Hardness of RNA Folding Problem With Four Symbols*

Yi-Jun Chang

Department of EECS, University of Michigan, Ann Arbor, MI, USA
cyijun@umich.edu

Abstract

An RNA sequence is a string composed of four types of nucleotides, A, C, G, and U. Given an RNA sequence, the goal of the RNA folding problem is to find a maximum cardinality set of crossing-free pairs of the form $\{A, U\}$ or $\{C, G\}$. The problem is central in bioinformatics and has received much attention over the years. Whether the RNA folding problem can be solved in $\mathcal{O}(n^{3-\epsilon})$ time remains an open problem. Recently, Abboud, Backurs, and Williams (FOCS'15) made the first progress by showing a conditional lower bound for a generalized version of the RNA folding problem based on a conjectured hardness of the k-clique problem. However, their proof requires alphabet size ≥ 36 to work, making the result biologically irrelevant. In this paper, by constructing the gadgets using a lemma of Bringmann and Künnemann (FOCS'15) and surrounding them with some carefully designed sequences, we improve upon the framework of Abboud et al. to handle the case of alphabet size 4, yielding a conditional lower bound for the RNA folding problem. We also investigate the Dyck edit distance problem. We demonstrate a reduction from RNA folding problem to Dyck edit distance problem of alphabet size 10, establishing a connection between the two fundamental string problems. This leads to a much simpler proof of the conditional lower bound for Dyck edit distance problem given by Abboud et al. and lowers the required alphabet size for the lower bound to work.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases RNA folding, Dyck edit distance, longest common subsequence, conditional lower bound, clique

Digital Object Identifier 10.4230/LIPIcs.CPM.2016.13

1 Introduction

An *RNA sequence* is a string composed of four types of nucleotides, A, C, G, and U. Given an RNA sequence, the goal of the *RNA folding* problem is to find a maximum cardinality set of crossing-free pairs of nucleotides, where all the pairs are either $\{A, U\}$ or $\{C, G\}$. The problem is central in bioinformatics and has found applications in many areas of molecular biology. For a comprehensive exposition of the topic, the reader is referred to e.g. [18].

It is well-known that the problem can be solved in cubic time by a simple dynamic programming method [9]. Due to the importance of RNA folding in practice, there has been a long line of research on improving the time complexity (See e.g. [3, 11, 12, 13, 18, 21]). Currently the best upper bound is $\mathcal{O}\left(\frac{n^3}{\log^2(n)}\right)$ [13, 18], which can be obtained by four-Russian method or fast min-plus multiplication (based on ideas from Valiant's CFG parser [19]).

* A more detailed version of the paper: http://arxiv.org/abs/1511.04731. Supported by NSF grants CCF-1217338, CNS-1318294, and CCF-1514383.

Whether the RNA folding problem can be solved in $\mathcal{O}(n^{3-\epsilon})$ time for some $\epsilon > 0$ is still a major open problem. Other than attempting to improve the upper bound, we should also approach the problem in the opposite direction, i.e. arguing why the problem is hard.

Conditional lower bounds

A popular way to show hardness of a problem is to demonstrate a lower bound conditioned on some widely accepted hypothesis.

▶ **Conjecture 1** (Strongly Exponential Time Hypothesis (SETH)). *There exists no $\epsilon, k_0 > 0$ such that k-SAT with n variables can be solved in time $\mathcal{O}(2^{(1-\epsilon)n})$ for all $k > k_0$.*

▶ **Conjecture 2.** *There exists no $\epsilon, k_0 > 0$ such that k-clique on graphs with n nodes can be solved in time $\tilde{\mathcal{O}}\left(n^{(\omega-\epsilon)k/3}\right)$ for all $k > k_0$, where $\omega < 2.373$ is the matrix multiplication exponent.*

For instance, assuming that SETH (Conjecture 1) holds, the following bounds are unattainable for any $\epsilon > 0$:

- an $\mathcal{O}(n^{k-\epsilon})$ algorithm for k-dominating set problem [14],
- an $\mathcal{O}(n^{2-\epsilon})$ algorithm for dynamic time warping, longest common subsequence, and edit distance [2, 6, 7],
- an $\mathcal{O}(m^{2-\epsilon})$ algorithm for $(3/2-\epsilon)$-approximating the diameter of a graph with m edges [15].

We note that such negative results allow us to have a better picture of the structure of polynomial time complexity, and identify the main obstacles to obtaining faster algorithms for various fundamental problems.

As remarked in [1], it is easy to reduce the longest common subsequence problem on binary strings to the RNA folding problem as following: Given two binary strings X, Y, we let $\hat{X} \in \{A, C\}^{|X|}$ be the string such that $\hat{X}[i] = A$ if $X[i] = 0$, $\hat{X}[i] = C$ if $X[i] = 1$, and we let $\hat{Y} \in \{G, U\}^{|Y|}$ be the string such that $\hat{Y}[i] = U$ if $Y[i] = 0$, $\hat{Y}[i] = G$ if $Y[i] = 1$. Then we have a 1-1 correspondence between RNA foldings of $\hat{X} \circ \hat{Y}^R$ (i.e. concatenation of $\hat{X}$ and the reversal of $\hat{Y}$) and common subsequences of X and Y. It has been shown in [7] that there is no $\mathcal{O}(n^{2-\epsilon})$ algorithm for longest common subsequence problem on binary strings conditioned on SETH, and we immediately get the same conditional lower bound for RNA folding from the simple reduction!

Very recently, based on a conjectured hardness of k-clique problem (Conjecture 2), a higher conditional lower bound was proved for a generalized version of the RNA folding problem (which coincides with the RNA folding problem when the alphabet size is 4) [1]:

▶ **Theorem 1** ([1]). *If the generalized RNA folding problem on sequences of length n with alphabet size 36 can be solved in $T(n)$ time, then $3k$-clique on graphs with $|V| = n$ can be solved in $\mathcal{O}\left(T\left(n^{k+2}\log(n)\right)\right)$ time.*

Therefore, a $\mathcal{O}(n^{\omega-\epsilon})$ time algorithm for the generalized RNA folding with alphabet size at least 36 will disprove Conjecture 2, yielding a breakthrough to the parameterized complexity of clique problem.

However, the above theorem is irrelevant to the RNA folding problem in real life (which has alphabet size 4). It is unknown whether the generalized RNA folding for alphabet size 4 admits a faster algorithm than the case for alphabet size > 4. In fact, there are examples of string algorithms whose running time scales with alphabet size (e.g. string matching with mismatched [5] and jumbled indexing [4, 8]). We also note that when the alphabet size is 2, the generalized RNA folding can be trivially solved in linear time.

In this paper, we improve upon Theorem 1 by showing the same conditional lower bound for the RNA folding problem:

▶ **Theorem 2.** *If the RNA folding problem on sequences in $\{A, C, G, U\}^n$ can be solved in $T(n)$ time, then $3k$-clique on graphs with $|V| = n$ can be solved in $\mathcal{O}\left(T\left(n^{k+1}\log(n)\right)\right)$ time.*

Note that we also get an $\mathcal{O}(n)$ factor improvement inside $T(\cdot)$, though it does not affect the conditional lower bound.

In the proof of Theorem 1 in [1], given a graph $G = (V, E)$, a sequence of length $\mathcal{O}(n^{k+2}\log(n))$ is constructed in such a way that we can decide whether G has a $3k$-clique according to the number of pairs in an optimal generalized RNA folding of S. The construction requires a large alphabet size to build various "walls" which prevent undesired pairings between different parts of the sequence. Extending their approach to handle the case with alphabet size 4 may not be easy without aid from other techniques and ideas.

Overview of our approach

At a high level, our reduction (from $3k$-clique problem to RNA folding problem) follows the approach in [1]: We enumerate all k-cliques, and each of them is encoded as some gadgets. All the gadgets are then put together to form an RNA sequence. The goal is to ensure that an optimal RNA folding corresponds to choosing three k-cliques that form a $3k$-clique, given that the underlying graph admits a $3k$-clique.

To achieve this result using 4 symbols, we implement the above construction using more efficient gadgets based on a key lemma in [7], whose original purpose is to prove that longest common subsequence and other edit distance problems are SETH-hard even on binary strings. We will treat it as a black box and apply it multiple times.

In the final RNA sequence, all clique gadgets are well-separated by some carefully designed sequences whose purpose is to "trap" all the clique gadgets except three of them. We will see that only these three clique gadgets can influence the number of matched pairs in an optimal RNA folding, and the number of matched pairs is maximized when these three clique gadgets correspond to a $3k$-clique. Therefore, we can infer whether the graph has a $3k$-clique from the optimal RNA folding of the RNA sequence.

Dyck Edit Distance

One other way to formulate the RNA folding problem is as follows: deleting the minimum number of letters in a given string to transform the string into a string in the language defined by the grammar $\mathbf{S} \to \mathbf{SS}, A\mathbf{S}U, U\mathbf{S}A, C\mathbf{S}G, G\mathbf{S}C, \epsilon$ (empty string). The *Dyck edit distance problem* [16, 17], which asks for the minimum number of edits to transform a given string to a well-balanced parentheses of s different types, has a similar formulation. Due to the similarity, the same conditional lower bound as Theorem 1 was also shown for the Dyck edit distance problem (with alphabet size ≥ 48) in [1]. In this paper, we improve and simplify their result by demonstrating a simple reduction from RNA folding to Dyck edit distance problem:

▶ **Theorem 3.** *If Dyck edit distance problem on sequences of length n with alphabet size 10 can be solved in $T(n)$ time, then the RNA folding problem on sequences in $\{A, C, G, U\}^n$ can be solved in $\mathcal{O}(T(n))$ time.*

▶ **Corollary 4.** *If the Dyck edit distance problem on sequences of length n with alphabet size 10 can be solved in $T(n)$ time, then $3k$-clique on graphs with $|V| = n$ can be solved in $\mathcal{O}\left(T\left(n^{k+1}\log(n)\right)\right)$ time.*

Interpretations of our results

The current state-of-art algorithm for $k-$clique, which takes $\mathcal{O}\left(n^{\omega k/3}\right)$ time, requires the use of fast matrix multiplication [10] which does not perform very efficiently in practice. For combinatorial, non-algebraic algorithm for $k-$clique, the current state-of-art has time complexity $\mathcal{O}\left(\frac{n^k}{\log^k(n)}\right)$ [20], which is only slightly better than the trivial approach.

Therefore, despite the current gap between the n^3 upper bound and the n^ω lower bound (neglecting polylog factors) for RNA folding and Dyck edit distance, it is unlikely to have an $n^{3-\epsilon}$ time "efficient" algorithm for these problems, unless there is a breakthrough in combinatorial algorithms for k-clique. As a result, our reductions (and the ones in [1]) imply that very likely the use of approximation or heuristic is necessary if one needs a faster algorithm.

2 Preliminaries

Given a set of letters Σ, the set Σ' is defined as $\{x' | x \in \Sigma\}$. We require that $\Sigma \cap \Sigma' = \emptyset$, and $\forall x, y \in \Sigma, (x \neq y) \rightarrow (x' \neq y')$. Therefore, we have $|\Sigma'| = |\Sigma|$ and $|\Sigma \cup \Sigma'| = 2|\Sigma|$.

For any $X = (x_1, \ldots, x_k) \in \Sigma^k$, we write $p(X)$ to denote $(x'_1, \ldots, x'_k)$ (the letter p stands for the prime symbol). We denote the reversal of the sequence X as X^R. The concatenation of two sequences X, Y is denoted as $X \circ Y$ (or simply XY). We write *substring* to denote a contiguous subsequence. Two pairs of indices (i_1, j_1), (i_2, j_2), with $i_1 < j_1$ and $i_2 < j_2$, form a *crossing pair* iff $(\{i_1, j_1\} \cap \{i_2, j_2\} \neq \emptyset) \vee (i_1 < i_2 < j_1 < j_2) \vee (i_2 < i_1 < j_2 < j_1)$.

Generalized RNA Folding

Given $S \in (\Sigma \cup \Sigma')^n$, the goal of the generalized RNA folding problem is to find a maximum cardinality set $A \subseteq \{(i,j) | 1 \leq i < j \leq n\}$ among all sets meeting the following conditions:

- A does not contain any crossing pair.
- For any $(i,j) \in A$, either (i) $S[i] \in \Sigma$ and $S[j] = S[i]'$ or (ii) $S[j] \in \Sigma$ and $S[i] = S[j]'$ is true.

We write $\text{RNA}(S) = |A|$.

Any set meeting the above conditions is called an *RNA folding* of S. If its cardinality equals $\text{RNA}(S)$, then it is said to be *optimal*.

In the paper we will only focus on the generalized RNA folding problem with four types of letters, i.e. $\Sigma = \{0, 1\}, \Sigma' = \{0', 1'\}$, which coincides with the RNA folding problem for alphabet $\{A, C, G, U\}$.

With a slight abuse of notation, sometimes we will write $(S[i], S[j])$ to denote a pair $(i,j) \in A$. The notation $\{\cdot, \cdot\}$ is used to indicate an unordered pair.

Longest Common Subsequence (LCS)

Given $X \in \Sigma^n$ and $Y \in \Sigma^m$, we define $\delta_{\text{LCS}}(X, Y) = n + m - 2k$, where $k =$ the length of the longest common subsequence of X and Y. It is easy to observe that $\text{RNA}(X \circ p(Y^R))$ equals the length of $\text{LCS} = (n + m - \delta_{\text{LCS}}(X, Y))/2$. In this sense, we can conceive of an LCS problem as an RNA folding problem with some structural constraint on the sequence.

In [7], a conditional lower bound for the LCS problem with $|\Sigma| = 2$ based on SETH was presented. A key technique in their approach is a function that transforms an instance of an alignment problem between two sets of sequences to an instance of the LCS problem, which is described below.

Alignments of two sets of sequences

Let $\mathbf{X} = (X_1, \ldots, X_n)$ and $\mathbf{Y} = (Y_1, \ldots, Y_m)$ be two linearly ordered sets of sequences of alphabet Σ. We assume that $n \geq m$. An *alignment* is a set $A = \{(i_1, j_1), (i_2, j_2), \ldots, (i_{|A|}, j_{|A|})\}$ with $1 \leq i_1 < i_2 < \ldots < i_{|A|} \leq n$ and $1 \leq j_1 < j_2 < \ldots < j_{|A|} \leq m$. An alignment A is called *structural* iff $|A| = m$ and $i_m = i_1 + m - 1$. That is, all sequences in $\mathbf{Y}$ are matched, and the matched positions in $\mathbf{X}$ are contiguous. The set of all alignments is denoted as $\mathcal{A}_{n,m}$, and the set of all structural alignments is denoted as $\mathcal{S}_{n,m}$.

The *cost* of an alignment A (with respect to $\mathbf{X}$ and $\mathbf{Y}$) is defined as:

$$\delta(A) = \sum_{(i,j) \in A} \delta_{\mathrm{LCS}}(X_i, Y_j) + (m - |A|) \max_{i,j} \delta_{\mathrm{LCS}}(X_i, Y_j).$$

That is, unaligned parts of $\mathbf{Y}$ are penalized by $\max_{i,j} \delta_{\mathrm{LCS}}(X_i, Y_j)$.

Given a sequence X, the *type* of X is defined as $(|X|, \sum_i X[i])$, where each letter is assumed to be a number. Note that when the alphabet is simply $\{0, 1\}$, $\sum_i X[i]$ is simply the number of occurrences of 1 in X.

The following key lemma was proved in [7] (Lemma 4.3 of [7]):

▶ **Lemma 5** ([7]). *Let $\mathbf{X} = (X_1, \ldots, X_n)$ and $\mathbf{Y} = (Y_1, \ldots, Y_m)$ be two linearly ordered sets of binary strings such that $n \geq m$, all X_i are of type $\mathcal{T}_X = (\ell_X, s_X)$, and all Y_i are of type $\mathcal{T}_Y = (\ell_Y, s_Y)$. There are two binary strings $S_X = \mathrm{GA}_X^{m,\mathcal{T}_Y}(X_1, \ldots, X_n)$, $S_Y = \mathrm{GA}_Y^{n,\mathcal{T}_X}(Y_1, \ldots, Y_m)$ and an integer C meeting the following requirements:*

- $\min_{A \in \mathcal{A}_{n,m}} \delta(A) \leq \delta_{LCS}(S_X, S_Y) - C \leq \min_{A \in \mathcal{S}_{n,m}} \delta(A)$.
- *The types of S_X, S_Y and the integer C only depend on $n, m, \mathcal{T}_X, \mathcal{T}_Y$.*
- *S_X, S_Y, and C can be calculated in time $\mathcal{O}((n+m)(\ell_X + \ell_Y))$. Hence $|S_X|$ and $|S_Y|$ are both $\mathcal{O}((n+m)(\ell_X + \ell_Y))$.*

Note that the term GA comes from the word gadget.

Intuitively, computing an optimal alignment (or an optimal structural alignment) of two sets of sequences is at least as hard as computing a longest common subsequence. The above lemma gives a reduction from the computation of a number s with $\min_{A \in \mathcal{A}_{n,m}} \delta(A) \leq s \leq \min_{A \in \mathcal{S}_{n,m}} \delta(A)$ (which can be regarded as an approximation of optimal alignments) to a single LCS instance.

In the next section, we will use the above lemma as a black box to devise two encodings, the clique node gadget $\mathrm{CNG}(t)$ and the clique list gadget $\mathrm{CLG}(t)$, for a k-clique t in a graph in such a way that we can decide whether two k-cliques t_1, t_2 form a $2k$-clique according the value of $\delta_{\mathrm{LCS}}(\mathrm{CNG}(t_1), \mathrm{CLG}(t_2))$.

When invoking the lemma, $\mathbf{X}$, $\mathbf{Y}$ are designed in such a way that we can test whether a condition is met (e.g. whether two given k-cliques form a $2k$-clique) by the value of $\min_{A \in \mathcal{A}_{n,m}} \delta(A)$. We will show that $\min_{A \in \mathcal{A}_{n,m}} \delta(A) = \min_{A \in \mathcal{S}_{n,m}} \delta(A)$ for the case we are interested in. Therefore, we can infer whether the condition we are interested in is met from the value of $\delta_{\mathrm{LCS}}(S_X, S_Y)$.

3 From Cliques to RNA Folding

The goal of this section is to prove Theorem 2.

Let $G = (V, E)$ be a graph, and let $n = |V|$. We write $\mathcal{C}_k$ to denote the set of k-cliques in G. We fix $\Sigma = \{0, 1\}$. As in [1], we will construct a sequence $S_G \in (\Sigma \cup \Sigma')^*$ such that we can decide whether G has a $3k$-clique according to the value of $\mathrm{RNA}(S_G)$.

As our framework of the construction of S_G is similar to the one in [1], we will give the building blocks for constructing S_G the same names as their analogues in [1], despite that they have different lower-level implementations.

3.1 Testing $2k$-cliques via LCS

We associate each vertex $v \in V$ a distinct integer in $\{0, 1, \ldots\ n-1\}$. Let s_v be the binary encoding of such integer with $|s_v| = \lceil \log(n) \rceil$. We define $\bar{v}$ to be the binary string resulted by replacing each 0 in s_v with 01 and replacing each 1 in s_v with 10. It is clear that for each $v \in V$, $\bar{v}$ is of type $\mathcal{T}_0 = (2\lceil \log(n) \rceil, \lceil \log(n) \rceil)$, and $\delta_{\mathrm{LCS}}(\bar{u}, \bar{v}) = 0$ iff $u = v$.

Our goal is to devise two encodings $\mathrm{CNG}(t), \mathrm{CLG}(t)$ for a k-clique t such that we can infer whether two k-cliques t_1, t_2 form a $2k$-clique from the value of $\delta_{\mathrm{LCS}}(\mathrm{CNG}(t_1), \mathrm{CLG}(t_2))$.

For each $v \in V$, the *list gadget* $\mathrm{LG}(v)$ and the *node gadget* $\mathrm{NG}(v)$ are defined as following:

- $\mathrm{LG}(v) = \mathrm{GA}_X^{1,\mathcal{T}_0}(\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_{|N(v)|}, 1^{\lceil \log(n) \rceil}0^{\lceil \log(n) \rceil}, \ldots, 1^{\lceil \log(n) \rceil}0^{\lceil \log(n) \rceil})$, where $N(v) = \{u_1, u_2, \ldots, u_{|N(v)|}\}$, and the number of occurrences of $1^{\lceil \log(n) \rceil}0^{\lceil \log(n) \rceil}$ is $n - |N(v)|$.
- $\mathrm{NG}(v) = \mathrm{GA}_Y^{n,\mathcal{T}_0}(\bar{v})$.

▶ **Lemma 6.** *There is a constant c_0, depending only on n, such that for any $v_1, v_2 \in V$, we have $\{v_1, v_2\} \in E$ iff $\delta_{LCS}(LG(v_1), NG(v_2)) = c_0 = \min_{v_1', v_2' \in V} \delta_{LCS}(LG(v_1'), NG(v_2'))$.*

Proof. We let $N(v_1) = \{u_1, u_2, \ldots, u_{|N(v_1)|}\}$.

Let $\mathbf{X} = (\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_{|N(v_1)|}, 1^{\lceil \log(n) \rceil}0^{\lceil \log(n) \rceil}, \ldots, 1^{\lceil \log(n) \rceil}0^{\lceil \log(n) \rceil})$, where the number of occurrences of $1^{\lceil \log(n) \rceil}0^{\lceil \log(n) \rceil}$ is $n - |N(v_1)|$, and let $\mathbf{Y} = (\bar{v}_2)$.

In view of Lemma 5, $\min_{A \in \mathcal{A}_{n,1}} \delta(A) \leq \delta_{\mathrm{LCS}}(\mathrm{LG}(v_1), \mathrm{NG}(v_2)) - C \leq \min_{A \in \mathcal{S}_{n,1}} \delta(A)$, for some C whose value depends on $|\mathbf{X}|, |\mathbf{Y}|$, and $\mathcal{T}_0$. As these parameters depend solely on n, the number C only depends on n.

Since $|\mathbf{Y}| = 1$, any non-empty alignment between $\mathbf{X}$ and $\mathbf{Y}$ is structural. This implies that $\delta_{\mathrm{LCS}}(\mathrm{LG}(v_1), \mathrm{NG}(v_2)) - C = \min_{A \in \mathcal{A}_{n,1}} \delta(A) = \min_{A \in \mathcal{S}_{n,1}} \delta(A)$.

When $\{v_1, v_2\} \in E$, since $\bar{v}_2$ is contained in $\mathbf{X}$, clearly $\min_{A \in \mathcal{S}_{n,m}} \delta(A) = 0$. When $\{v_1, v_2\} \notin E$, $\bar{v}_2$ does not appear in $\mathbf{X}$, so $\min_{A \in \mathcal{S}_{n,m}} \delta(A) > 0$. Note that $1^{\lceil \log(n) \rceil}0^{\lceil \log(n) \rceil} \neq \bar{v}$, for any $v \in V$.

Hence $\{v_1, v_2\} \in E$ iff $\delta_{\mathrm{LCS}}(\mathrm{LG}(v_1), \mathrm{NG}(v_2)) = C = \min_{v_1', v_2' \in V} \delta_{\mathrm{LCS}}(\mathrm{LG}(v_1'), \mathrm{NG}(v_2'))$. Therefore, it suffices to set $c_0 = C$. ◀

We let $\mathcal{T}_X$ be the type of the list gadgets, and we let $\mathcal{T}_Y$ be the type of the node gadgets. For each k-clique $t = \{u_1, u_2, \ldots, u_k\}$, we define the *clique node gadget* $\mathrm{CNG}(t)$ and the *clique list gadget* $\mathrm{CLG}(t)$ as following:

- $\mathrm{CLG}(t) = \mathrm{GA}_X^{k^2,\mathcal{T}_Y}(\mathrm{LG}(u_1), \ldots, \mathrm{LG}(u_1), \mathrm{LG}(u_2), \ldots, \mathrm{LG}(u_2), \ldots, \mathrm{LG}(u_k), \ldots, \mathrm{LG}(u_k))$, where the number of occurrences of each $\mathrm{LG}(u_i)$ is k.
- $\mathrm{CNG}(t) = \mathrm{GA}_Y^{k^2,\mathcal{T}_X}(\mathrm{NG}(u_1), \mathrm{NG}(u_2), \ldots, \mathrm{NG}(u_k), \mathrm{NG}(u_1), \mathrm{NG}(u_2), \ldots, \mathrm{NG}(u_k), \ldots,$ $\mathrm{NG}(u_1), \mathrm{NG}(u_2), \ldots, \mathrm{NG}(u_k))$, where the number of occurrences of each $\mathrm{NG}(u_i)$ is k.

We are ready to prove the main lemma in the subsection:

▶ **Lemma 7.** *There is a constant c_1, depending only on n, k, such that for any $t_1, t_2 \in \mathcal{C}_k$, $t_1 \cup t_2$ is a $2k$-clique iff $\delta_{LCS}(CLG(t_1), CNG(t_2)) = c_1 = \min_{t'_1,t'_2 \in \mathcal{C}_k} \delta_{LCS}(CLG(t'_1), CNG(t'_2))$.*

Proof. Let $t_1 = \{u_1, u_2, \dots, u_k\}$, and let $t_2 = \{v_1, v_2, \dots, v_k\}$.

Let $\mathbf{X} = (\mathrm{LG}(u_1), \dots, \mathrm{LG}(u_1), \mathrm{LG}(u_2), \dots, \mathrm{LG}(u_2), \dots, \mathrm{LG}(u_k), \dots, \mathrm{LG}(u_k))$, where each $\mathrm{LG}(u_i)$ appears k times, and let $\mathbf{Y} = (\mathrm{NG}(v_1), \mathrm{NG}(v_2), \dots, \mathrm{NG}(v_k), \mathrm{NG}(v_1), \mathrm{NG}(v_2), \dots,$ $\mathrm{NG}(v_k), \dots, \mathrm{NG}(v_1), \mathrm{NG}(v_2), \dots, \mathrm{NG}(v_k))$, where each $\mathrm{NG}(v_i)$ appears k times.

In view of Lemma 6, we have $\min_{w_1,w_2 \in V} \delta_{\mathrm{LCS}}(\mathrm{LG}(w_1), \mathrm{NG}(w_2)) \geq c_0$, so we can lower bound $\min_{A \in \mathcal{A}_{k^2,k^2}} \delta(A)$ by $k^2 c_0$.

If $\max_{i,j} \delta_{\mathrm{LCS}}(X_i, Y_j) = c_0$, any alignment has cost $k^2 c_0$. When $\max_{i,j} \delta_{\mathrm{LCS}}(X_i, Y_j) > c_0$, it is easy to observe that in order to achieve $\delta(A) = k^2 c_0$, all sequences in $\mathbf{Y}$ must be aligned (as the cost for any unaligned sequence in $\mathbf{Y}$ is now $> c_0$). Therefore, any alignment A with $\delta(A) = k^2 c_0$ must be $A = \{(i,i) | i \in \{1, 2, \dots, k^2\}\}$ with $\delta_{\mathrm{LCS}}(X_i, Y_i) = c_0$, for all $i \in \{1, 2, \dots, k^2\}$.

In view of the above, $\min_{A \in \mathcal{A}_{k^2,k^2}} \delta(A) = k^2 c_0$ iff $\delta_{\mathrm{LCS}}(X_i, Y_i) = c_0$ for all $i \in \{1, 2, \dots, k^2\}$.

Since $A = \{(i,i) | i \in \{1, 2, \dots, k^2\}\}$ is structural, $\min_{A \in \mathcal{A}_{k^2,k^2}} \delta(A) = k^2 c_0$ iff $\min_{A \in \mathcal{S}_{k^2,k^2}} \delta(A) = k^2 c_0$. Therefore, in view of Lemma 5, there exists a constant C such that:

- If $\min_{A \in \mathcal{A}_{k^2,k^2}} \delta(A) = k^2 c_0$, then $\delta_{\mathrm{LCS}}(\mathrm{CLG}(t_1), \mathrm{CNG}(t_2)) = k^2 c_0 + C$.
- If $\min_{A \in \mathcal{A}_{k^2,k^2}} \delta(A) > k^2 c_0$, then $\delta_{\mathrm{LCS}}(\mathrm{CLG}(t_1), \mathrm{CNG}(t_2)) > k^2 c_0 + C$.

Moreover, the value of C depends only on $|\mathbf{X}|, |\mathbf{Y}|, \mathcal{T}_X, \mathcal{T}_Y$. As these parameters depend solely on n, k, the number C only depends on n, k.

When $t_1 \cup t_2$ is a $2k$-clique, all vertices in t_1 are adjacent to all vertices in t_2. In view of Lemma 6, $\forall_{i,j} \delta_{\mathrm{LCS}}(X_i, Y_j) = c_0$. Hence $\min_{A \in \mathcal{A}_{k^2,k^2}} \delta(A) = k^2 c_0$, implying that $\delta_{\mathrm{LCS}}(\mathrm{CLG}(t_1), \mathrm{CNG}(t_2)) = k^2 c_0 + C$.

When $t_1 \cup t_2$ is not a $2k$-clique, there exist $u_i \in t_1, v_j \in t_2$ such that $\{u_i, v_j\} \notin E$. According to our definition of $\mathbf{X}$ and $\mathbf{Y}$, we have $X_{j+k(i-1)} = \mathrm{LG}(u_i)$, $Y_{j+k(i-1)} = \mathrm{NG}(v_j)$, and hence $\delta_{\mathrm{LCS}}(X_{j+k(i-1)}, Y_{j+k(i-1)}) > c_0$. This implies that $\min_{A \in \mathcal{A}_{k^2,k^2}} \delta(A) > k^2 c_0$, which leads to $\delta_{\mathrm{LCS}}(\mathrm{CLG}(t_1), \mathrm{CNG}(t_2)) > k^2 c_0 + C$.

As a result, $t_1 \cup t_2$ is a $2k$-clique iff $\delta_{\mathrm{LCS}}(\mathrm{CLG}(t_1), \mathrm{CNG}(t_2)) = k^2 c_0 + C = \min_{t'_1,t'_2 \in \mathcal{C}_k} \delta_{\mathrm{LCS}}(\mathrm{CLG}(t'_1), \mathrm{CNG}(t'_2))$. Setting $c_1 = k^2 c_0 + C$ suffices. ◀

▶ **Lemma 8.** *There exist four integers $\ell_{CNG,0}, \ell_{CNG,1}, \ell_{CLG,0}, \ell_{CLG,1} \in \mathcal{O}(k^2 n \log(n))$, such that for any $t \in \mathcal{C}_k$,*

- *$\ell_{CNG,b}$ = the number of occurrences of b in $CNG(t)$, $b \in \{0, 1\}$.*
- *$\ell_{CLG,b}$ = the number of occurrences of b in $CLG(t)$, $b \in \{0, 1\}$.*

Proof. As a consequence of Lemma 5, all $\mathrm{CNG}(t)$ have the same type, and all $\mathrm{CLG}(t)$ have the same type. Therefore, the existence of these four integers is guaranteed.

In view of Lemma 5, for all $v \in V$, both $\mathrm{LG}(v)$ and $\mathrm{NG}(v)$ have length at most $(n+1) \cdot (2\lceil\log(n)\rceil + 2\lceil\log(n)\rceil) = \mathcal{O}(n \log(n))$. Applying Lemma 5 again, the length of both $\mathrm{CNG}(t)$ and $\mathrm{CLG}(t)$ for all $t \in \mathcal{C}_k$ is $(k^2 + k^2)(\mathcal{O}(n \log(n)) + \mathcal{O}(n \log(n))) = \mathcal{O}(k^2 n \log(n))$.

As a result, the four integers can be bounded by $\mathcal{O}(k^2 n \log(n))$. ◀

3.2 The RNA sequence S_G

In this subsection, we define the RNA sequence S_G and show that we can decide whether G has a $3k$-clique according to $\mathrm{RNA}(S_G)$.

Based on the parameters in Lemma 8, we define $\ell_0 = \ell_{\mathrm{CNG},0} + \ell_{\mathrm{CNG},1} + \ell_{\mathrm{CLG},0} + \ell_{\mathrm{CLG},1} = \mathcal{O}(k^2 n \log(n))$; for $i \in \{1, 2, 3\}$, we set $\ell_i = 100\ell_{i-1}$; and $\ell_4 = 100|\mathcal{C}_k|\ell_3 = \mathcal{O}(k^2 n^{k+1} \log(n))$.

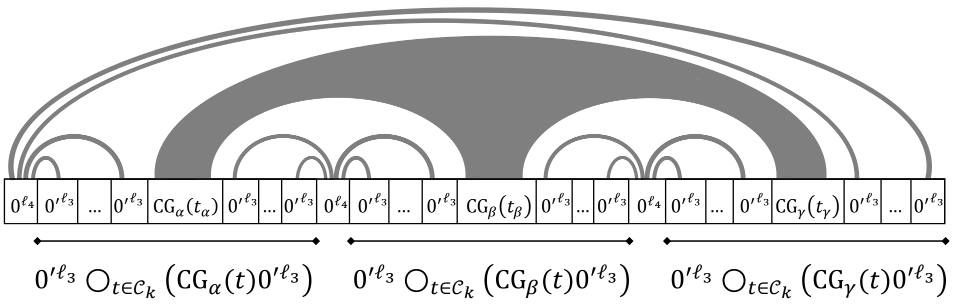
0^{ℓ_4}
$0'^{\ell_3}$
...
$CG_\alpha(t_\alpha)$
$CG_\beta(t_\beta)$
$CG_\gamma(t_\gamma)$
$0'^{\ell_3} \bigcirc_{t\in\mathcal{C}_k} (CG_\alpha(t)0'^{\ell_3})$
$0'^{\ell_3} \bigcirc_{t\in\mathcal{C}_k} (CG_\beta(t)0'^{\ell_3})$
$0'^{\ell_3} \bigcirc_{t\in\mathcal{C}_k} (CG_\gamma(t)0'^{\ell_3})$

Figure 1 The three selected clique gadgets and the matchings between $0'^{\ell_3}$ and 0^{ℓ_4}.

The RNA sequence S_G is then defined as below:

$$S_G = 0^{\ell_4}\left[0'^{\ell_3} \bigcirc_{t\in\mathcal{C}_k}\left(\mathrm{CG}_\alpha(t)0'^{\ell_3}\right)\right]0^{\ell_4}\left[0'^{\ell_3} \bigcirc_{t\in\mathcal{C}_k}\left(\mathrm{CG}_\beta(t)0'^{\ell_3}\right)\right]0^{\ell_4}\left[0'^{\ell_3} \bigcirc_{t\in\mathcal{C}_k}\left(\mathrm{CG}_\gamma(t)0'^{\ell_3}\right)\right],$$

where

$$\mathrm{CG}_\alpha(t) = 1'^{2\ell_2}p(\mathrm{CLG}(t)^R)0'^{\ell_1}1^{\ell_2}0^{\ell_1}\mathrm{CNG}(t)1^{\ell_2},$$
$$\mathrm{CG}_\beta(t) = 1'^{\ell_2}p(\mathrm{CLG}(t)^R)0'^{\ell_1}1'^{2\ell_2}0'^{\ell_1}p(\mathrm{CNG}(t))1'^{\ell_2},$$
$$\mathrm{CG}_\gamma(t) = 1^{\ell_2}\mathrm{CLG}(t)^R0^{\ell_1}1^{\ell_2}0^{\ell_1}\mathrm{CNG}(t)1^{2\ell_2}.$$

For any $t \in \mathcal{C}_k$, $x \in \{\alpha,\beta,\gamma\}$, the string $\mathrm{CG}_x(t)$ is called a *clique gadget*.
Note that $\mathrm{CG}_\alpha(t) \in (\Sigma\cup\Sigma')^*$, $\mathrm{CG}_\beta(t) \in \Sigma'^*$, and $\mathrm{CG}_\gamma(t) \in \Sigma^*$.
It is obvious that $|S_G| = \mathcal{O}(|\mathcal{C}_k|\ell_0) = \mathcal{O}(k^2n^{k+1}\log(n))$.

▶ **Lemma 9.** *$RNA(S_G) = f(n,k) - \frac{Q}{2}$, for $Q = \min_{t_\alpha,t_\beta,t_\gamma\in\mathcal{C}_k}(\delta_{LCS}(CLG(t_\alpha), CNG(t_\beta)) + \delta_{LCS}(CLG(t_\alpha), CNG(t_\gamma)) + \delta_{LCS}(CLG(t_\beta), CNG(t_\gamma)))$, and $f(n,k) = 6\ell_2 + 3\ell_1 + \frac{3}{2}\ell_0 + 3(|\mathcal{C}_k|+1)\ell_3 + (|\mathcal{C}_k|-1)(2\ell_1 + 2\ell_2 + \min(\ell_{CLG,1},\ell_{CNG,1}) + \ell_{CLG,0} + \ell_{CNG,0})$.*

Proof (Sketch). Due to the page limit, we only demonstrate an example of an RNA folding matching this bound, omitting the proof of optimality:

- We link all $0'$ in all $0'^{\ell_3}$ to some 0 in some 0^{ℓ_4} in such a way that all clique gadgets are "blocked" (a clique gadget is blocked if its letters can only link to letters in the same clique gadget or some 0 in some 0^{ℓ_4}) except $\mathrm{CG}_\alpha(t_\alpha)$, $\mathrm{CG}_\beta(t_\beta)$, and $\mathrm{CG}_\gamma(t_\gamma)$. This gives us $3(|\mathcal{C}_k|+1)\ell_3$ amount of pairs. See Fig. 1.
- For a clique gadget that is "blocked", our design of S_G ensures that the optimal number of pairs involving letters in the clique gadget is irrelevant to its corresponding k-clique:
  - For a blocked $\mathrm{CG}_\alpha(t)$, since ℓ_2 is significantly larger than ℓ_1, ℓ_0, an optimal way to pair up the letters is to match as many $\{1', 1\}$ as possible. This gives us $2\ell_2 + \min(\ell_{\mathrm{CLG},1}, \ell_{\mathrm{CNG},1})$ pairs.
  - For a blocked $\mathrm{CG}_\beta(t)$, since we do not have any 1 here, the best we can do is to match all $0'$ to some 0^{ℓ_4}. This gives us $2\ell_1 + \ell_{\mathrm{CLG},0} + \ell_{\mathrm{CNG},0}$ pairs.
  - For a blocked $\mathrm{CG}_\gamma(t)$, no matching can be made.

  The total amount of pairs involving blocked clique gadgets is $(|\mathcal{C}_k|-1)(2\ell_1 + 2\ell_2 + \min(\ell_{\mathrm{CLG},1}, \ell_{\mathrm{CNG},1}) + \ell_{\mathrm{CLG},0} + \ell_{\mathrm{CNG},0})$. See Fig. 2 for an illustration.
- For the three clique gadgets that are not blocked, the matching described in Fig. 3 has cardinality $6\ell_2+3\ell_1+\frac{1}{2}(\ell_0 - \delta_{\mathrm{LCS}}(\mathrm{CLG}(t_\alpha), \mathrm{CNG}(t_\beta)))+\frac{1}{2}(\ell_0 - \delta_{\mathrm{LCS}}(\mathrm{CLG}(t_\alpha), \mathrm{CNG}(t_\gamma)))+\frac{1}{2}(\ell_0 - \delta_{\mathrm{LCS}}(\mathrm{CLG}(t_\beta), \mathrm{CNG}(t_\gamma)))$. Recall that $\frac{1}{2}(\ell_0 - \delta_{\mathrm{LCS}}(\mathrm{CLG}(t_x), \mathrm{CNG}(t_y)))$ is the length of the LCS between $\mathrm{CLG}(t_x)$ and $\mathrm{CNG}(t_y)$. ◀

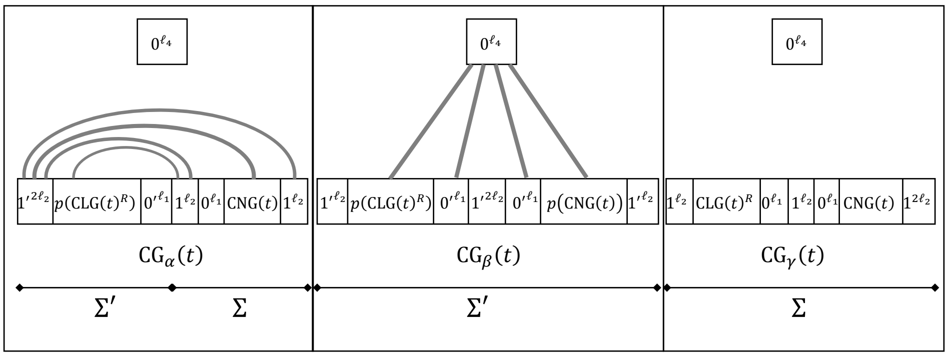
0^{ℓ_4}
0^{ℓ_4}
0^{ℓ_4}
$1'^{2\ell_2}$ | $p(\mathrm{CLG}(t)^R)$ | $0'^{\ell_1}$ | 1^{ℓ_2} | 0^{ℓ_1} | $\mathrm{CNG}(t)$ | 1^{ℓ_2}
$1'^{\ell_2}$ | $p(\mathrm{CLG}(t)^R)$ | $0'^{\ell_1}$ | $1'^{2\ell_2}$ | $0'^{\ell_1}$ | $p(\mathrm{CNG}(t))$ | $1'^{\ell_2}$
1^{ℓ_2} | $\mathrm{CLG}(t)^R$ | 0^{ℓ_1} | 1^{ℓ_2} | 0^{ℓ_1} | $\mathrm{CNG}(t)$ | $1^{2\ell_2}$
$\mathrm{CG}_\alpha(t)$
$\mathrm{CG}_\beta(t)$
$\mathrm{CG}_\gamma(t)$
Σ'
Σ
Σ'
Σ

Figure 2 The matchings between a blocked clique gadget and 0^{ℓ_4}.

By Lemma 7, there exists a number c_1 such that:

- the number c_1 depends only on n, k, and $Q \geq 3c_1$.
- If $Q = 3c_1$, then there exist $t_\alpha, t_\beta, t_\gamma \in \mathcal{C}_k$ such that $t_\alpha \cup t_\beta$, $t_\alpha \cup t_\gamma$, $t_\beta \cup t_\gamma$ are three $2k$-cliques. This implies that $t_\alpha \cup t_\beta \cup t_\gamma$ is a $3k$-clique.
- If $Q > 3c_1$, then the graph has no $3k$-clique.

Hence we can decide whether G has a $3k$-clique according to $\mathrm{RNA}(S_G)$, which can be calculated in time $T\left(\mathcal{O}\left(k^2 n^{k+1}\log(n)\right)\right) = \mathcal{O}\left(T\left(n^{k+1}\log(n)\right)\right)$ (k is a constant, and $T(\cdot)$ is the time complexity of computing optimal RNA folding). Theorem 2 is concluded.

4 Hardness of Dyck Edit Distance Problem

In this section, we shift our focus to the Dyck edit distance problem. We will present a simple reduction from RNA folding problem (with alphabet size 4) to Dyck edit distance problem (with alphabet size 10). This leads to a much simplified and improved proof for a conditional lower bound of Dyck edit distance based on the conjectured hardness k-clique. Recall that the previous proof in [1] requires 48 symbols.

Given $S \in (\Sigma \cup \Sigma')^n$, the goal of the Dyck edit distance problem is to find a minimum number of edit operations (insertion, deletion, and substitution) that transform S into a string in the Dyck context free language defined by the grammar: $\mathbf{S} \to \mathbf{SS}$, $\forall x \in \Sigma, \mathbf{S} \to x\mathbf{S}x'$, and $\mathbf{S} \to \epsilon$ (empty string).

An alternative definition of the Dyck edit distance problem is described as follows: Given a sequence $S \in (\Sigma \cup \Sigma')^n$, find a minimum cost set $A \subseteq \{(i,j) | 1 \leq i < j \leq n\}$ satisfying the following conditions:

- $A = A_M \uplus A_S$ has no crossing pair.
- A_M contains only pairs of the form (x, x'), $x \in \Sigma$ (i.e. for all $(i,j) \in A_M$, we have $S[i] = x$, $S[j] = x'$, for some $x \in \Sigma$). A_M corresponds to the set of matched pairs.
- A_S does not contain any pair of the form (y', x), $x, y \in \Sigma$ (i.e. for all $(i,j) \in A_S$ we have either $S[i] \in \Sigma$ or $S[j] \in \Sigma'$). A_S corresponds to the set of pairs that can be fixed by one substitution operation per each pair.
- Let D be the set of letters in S that do not belong to any pair in A. Each letter in D requires one deletion/insertion operation to fix.

The cost of A is then defined as $|A_S| + |D|$, and the Dyck edit distance of the string S is the cost of a minimum cost set meeting the above conditions.

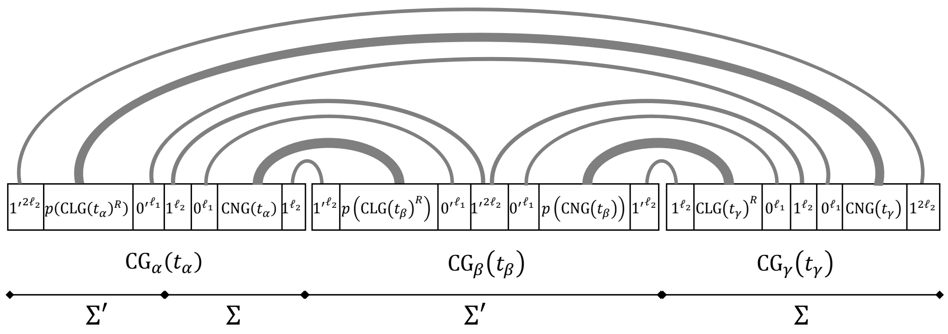

Figure 3 The matchings within the three selected clique gadgets.

We can view Dyck edit distance problem as an asymmetric version of RNA folding (both (x, x') and (x', x) are legit aligned pairs in RNA folding) that also handles substitution (in addition to deletion and insertion). Intuitively, Dyck edit distance is more complicated than RNA folding. Indeed, the same conditional lower bound as Theorem 1 for Dyck edit distance problem shown in [1] requires a bigger alphabet size (48 instead of 36) and a longer proof. In the next, we prove Theorem 3 by showing a simple reduction from RNA folding to Dyck edit distance with alphabet size 10. This improves upon the hardness result in [1], and justifies the intuition that Dyck edit distance is a harder problem than RNA folding.

Proof of Theorem 3. For notational simplicity, we let the alphabet for the RNA folding problem be $\Sigma \cup \Sigma' = \{0, 0', 1, 1'\}$ (instead of $\{A, C, G, U\}$). Let S be any string in $(\Sigma \cup \Sigma')^n$. We define the string S_{Dyck} as the result of applying the following operations on S:

- Replace each letter 0 with the sequence $S_0 = aeb'aeb'$.
- Replace each letter $0'$ with the sequence $S_{0'} = bba'a'$.
- Replace each letter 1 with the sequence $S_1 = ced'ced'$.
- Replace each letter $1'$ with the sequence $S_{1'} = ddc'c'$.

It is clear that S_{Dyck} is a sequence of length at most $6n$ on the alphabet $\{a, b, c, d, e\} \cup \{a', b', c', d', e'\}$, though the letter e' is not used. We claim that the Dyck edit distance of S_{Dyck} is $\frac{|S_{\text{Dyck}}|}{2} - 2\text{RNA}(S)$.

First, we show that the Dyck edit distance of S_{Dyck} is at most $\frac{|S_{\text{Dyck}}|}{2} - 2\text{RNA}(S)$. Given an optimal RNA folding of S, we construct a crossing-free matching A with cost $\frac{|S_{\text{Dyck}}|}{2} - 2\text{RNA}(S)$ as follows:

- For matched pairs in the RNA folding of S:
  - For each matched pair $(0, 0')$ in the RNA folding of S, we add two pairs $(a, a'), (a, a')$ to A_M, and add three pairs $(e, b'), (e, b'), (b, b)$ to A_S in its corresponding pair of substrings $(S_0 = \mathbf{a}(eb')\mathbf{a}(eb'), S_{0'} = (bb)\mathbf{a}'\mathbf{a}')$ in S_{Dyck}.
  - For each matched pair $(0', 0)$ in the RNA folding of S, we add two pairs $(b, b'), (b, b')$ to A_M, and add three pairs $(a', a'), (a, e), (a, e)$ to A_S in its corresponding pair of substrings $(S_{0'} = \mathbf{bb}(a'a'), S_0 = (ae)\mathbf{b}'(ae)\mathbf{b}')$ in S_{Dyck}.
  - Similarly, for each matched pair $(1, 1'), (1', 1)$ in the RNA folding of S, we can add two pairs to A_M and three pairs to A_S.
- For unmatched letters in S:
  - For each unmatched letter 0 in S, we add three pairs $(a, b'), (e, b'), (a, e)$ to A_S in its corresponding substring $S_0 = (a(eb')(ae)b')$. Similarly, for each unmatched letter 1, we can add three pairs to A_S.

- For each unmatched letter $0'$ in S, we add two pairs $(b, b), (a', a')$ to A_S in its corresponding substring $S_0 = (bb)(a'a')$. Similarly, for each unmatched letter $1'$, we can add two pairs to A_S.

The set A_M has size $2\text{RNA}(S)$, the set A_S has size $\frac{|S_{\text{Dyck}}|-4\text{RNA}(S)}{2}$, and D is an empty set. Therefore, the cost of A is $\frac{|S_{\text{Dyck}}|-4\text{RNA}(S)}{2} = \frac{|S_{\text{Dyck}}|}{2} - 2\text{RNA}(S)$.

Second, we show that the Dyck edit distance of S_{Dyck} is at least $\frac{|S_{\text{Dyck}}|}{2} - 2\text{RNA}(S)$. Given a crossing-free matching A (on the string S_{Dyck}) of cost C, we recover an RNA folding of S that has $\geq \frac{|S_{\text{Dyck}}|}{4} - \frac{C}{2}$ number of matched pairs.

We build a multi-graph $G = (V, E)$ such that V is the set of all substrings $S_0, S_{0'}, S_1, S_{1'}$ that constitute S_{Dyck}, and the number of edges between two substrings in V is the number of pairs in A_M linking letters between these two substrings. Note that $|V| = n, |E| = A_M$. It is clear that $C \geq \frac{|S_{\text{Dyck}}|-2|E|}{2}$, since $|A_S| + |D| \geq \frac{|S_{\text{Dyck}}|-2|A_M|}{2} = \frac{|S_{\text{Dyck}}|-2|E|}{2}$. Therefore, we are done if we can recover an RNA folding of size $\geq \frac{|E|}{2}$, since $\frac{|E|}{2} \geq \frac{|S_{\text{Dyck}}|}{4} - \frac{C}{2}$.

We observe the following:

- G has degree at most 2 (due to our definition of $S_0, S_{0'}, S_1, S_{1'}$, at most two letters in such a substring can participate in pairings of the form (x, x'), $x \in \{a, b, c, d\}$, without crossing).
- In the graph G, any edge must either links an S_0 with an $S_{0'}$ or links an S_1 with an $S_{1'}$ (due to our definition of $S_0, S_{0'}, S_1, S_{1'}$, any pairing of the form (x, x'), $x \in \{a, b, c, d\}$, must be made between $S_0, S_{0'}$ or between $S_1, S_{1'}$).
- G does not contain any cycle of odd length (due to the above observation).

In view of the above second observation, a (graph-theoretic) matching $M \subseteq E$ of G naturally corresponds to a (size $|M|$) RNA folding of S: for each edge (a pair of substrings in S_{Dyck}) in M, we add its corresponding pair of letters in S to the RNA folding. Since a maximum matching has size $\geq \frac{|E|}{2}$ in a graph of maximum degree 2 without odd cycles, we conclude the proof. ◀

We note that for the case substitution is not allowed, the letter e in the above proof is not needed, and this lowers the required alphabet size to 8.

Acknowledgements. The author thanks Seth Pettie and anonymous reviewers for helpful comments.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is Valiant's parser. In *Proceedings of 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 98–117, 2015.

2 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *Proceedings of 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 59–78, 2015.

3 Tatsuya Akutsu. Approximation and exact algorithms for RNA secondary structure prediction and recognition of stochastic context-free languages. *Journal of Combinatorial Optimization*, 3(2):321–336, 1999.

4 Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In *Proceedings of 41st International Colloquium Automata, Languages, and Programming (ICALP)*, pages 114–125, 2014.

5 Amihood Amir and Gad M. Landau. Fast parallel and serial multidimensional approximate array matching. *Theoretical Computer Science*, 81(1):97–115, 1991.

6 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 51–58, 2015.

7 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proceedings of 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 79–97, 2015.

8 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proceedings of 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 31–40, 2015.

9 Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme J. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

10 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1):57–67, 2004.

11 Yelena Frid and Dan Gusfield. A simple, practical and complete $\mathcal{O}(\frac{n^3}{\log n})$-time algorithm for RNA folding using the Four-Russians speedup. *Algorithms for Molecular Biology*, 5(1):1–8, 2010.

12 Tamar Pinhas, Dekel Tsur, Shay Zakov, and Michal Ziv-Ukelson. Edit distance with duplications and contractions revisited. In *Proceedings of 22nd Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 441–454. Springer Berlin Heidelberg, 2011.

13 Tamar Pinhas, Shay Zakov, Dekel Tsur, and Michal Ziv-Ukelson. Efficient edit distance with duplications and contractions. *Algorithms for Molecular Biology*, 8(1):1–28, 2013.

14 Mihai Pătraşcu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proceedings of 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1065–1075, 2010.

15 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of 45th ACM Symposium on Theory of Computing (STOC)*, pages 515–524, 2013.

16 Barna Saha. The Dyck language edit distance problem in near-linear time. In *Proceedings of 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 611–620, 2014.

17 Barna Saha. Language edit distance and maximum likelihood parsing of stochastic grammars: Faster algorithms and connection to fundamental graph problems. In *Proceedings of 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 118–135, 2015.

18 Yinglei Song. Time and space efficient algorithms for RNA folding with the Four-Russians technique. Technical Report arXiv:1503.05670, 2015.

19 Leslie G. Valiant. General context-free recognition in less than cubic time. *Journal of Computer and System Sciences*, 10(2):308–315, 1975.

20 Virginia Vassilevska. Efficient algorithms for clique problems. *Information Processing Letters*, 109(4):254–257, 2009.

21 Balaji Venkatachalam, Dan Gusfield, and Yelena Frid. Faster algorithms for RNA-folding using the Four-Russians method. *Algorithms for Molecular Biology*, 9(1):1–12, 2014.