston, MA, 1972, pp. 85–103, https://doi.org/10.1007/978-1-4684-2001-2_9.

[20] M. Lafond, B. Zhu, and P. Zou, *The tandem duplication distance is NP-hard*, in Proceedings of STACS'2020, LIPIcs, vol. 154, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2020, pp. 15:1–15:15.

[21] G. M. Landau, J. P. Schmidt, and D. Sokol, *An algorithm for approximate tandem repeats*, Journal of Computational Biology, 8 (2001), pp. 1–18, https://doi.org/10.1089/106652701300099038.

[22] J. Leech, *A problem on strings of beads*, The Mathematical Gazette, 41 (1957), pp. 277–278.

[23] I. Letunic, R. R. Copley, and P. Bork, *Common exon duplication in animals and its role in alternative splicing*, Human Molecular Genetics, 11 (2002), pp. 1561–1567, https://doi.org/10.1093/hmg/11.13.1561.

[24] P. Leupold, C. Martin-Vide, and V. Mitrana, *Uniformly bounded duplication languages*, Discrete Applied Mathematics, 146 (2005), pp. 301–310, https://doi.org/10.1016/j.dam.2004.10.003.

[25] P. Leupold, V. Mitrana, and J. M. Sempere, *Formal languages arising from gene repeated duplication*, in Aspects of Molecular Computing, G. P. Natasa Jonoska and G. Rozenberg, eds., vol. 2950 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2004, pp. 297–308, https://doi.org/10.1007/978-3-540-24635-0_22.

[26] M. Macdonald, et al., and P. S. Harper, *A novel gene containing a trinucleotide repeat that is expanded and unstable on huntington's disease*, Cell, 72 (1993), pp. 971–983, https://doi.org/10.1016/0092-8674(93)90585-E.

[27] L. Oesper, A. M. Ritz, S. J. Aerni, R. Drebin, and B. J. Raphael, *Reconstructing cancer genomes from paired-end sequencing data*, BMC Bioinformatics, 13 (2012), p. S10, https://doi.org/10.1186/1471-2105-13-S6-S10.

[28] L. Qingge, X. He, Z. Liu, and B. Zhu, *On the minimum copy number generation problem in cancer genomics*, in Proceedings of the 10th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB'18, New York, NY, 2018, ACM Press, pp. 260–269, https://doi.org/10.1145/3233547.3233586.

[29] D. Sankoff, *Gene and genome duplication*, Current opinion in genetics and development, 11 (2001), pp. 681–684, https://doi.org/10.1016/S0959-437X(00)00253-7.

[30] A. J. Sharp, et al., and E. E. Eichler, *Segmental duplications and copy-number variation in the human genome*, The American J. of Human Genetics, 77 (2005), pp. 78–88, https://doi.org/10.1086/431652.

[31] J. W. Szostak and R. Wu, *Unequal crossing over in the ribosomal dna of saccharomyces cerevisiae*, Nature, 284 (1980), pp. 426–430, https://doi.org/10.1038/284426a0.

[32] O. Tremblay-Savard, D. Bertrand, and N. El-Mabrouk, *Evolution of orthologous tandemly arrayed gene clusters*, BMC Bioinformatics, 12 (2011), p. S2, https://doi.org/10.1186/1471-2105-12-S9-S2.

[33] M.-W. Wang, *On the irregularity of the duplication closure*, Bulletin of the EATCS, 70 (2000), pp. 162–163.

[34] B. Zhu, *Tandem duplication, segmental duplications and deletions, and their applications*, in Proceedings of the 15th International Computer Science Symposium in Russia (CSR'20), LNCS, vol. 12159, Springer, 2020, pp. 79–102.