Computing similarity between RNA structures *

Kaizhong Zhang[1], Lusheng Wang[2], and Bin Ma[3]

[1] Dept. of Computer Science, University of Western Ontario,
London, Ont. N6A 5B7, Canada
kzhang@csd.uwo.ca

[2] Dept. of Computer Science, City University of Hong Kong,
83 Tat Chee Avenue, Kowloon, Hong Kong
lwang@cs.cityu.edu.hk

[3] Dept. of Mathematics, Peking University,
Beijing 100871, P.R. China
bma@sxx0.math.pku.edu.cn

Abstract. The primary structure of a ribonucleic acid (RNA) molecule is a sequence of nucleotides (bases) over the 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 in general this problem is NP-hard for tertiary structures. We present algorithms for the case where at least one of the RNA involved is of secondary structures. We then show that this algorithm might be used to deal with the practical application. We also show an approximation algorithm.

1 Introduction

Ribonucleic Acid (RNA) is an important molecule which performs a wide range of functions in 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.

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

* Research supported partially by the Natural Sciences and Engineering Research Council of Canada under Grant No. OGP0046373.

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 NP-complete. We present an algorithm for the case where at least one of the RNA involved is of secondary structure. We then show 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 tree-like structure, there are works considering comparison using tree comparison [7, 4, 5, 8, 3]. 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, the method we propose in this paper directly use base-paired and unpaired nucleotides in the representation and apply some basic operations on them.

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 treat a base-pair as a whole entity. For example, in the comparison of two RNAs, a base-pair from one RNA can have one nucleotide deleted while the other nucleotide matched to nucleotide (unpaired or even paired) in the other RNA. Our method treat 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 [14–16], we assume a model where there is no knots in the secondary structure. This means that for the secondary structure, the bonds are non-crossing. For 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.

$$S(R) = \begin{array}{l} \{(i,j) | i < j \ \textit{and} \ (R[i], R[j]) \ \textit{is a base pair in} \ R\} \\ \cup\{(i,i) | R[i] \ \textit{is not involved in any base pair in} \ R\} \end{array}$$

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

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

For $r = (i,j) \in S(R)$, we use $label_R(r)$ to represent 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. We define $left(r) = i$ and $right(r) = j$.

Following the tradition in sequence comparison [6, 9, 10], 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 to delete 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 to insert 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 [11, 13], we represent an edit operation as $a \rightarrow b$, 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 $a \rightarrow b$ 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 $s_1, ..., s_k$ of edit operations. An S-derivation from RNA structure A to RNA structure B is a sequence of RNA structures $A_0, ..., A_k$ such that $A = A_0$, $B = A_k$, and $A_{i-1} \rightarrow A_i$ via s_i for $1 \leq i \leq k$.

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

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 $r_l < s_l$ and $r_r < s_r$.

Let R_1 and R_2 be two RNA structures. 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:

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

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) $\gamma(M_1 \circ M_2) \le \gamma(M_1) + \gamma(M_2)$.*

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 $\gamma(x \to y) \le \gamma(x \to z) + \gamma(z \to y)$. □

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

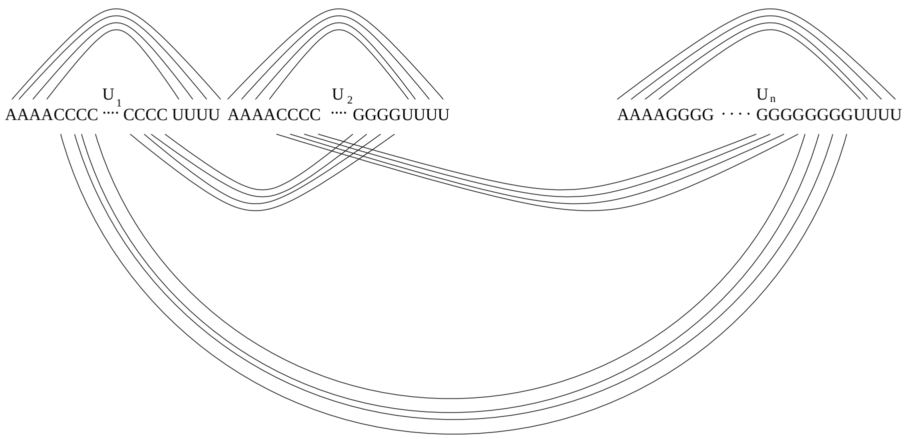

Fig. 1. RNA structure 1

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 $\gamma(M) \leq \gamma(S)$. 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 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 $\gamma(M_1) \leq \gamma(S_1)$. Let M_2 be the mapping for s_k. From lemma 1, we have $\gamma(M_1 \circ M_2) \leq \gamma(M_1) + \gamma(M_2) \leq \gamma(S)$. □

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

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

Proof. Immediately from lemma 2. □

3 NP-hard result

We now consider the problem of comparing RNA structures where both structures are tertiary structures. We show that this is in general NP-hard.

We will reduce the 3-SAT problem to this problem.

Problem of 3-SAT

Let $S = C_1 \cdot C_2 ... C_n$, where $C_i = (v_{i_1} \cup v_{i_2} \cup v_{i_3})$, be an instance of 3-SAT problem. We will construct two RNA structures R_1 and R_2 as in Figure 1 and Figure 2.

In R_1, there are n segments each of which is enclosed by four base pairs. These base pairs are all AU pairs. And each segment is connected to every other

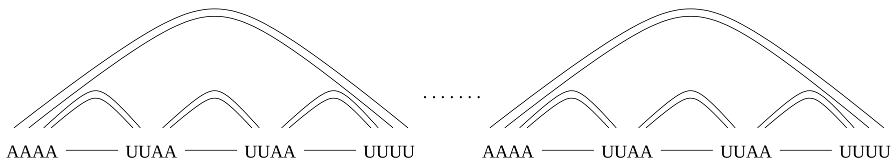

Fig. 2. RNA structure 2

segment by four base pairs of CG type. Note that the number of base pairs in R_1 is $4 \cdot n \cdot (1 + (n-1)/2)$.

In R_2, for each v_{i_k}, there is a corresponding segment which is enclosed by two base pairs of AU type. Each clause C_i is then represented by segments of $v_{i_1}, v_{i_2}, v_{i_3}$ and is enclosed by another two base pairs of AU type.

We now consider base pairs between segments in R_2. For each v_{i_k}, define s_{i_k} as follows.

$$s_{i_k} = \{v_{j_l} | i \neq j \text{ } and \text{ } v_{j_l} \text{ } is \text{ } not \text{ } complement \text{ } of \text{ } v_{i_k}\}$$

For each v_{j_l} in s_{i_k} there are four bases in the segment for v_{i_k}. If $j < i$ then these bases are G's, otherwise these are C's. Suppose that v_{j_l} and v_{g_h} are in s_{i_k}, then the bases for v_{j_l} is before the bases for v_{g_h} if either $j > g$ or $j = g$ and $l > h$. Note that if v_{j_l} is in s_{i_k} then v_{i_k} is also in s_{j_l}. Now suppose that $i < j$, then the bases in segment v_{i_k} for v_{j_l} are C's and the bases in segment v_{j_l} for v_{i_k} are G's. In the RNA structure R_2, they form base pairs. Figure 3 shows an example involving two clauses. Let N be the number of base pairs in R_2, then $N = 8 \cdot n + 2 \cdot \sum_{i=1}^{n} \sum_{k=1}^{3} |s_{i_k}|$.

It is clear that R_1 and R_2 can be constructed in polynomial time from an instance of 3-SAT problem S. In the following, we assume that each operation has unit cost. We will show that S can be satisfied if and only if $D(R_1, R_2) = N - 4 \cdot n \cdot (1 + (n-1)/2)$.

Let S be an instance of 3-SAT problem and R_1 and R_2 be as in Figure 1 and Figure 2. The following lemmas give the relationship between S and $D(R_1, R_2)$.

Lemma 3. *If S can be satisfied, then $D(R_1, R_2) = N - 4 \cdot n \cdot (1 + (n-1)/2)$.*

Proof. If S can be satisfied, then for each clause C_i there is at least one v_{i_k} whose value is true. Consider R_2, for each clause, we can first delete any segment which does not correspond to v_{i_k} and its enclosing base pairs. For the segment of v_{i_k}, we can delete the bases which base paired with these segments that have already been deleted. The resulting structure after these deletions is exactly the same as R_1. Therefore $D(R_1, R_2) = N - 4 \cdot n \cdot (1 + (n-1)/2)$ since the number of base pairs in R_2 is N and in R_1 is $4 \cdot n \cdot (1 + (n-1)/2)$. □

Lemma 4. *If $D(R_1, R_2) = N - 4 \cdot n \cdot (1 + (n-1)/2)$, then S can be satisfied.*

Proof. In this case, every base pair in R_1 is in the optimal mapping M. In addition, each base pair in R_1 is matched to an identical base pair in R_2. This

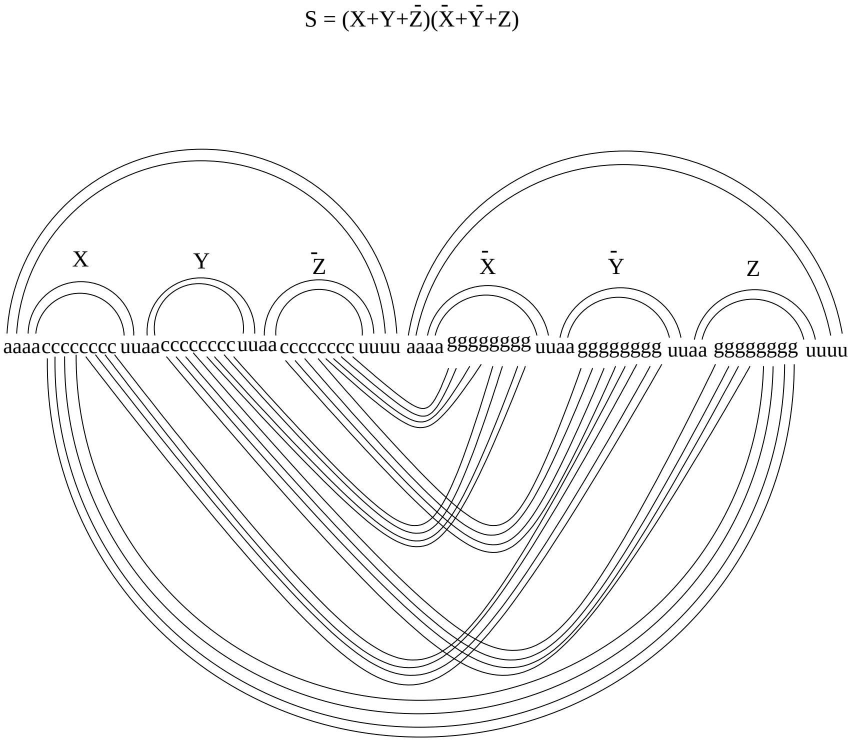

Fig. 3. An example

means that the four base pairs enclosing each segment must map to four base pairs for each clause in R_2. The only possibility for this to happen is that for each clause two (out of three) segments have to be deleted. Therefore mapping M in R_2 keeps one variable in each clause. And this variable is connected to all the variables left in other clauses by means of base pairing. So for any two variables left there is no conflict. Hence we can assign the value true to all the variables left and S is satisfied. □

Theorem 2. *The problem of determining if* $D(R_1, R_2) \leq k$ *is NP-complete.*

Proof. This problem is clearly in NP since one can guess a mapping in R_1 and R_2 and check to see if the cost is less or equal to k or not.

By lemma 3 and lemma 4, S is satisfied if and only if $D(R_1, R_2) = N - 4 \cdot n \cdot (1 + (n-1)/2)$. Therefore this problem is NP-hard.

Hence this problem is NP-complete. □

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 [12, 13].

We now consider the case where at most one of the RNA involved is 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.

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

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

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

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

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 5. *If $left_1[i] \neq 0$, $left_2[j] \neq 0$, $cross\_left[i] \neq 0$, or $cross\_left[j] \neq 0$, then*

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

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 $D(i,j) = D_1(i,j-1) + \gamma(\lambda \rightarrow label_{R_2}(S_2[j]))$. 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 structure, there is no crossing. Also if $S_1[i]$ and $S_2[j]$ are both single bases, both cross_weights are zero. □

Lemma 6. *If $left_1[i] = 0$, $left_2[j] = 0$, $cross\_left[i] = 0$, and $cross\_left[j] = 0$, then*

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

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 $R_2[l_2..t_2]$. The first two cases are similar to lemma 5. For the last case, since there is no pair before or cross_before $S_1[i]$ or $S_2[j]$, $S_1[k]$, $1 \leq k < i$, is inside $S_1[i]$ and $S_2[k]$, $1 \leq k < j$, is inside $S_2[j]$. Therefore $D_1(i,j)$ $= D_1(i-1,j-1) + \gamma(label_{R_1}(S_1[i]) \rightarrow label_{R_2}(S_2[j])) + cross\_weight_1[i] + cross\_weight_2[j]$. □

From the above lemmas, we can compute $D(R_1,R_2)$ using 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 5 and 6, 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 6, 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-product 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 $(i,j), (i+1,j-1), ...(i+k-1, j-k+1)$ are all base pairs in $R(S)$ and $(i-1,j+1)$ and $(i+k, j-k)$ are not base pairs in $R(S)$.

4.2 Algorithm

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 $D(R_1[i_1,j_1], R_2[i_2,j_2])$. Figure 5 shows the algorithm. We use lemma 5 and 6 to compute $D(R_1[i_1,j_1], R_2[i_2,j_2])$. Figure 4 shows this computation.

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

compute a sorted list S1 of base pairs inside (i1, j1);
compute a sorted list S2 of base pairs inside (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 5
        else
             Compute D1(i, j) as in Lemma 6
```

Fig. 4. Procedure: Computing $D(R_1[i_1, j_1], R_2[i_2, j_2])$

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 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 $O(|S(R_1)| \times |S(R_2)|) = O(m \times n)$.

Note that when one of the RNA is secondary structure, this algorithm compute the optimal solution of the problem. This algorithm can be modified to handle the case where the input RNAs are tertiary structures with H-type pseudoknots (a stem crosses with at most one other stem).

If we represent the secondary structure by a forest, then by using the technique of Klein [3] we can compute similarity between a secondary structure and a tertiary structure in $O(m^2 n \log n)$ time where $m \leq n$.

Note also that since the number of tertiary interactions is relatively small compared with the number of secondary interactions, 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.

```
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. 5. An algorithm: Computing $D(R_1, R_2)$

5 Approximation Algorithms

In this section, we consider a maximization version of the problem. Let M be a mapping from R_1 to R_2. The value $\delta(M)$ of M is defined to be the number of identical pairs of base-pairs in M. Suppose that we define $\gamma(a, b)$ to be 0 if a and b are identical; 2 if a and b are non-identical base-pairs; and 1 if one of them is λ. Then $\delta(M) + \gamma(M) = n_1 + n_2$, where n_1 and n_2 are the number of base-pairs in $S(R_1)$ and $S(R_2)$. Instead of finding a M with the smallest cost $\gamma(M)$, we want to find a M with the largest value $\delta(M)$. Obviously, the maximization version is also NP-complete.

We give an ratio-$(b-1) + \frac{2}{b+1}$ approximation algorithm for the case where each base-pair *crosses* with at most b other base-pairs. Due to space limitation, we only present the basic idea here.

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), ..., 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), ..., and (i_b, j_b) *a b-component* for $S(R_1)$. We use (i', j'), (i'_1, j'_1), (i'_2, j'_2), ..., 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 Figure 6.)

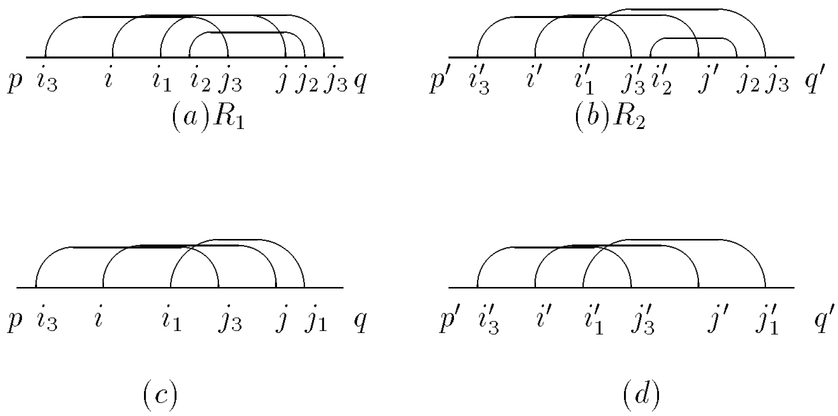

Fig. 6. (a) the set of specified links for R_1. (b) the set of specified links for R_2. (c) the preserved links for R_1 in a match. (d) the preserved links for R_2 in a match. (i, j) matches (i', j') and (i_l, j_l) matches (i'_l, j'_l) for $l = 1$ and 3. Such a match form 7 matched segments for both R_1 and R_2.

References

1. V. Bafna, S. Muthukrishnan, and R. Ravi, 'Comparing similarity between RNA strings', Proc. Combinatorial Pattern Matching Conf. 95, LNCS 937, pp.1-14, 1995
2. F. Corpet and B. Michot, 'RNAlign program: alignment of RNA sequences using both primary and secondary structures', *Comput. Appl. Biosci* vol. 10, no. 4, pp.389-399, 1995
3. P.N. Klein, 'Computing the edit-distance between unrooted ordered trees', Proc. Annual European Symposium on Algorithms 98 LNCS 1461, pp.91-102, 1998.
4. S.Y. Le, R. Nussinov and J.V. Mazel, 'Tree graphs of RNA secondary structures and their comparisons' *Comput. Biomed. Res.* vol. 22, pp.461-473, 1989
5. S.Y. Le, J. Owens, R. Nussinov, J.H. chen, B. Shapiro, and J.V. Mazel, 'RNA secondary structures: comparisons and determination of frequently recurring substructures by consensus', *Comput. Appl. Biosci* vol. 5, pp.205-210, 1989
6. S.E. Needleman and C.D. Wunsch, 'A general method applicable to the search for similarities in the amino-acid sequences of two proteins', *J. Mol. Bio.*, 48, pp.443-453, 1970
7. B. Shapiro, 'An algorithm for comparing multiple RNA secondary structures', *Comput. Appl. Biosci* vol. 4, no. 3, pp.387-393, 1988
8. B. Shapiro and K. Zhang, 'Comparing multiple RNA secondary structures using tree comparisons', *Comput. Appl. Biosci* vol. 6, no.4, pp.309-318, 1990
9. T.F. Smith and M.S. Waterman, 'The identification of common molecular subsequences', *J. Mol. Bio.* 147, pp.195-197, 1981

10. T.F. Smith and M.S. Waterman, 'Comparison of biosequences', *Adv. in Appl. Math.* 2, pp.482-489, 1981
11. K.C. Tai, 'The tree to tree correction problem', *JACM* vol.26, no.3, pp.422-433, 1979
12. Kaizhong Zhang, 'Computing similarity between RNA secondary structures', *Proceedings of IEEE International Joint Symposia on Intelligence and Systems*, Rockville, Maryland, May 1998, pp. 126-132.
13. K. Zhang and D. Shasha, 'Simple fast algorithms for the editing distance between trees and related problems', *SIAM J. Computing* vol. 18, no. 6, pp.1245-1262, 1989
14. M. Zuker, 'On finding all suboptimal foldings from of an RNA molecule', *Science* 244, pp.48-52, 1989
15. M. Zuker and D. Sankoff, 'RNA secondary structure and their prediction', *Bull. Math. Biol.* 46, pp.591-621, 1984
16. M. Zuker and P. Stiegler, 'Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information', *Nucleic Acid Res.* 9, pp.133-148, 1981