ELSEVIER

Theoretical Computer Science 276 (2002) 111–132

Theoretical Computer Science

www.elsevier.com/locate/tcs

Computing similarity between RNA structures

Bin Ma[a], Lusheng Wang[b,*], Kaizhong Zhang[c]

[a] *Department of Mathematics, Peking University, Beijing 100871, People's Republic of China*
[b] *Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong*
[c] *Department of Computer Science, University of Western Ontario, London, Ont. Canada N6A 5B7*

Received January 2000; accepted March 2001
Communicated by D.-Z. Du

Abstract

The primary structure of a ribonucleic acid (RNA) molecule is a sequence of nucleotides (bases) over the four-letter alphabet $\{A, C, G, U\}$. The secondary or tertiary structure of an RNA is a set of base-pairs (nucleotide pairs) which forms bonds between $A-U$ and $C-G$. For secondary structures, these bonds have been traditionally assumed to be one to one and non-crossing. This paper considers a notion of similarity between two RNA molecule structures taking into account the primary, the secondary and the tertiary structures. We show that, for tertiary structures, it is Max SNP-hard for both minimization and maximization versions. We show a stronger result for the maximization version where it cannot be approximated within ratio $2^{\log^{\delta} n}$ in polynomial time, unless NP $\subseteq$ DTIME$[2^{\text{poly log}n}]$. We then present an algorithm that can be used for practical application. Our algorithm will produce an optimal solution for the case where at least one of the RNA involved is of a secondary structure. We also show an approximation algorithm.

Keywords: Molecular biology; RNA structures; Similarity; Inapproximability

1. Introduction

Ribonucleic acid (RNA) is an important molecule which performs a wide range of functions in the biological system. In particular, it is RNA (not DNA) that contains genetic information of virus such as HIV and therefore regulates the functions of such virus. RNA has recently become the center of much attention because of its catalytic properties, leading to an increased interest in obtaining structural information.

* Corresponding author.
E-mail addresses: bma@sxx0.math.pku.edu.cn (B. Ma), lwang@cs.cityu.edu.hk (L. Wang), kzhang@csd.uwo.ca (K. Zhang).

0304-3975/02/$ - see front matter
PII: S0304-3975(01)00192-X

It is well known that secondary and tertiary structural features of RNAs are important in the molecular mechanism involving their functions. The presumption, of course, is that to a preserved function there corresponds a preserved molecular confirmation and, therefore, a preserved secondary and tertiary structure. Therefore the ability to compare RNA structures is useful.

In RNA secondary or tertiary structure, a bonded pair of bases (base-pair) is usually represented as an edge between the two complementary bases involved in the bond. It is assumed that any base participates in at most one such pair. For the secondary structure, the edges of the bonded pairs are non-crossing.

Following the notion of similarity in comparing sequences, we define a similarity between two RNA molecule structures taking into account the primary, the secondary and the tertiary structures.

Results. We show that computing this similarity between RNA tertiary structures is Max SNP-hard. This means that there is no polynomial time approximation scheme (PTAS) for this problem unless $P = NP$. For the maximization version, we show that it cannot be approximated within ratio $2^{\log^{\delta} n}$ in polynomial time, unless $NP \subseteq DTIME[2^{\text{poly } \log n}]$. We present an algorithm for the case where at least one of the RNA involved is of a secondary structure. Our algorithm can be extended to handle simple tertiary interactions known as H-type pseudo-knots. We then show that this algorithm could be used to compare tertiary structures in practical application. Finally, we will give an approximation algorithm.

Related work. Since the secondary structure appears as a tree-like structure, there are works considering comparisons using tree comparisons [9, 5, 6, 10, 4]. However, these methods do not directly use base-paired nucleotides and unpaired nucleotides. Instead loops and stems (stacked pairs) are used as the basic unit making it difficult to define the semantic meaning in the process of converting one RNA into another. To overcome this difficulty, we proposed a method [14] which defines some basic operations directly on base-paired and unpaired nucleotides and then use these operations to define the similarity measure. In this paper we extend this method from secondary structures to tertiary structures.

Another line of works are primary structure based where the comparison is basically done on the primary structure while trying to incorporate secondary structure data [1, 2]. The weakness of this approach is that it does not give a clear definition on how to treat base-pairs. For example, in the comparison of two RNAs, a base-pair from one RNA can be considered as a whole entity by matching it to a base-pair or it can be considered as two single bases by matching them to two bases (unpaired or even paired) in the other RNA. Our method treats base-pair as a unit, it can be matched to another base-pair, it can be deleted, or it can be inserted. This is closer to the spirit of the comparative analysis method currently being used in the analysis of RNA secondary structures either manually or automatically.

2. Comparing two RNA structures

2.1. RNA structures and basic operations

The primary structure of a ribonucleic acid (RNA) molecule is a sequence of nucleotides (bases) over the four-letter alphabet $\sum = \{A, C, G, U\}$. The secondary or tertiary structure of an RNA is a set of base-pairs (nucleotide pairs) which formed bonds between $A-U$ and $C-G$. Following Zuker [16–18], we assume a model where there are no knots in the secondary structure. This means that for the secondary structure, the bonds are non-crossing. For the tertiary structure, there is no restriction of non-crossing.

Given an RNA structure R, we use $R[i]$ to represent the ith nucleotide of R. We use $R[i..j]$ to represent the sequence of nucleotides from $R[i]$ to $R[j]$.

We use $S(R)$ to represent the set of structural elements consisting of both its set of base-pairs and the remaining unpaired nucleotides.

$$S(R) = \{(i,j) \mid i < j \text{ and } (R[i], R[j]) \text{ is a base pair in } R\}$$
$$\cup \{(i,i) \mid R[i] \text{ is not involved in any base pair in } R\}.$$

We use $S(R)[i..j]$ to represent the set of structural elements in sequence $R[i..j]$.

$$S(R)[i..j] = \{r \mid r = (k,l) \in S(R),\ i \leqslant k, l \leqslant j\}.$$

For $r = (i,j) \in S(R)$, we use $label_R(r)$ to represent the label of r in R. If $i = j$, then $label_R(r) = R[i] = R[j]$, otherwise $label_R(r) = R[i]R[j]$. For $r = (i,j) \in S(R)$, i and j are often called the $5'$ end and $3'$ end of r, respectively. We define $left(r) = i$ and $right(r) = j$.

Following the tradition in sequence comparison [7, 11, 12], we define three operations, relabel, delete, and insert, on RNA structures. For a given RNA structure R, each operation can be applied to either a base-pair in $S(R)$ or an unpaired base. Relabelling a base-pair is to replace one base-pair in $S(R)$ with another. This means that at the sequence level, two bases may be changed at the same time. Deleting a base-pair is to delete the pair from $S(R)$. At the sequence level, this means deleting two bases at the same time. Inserting a base-pair is to insert a new base-pair into $S(R)$. At the sequence level, this means inserting two bases at the same time. Relabelling an unpaired base is to replace it with another base. Deleting an unpaired base is to delete the base from the sequence. Inserting a base is to insert a new base into the sequence as an unpaired base. Note that there is no relabel operation that can change a base-pair to an unpaired base or vice versa.

Following [13, 15], we represent an edit operation as $a \to b$, where a and b are either λ or labels of base-pair from $\{A, C, G, U\} \times \{A, C, G, U\}$, or unpaired base from $\{A, C, G, U\}$.

We call $a \to b$ a change operation if $a \neq \lambda$ and $b \neq \lambda$; a delete operation if $b = \lambda$; and an insert operation if $a = \lambda$.

Let S be a sequence $s_1,\ldots,s_k$ of edit operations. An S-derivation from RNA structure A to RNA structure B is a sequence of RNA structures $A_0,\ldots,A_k$ such that $A=A_0$, $B=A_k$, and $A_{i-1}\to A_i$ via s_i for $1\leqslant i\leqslant k$.

Let γ be a cost function which assigns to each edit operation $a\to b$ a non-negative real number $\gamma(a\to b)$. We constrain γ to be a distance metric. That is, (i) $\gamma(a\to b)\geqslant 0$, $\gamma(a\to a)=0$; (ii) $\gamma(a\to b)=\gamma(b\to a)$; and (iii) $\gamma(a\to c)\leqslant\gamma(a\to b)+\gamma(b\to c)$.

We extend γ to a sequence of edit operations S by letting $\gamma(S)=\sum_{i=1}^{|S|}\gamma(s_i)$.

The *edit distance* between two RNA structures is defined by considering the minimum cost edit operation sequence that transforms one structure to the other. Formally, the edit distance between R_1 and R_2 is defined as

$D(R_1,R_2)=\min_S\{\gamma(T)\mid T$ is an edit operation sequence taking $S(R_1)$ to $S(R_2)\}$.

2.2. Mapping between RNA structures

Let $r=(r_l,r_r)$ and $s=(s_l,s_r)$ be two elements in $S(R)$ of an RNA R, we define the relation between r and s as follows. We say r is *before* s if $r_r<s_l$. We say r is *inside* s if $s_l<r_l$ and $r_r<s_r$. We say r is *cross-before* s if $r_l<s_l<r_r<s_r$.

Let R_1 and R_2 be two RNA structures. Formally, we define a triple (M,R_1,R_2) to be a mapping from R_1 to R_2, where M is a binary relation on $S(R_1)\times S(R_2)$ such that

(1) For any (r,s) in M,
    r is a base-pair in R_1 if and only if s is a base-pair in R_2.
(2) For any pair of (r_1,s_1) and (r_2,s_2) in M,
    (a) $r_1=r_2$ if and only if $s_1=s_2$ (one-to-one)
    (b) r_1 is *before* r_2 if and only if s_1 is *before* s_2.
    (c) r_1 is *inside* r_2 if and only if s_1 is *inside* s_2.
    (d) r_1 is *cross_before* r_2 if and only if s_1 is *cross_before* s_2.

We will use M instead of (M,R_1,R_2) if there is no confusion. Let M be a mapping from R_1 to R_2. Then we can similarly define the cost of M:

$$\gamma(M)=\sum_{(r,s)\in M}\gamma(label_{R_1}(r)\to label_{R_2}(s))+\sum_{r\notin M}\gamma(label_{R_1}(r)\to\lambda)$$

$$+\sum_{s\notin M}\gamma(\lambda\to label_{R_2}(s)).$$

Mappings can be composed. Let M_1 be a mapping from R_1 to R_2 and M_2 be a mapping from R_2 to R_3. Define

$$M_1\circ M_2=\{(r,t)\mid\exists s\ s.t.\ (r,s)\in M_1\ and\ (s,t)\in M_2\}.$$

Lemma 1. (1) *$M_1\circ M_2$ is a mapping between R_1 and R_3.* (2) *$\gamma(M_1\circ M_2)\leqslant\gamma(M_1)+\gamma(M_2)$.*

Proof. (1) follows from the definition of mapping. Let us check condition (2) only. Suppose that (r_1,t_1) and (r_2,t_2) are in $M_1\circ M_2$, by definition of mapping, there exist s_1

and s_2 such that (r_1,s_1) and (r_2,s_2) are in M_1 and (s_1,t_1) and (s_2,t_2) are in M_2. If r_1 is before r_2, then by the definition of mapping, s_1 is before s_2. Therefore, t_1 is before t_2, again by the definition of mapping. Similarly if r_1 is inside r_2 or r_1 is cross-before r_2, then if t_1 is inside t_2 or t_1 is cross-before t_2.

(2) Let M_1 be the mapping from R_1 to R_2, M_2 be the mapping from R_2 to R_3, and $M_1 \circ M_2$ be the composed mapping from R_1 to R_3. Three general situations occur. $(r,s) \in M_1 \circ M_2$, $r \notin M_1$, or $s \notin M_2$. In each case this corresponds to an edit operation $\gamma(x \to y)$ where x and y may be labels or may be λ. In all such cases, the triangle inequality on the distance metric γ ensures that $\gamma(x \to y) \leqslant \gamma(x \to z) + \gamma(z \to y)$. □

The relation between a mapping and a sequence of edit operations is as follows:

Lemma 2. *Given S, a sequence $s_1,\ldots,s_k$ of edit operations from R_1 to R_2, there exists a mapping M from R_1 to R_2 such that $\gamma(M) \leqslant \gamma(S)$. Conversely, for any mapping M_e, there exists a sequence of edit operations such that $\gamma(S) = \gamma(M)$.*

Proof. The first part can be proved by induction on k. The base case is $k = 1$. This case holds because any single edit operation preserves the mapping conditions. In a general case, let S_1 be the sequence $s_1,\ldots,s_{k-1}$ of edit operations. There exist a mapping M_1 such that $\gamma(M_1) \leqslant \gamma(S_1)$. Let M_2 be the mapping for s_k. From Lemma 1, we have $\gamma(M_1 \circ M_2) \leqslant \gamma(M_1) + \gamma(M_2) \leqslant \gamma(S)$. □

Based on the lemma, the following theorem states the relation between the distance and the mappings.

Theorem 1. $D(R_1,R_2) = \min_M \{\gamma(M) \mid M \text{ is a mapping from } R_1 \text{ to } R_2\}$.

Proof. Immediately from Lemma 2. □

3. Inapproximability

We now consider the problem of comparing RNA structures where both structures are tertiary structures. We first show that it is Max SNP-hard for both minimization and maximization versions. Using a technique in [3], we show that the maximization version cannot be approximated within ratio $2^{\log^{\delta} n}$ in polynomial time, unless $\mathrm{NP} \subseteq \mathrm{DTIME}[2^{\mathrm{poly}\ \log n}]$.

Theorem 2. *The problem of calculating $D(R_1,R_2)$ is* Max-SNP *hard.*

Proof. We give an L-reduction from Max-Cut to the minimization version of the problem, i.e., the edit distance between two RNA structures. Max-Cut is Max-SNP hard even when the degrees of the vertices in the graph are bounded by 3 [8]. Suppose that we are given a graph $G = \langle V,E \rangle$, where $|V| = \{v_1,v_2,\ldots,v_n\}$, $|E| = m$, and $deg(v) \leqslant 3$

for any $v \in V$. Without loss of generality, we assume that the graph G contains one component.

Let K be the number of edges for a maximum cut of G. Now, we construct two structures R_1 and R_2 as follows:

(1) For any $v_i \in V$, we have two pieces $l(p_i)$ and $r(p_i)$ in R_1 and four pieces $l(q_i)$, $l(q_i')$, $r(q_i)$ and $r(q_i')$ in R_2, where $l(p_i) = a^{24}g^{deg(v_i)}$, $r(p_i) = u^{24}g^{deg(v_i)}$, $l(q_i) = a^{24}g^{deg(v_i)}$, $l(q_i') = u^{24}$, $r(q_i) = u^{24}$, $r(q_i') = u^{24}g^{deg(v_i)}$ and $deg(v_i)$ is the degree of node v_i in G that is bounded by 3.

(2) In R_1, the 24 a's in $l(p_i)$ and the 24 u's in $r(p_i)$ form 24 nested base-pairs. Besides, there are 3 nested $(\mathtt{z},\mathtt{z})$ pairs that are cross-before the 24 nested $(\mathtt{a},\mathtt{u})$ pairs. Moreover, the 3 nested $(\mathtt{z},\mathtt{z})$ pairs are nested with other $(\mathtt{a},\mathtt{u})$ pairs in R_1.

(3) In R_2, the 24 a's in $l(q_i)$ and the 24 u's in $r(q_i)$ form 24 nested base-pairs and symmetrically, the 24 a's in $l(q_i')$ and the 24 u's in $r(q_i')$ form 24 nested base-pairs. Besides, there are 3 nested $(\mathtt{z},\mathtt{z})$ pairs that are cross-before the 48 nested $(\mathtt{a},\mathtt{u})$ pairs. Moreover, the 3 nested $(\mathtt{z},\mathtt{z})$ pairs are nested with other $(\mathtt{a},\mathtt{u})$ pairs in R_2.

(4) Organize the node in V in an order

$$v_1 v_2 \ldots v_n$$

such that v_i is adjacent to neither v_{i-1} nor v_{i+1} in G. Such an order does exist when n is big enough, say, $n \geqslant 36$. In this case, we can find an independent set for G with size 9 by selecting a node, deleting at most 3 nodes, and repeating this process, since each node v in G is adjacent to at most 3 nodes. We order the 9 nodes of the independent set as $v_1, v_2, \ldots, v_9$. Call the first 9 nodes the *good part*. Given an order with $v_1, v_2, \ldots, v_9$ as the first 9 nodes, for each v in V, we can find two consecutive v_i and v_{i+1} with v_i and v_{i+1} in the good part such that v is adjacent to neither v_i nor v_{i+1} (there are at most 6 pairs in the good part that do not satisfy this condition and there are at least 9 nodes in the good part) and insert v between v_i and v_{i+1} in the sequence. Now the good part contains one more node. Repeat the process at most $n-9$ times, we can get a satisfying order.

(5) Those $l(p_i)$ and $r(p_i)$ in R_1 are organized in the order

$$l(p_1)l(p_2)\cdots l(p_n)r(p_n)r(p_{n-1})\cdots r(p_1).$$

(6) Those $l(q_i)$, $l(q_i')$, $r(q_i)$ and $r(q_i')$ are organized in the order

$$l(q_1)l(q_1')l(q_2)l(q_2')\cdots l(q_n)l(q_n')r(q_n')r(q_n)r(q_{n-1}')r(q_{n-1})\cdots r(q_1')r(q_1).$$

(7) For any $e = \langle v_i, v_j \rangle \in E$, we arbitrarily choose an unpaired g from each of $l(p_i)$ and $r(p_j)$ and each of $r(p_i)$ and $l(p_j)$, pair them up. At the same positions where we choose g from $l(p_i)$, $r(p_j)$, $r(p_i)$ and $l(p_j)$, we pair the two g of $l(q_i)$ and $r(q_j')$, and pair the two g of $r(q_i')$ and $l(q_j)$. (Note that we use $(\mathtt{g},\mathtt{g})$ for convenience, for real RNA's we can use $(\mathtt{a},\mathtt{u})$.

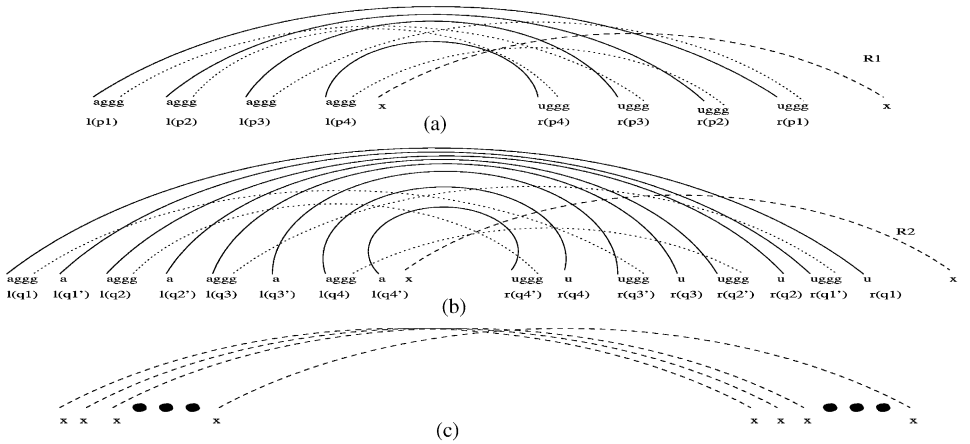

Fig. 1. The construction of R_1 and R_2. The dotted lines correspond to the edges in G. Each solid line represents 24 nested $(\mathtt{a},\mathtt{u})$ pairs. Each dashed line represents $216n$ twisted $(\mathtt{x},\mathtt{x})$ pairs in (a) and (b). The $(\mathtt{z},\mathtt{z})$ pairs are omitted in (a) and (b) for clarity of the figure. (c) shows the $216n$ twisted $(\mathtt{x},\mathtt{x})$ pairs that can be viewed as a super base-pair to replace the dashed pairs in (a) and (b).

(8) Finally, in both R_1 and R_2 we add $216n$ twisted $(\mathtt{x},\mathtt{x})$ pairs in Fig. 1. (The $(\mathtt{x},\mathtt{x})$ pairs are not necessary for the Max SNP-hard proof. They are crucial for the proof of Theorem 4.)

Now, we assume that a match between any pair of base-pairs costs 0 and an insertion/deletion of a base-pair costs 1. Note that under this cost scheme, the labels of the pairs do not affect the cost. This is identical to relabelling all pairs in the RNA structures with same labels, say, $(\mathtt{a},\mathtt{u})$. However, in the proofs throughout the section, we still use different labels for convenience.

Lemma 4. *Given a mapping between R_1 and R_2, we can construct in polynomial time another mapping with equal or smaller cost with the following properties*:

(1) *all* $(\mathtt{x},\mathtt{x})$ *pairs in* R_1 *match all* $(\mathtt{x},\mathtt{x})$ *pairs in* R_2.
(2) *all* $(\mathtt{a},\mathtt{u})/(\mathtt{z},\mathtt{z})$ *pairs in* R_1 *are in the mapping that map to* $(\mathtt{a},\mathtt{u})/(\mathtt{z},\mathtt{z})$ *pairs.*
(3) $l(p_i)$ *is mapped to either* $l(q_i)$ *or* $l(q_i')$ *and correspondingly* $r(p_i)$ *is mapped to either* $r(q_i)$ *or* $r(q_i')$.
(4) *if* $l(p_i)$ *is mapped to* $l(q_i)$, *then* $r(p_i)$ *is mapped to* $r(q_i)$; *if* $l(p_i)$ *is mapped to* $l(q_i')$, *then* $r(p_i)$ *is mapped to* $r(q_i')$.
(5) *if* $l(p_i)$ *and* $l(p_j)$ *are mapped to* $l(q_i)$ *and* $l(q_j')$ *or* $l(q_i')$ *and* $l(q_j)$, *then one of the* $(\mathtt{g},\mathtt{g})$ *pairs for* v_i *and* v_j *is in the mapping.*

Proof. First, we want to show that given a mapping M, we can modify M in polynomial time without increasing the cost such that all $(\mathtt{x},\mathtt{x})$ pairs in R_1 match all $(\mathtt{x},\mathtt{x})$ pairs in R_2.

Note that we have $216n$ $(\mathtt{x},\mathtt{x})$ pairs in both R_1 and R_2. If no $(\mathtt{x},\mathtt{x})$ pair matches $(\mathtt{x},\mathtt{x})$ pair in the given mapping, then the total number of matched pairs in the mapping

is at most $24n + 24n + 6n + 3n$. Thus, we can simply match $216n$ $(\mathtt{x},\mathtt{x})$ pairs in both R_1 and R_2. This will not increase the cost. If one $(\mathtt{x},\mathtt{x})$ pair matches another $(\mathtt{x},\mathtt{x})$ pair in the mapping, we can simply match all the $216n$ $(\mathtt{x},\mathtt{x})$ pairs in both R_1 and R_2.

The $(\mathtt{z},\mathtt{z})$ pairs in the construction ensure that for the two $(\mathtt{g},\mathtt{g})$ pairs of edge (v_i, v_j), at most one $(\mathtt{g},\mathtt{g})$ pair can be in a mapping if the $(\mathtt{a},\mathtt{u})$ pairs in both $l(p_i)$ and $l(p_j)$ are mapped to $(\mathtt{a},\mathtt{u})$ pairs in the mapping. Again, from the construction, if a $(\mathtt{g},\mathtt{g})$ pair between v_i and v_j is in the mapping and the $(\mathtt{a},\mathtt{u})$ pairs in both $l(p_i)$ and $l(p_j)$ are mapped to $(\mathtt{a},\mathtt{u})$ pairs in the mapping, then $l(p_i)$ and $l(p_j)$ are mapped to either $l(q_k)$ and $l(q'_l)$ or $l(q'_k)$ and $l(q_l)$, and vice versa.

Let P be a set of $l(p_i)$ such that their $(\mathtt{a},\mathtt{u})$ pairs are matched to $(\mathtt{a},\mathtt{u})$ pairs in the mapping. Each $l(p_i)$ in P is mapped to either a $l(q_k)$ or a $l(q'_k)$. Now we construct a new mapping as follows: if $l(p_i)$ was mapped to some $l(q_k)$, we map $l(p_i)$ to $l(q_i)$; if $l(p_i)$ was mapped to some $l(q'_k)$, we map $l(p_i)$ to $l(q'_i)$. And of course we map all the 24 $(\mathtt{a},\mathtt{u})$ pairs between $l(p_i)$ and $r(p_i)$. For the $l(p_i)$ not in P, we simply match $l(p_i)$ to $l(q_i)$.

Now, we want to show that the number of mapped pairs is not reduced in the new mapping.

For any $(\mathtt{g},\mathtt{g})$ pair between v_i and v_j in P, if $l(p_i)$ and $l(p_j)$ are mapped to either $l(q_i)$ and $l(q'_j)$ or $l(q'_i)$ and $l(q_j)$, we can add it into the new mapping. For any $(\mathtt{g},\mathtt{g})$ pair between v_i and v_j in P which was in the old mapping, then $l(p_i)$ and $l(p_j)$ are mapped to either $l(q_k)$ and $l(q'_l)$ or $l(q'_k)$ and $l(q_l)$ for some k and l. This means that they are now mapped to $l(q_i)$ and $l(q'_j)$ or $l(q'_i)$ and $l(q_j)$. Therefore this $(\mathtt{g},\mathtt{g})$ pair is now in the new mapping.

Now, consider those $(\mathtt{g},\mathtt{g})$ pairs that are mapped to $(\mathtt{a},\mathtt{u})$ pairs in the given mapping. Note that we have ordered the nodes in V so that the two $(\mathtt{g},\mathtt{g})$ pairs for (v_i, v_j) $(i<j)$ in R_1 are crossing either the $(\mathtt{a},\mathtt{u})$ pairs for $l(p_{i-1})$ or the $(\mathtt{a},\mathtt{u})$ pairs for $l(p_{i+1})$ in R_1. Since all $(\mathtt{a},\mathtt{u})$ pair in both R_1 and R_2 are nested, if a $(\mathtt{g},\mathtt{g})$ pair for (v_i, v_j) $(i<j)$ in R_1 is mapped to some $(\mathtt{a},\mathtt{u})$ pair in R_2 in the given mapping, then either no $(\mathtt{a},\mathtt{u})$ pair for $l(p_{i-1})$ can match $(\mathtt{a},\mathtt{u})$ pair in R_2, or no $(\mathtt{a},\mathtt{u})$ pair for $l(p_{i+1})$ can match $(\mathtt{a},\mathtt{u})$ pair in R_2. The condition for the $(\mathtt{g},\mathtt{g})$ pairs in R_2 is similar.

Let p_1 be the number of $(\mathtt{g},\mathtt{g})$ pairs in R_1 that are mapped to $(\mathtt{a},\mathtt{u})$ pairs in R_2 and p_2 be the number of $(\mathtt{g},\mathtt{g})$ pairs in R_2 that are mapped to $(\mathtt{a},\mathtt{u})$ pairs in R_1. Let $p = p_1 + p_2$. Then there are at least $p/(2\times 6)$ $l(p_i)$ (or $l(q_i)$ and $l(q'_i)$) such that the $(\mathtt{a},\mathtt{u})$ pairs for those $p/(2\times 6)$ $l(p_i)$ (or $l(q_i)$ and $l(q'_i)$) cannot match any $(\mathtt{a},\mathtt{u})$ pairs in R_2 (or R_1). The reason is that each $l(p_i)$ (or $l(q_i)$ and $l(q'_i)$) has at most 6 (g,g) pairs in R_1 (or R_2) and each $l(p_i)$ (or $l(q_i)$ and $l(q'_i)$) might be counted for $(\mathtt{g},\mathtt{g})$ pairs from both $l(p_{i-1})$ (or $l(q_{i-1})$ and $l(q'_{i-1})$) and $l(p_{i+1})$ (or $l(q_{i+1})$ and $l(q'_{i+1})$).

In the new mapping, we have at least $p/(2\times 6)$ $l(p_i)$ that are newly mapped to $l(q_i)$, each contains 24 nested $(\mathtt{a},\mathtt{u})$–$(\mathtt{a},\mathtt{u})$ pairs. Thus, we have $2p$ new matched pairs in the new mapping that can compensate the p $(\mathtt{g},\mathtt{g})$–$(\mathtt{a},\mathtt{u})$ pairs plus possibly $6\times p/(2\times 6)$ $(\mathtt{g},\mathtt{g})$–$(\mathtt{g},\mathtt{g})$ pairs corresponding to those $p/(2\times 6)$ $l(p_i)$ in the old mapping. Hence by this construction, we get a better mapping.

Similarly, we can show that $(\mathtt{g},\mathtt{g})$ pairs cannot match any $(\mathtt{z},\mathtt{z})$ pairs in the mapping.

Finally, it is easy to see that if $(\mathtt{a},\mathtt{u})$ pairs match some $(\mathtt{z},\mathtt{z})$ pairs, we can get better mapping by matching $(\mathtt{a},\mathtt{u})$ pairs with $(\mathtt{a},\mathtt{u})$ pairs and $(\mathtt{z},\mathtt{z})$ pairs with $(\mathtt{z},\mathtt{z})$ pairs. □

Lemma 4. *Let M be a mapping satisfying properties in Lemma* 3, *V' be the set of v_i's such that $l(p_i)$ is mapped to $l(q_i)$ and V'' be the set of v_i's such that $l(p_i)$ is mapped to $l(q_i')$. We have $\gamma(M)=24n+4m-2k$, where k is the number of edges between V' and V''.*

Proof. Since only one of $l(q_i)$ and $l(q_i')$ is mapped, we need to delete $24n$ $(\mathtt{a},\mathtt{u})$ pairs from R_2. There are $2m$ $(\mathtt{g},\mathtt{g})$ pairs in R_1 and there are $2m$ $(\mathtt{g},\mathtt{g})$ pairs in R_2. For each edge between V' and V'', there is a $(\mathtt{g},\mathtt{g})$–$(\mathtt{g},\mathtt{g})$ match in the mapping, whereas no other $(\mathtt{g},\mathtt{g})$ pairs can be in the mapping. Therefore we need to delete $4m-2k$ $(\mathtt{g},\mathtt{g})$ pairs. Since all the other pairs are in the mapping, we have $\gamma(M)=24n+4m-2k$. □

From Lemmas 3 and 4, it is clear that $D(R_1,R_2)=24n+4m-2K$.

We only have to verify that our construction satisfies the two conditions for L-reductions [3]. Since $K\geqslant n/2$ and $m\leqslant 3n/2$, we have

$$d_{\mathrm{opt}} \leqslant 24n+4m-2K \leqslant 60n/2-2K \leqslant 58K.$$

Given a mapping with cost d, from Lemmas 3 and 4, we can construct, in polynomial time, a new mapping with cost $24n+4m-2k\leqslant d$. Moreover, this new mapping gives us a cut of the graph G with value k. Therefore, we have,

$$d-d_{\mathrm{opt}} \geqslant 24n+4m-2k-(24n+4m-2K)=2(K-k).$$

Thus, our reduction is an L-reduction. This completes the proof.

Now, we consider the maximization version of the problem. Assume that a matching between any two pairs contributes cost by 1, and any other matching contributes cost by 0. We want to find a mapping with the *maximum* cost. We use $m(R_1,R_2)$ to denote the cost of the optimal mapping between R_1 and R_2.

Theorem 3. *The maximization version of the problem is also* Max SNP-*hard.*

Proof. From the proof of Theorem 2, it is easy to see that for the same construction, the cost of the mapping is $216n+24n+3n+k$ if there is a cut of size k in G and vice versa. Since G is connected and the degree of each node in V is bounded by 3, the size of the maximum cut is $\mathrm{O}(n)$. Thus, it is easy to verify that the same construction is also an L-reduction for the maximization version. □

Now, we use Theorem 3 to prove a stronger hardness result for the maximization version of the problem.

Theorem 4. *For any constant $\delta<1$, the maximization version of the problem cannot be approximated within ratio $2^{\log^{\delta} n}$ in polynomial time, unless* $\mathrm{NP}\subseteq\mathrm{DTIME}[2^{\mathrm{poly}\log n}]$.

Proof. Let R_1 and R_2 be the structures constructed in the proof of Theorem 2. Call $zzzl(p_1)zzzl(p_2)\cdots zzzl(p_n)x^{216n}$ the *left segment* of R_1 and $zzzr(p_n)\ zzzr(p_{n-1})\cdots zzzr(p_1)x^{216n}$ the *right segment* of R_1. Symmetrically, call $zzzl(q_1)l(q_1')\ zzzl(q_2)l(q_2')\cdots zzzl(q_n)l(q_n')x^{216n}$ the *left segment* of R_2 and $zzzr(q_n')r(q_n)\ zzzr(q_{n-1}')r(q_{n-1})\cdots zzzr(q_1')r(q_1)x^{216n}$ the *right segment* of R_2. Let $(\mathtt{t_i},\mathtt{t_j})$ be a pair in $R\in\{R_1,R_2\}$ such that t_i and t_j are the ith letter and jth letter in R and $i<j$. From the construction, t_i is in the left segment and t_j is in the right segment. Let R' be a structure identical to R. Note that R' can be viewed as a *super* base-pair with one base as the left segment and another base as the right segment. The product R^2 $(l=1,2)$ is a structure obtained from R by substituting each pair $(\mathtt{t_i},\mathtt{t_j})$ in R with a structure R'. In other words, we insert the left segment of R' between t_i and t_{i+1} and the right segment of R' between t_{j-1} and t_j for every pair $(\mathtt{t_i},\mathtt{t_j})$ in R, keep the pairs in all R' (one R' for each base-pair $(\mathtt{t_i},\mathtt{t_j})$ in R) and delete the original bases and base-pairs in R. From the proof of Theorem 2, the labels on nodes and base-pairs in the structures do not matter. $R^k=R\times R^{k-1}$ is obtained from R by substituting each base-pair $(\mathtt{t_i},\mathtt{t_j})$ in R with the structure R^{k-1}.

A *k-level structure* for R^{k+1} is a substructure of R^{k+1} that is identical to R^k. Replacing each pair in R by R^k, one can obtain R^{k+1}. Here R is called the *outer structure* of R^{k+1}.

Claim 5. *In R_1^{k+1}, if $q\geqslant 2$ k-level structures match one k-level structure in the opposite structure, the cost is at most $1.2^{\lfloor\log_2(q-1)\rfloor}(216n+24n+3n+K)^k$, where K is the number of $(\mathtt{g},\mathtt{g})$–$(\mathtt{g},\mathtt{g})$ pairs in a best mapping between R_1 and R_2 satisfying the requirements in Lemma* 3 *and induced by the given mapping between R_1^{k+1} and R_2^{k+1}.*

Proof. First, we note that if the outer structure of R_1^{k+1} matches the outer structure of a R_2^k, then the total number of matched pairs is at most $(1.2(216n+24n+3n))^k\leqslant(216n+24n+3n)^{k+1}$, when $k\leqslant\log^{\delta} n$ for any $0<\delta<1$, where $\log^{\delta} n$ is the desired value of k to ensure the ratio in Theorem 4. Thus, it is better to map each level of structures according to the requirements in Lemma 3 and in that case there are at least $(216n+24n+3n)^{k+1}$ pairs which are matched.

Now, we show that in R_1^{k+1} if 3 k-level structures match the same k-level structure in the opposite structure, the cost is at most $(216n+24n+24n+6n+3n)(216n+24n+3n+K)^{k-1}\leqslant 1.2(216n+24n+3n+K)^k$, where K is defined in the claim. In order to keep the cost of the mapping large enough, the three k-level structures must form the configuration in Fig. 2, where A (a k-level structure) serves as $(\mathtt{x},\mathtt{x})$ pairs in the opposite k-level structure, and the $(\mathtt{a},\mathtt{u})$ pairs in B structure plus the $(\mathtt{a},\mathtt{u})$ pairs in the C structure match the $24\times 2n$ $(\mathtt{a},\mathtt{u})$ pairs in the opposite structure. Thus, at the $(k-1)$ level, each $(k-1)$-level structure can match at most one $(k-1)$-level structure. In this case, it is better to match the $(k-1)$-level structures such that every

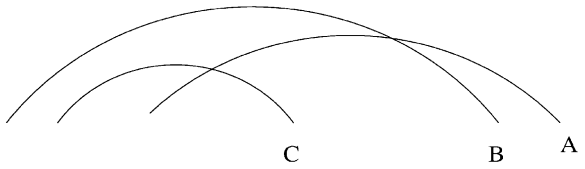

Fig. 2. The configuration of three k-level structures. Pair A serves as $(\mathtt{x},\mathtt{x})$ pairs in the structure.

level in the $(k-1)$-level structure satisfies the requirements in Lemma 3. (The above statement is, in fact, what Lemma 5 says for any value of $k+1$. Here we assume that it is true for $(k-1)$-level. The reason is that (1) the statement is true for $k=1$ and 2 since $(k-1)$-level structures degenerate to base-pairs when $k=1$ and $(k-1)$-level structures are identical to R_1 or R_2 when $k=2$; (2) Lemma 5 ensures that it is true for $(k+1)$-level. Here an induction on k is used.) Therefore, the cost is as desired.

We want to emphasize that the $(\mathtt{a},\mathtt{u})$ pairs in the structure dominate the $(\mathtt{g},\mathtt{g})$ pairs. Thus, it is impossible to increase the number of matched $(\mathtt{g},\mathtt{g})$ pairs without matching more than $24n$ $(\mathtt{a},\mathtt{u})$ pairs. (If we match $24n$ $(\mathtt{a},\mathtt{u})$ pairs according to the requirements in Lemma 3, it means that a k-level structure matches another k-level structure in the opposite structure. Only in this case, increasing the number of matched $(\mathtt{g},\mathtt{g})$ pairs means to increase the number of cutting edges in the Max Cut problem.) Note that, the $(\mathtt{x},\mathtt{x})$ pairs are twisted, they cannot match nested $(\mathtt{a},\mathtt{u})$ pairs. Therefore, to match $24\times n$ $(\mathtt{a},\mathtt{u})$ pairs in the outer structure of a k-level structure in R_2^{k+1}, one has to use three structures as shown in Fig. 2.

It is possible that the cost of the mapping is greater than $1.2(216n+24n+3n+K)^k$. This happens (with small chances) when there are pairs nested with C so that 2 (or more, say, $2^2, 2^3, \ldots$) $(\mathtt{a},\mathtt{u})$ pairs (corresponding to $(k-1)$-level structures) match one $(\mathtt{a},\mathtt{u})$ pair. In this case, it is possible that another pair crosses both the $(\mathtt{a},\mathtt{u})$ pairs to form the configuration in Fig. 2. This causes several $(k-1)$-level structures to match one $(k-1)$-level structure and the cost could be increased by 1.2 times. Note that only $(\mathtt{a},\mathtt{u})$ pairs can be involved in such a mapping. Those $(\mathtt{x},\mathtt{x})$ cannot join such a mapping since they are twisted and $(\mathtt{z},\mathtt{z})$ pairs cannot join such a mapping since there are 3 nested $(\mathtt{z},\mathtt{z})$ pairs for each v_i in G.

This process may continue to the lower levels. However, it needs to match more (2^2, 2^3, …) k-level structures to one k-level structure. This completes the proof of the claim. □

From Claim 5, we can conclude

Corollary 6. *If q k-level structures match one k-level structure, we can assume that one of the q k-level structures contributes to the cost by at most $(216n+24n+3n+K)^k$, each of the rest $(q-1)$ k-level structures increases the cost by at most $0.2(216n+24n+3n+K)^k$, where K is defined in Claim 5.*

Proof. This can be seen as follows: for any integer r,

$$(1+0.2)^r = 1 + C_r^1 0.2 + C_r^2 0.2^2 + C_r^3 0.2^3 + \cdots + 0.2^r \leqslant 1 + 0.2 \times 2^r.$$

Since we have $1+2^r$ structures, thus, one structure contributes cost $(216n+24n+3n+K)^k$, each of the other 2^r structures contributes at most $0.2(216n+24n+3n+K)^k$. □

Now, we will show the following lemma.

Lemma 5. *Given a mapping M of cost c between R_1^{k+1} and R_2^{k+1}, without decreasing the cost, one can find in polynomial time a mapping M' between R_1^{k+1} and R_2^{k+1} such that at each level M' satisfies the requirements in Lemma* 3.

Proof. We show the lemma by induction on k. Now, we show that given a mapping between R_1^{k+1} and R_2^{k+1}, without desreasing the cost, one can find in polynomial time a mapping M' between R_1^{k+1} and R_2^{k+1} such that one k-level structure can match at most one k-level structure in the other structure. Moreover, the mapping between the outer structures of R_1^{k+1} and R_2^{k+1} induced by M' satisfies the requirements in Lemma 3. Thus, by induction assumption, at every level M' satisfies the requirements in Lemma 3.

A k-level structure corresponding to an (e, f) pair in the outer structure of R_1^{k+1}/R_2^{k+1} is denoted as a k-(e, f) pair.

First, we show that the K-(g, g) pairs cannot match any of the k-(a, u) pairs in the opposite structure.

Let p_1 be the number of k-(g, g) pairs in R_1^{k+1} that are mapped to some k-(a, u) pairs in R_2^{k+1} and p_2 be the number of k-(g, g) pairs in R_2^{k+1} that are mapped to the k-(a, u) pairs in R_1^{k+1}. (q k-level structures can be mapped to one k-level structure.) Let $p = p_1 + p_2$. Then there are at least $p/(2\times 6)$ $l(p_i)$ in the outer structure of R_1^{k+1} (or $l(q_i)$ and $l(q_i')$) such that each of the (a, u) pairs for those $p/(2\times 6)$ $l(p_i)$ (or $l(q_i)$ and $l(q_i')$) cannot match any single (a, u) pairs in the outer structure of R_2^{k+1} (or R_1^{k+1}). (It is possible for those k-(a, u) to join a q to 1 match.) The reason is that each $l(p_i)$ (or $l(q_i)$ and $l(q_i')$) has at most 6 (g, g) pairs in R_1 (or R_2) and each $l(p_i)$ (or $l(q_i)$ and $l(q_i')$) might be counted for (g, g) pairs from both $l(p_{i-1})$ (or $l(q_{i-1})$ and $l(q_{i-1}')$) and $l(p_{i+1})$ (or $l(q_{i+1})$ and $l(q_{i+1}')$). Call those (a, u) pairs and $l(p_i)/l(q_i)$ *forbidden*.

We can form a new mapping such that all $24n$ (a, u) pairs in the outer structure of R_1^{k+1} match $24n$ (a, u) pairs in the opposite structure. In the new mapping, we have at least $p/(2\times 6)$ $l(p_i)$ that are newly mapped to $l(q_i)/l(q_i')$, each contains 24 nested (a, u)–(a, u) pairs. Thus, totally, we have $2p(216n+24n+K)^k$ newly mapped pairs in the new mapping. Moreover, those p (a, u) − (g, g) pairs and possibly some (g, g)–(g, g) pairs corresponding to those forbidden $l(p_i)/l(q_i)$ in the old mapping are deleted. The total cost of deleted pairs in the old mapping is at most $(p+2p\times 0.2)(216n+24n+K)^k + 6\times p/(2\times 6)(216n+24n+K)^k < 2p(216n+24n+K)^k$, where each of those $2p$ forbidden (a, u) pairs contributes a cost of less than $0.2(214n+24n+K)^k$ in

the old mapping, and there may be 6 $(\mathtt{g},\mathtt{g})$ pairs for each forbidden $l(p_i)$. Thus, the new mapping has a better cost.

Similarly, we can show that the k-$(\mathtt{g},\mathtt{g})$ pairs cannot match any k-$(\mathtt{z},\mathtt{z})$ or k-$(\mathtt{x},\mathtt{x})$ pairs.

From Corollary 6 and the induction assumption, we know that each of the k-$(\mathtt{x},\mathtt{x})$ pairs, k-$(\mathtt{z},\mathtt{z})$ pairs and k-$(\mathtt{a},\mathtt{u})$ pairs in R_1^{k+1} can contribute to a cost by at most $(216n + 24n + 3n + K)^k$. Now, we simply match the $216n$ k-$(\mathtt{x},\mathtt{x})$ pairs and the $3n$ k-$(\mathtt{z},\mathtt{z})$ pairs in both R_1^{k+1} and R_2^{k+1}, and the 24 k-$(\mathtt{a},\mathtt{u})$ pairs for every node $v_i \in V$ in R_1^{k+1} into one of the two groups of 24 k-$(\mathtt{a},\mathtt{u})$ pairs for v_i in R_2^{k+1} as in Lemma 3. By doing this, each of the k-$(\mathtt{x},\mathtt{x})$ pairs, k-$(\mathtt{z},\mathtt{z})$ pairs and k-$(\mathtt{a},\mathtt{u})$ pairs in R_1^{k+1} can contribute to the cost by $(216n+24n+3n+K)^k$. The remaining is to show that the number of k-$(\mathtt{g},\mathtt{g})$–k-$(\mathtt{g},\mathtt{g})$ pairs are not reduced in the new mapping. Consider a k-$(\mathtt{g},\mathtt{g})$–k-$(\mathtt{g},\mathtt{g})$ pair for edge (v_i, v_j) in G in the old mapping. If the corresponding 24 k-$(\mathtt{a},\mathtt{u})$ pairs for both v_i and v_j in R_1^{k+1} are mapped into one of the two groups of the 24 k-$(\mathtt{a},\mathtt{u})$ pairs in the opposite structure, then in the new mapping, the k-$(\mathtt{g},\mathtt{g})$–k-$(\mathtt{g},\mathtt{g})$ pair can be kept by selecting the right group of the 24 k-$(\mathtt{a},\mathtt{u})$ pairs in the opposite structure. (This is similar to Lemma 3.) If the 24 k-$(\mathtt{a},\mathtt{u})$ pairs for v_i or v_j in R_1^{k+1} are *not* mapped into k-$(\mathtt{a},\mathtt{u})$ pairs in the opposite structure, the number of matched pairs are increased since each of the corresponding 24 k-$(\mathtt{a},\mathtt{u})$ pairs in R_1^{k+1} contributes to the cost by $(216n + 24n + 3n + K)^k$. □

Lemma 6. *Let R_1 and R_2 be the two structures constructed in the proof of Theorem 2. Then*

$$m(R_1^{k+1}, R_2^{k+1}) \geqslant m(R_1^k, R_2^k) \times m(R_1, R_2).$$

Moreover, given a mapping M of cost c between R_1^{k+1} and R_2^{k+1}, one can find in polynomial time a mapping M_1 of cost c_1 between R_1^k and R_2^k and a mapping M_2 of cost c_2 between R_1 and R_2 such that $c_1 \times c_2 \geqslant c$.

Proof. Given a mapping of cost c_1 between R_1^k and R_2^k, and a mapping of cost c_2 between R_1 and R_2, we can construct a mapping of cost $c_1 \times c_2$ between R_1^{k+1} and R_2^{k+1} by substitution. Thus, we have

$$m(R_1^2, R_2^{k+1}) \geqslant m(R_1^k, R_2^k) \times m(R_1, R_2).$$

Given a mapping M of cost c between R_1^{k+1} and R_2^{k+1}, Lemma 5 implies that we can find in polynomial time a mapping M_1 of cost c_1 between R_1^k and R_2^k and a mapping M_2 of cost c_2 between R_1 and R_2 such that $c_1 \times c_2 \geqslant c$. □

From Lemma 6 and the same argument as in [3], we can show that the theorem is true. □

4. Algorithms

When both RNAs are secondary structures, since there is no crossing, we can represent RNA structures as ordered forests and then use the tree edit distance algorithm to solve this problem [14, 15].

We now consider the case where at most one of the RNAs involved is a tertiary structure. We present an algorithm which solves this problem. An extension of our algorithm can handle the case where both RNAs are tertiary structures with H-type pseudo-knots. Our algorithm can also be used for comparing tertiary structures in practical application.

4.1. Properties

We use a bottom-up approach. We consider smaller substructures first and eventually consider the whole structure. We can now consider how to compute $D(R_1[l_1..r_1], R_2[l_2..r_2])$.

Let $S_1[1..m]$ and $S_2[1..n]$ be arrays containing pairs in $S(R_1)[l_1..r_1]$ and $S(R_2)[l_2..r_2]$, sorted by $3'$ end.

Let $S_1[i]=(s_1,t_1)$ and $S_2[j]=(s_2,t_2)$, we define $left_1[i]$, $cross\_left_1[i]$ and $cross\_weight_1[i]$ as follows. $left_2[j]$, $cross\_left_2[j]$ and $cross\_weight_2[j]$ are defined similarly.

$$left_1[i] = \begin{cases} \max\{k\}, & S_1[k]\text{'s}3' \textit{ end is less than } s_1, \\ 0, & \textit{if no such } k \textit{ exists}, \end{cases}$$

$$cross\_left_1[i] = \begin{cases} 1 & \textit{if there exists a } k < i, \textit{ such that } S_1[k] \\ & \quad cross\_before\ S_1[i], \\ 0 & \textit{if no such } k \textit{ exists}, \end{cases}$$

$$cross\_weight_1[i] = \sum_{1 \leqslant k < i,\, S_1[k] cross\_before S_1[i]} \gamma(label_{R_1}(S_1[k]) \to \lambda).$$

Again let $S_1[i]=(s_1,t_1)$ and $S_2[j]=(s_2,t_2)$, we now define $D_1(i,j)$ and $D_2(i,j)$ as follows.

$$D_1(i,j) = D(R_1[l_1..t_1], R_2[l_2..t_2]),$$

$$D_2(i,j) = D(R_1[s_1..t_1], R_2[s_2..t_2]).$$

Lemma 7. *Suppose that $S_1[i]$ is a single base and $S_2[j]$ is a base pair or vice versa, then*

$$D_1(i,j) = \min \begin{cases} D_1(i-1,j) + \gamma(label_{R_1}(S_1[i]) \to \lambda), \\ D_1(i,j-1) + \gamma(\lambda \to label_{R_2}(S_2[j])), \end{cases}$$

Proof. Since a single base cannot be matched to a base pair, we can delete either the single base or the base pair. □

Lemma 8. *Suppose that* $S_1[i]$ *and* $S_2[j]$ *are both single bases, then*

$$D_1(i,j) = \min \begin{cases} D_1(i-1,j) + \gamma(label_{R_1}(S_1[i]) \to \lambda), \\ D_1(i,j-1) + \gamma(\lambda \to label_{R_2}(S_2[j])), \\ D_1(i-1,j-1) + \gamma(label_{R_1}(S_1[i]) \to label_{R_2}(S_2[j])). \end{cases}$$

Proof. In this case, one can either delete one of the single bases or match them together. □

Lemma 9. *Suppose that* $S_1[i]$ *and* $S_2[j]$ *are both base pairs. If* $left_1[i] \neq 0$, $left_2[j] \neq 0$, $cross\_left[i] \neq 0$, *or* $cross\_left[j] \neq 0$, *then*

$$D_1(i,j) = \min \begin{cases} D_1(i-1,j) + \gamma(label_{R_1}(S_1[i]) \to \lambda), \\ D_1(i,j-1) + \gamma(\lambda \to label_{R_2}(S_2[j])), \\ D_1(left_1[i], left_2[j]) + D_2(i,j) + cross\_weight_1[i] \\ \quad + cross\_weight_2[j]. \end{cases}$$

Proof. Let $S_1[i] = (s_1, t_1)$ and $S_2[j] = (s_2, t_2)$. Consider the best mapping between $R_1[l_1..t_1]$ and $R_2[l_2..t_2]$. If $S_1[i] = (s_1, t_1)$ is not in the mapping, then $D_1(i,j) = D_1(i-1,j) + \gamma(label_{R_1}(S_1[i]) \to \lambda)$. If $S_2[j] = (s_2, t_2)$ is not in the mapping, then $D(i,j) = D_1(i,j-1) + \gamma(\lambda \to label_{R_2}(S_2[j]))$. If both $S_1[i] = (s_1, t_1)$ and $S_2[j] = (s_2, t_2)$ are in the mapping, then they should map to each other by the definition of mapping. In this case, since one of the structures is a secondary structure, any base pair cross_before $S_1[i]$ or $S_2[j]$ will not be in the mapping and should be deleted. Therefore, if $left_1[i] \neq 0$, or $left_2[j] \neq 0$, $D(i,j) = D_1(left_1[i], left_2[j]) + D_2(i,j) + cross\_weight_1[i] + cross\_weight_2[j]$. If $left_1[i] = 0$ and $left_2[j] = 0$, and $cross\_left[i] \neq 0$, or $cross\_left[j] \neq 0$, then $D(i,j) = D_2(i,j) + cross\_weight_1[i] + cross\_weight_2[j]$. If we define $D(0,0) = 0$, then we can combine the above two cases. Note that one of the cross_weights is zero since in secondary structures there is no crossing. Also if $S_1[i]$ and $S_2[j]$ are both single bases, both cross_weights are zero. □

Lemma 10. *Suppose that* $S_1[i]$ *and* $S_2[j]$ *are both base pairs. If* $left_1[i] = 0$, $left_2[j] = 0$, $cross\_left[i] = 0$, *and* $cross\_left[j] = 0$, *then*

$$D_1(i,j) = \min \begin{cases} D_1(i-1,j) + \gamma(label_{R_1}(S_1[i]) \to \lambda), \\ D_1(i,j-1) + \gamma(\lambda \to label_{R_2}(S_2[j])), \\ D_1(i-1,j-1) + \gamma(label_{R_1}(S_1[i]) \to label_{R_2}(S_2[j])). \end{cases}$$

```
To compute D(R1[i1, j1], R2[i2, j2])

compute a sorted list S1 of pairs in S(R1[i1, j1]];
compute a sorted list S2 of pairs in S(R2[i2, j2]];

compute left1[] and left2[];
compute cross_left1[] and cross_left2[];
compute cross_weight1[] and cross_weight2[];

D1(0, 0) = 0
for i := 1 to |S1|
    for j := 1 to |S2|
        if left1[i] ≠ 0 or cross_left1[i] ≠ 0 or
           left2[j] ≠ 0 or cross_left2[j] ≠ 0 then
            Compute D1(i, j) as in Lemma 9
        else
            Compute D1(i, j) as in Lemma 10
```

Fig. 3. Procedure: Computing $D(R_1[i_1,j_1],R_2[i_2,j_2])$.

Proof. Let $S_1[i]=(s_1,t_1)$ and $S_2[j]=(s_2,t_2)$. Consider the best mapping between $R_1[l_1..t_1]$ and $R_2[l_2..t_2]$. The first two cases are similar to Lemma 9. For the last case, since there is no pair before or cross_before $S_1[i]$ or $S_2[j]$, $S_1[k]$, $1\leqslant k<i$, is inside $S_1[i]$ and $S_2[k]$, $1\leqslant k<j$, is inside $S_2[j]$. Therefore $D_1(i,j)=D_1(i-1,j-1)+\gamma(label_{R_1}(S_1[i])\rightarrow label_{R_2}(S_2[j]))$. □

4.2. Algorithm

From the above lemmas, we can compute $D(R_1,R_2)$ using a bottom-up approach. Moreover, it is clear that we do not need to compute all $D(R_1[l_1..r_1], R_2[l_2..r_2])$. Since we only use $D_2(i,j)$ in Lemma 9, we only need to compute these $D(R_1[l_1..r_1],R_2[l_2..r_2])$ such that (l_1,r_1) is a base-pair in R_1 and (l_2,r_2) is a base-pair in R_2. Furthermore, by Lemma 10, if (l_1,r_1) and (l_1+1,r_1-1) are both base-pairs in R_1 and (l_2,r_2) and (l_2+1,r_2-1) are both base-pairs in R_2, then we only need to compute $D(R_1[l_1..r_1],R_2[l_2..r_2])$. $D(R_1[l_1..r_1],R_2[l_2+1..r_2-1])$, $D(R_1[l_1+1..r_1-1],R_2[l_2..r_2])$, and $D(R_1[l_1+1..r_1-1],R_2[l_2+1..r_2-1])$ will be by-products of the computation of $D(R_1[l_1..r_1],R_2[l_2..r_2])$.

These base-pairs are called stacked pairs. A stem in an RNA R is a set of stack pairs of maximum size. More formally, we say $s=(i,j,k)$ is a stem in $R(S)$ if (i,j), $(i+1,j-1),\ldots,(i+k-1,j-k+1)$ are all base-pairs in $R(S)$ and $(i-1,j+1)$ and $(i+k,j-k)$ are not base-pairs in $R(S)$. From the above discussion, we can reduce the computation from each pair of base-pairs to each pair of stems.

Given R_1 and R_2, we can first compute sorted stem lists L_1 for R_1 and L_2 for R_2. It follows from the above discussion that, for each pair of stems $L_1[i]=(i_1,j_1,k_1)$ and $L_2[j]=(i_2,j_2,k_2)$, we have to compute $D(R_1[i_1,j_1],R_2[i_2,j_2])$. Fig. 3 shows the

```
Input: R1[1..m] and R2[1..n].

Compute a sorted (by 3′ end) stem list L1 for R1.
Compute a sorted (by 3′ end) stem list L2 for R2.

for i := 1 to |L1|
    for j := 1 to |L2|
        let L1[i] = (i1, j1, k1)
        let L1[j] = (i2, j2, k2)
        compute D(R1[i1, j1], R2[i2, j2])

compute D(R1[1, m], R2[2, n])
```

Fig. 4. An algorithm: computing $D(R_1, R_2)$

algorithm. We use Lemmas 7–10 to compute $D(R_1[i_1, j_1], R_2[i_2, j_2])$. Fig. 4 shows this computation.

Let $R_1[1..m]$ and $R_2[1..n]$ be the two given RNA structures. Let $stem(R_1)$ and $stem(R_2)$ be the number of stems in R_1 and R_2, respectively. The time to compute $D(R_1[i_1, j_1], R_2[i_2, j_2])$ is bounded by $O(|S(R_1)| \times |S(R_2)|)$. Since $|S(R_1)| < m$ and $|S(R_2)| < n$, the time complexity of the algorithm is $O(stem(R_1) \times stem(R_2) \times m \times n)$. The space complexity of the algorithm is $O(|S(R_1)| \times |S(R_2)|) = O(m \times n)$ since we only need one array to hold D_1 and another to hold D_2.

If we represent the secondary structure by a forest, then by using the technique of Klein [4] we can compute the similarity between a secondary structure and a tertiary structure in $O(m^2 n \log n)$ time where n is the size of the secondary structure. However, it seems that the space complexity of this solution is higher than quadratic.

4.3. Discussion and extensions

The essential idea of our algorithm is that although the input may include tertiary elements, the mappings our algorithm minimizes contain only base-pairs with no crossings. Let the output of our algorithm be $D_T(R_1, R_2)$ when both input RNAs are tertiary structures, Lemma 11 establishes the relation between $D_T(R_1, R_2)$ and $D(R_1, R_2)$. Therefore, when one of the inputs is a secondary structure, this algorithm computes the optimal solution.

Lemma 11. *Given two RNA tertiary structures R_1 and R_2, let P_1 and P_2 be their sets of base-pairs. Let $T \subseteq P_1$ be a set with minimum cardinality such that $P_1 - T$ has no crossings, then*

$$D_T(R_1, R_2) - 2\sum_{r \in T} \gamma(r \to \lambda) \leqslant D(R_1, R_2) \leqslant D_T(R_1, R_2).$$

Proof. Since in our algorithm we require that in the mapping there is no crossing, it is clear that $D(R_1, R_2) \leqslant D_T(R_1, R_2)$.

Consider the optimal mapping M between R_1 and R_2. Let M_1 be a subset of T and for any $r \in M_1$ there exists an s such that $(r,s) \in M$. Then by the definition of mappings and triangle inequality, we have

$$\begin{aligned} D(R_1,R_2) &= \gamma(M) - \sum_{r\in M_1} (\gamma(r \to s) - \gamma(r \to \lambda) - \gamma(\lambda \to s)) \\ &\quad + \sum_{r\in M_1} (\gamma(r \to s) - \gamma(r \to \lambda) - \gamma(\lambda \to s)) \\ &\geqslant D_T(R_1,R_2) + \sum_{r\in M_1} (\gamma(r \to s) - \gamma(r \to \lambda) - \gamma(\lambda \to s)) \\ &\geqslant D_T(R_1,R_2) - 2\sum_{r\in M_1} \gamma(r \to \lambda) \\ &\geqslant D_T(R_1,R_2) - 2\sum_{r\in T} \gamma(r \to \lambda). \qquad \square \end{aligned}$$

In real applications, the input usually contains tertiary interactions. However, the number of tertiary interactions is always relatively small compared with the number of secondary interactions. Therefore, we can also use this algorithm to compute the similarity when both structures are tertiary structures. Essentially, the algorithm tries to find the best secondary structures to match and delete tertiary interactions. Although this is not an optimal solution, in practice it would produce a reasonable result by matching most of the base pairs. A post-processing step can be applied to add some matching tertiary interactions which satisfy the mapping constraints.

The simplest tertiary interaction is known as an H-type pseudo-knots where a stem crosses with at most one other stem. We can extend our algorithm by allowing these kind of crossings in the mappings. With this extension, when the inputs are RNA structures with H-type pseudo-knots, our algorithm can find the optimal solution with the same time complexity. When one of the inputs is a general tertiary structure and the other one is a tertiary structure with only H-type pseudo-knots, our algorithm can find the optimal solution with higher, but still polynomial, complexity.

5. Approximation algorithms

In this section, we consider a maximization version of the problem. Assume that a matching between any two identical pairs contributes to the cost by 1, and any other matching contributes to the cost by 0. We want to find a mapping with the *maximum* cost. We use $\delta(R_1,R_2)$ to denote the cost of the optimal mapping between R_1 and R_2. This maximization version is similar to the longest common subsequence problem for

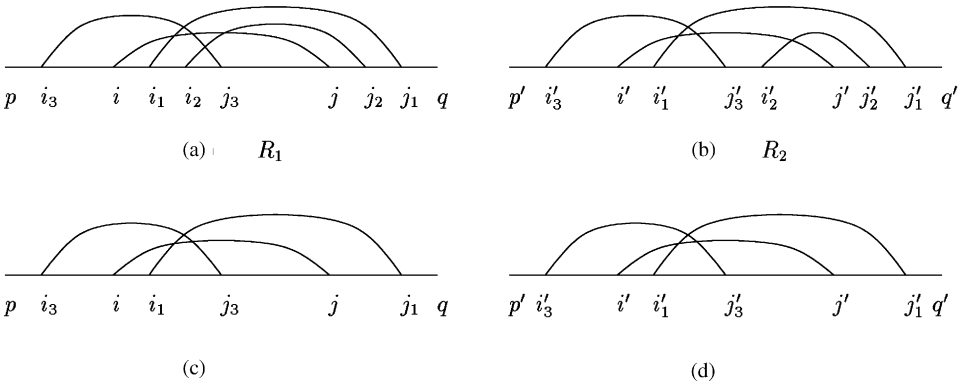

Fig. 5. (a) The set of specified links for R_1. (b) The set of specified links for R_2. (c) The preserved links for R_1 in a match. (d) The preserved links for R_2 in a match. (i,j) matches (i',j') and (i_l,j_l) matches (i'_l,j'_l) for $l=1$ and 3. Such a match form 7 matched segments for both R_1 and R_2.

sequences. In Section 3, we proved that this problem cannot be approximated within ratio $2^{\log^{\delta} n}$ in polynomial time, unless $\mathrm{NP} \subseteq \mathrm{DTIME}[2^{\mathrm{poly}\ \log n}]$.

We provide a ratio-$(b-1)+2/(b+1)$ approximation algorithm for the case where each base-pair *crosses* with at most b other base-pairs.

Our *basic idea* is as follows: We start with an arbitrary base-pair (i,j) in $S(R_1)$ and consider (i,j) and the other at most b base-pairs $(i_1,j_1),(i_2,j_2),\ldots,$ and (i_b,j_b) crossing (i,j) in $S(R_1)$. Call the $b+1$ base-pairs $(i,j),(i_1,j_1),(i_2,j_2),\ldots,$ and (i_b,j_b) *a b-component* for $S(R_1)$. We use $(i',j'),(i'_1,j'_1),(i'_2,j'_2),\ldots,$ and (i'_b,j'_b) to denote a b-component for $S(R_2)$. For each pair of subsequences $R_1[p..q]$ and $R_2[p'..q']$, we consider all pairs of b-components for them. A *match* between the two b-components contains $k+1$ matched pairs of base-pairs such that (i,j) matches (i',j') and the $k+1$ matched pairs of base-pairs satisfy (a)–(d) in the definition of a mapping. (i,j) and (i',j') form an *imposed* pair of base-pairs. The $k+1$ base-pairs form $2(k+1)$ positions in both $R_1[p..q]$ and $R_2[p'..q']$ that decompose both $R_1[p..q]$ and $R_2[p'..q']$ into $2k+3$ segments, called *matched* segments. For each pair of b-components for $R_1[p..q]$ and $R_2[p'..q']$, we try all possible matches between the two b-components. For each match, we forbid any other base-pairs not in the b-components to cross any base-pair in the b-components. The match between the corresponding matched segments are computed recursively (see Fig. 5).

For computation, we define $d[p,q,p',q',i,j,i_1,j_1,\ldots,i_b,j_b,i',j',i'_1,j'_1,\ldots,i'_b,j'_b]$ and $d[p,q,p',q']$, recursively.

1. If no matched segment of $R_1[p..q]$ or $R_2[p'..q']$ for the two b-components has any base-pair, the $d[i,j,i_1,j_1,\ldots,i_b,j_b,i',j',i'_1,j'_1,\ldots,i'_b,j'_b]$ is defined to be the biggest number of matched pairs of base-pairs between the two b-components among all possible matches.
2. Otherwise,

$$d[p,q,p',q',i,j,i_1,j_1,\ldots,i_b,j_b,i',j',i'_1,j'_1,\ldots,i'_b,j'_b]$$
$$= \max_{any\ \ possible\ \ match} \left\{ cost(match) + \sum_{matched\ \ segments\ \ R_1[p_1,q_1]\ \ and\ \ R_2[p_1,q_2]} d[p_1,q_1,p_2,q_2] \right\}, \quad (1)$$

where $cost(match)$ is the number of preserved pairs of base-pairs in the match and $d[p,q,p',q']$ is defined as

$$d[p,q,p',q'] = \max_{\substack{i,j,i_1,j_1,\ldots,i_b,j_b \\ i',j',i'_1,j'_1,\ldots,i'_b,j'_b}} d[p,q,p',q',i,j,i_1,j_1,\ldots,i_b,j_b,i',j',i'_1,j'_1,\ldots,i'_b,j'_b]. \quad (2)$$

In (2), we take the maximum over all pairs of b-component.

We can compute the values of $d[p,q,p'q']$'s and $d[p,q,p',q',i,j,i_1,j_1,\ldots,i_b,j_b,i',j',i'_1,j'_1,\ldots,i'_b,j'_b]$'s bottom-up. Computing each $d[p,q,p',q',i,j,i_1,j_1,\ldots,i_b,j_b,i',j',i'_1,j'_1,\ldots,i'_b,j'_b]$ requires to consider $(b+1)!$ matches and thus requires $O((b+1)!)$ time. Computing each $d[p,q,p'q']$ requires to consider all pairs of b-components, which is bounded by $O(n^2)$, where n is the number of base-pairs in $S(R_1)$ and $S(R_2)$. Therefore, the total time required is $O(m^4(b+1)!n^2)$, where m is the length of the sequences.

Theorem 7. *The performance ratio of the algorithm is* $(b-1)+2/(b-1)$.

Proof. Consider a match for two b-components with base-pairs (i,j), $(i_1,j_1),\ldots,(i_b,j_b)$ and (i',j'), $(i'_1,j'_1),\ldots,(i'_b,j'_b)$. Recall that (i,j) always matches (i',j'). Assume that k pairs of base-pairs are preserved besides the imposed pair. Each base-pair (i_l,j_l) for $l=1,2,\ldots b$ may cross at most b_1 other base-pairs not in the b-component. Those $b-1$ base-pairs are forbidden to be included in our approximation solution. Therefore, the performance ratio ρ is upper bounded as follows:

$$\rho = \frac{1+k+k(b-1)}{k+1} = \frac{1+k-(b-1)+k(b-1)+(b-1)}{k+1} \quad (3)$$

$$= \frac{(k+1)(b-1)+1+k-(b-1)}{k+1} = (b-1)+\frac{1+k-(b-1)}{k+1}. \quad (4)$$

When $k \leqslant b-2$, $\rho \leqslant b-1$. When $k > b-2$, i.e., $k=(b-1)$ or $k=b$,

$$\rho \leqslant \max\left\{(b-1)+\frac{1}{b},(b-1)+\frac{2}{b+1}\right\} = (b-1)+\frac{2}{b+1}. \quad \square \quad (5)$$

Not that it is important to insist that base-pair (i,j) matches base-pair (i',j'). Otherwise, the performance ratio could be b.

6. Conclusion

We have presented a similarity measure between RNA structures. We show that in general this problem is Max SNP-hard. We show a stronger inapproximability result for the maximization version. We then present algorithms which can be used in practical applications. We also show an approximation algorithm.

Acknowledgements

Lusheng Wang is fully supported by the Hong Kong CERG Grant 9040351. Kaizhong Zhang is partially supported by the Natural Sciences and Engineering Research Council of Canada under Grant No. OGP0046373.

References

[1] V. Bafna, S. Muthukrishnan, R. Ravi, Comparing similarity between RNA strings, Proc. Combinatorial Pattern Matching Conference 95, Lecture Notes in Computer Science, vol. 937, 1995, pp. 1–14.
[2] F. Corpet, B. Michot, RNAlign program: alignment of RNA sequences using both primary and secondary structures, Comput. Appl. Biosci. 10 (4) (1995) 389–399.
[3] J. Hein, T. Jiang, L. Wang, K. Zhang, On the complexity of comparing evolutionary trees, Discrete Appl. Math. 71 (1996) 153–169.
[4] P.N. Klein, Computing the edit-distance between unrooted ordered trees, Private communication, 1998.
[5] S.Y. Le, R. Nussinov, J.V. Mazel, Tree graphs of RNA secondary structures and their comparisons, Comput. Biomed. Res. 22 (1989) 461–473.
[6] S.Y. Le, J. Owens, R. Nussinov, J.H. Chen, B. Shapiro, J.V. Mazel, RNA secondary structures: comparisons and determination of frequently recurring substructures by consensus, Comput. Appl. Biosci. 5 (1989) 205–210.
[7] S.E. Needleman, C.D. Wunsch, A general method applicable to the search for similarities in the amino-acid sequences of two proteins, J. Mol. Biol. 48 (1970) 443–453.
[8] C.H. Papadimitriou, M. Yannakakis, Optimization, approximation, and complexity classes, J. Comput. System Sciences 43 (1991) 425–440.
[9] B. Shapiro, An algorithm for comparing multiple RNA secondary structures, Comput. Appl. Biosci. 4 (3) (1988) 387–393.
[10] B. Shapiro, K. Zhang, Comparing multiple RNA secondary structures using tree comparisons, Comput. Appl. Biosci. 6 (4) (1990) 309–318.
[11] T.F. Smith, M.S. Waterman, The identification of common molecular subsequences, J. Mol. Biol. 147 (1981) 195–197.
[12] T.F. Smith, M.S. Waterman, Comparison of biosequences, Adv. Appl. Math. 2 (1981) 482–489.
[13] K.C. Tai, The tree to tree correction problem, J. ACM 26 (3) (1979) 422–433.
[14] K. Zhang, Computing similarity between RNA secondary structures, Proc. IEEE Internat. Joint Symp. on Intelligence and Systems, Rockville, Maryland, May 1998, pp. 126–132.
[15] K. Zhang, D. Shasha, Simple fast algorithms for the editing distance between trees and related problems, SIAM J. Comput. 18 (6) (1989) 1245–1262.

[16] M. Zuker, On finding all suboptimal foldings from of an RNA molecule, Science 244 (1989) 48–52.
[17] M. Zuker, D. Sankoff, RNA secondary structure and their prediction, Bull. Math. Biol. 46 (1984) 591–621.
[18] M. Zuker, P. Stiegler, Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information, Nucl. Acid Res. 9 (1981) 133–148.