A new method for computing similarity between RNA structures*

Greg Collins
Dept. of Computer Science
University of Western Ontario
London, Ont. N6A 5B7
Canada
greg@csd.uwo.ca

Shuyun Le
Lab. of Exp. and Comp. Biology
National Cancer Instituite, NIH
Frederick, Maryland 21702
USA
shuyun@ncifcrf.gov

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

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.

We consider the edit distance between two RNA structures. This is a notion of similarity, introduced in [9], between two RNA molecule structures taking into account the primary, the secondary and the tertiary structures. In general this problem is NP-hard for tertiary structures. In this paper we consider this notion under some constraints. We present an algorithm and then show how to use this algorithm for practical applications.

1 Introduction

Ribonucleic Acid (RNA) is an important molecule which performs a wide range of functions in biological systems. In particular it is RNA (not DNA) that contains genetic information of viruses such as HIV and therefore regulates the functions of such viruses. 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 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.

In [7, 9], a similarity measure between RNA structures was introduced. First three basic operations, insertion, deletion, and relabel, were defined on base pairs and single bases. The measure between two RNA's is then defined to be the minimum number of (weighted) operations that can transform one RNA into the other. If both structures are secondary, then using the solution in [7], we can compute the optimal solution using ordered tree edit distance algorithms [8]. If at least one of the structures is secondary, then using the solution in [9], one can find the optimal solution. When both structures are tertiary, then the problem is NP-hard [9].

In this paper we consider this problem under some constraints. Given two RNA's R_1 and R_2, a set of base pairs $\{p_{1_1} \cdots p_{1_K}\}$ from R_1, and a set of base pairs $\{p_{2_1} \cdots p_{2_K}\}$ from R_2, We would like to find the minimum number of (weighted) operations that can transform R_1 into R_2 under the condition that p_{1_i} has be matched to p_{2_i} for $1 \le i \le K$.

We will present an algorithm for this problem. We will show that this algorithm can be used to accommodate the situation where both structures are tertiary structures.

2 Comparing two RNA structures

2.1 RNA structures and edit distance

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 [10, 11], 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 tertiary structure, there is no restriction of non-crossing.

An RNA structure is represented by $R(P)$, where R is a sequence of nucleotides and $P \subset \{1, 2, \cdots |R|\}^2$ is a set of pairs of which each element (i, j) represents a base pair $(R[i], R[j])$. 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 assume

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

that base pairs in R do not share participating bases. Formally for any (i_1, j_1) and (i_2, j_2) in P, $j_1 \neq i_2$, $i_1 \neq j_2$, and $i_1 = i_2$ if and only if $j_1 = j_2$.

Given an RNA structure $R(P)$, we use $S(R, P)$ to represent the set of structural elements consisting of both its set of base-pairs and the remaining unpaired nucleotides.

$$S(R, P) = P \cup \{(i, i) | R[i] \text{ } is \text{ } a \text{ } single \text{ } base \text{ } in \text{ } R\}$$

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

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

For $r = (i, j) \in S(R, P)$, 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 $r = (i, j) \in S(R, P)$, 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, 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 $S(R, P)$ or an unpaired base. Relabelling a base-pair is to replace one base-pair in $S(R, P)$ 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 $S(R, P)$. 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 $S(R, P)$. 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 [8], 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 $s_1, ..., s_k$ 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 $A_{i-1} \to A_i$ via s_i for $1 \leq i \leq k$.

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

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 $R_1(P_1)$ and $R_2(P_2)$ is defined as:

$$D(R_1(P_1), R_2(P_2)) = \min_S \{\gamma(T) \mid T \text{ is an edit operation sequence taking } S(R_1, P_1) \text{ to } S(R_2, P_2)\}$$

2.2 Mapping between RNA structures

Let $r = (r_l, r_r)$ and $s = (s_l, s_r)$ be two elements in $S(R, P)$ of an RNA $R(P)$, 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 $R_1(P_1)$ and $R_2(P_2)$ be two RNA structures. Formally we define a triple (M, R_1, R_2) to be a mapping from $R_1(P_1)$ to $R_2(P_2)$, where M is a binary relation on $S(R_1, P_1) \times S(R_2, P_2)$ such that

(1) For any (r, s) in M,
  r is a base-pair in P_1 if and only if s is a base-pair in P_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 $R_1(P_1)$ to $R_2(P_2)$. Then we can similarly define the cost of M:

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

Mappings can be composed. Let M_1 be a mapping from $R_1(P_1)$ to $R_2(P_2)$ and M_2 be a mapping from $R_2(P_2)$ to $R_3(P_3)$. Define

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

Lemma 1 *[9] 1) $M_1 \circ M_2$ is a mapping between $R_1(P_1)$ and $R_3(P_3)$. 2) $\gamma(M_1 \circ M_2) \leq \gamma(M_1) + \gamma(M_2)$.* The relation between a mapping and a sequence of edit operations is as follows:

Lemma 2 *[9] Given S, a sequence $s_1, \ldots, s_k$ of edit operations from $R_1(P_1)$ to $R_2(P_2)$, there exists a mapping M from $R_1(P_1)$ to $R_2(P_2)$ such that $\gamma(M) \leq \gamma(S)$. Conversely, for any mapping M_e, there exists a sequence of edit operations such that $\gamma(S) = \gamma(M)$.*

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

Theorem 1 *[9]* $D(R_1(P_1), R_2(P_2)) = \min_M \{\gamma(M) \mid M \text{ } is \text{ } a \text{ } mapping \text{ } from \text{ } R_1(P_1) \text{ } to \text{ } R_2(P_2)\}$

2.3 Constrained edit distance

We now consider the edit distance between two RNAs with constraints. Given $R_1(P_1)$ and $R_2(P_2)$ and a mapping M between them, we would like to find the edit distance between them with the constraint that for any $(r, s) \in M$, r has to match to s.

Formally given $R_1(P_1)$ and $R_2(P_2)$ and a mapping M between them, we define the constrained edit distance between $R_1(P_1)$ and $R_2(P_2)$ with constraint M as follows.

$$D(R_1(P_1), R_2(P_2), M) = \min_{M'} \{\gamma(M') \mid M \subset M' \text{ and } M' \text{ is a mapping from } R_1(P_1) \text{ to } R_2(P_2)\}$$

The following lemma is immediate from this definition and Theorem 1.

Lemma 3
$D(R_1(P_1), R_2(P_2)) = D(R_1(P_1), R_2(P_2), \emptyset)$
$D(R_1(P_1), R_2(P_2)) \leq D(R_1(P_1), R_2(P_2), M)$

We consider the properties of mappings. Suppose that M is a mapping between $R_1(P_2)$ and $R_2(P_2)$, we define a binary relation S_M on $\{0, 1, \cdots |R_1| + 1\} \times \{0, 1, \cdots |R_2| + 1\}$ as follows.
$S_M = \{(0, 0), (|R_1| + 1, |R_2| + 1)\} \cup$
$\{(left(r), left(s)), (right(r), right(s)) | (r, s) \in M\}$

Lemma 4 *Given a mapping M between $R_1(P_2)$ and $R_2(P_2)$, then S_M satisfies the following condition.*
For any (i_1, j_1) and (i_2, j_2) in S_M,

$$i_1 < i_2 \text{ if and only if } j_1 < j_2,$$

From this lemma, S_M can be represented as a sequence of ordered pairs as follows, where $i_k < i_{k+1}$, $j_k < j_{k+1}$ for $0 \leq k < K$.
$S_M = ((i_0 = 0, j_0 = 0), (i_1, j_1), \cdots (i_{K-1}, j_{K-1}),$
$(i_K = |R_1| + 1, j_K = |R_2| + 1))$.

Let (r, s) be a pair in a mapping M' from $R_1(P_1)$ to $R_2(P_2)$, We say (r, s) is compatible with M or S_M if there are k and l such that $i_k < left(r) < i_{k+1}$, $j_k < left(s) < j_{k+1}$, $i_l < right(r) < i_{l+1}$, and $j_l < right(s) < j_{l+1}$.

Lemma 5 *Suppose that M is a mapping between $R_1(P_1)$ and $R_2(P_2)$ and $S_M = ((i_0, j_0), (i_1, j_1), \cdots (i_K, j_K))$. Let M' be another mapping between $R_1(P_1)$ and $R_2(P_2)$ such that $M \cap M' = \emptyset$, then $M \cup M'$ is a mapping if and only if for any $(r, s) \in M'$, (r, s) is compatible with S_M.*

Proof: Suppose that the conditions are true. Let (r_1, s_1) and (r_2, s_2) be in $M \cup M'$, we want to show that they satisfy the mapping condition. We will only consider the case where (r_1, s_1) is in M and (r_2, s_2) is in M' since M and M' are both mappings. We now assume that r_1 and r_2 are both base pairs. In this case s_1 and s_2 are also base pairs. Other cases where r_1 or r_2 is a single base are easier to analyse and will be omitted.

Let $i_k = left(r_1)$ and $i_l = right(r_1)$, then $j_k = left(s_1)$ and $j_l = right(s_1)$. By the definition of compatible, we have, for $t = k, l$, $left(r_2) < i_t$ if and only if $left(s_2) < j_t$, and $right(r_2) < j_t$ if and only if $right(s_2) < j_t$. This means that the relation between r_1 and r_2 and the relation between s_1 and s_2 are the same. This means that $M \cup M'$ is a mapping.

On the other hand, suppose that $M \cup M'$ is a mapping but the condition is violated, then there is a (r, s) in M' and a k such that either $left(r) < i_k$ and $left(s) > j_k$, or $right(r) < i_k$ and $right(s) > j_k$. Because base pairs do not share end bases and $M \cap M' = \emptyset$, we do not have to consider the case where, for some k, $left(r) = i_k$, $left(s) = j_k$, $right(r) = i_k$, or $right(s) = j_k$.

This means that there is some $(r_1, s_1) \in M$ such that (r, s) and (r_1, s_1) do not satisfy mapping conditions. Therefore $M \cup M'$ is not a mapping which is a contradiction. □

3 Algorithms

In this section, we assume that $R_1(P_1)$, $R_2(P_2)$, and M are given and $S_M = ((i_0, j_0), (i_1, j_1), \cdots (i_K, j_K))$.

Let M_1 be $\{s = (l, r) | l \neq r, \exists t,$ such that $(s, t) \in M\}$, and M_2 be $\{t = (l, r) | l \neq r, \exists s,$ such that $(s, t) \in M\}$. We now consider the case where at most one of the $P_1 - M_1$ and $P_2 - M_2$ is tertiary structure. We present an algorithm which solves this problem Our algorithm can be used for comparing tertiary structures in practical application.

3.1 Properties

We use a bottom-up approach. We consider smaller substructures first and eventually consider the whole structure.

We first consider how the given mapping M will limit the substructures we have to consider. We say i and j are consistent with respect to S_M if there is a k such that either $i = i_k$ and $j = j_k$, or $i_k < i < i_{k+1}$ and $j_k < j < j_{k+1}$.

For any $D(R_1[l_1..r_1], R_2[l_2..r_2])$ such that either l_1 and l_2 are not consistent or r_1 and r_2 are not consistent, there exists a k such that either $l_1 \leq i_k \leq r_1$ but $j_k < l_2$ or $j_k > r_2$, or $l_2 \leq j_k \leq r_2$ but $i_k < l_1$ or $i_k > r_1$. In both cases, $D(R_1[l_1..r_1], R_2[l_2..r_2])$ is not useful since one of the matching element in M has to be deleted.

We can now consider how to compute $D(R_1[l_1..r_1], R_2[l_2..r_2])$ where (l_1, l_2) and (r_1, r_2) are consistent with respect to S_M.

Let $S_1[1..m]$ and $S_2[1..n]$ be arrays containing pairs in $S(R_1, P_1 - M_1)[l_1..r_1]$ and $S(R_2, P_2 - M_2)[l_2..r_2]$, sorted by $3'$ end.

Let $S_1[i] = (s_1, t_1)$ and $S_2[j] = (s_2, t_2)$, $left_1[i]$ and $left_2[j]$, $cross\_left_1[i]$ and $cross\_left_2[j]$, and $cross\_weight_1[i]$ and $cross\_weight_2[j]$ are define as in [9].

We now define $D_1(i, j)$ and $D_2(i, j)$ as follows.

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

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

Lemma 6 *Suppose that $i = i_k$ and $j = j_k$, then* $D_1(i, j) = D_1(i - 1, j - 1)$

Proof: Because we require that M be part of the mapping between $R_1(P_1)$ and $R_2(P_2)$, the actual costs of elements in M do not affect the final solution. Therefore we can pre-calculate the cost of the constraints and assume that the cost is zero in our algorithm. □

Lemma 7 *Suppose that $i = i_k$ and $j_k < j < j_{k+1}$, then $D_1(i, j) = D_1(i, j - 1) + \gamma(\lambda \to label_{R_2}(S_2[j]))$*

Lemma 8 *Suppose that $i_k < i < i_{k+1}$ and $j = j_k$, then $D_1(i, j) = D_1(i - 1, j) + \gamma(label_{R_1}(S_1[i]) \to \lambda)$*

In the rest of section 3.1, we assume that there is no k such that $i = i_k$ or $j = j_k$.

Lemma 9 *Suppose that $S_1[i]$ and $S_2[j]$ are both base pairs and $S_1[i]$ and $S_2[j]$ are not compatible with S_M, then*

$D_1(i,j) = \min$

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

Proof: In this case, one can delete either $S_1[i]$ or $S_2[j]$ because matching $S_1[i]$ with $S_2[j]$ will violate mapping conditions. $\square$

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

$D_1(i,j) = \min$

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

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 $D_1(i,j) = D_1(i-1,j) + \gamma(label_{R_1}(S_1[i]) \rightarrow \lambda)$. 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 structures there is no crossing. Also if $S_1[i]$ and $S_2[j]$ are both single bases, both cross_weights are zero. $\square$

Lemma 11 *Suppose that $S_1[i]$ and $S_2[j]$ are both base pairs and $S_1[i]$ and $S_2[j]$ are compatible with S_M. 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 10. 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 $D_1(i,j) = D_1(i-1, j-1) + \gamma(label_{R_1}(S_1[i]) \rightarrow label_{R_2}(S_2[j]))$. $\square$

3.2 Algorithm

From the above lemmas, we can compute $D(R_1(P_1), R_2(P_2), M)$ using bottom up approach. Since we only use $D_2(i,j)$ in lemma 10, we only need to compute these $D(R_1[l_1..r_1], R_2[l_2..r_2], M)$ such that $(l_1, r_1) \in P_1$ and $(l_2, r_2) \in P_2$. Furthermore, by lemma 11, we can reduce the computation from each pair of base pairs to each pair of stems, where a stem in an RNA is a set of stack pairs of maximum size.

Given $R_1(P_1)$ and $R_2(P_2)$, 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 $L_1[i] = (l_1, r_1, k_1)$ and $L_2[j] = (l_2, r_2, k_2)$ such that (l_1, l_2) and $r_1, r_2)$ are consistent with S_M, we have to compute $D(R_1[l_1, r_1], R_2[l_2, r_2], M)$. We use lemmas 9, 10 and 11 to compute $D(R_1[l_1, r_1], R_2[l_2, r_2], M)$.

Let $R_1[1..m](P_1)$ and $R_2[1..n](P_2)$ 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. For the time complexity , the worst case would be that the constraint M is empty. The time to compute $D(R_1[l_1, r_1], R_2[l_2, r_2], M)$ 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.

4 Applications

In the real applications, the input usually contains secondary and tertiary interactions in the form of $R(S \cup T)$, where S is a set of base pairs with no crossing forming the secondary structure of R and T is a set of tertiary base pairs. However the number of tertiary interactions is always relatively small compared with the number of secondary interactions. In addition when comparing $R_1(S_1 \cup T_1)$ and $R_2(S_2 \cup T_2)$, one can assume that base pairs in S_1 should match to base pairs in S_2 and base pairs in T_1 should match to base pairs in T_2.

Using our algorithm, the following method computes the similarity when both RNAs contain tertiary interactions.

Method

0 Input: $R_1(S_1 \cup T_1)$ and $R_2(S_2 \cup T_2)$.

1 compute $D(R_1(S_1), R_2(S_2))$.

2 Let M be the optimal mapping from step 1. compute $D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2), M)$.

Given $R_1(S_1 \cup T_1)$ and $R_2(S_2 \cup T_2)$, we first compute $D(R_1(S_1), R_2(S_2))$. Since there is no tertiary interaction, we can compute the optimal solution and produce the optimal mapping M. We then use M as a constraint and compute $D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2), M)$.

Essentially this method tries to find the best secondary structure matching first and ignores the tertiary interactions. Once this matching is found and fixed, we then try to add in tertiary interactions by using this matching as a constraint and put in another set of base pairs which within themselves have no crossing but can have crossing with the constrained base pairs in M. Although this is not an optimal solution, in practice it would produce a reasonable result by matching most of the base pairs.

In the following, we assume that the cost of an operation on a base pair is no less than the sum of the costs of the same operation on the two bases in the base pair. This means that given base pairs r and s, we assume that $\gamma(r \to s) \geq \gamma(left(r) \to left(s)) + \gamma(right(r) \to right(s))$, $\gamma(r \to \lambda) \geq \gamma(left(r) \to \lambda) + \gamma(right(r) \to \lambda)$, and $\gamma(\lambda \to s) \geq \gamma(\lambda \to left(s)) + \gamma(\lambda \to right(s))$.

We now prove the following lemmas which give bounds of our method.

Lemma 12 *Given $R_1(S_1 \cup T_1)$ and $R_2(S_2 \cup T_2)$, then $D(R_1(S_1), R_2(S_2)) \leq D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2))$.*

Proof: Consider the best mapping between $R_1(S_1 \cup T_1)$ and $R_2(S_2 \cup T_2)$, we want to construct a mapping M between $R_1(S_1)$ and $R_2(S_2)$.

For any (r, s) where $r \notin S_1 \cup T_1$ and $s \notin S_2 \cup T_2$ (r and s are single bases), we add it to M. For any (r, s) where $r \in S_1$ and $s \in S_2$, we add it to M. For any (r, s) where $r \in T_1$ and $s \in T_2$, we can break the base pairs into single bases and put $left(r), left(s))$ and $(right(r), right(s))$ into M.

It is clear from the construction and our assumption that $\gamma(M) \leq D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2))$. Therefore $D(R_1(S_1), R_2(S_2)) \leq D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2))$. □

Lemma 13 *Given $R_1(S_1 \cup T_1)$ and $R_2(S_2 \cup T_2)$, let M be the optimal mapping between $R_1(S_1)$ and $R_2(S_2)$ from the computation of $D(R_1(S_1), R_2(S_2))$, then $D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2)) \leq D(R_1(S_1 \cup T_1), R_2(S_2 \cup T_2), M)$.*

We tested our method by doing experiments with five RNA structures taken from Ribonuclease P database [2]. Ribonuclease P is the ribonucleoprotein endonuclease that cleaves transfer RNA precursors, removing 5' precursor sequences and generating the mature 5' terminus of the tRNA.

We use a very simple score function, where for single bases $\gamma(a \to b) = 0$ if $a = b$, $\gamma(a \to b) = 1$ if $a \neq b$; and for base pairs, $\gamma(a \to b) = 0$ if $a = b$, $\gamma(a \to b) = 2$ if $a \neq b$.

The results are shown in Table 4 where for each pair of RNAs, there are two numbers. The bold face number indicates the edit distance measure using our method and the other number indicates the lower bound ($D(R_1(S_1), R_2(S_2))$) produced by ignoring tertiary interactions. In general, the lower bounds are very close which means that the measure produced by our method are very close to optimal. In some cases we actually reach the optimal since these two numbers are identical.

5 Conclusion

We have presented a new algorithm to compute similarity measure between RNA structures. We show that using this algorithm our method can compute the measure in practical application even when input are RNA tertiary structures.

	R_1	R_2	R_3	R_4	R_5
R_1	**0**	240 **244**	218 **229**	252 **253**	225 **226**
R_2		**0**	218 **237**	201 **202**	206 **210**
R_3			**0**	214 **217**	228 **233**
R_4				**0**	204 **210**
R_5					**0**

Table 1: Experimental results

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] J.W. Brown, 'The Ribonuclease P Database', *Nucleic Acids Research*, 27:314, 1999.

[3] 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

[4] 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

[5] B. Shapiro, 'An algorithm for comparing multiple RNA secondary structures', *Comput. Appl. Biosci* vol. 4, no. 3, pp.387-393, 1988

[6] B. Shapiro and K. Zhang, 'Comparing multiple RNA secondary structures using tree comparisons', *Comput. Appl. Biosci* vol. 6, no.4, pp.309-318, 1990

[7] 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.

[8] 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

[9] K. Zhang, L. Wang, and B. Ma, 'Computing similarity between RNA structures' *Proceedings of the Tenth Symposium on Combinatorial Pattern Matching. Springer-Verlag's Lecture Notes in Computer Science 1645*, pp. 281-293, 1999.

[10] M. Zuker, 'On finding all suboptimal foldings from an RNA molecule', *Science* 244, pp.48-52, 1989

[11] M. Zuker and D. Sankoff, 'RNA secondary structures and their prediction', *Bull. Math. Biol.* 46, pp.591-621, 1984