

Multiple RNA Structure Alignment

Zhuozhi Wang *

Experimental Therapeutics, Research
University Health Network
620 University Health Network, Suite 703
Toronto, Ontario, Canada
zwang@uhnres.utoronto.ca

Kaizhong Zhang †

Department of Computer Science
University of Western Ontario
London, Ontario, Canada

kzhang@csd.uwo.ca

Abstract

RNA structures can be viewed as a kind of special strings with some characters bonded with each other. The question of aligning two RNA structures has been studied for a while, and there are several successful algorithms that are based upon different models. In this paper, by adopting the model introduced in [18], we propose two algorithms to attack the question of aligning multiple RNA structures. We reduce the multiple RNA structure alignment problem to the problem of aligning two RNA structure alignments.

1. Introduction

RNAs (shorthand for Ribonucleic Acids) are a kind of relatively short biological sequences. Like DNA sequences, the RNA sequences consist of four kinds of bases, *adenine*, *cytosine*, *guanine* and *uracil*. Usually, we use four letters **A**, **C**, **G** and **U** to represent those four bases respectively. Thus, we can view an RNA sequence as a string over letter set {A, C, G, U}. In this paper, we will use base and character interchangeably.

RNAs are different from DNAs that they are usually in their single-stranded state, but when they are performing their functionalities, they tend to fold onto themselves forming higher-order structure by base pairing. It is this folded higher-order structure that determines how the RNAs play the game. Various kinds of base pairings have been detected, three of them occur more frequently than the others, **A-U (U-A)**, **G-C (C-G)**, and **U-G (G-U)**. The former two kinds are called *Watson-Crick* base pair and the latter

one is called *Wobble* base pair. Usually, we call those three kinds of base pairing *canonical* base pair. The real folded higher-order structure of an RNA sequence is called *tertiary structure* which is very difficult to obtain and display, therefore researchers proposed a simplified model called *secondary structure*. The secondary structure is a kind of planar structure, whereas the tertiary structure is a kind of three-dimension structure. We will see the differences of them in section 2 shortly.

Since 1970s, many research work have been conducted on comparing biological sequences, such as DNA sequences and proteins. There are also many papers addressing the problem of comparing protein structures. In terms of the number of papers, it is not overstated that comparing RNA structures has not received much attention in the Bioinformatics research field. To the best of our knowledge, most of the RNA research work focused upon predicting RNA secondary structures and comparing two RNA secondary structures. The answer of aligning multiple RNA structures was considered as a by-product of the algorithms of comparing two RNA secondary structures. For example, in [1, 2, 5, 13] the authors claimed to solve the problem of aligning multiple RNA structures, but the algorithmic details were not clear and the explanations concentrated on the pairwise RNA structure alignment.

In this paper, we will propose two heuristic algorithms to compare multiple RNA structures based on the computational model introduced in [18]. The major advantage of that model is that it allows us to compare RNA structures by following sequence alignment paradigm, resulting in simple and fast algorithms. And furthermore, some of the multiple sequence alignment algorithms can also be used to solve multiple RNA structure alignment problem. We will solve the multiple RNA structure alignment problem by solving a new problem "*aligning two RNA structure alignments*". This new question is similar to the question of "*aligning sequences alignments*" (or "*aligning alignments*") discussed in [3, 8, 12].

*This research was done when the author was a Ph.D candidate at University of Western Ontario.

†This research is supported in part by the Natural Sciences and Engineering Research Council of Canada under research grant No. OGP0046373, a research fellowship from Simon Fraser University and a Sharcnet research fellowship.

Why do we need to align multiple RNA structures? There are several reasons, the most important one is to determine the structure patterns in an RNA family. Here, we made the assumption that during the long time of revolution, the RNA sequences of an RNA family might have changed greatly; however, the shape of their tertiary structures might have been preserved, thus they have similar functionalities. Another application is that we can determine the frequently recurring substructures of an RNA sequence as discussed in [13]. The scenario is as the following: given an RNA sequence, we use existing folding program (e.g. *mfold* by Zuker) to generate a set of possible RNA secondary structures, and then we can determine the frequently occurred substructures of an RNA sequence by aligning those generated structures.

Traditionally, an RNA secondary structure was modeled as a tree, therefore comparing two RNA secondary structures becomes the question of comparing two trees [1, 2, 5, 13]. Thus the most important part for this kind of representation is how to compare two trees efficiently [20]. A major drawback of this representation is that it is hard to model RNA tertiary structure which is a graph. Later, in 1995, Bafna et al. proposed a set of algorithms to compare two RNA structures by treating the RNA secondary structures as a kind of special string (we call this *string-model*) [16], but the time complexity of the proposed algorithms is $O(n^4)$ which prohibits them from being applied in real world. Since then, several algorithms based upon variants of *string-model* have been proposed [6, 11, 15, 18, 19], all of which can solve the pairwise RNA structure alignment in reasonable time. At the same time, there are also some other proposed methods to compute pairwise RNA structure alignment, such as algorithm based upon *Stochastic Context-Free Grammar* [10]. Our model is a variant of *string-model*.

2. Preliminaries

In this section, we first give the definition of RNA secondary structure and then present the computational model we are about to adopt.

2.1. RNA Structures

We use R to represent an RNA sequence, r_i to represent the i th base in R , and $r_i \cdot r_j$ to represent a base pair occurred between bases r_i and r_j . We define an RNA secondary structure from a mathematics point of view, which is introduced in [14].

DEFINITION An RNA *secondary structure* is a set of base pairs, say S , which should satisfy the following conditions:

1. For any base pair $r_i \cdot r_j$, it must be one of the three *canonical* base pairs, $A-U$ ($U-A$), $G-C$ ($C-G$) or $G-U$ ($U-G$).
2. For any two base pairs $r_{i_1} \cdot r_{j_1}$ and $r_{i_2} \cdot r_{j_2}$, either $i_1 = i_2$ and $j_1 = j_2$, or $i_1 \neq i_2$, j_2 and $j_1 \neq i_2, j_2$.
3. If $h < i < j < k$, then S cannot contain both $r_h \cdot r_j$ and $r_i \cdot r_k$.
4. If S contains $r_i \cdot r_j$, then $|j - i| \geq 4$.

An RNA secondary structure should not contain *crossing* base pairs as required by 3, and a base can be paired with one and only one other base pairs as required by 2. For an RNA *tertiary structure*, conditions 1, 2 and 3 do not hold any more. Some unusual base pairs can be included in an RNA *tertiary structure* such as $A-A$, a base can pair with more than one bases which results in “*triple*”, and two base pairs can cross resulting in “*pseudoknot*”.

2.2. Sum-of-Pair Scoring Function

The multiple sequence alignment problem is not only hard in terms of computation, but also in terms of choosing scoring function. There is no accepted way to score the alignment of multiple sequences [9]. But there are some popular evaluation functions, *sum-of-pair* is one of the frequently used functions to score the multiple sequence alignment, which is defined as:

$$sp_score(A) = \sum_{i=1}^n ind_pair(A_i, A_j)$$

where A is an alignment, A_i and A_j are the i th and j th sequence in the alignment, and function *ind_pair* calculates the score of the *induced* pairwise alignment of A_i and A_j . The *induced alignment* of A_i and A_j is defined by taking A_i and A_j from alignment A , and deleting those opposing ‘-’ columns (e.g. $\begin{smallmatrix} - \\ - \end{smallmatrix}$). The multiple sequence alignment problem with respect to *sum-of-pair* is NP-Complete [17], thus we do not have polynomial time solutions to this problem under our current knowledge. In this paper, we use *sum-of-pair* to evaluate how well the alignment of multiple RNA structures is.

2.3. Computational Model of RNA Structure Alignment

When determining how similar two sequences are, one of the most popular measures is to use *Levenshtein distance* or *edit distance*. Given two sequences s and t , we allow three operations to be operated on those sequences, *deletion*, *insertion* and *substitution*. A deletion allows us to delete a character from s , an insertion allows us to insert

a character of t into s , and a substitution allows us to substitute a character of s with another one of t . The edit distance problem is to ask for the minimum number of edit operations transforming s into t .

We would like to extend those edit operations on RNA structures. In addition to single bases, we allow them to be operated upon base pairs in an RNA structure. A deletion of a base pair in an RNA structure results in deletion of those two bases that consist of the base pair, an insertion of a base pair results in inserting both of the two bases consisting of the base pair, and a substitution of a base pair means to replace both of the bases by another pair of bases. Figure 1 shows some of the possible operations that are operated on the pairwise RNA structure alignment.

The *edit distance* is a good measure to evaluate how similar two sequences are. However, biologists believe that a mutation event occurred on a sequence usually involves k ($k \geq 1$) characters, which is modeled by the notation of *gap*. A *gap* in an alignment is the maximum run of spaces ('-') in a sequence [9]. *Affine gap penalty* is one of the effective ways to model the deletion or insertion of k characters. Generally, we use a function like $a + b \times l$ to model the affine gap penalty, where a is the initiation penalty of the gap, b is the extension penalty of the gap, and l is the length of the gap.

Let \mathcal{A} be an alignment of multiple RNA structures, we use $\mathcal{A}[i]$ to represent the i th column of \mathcal{A} , $\mathcal{A}[i, j]$ to represent the columns from $\mathcal{A}[i]$ to $\mathcal{A}[j]$ inclusively, \mathcal{A}_k to represent the k th structure of \mathcal{A} , $\mathcal{A}_k[i]$ to represent the i th character of the k th sequence of \mathcal{A} and $|\mathcal{A}|$ to represent the length of alignment \mathcal{A} .

DEFINITION Given k RNA structures S_1, S_2, \dots, S_k , we define the alignment of the k RNA structures $(S'_1, S'_2, \dots, S'_k)$ as following:

1. Each S'_i is S_i with some '-' inserted, and $|S'_1| = |S'_2| = \dots = |S'_k| = m$. We juxtapose each structure S'_i in a row, so that each element of S'_i occupies a different column, thus we have a $k \times m$ matrix.
2. At position i , for each structure S'_t ($1 \leq t \leq k$), either $S'_t[i]$ is a base or a space '-'. We do not allow a column consisting entirely of spaces to exist.
3. At positions i and j , if for some structure S'_t , $(S'_t[i], S'_t[j])$ is a base pair, then for any other structures S'_t ($1 \leq t \leq k$ and $t \neq l$), either $(S'_t[i], S'_t[j])$ is a base pair too, or $S'_t[i]=S'_t[j]='-'$.

Constraint 3 requires that a base pair should only be aligned with a base pair or ('-', '-'). Our object is to find an alignment of S_1, S_2, \dots, S_n which minimizes the sum of pair score among all the possible alignments of

S_1, S_2, \dots, S_n . There is no doubt that this question is NP-Complete, since we can view an RNA sequence as an RNA secondary structure without any base pairings.

For an alignment \mathcal{A} , if all the characters contained in $\mathcal{A}[i]$ are '-', we call that column *space column*. If all the characters contained in $\mathcal{A}[i]$ are either single base or '-', then we call $\mathcal{A}[i]$ *single column* or simply *s-column*. Two columns $\mathcal{A}[i]$ and $\mathcal{A}[j]$ are called *pairing column* or *p-column* if for $1 \leq t \leq K$, either $(\mathcal{A}_t[i], \mathcal{A}_t[j])$ is a base pair or ('-', '-'). Suppose that $i < j$, we call $\mathcal{A}[i]$ *5'-end column* and $\mathcal{A}[j]$ *3'-end column*.

To decrease the difficulty of the problem, we would like to reduce the *multiple RNA structure alignment problem* to *aligning two RNA structure alignments* similar to the question discussed in [3, 8, 12], we define the question of *aligning two RNA alignments* as following.

DEFINITION Suppose that we are given two alignments of RNA structures \mathcal{A} and \mathcal{B} , the alignment of them is represented by $(\mathcal{A}', \mathcal{B}')$, which should satisfy the following constraints:

1. \mathcal{A}' is \mathcal{A} with some space columns inserted, \mathcal{B}' is \mathcal{B} with some space columns inserted, and $|\mathcal{A}'| = |\mathcal{B}'|$.
2. If $\mathcal{A}'[i]$ is a single column of \mathcal{A}' , then $\mathcal{B}'[i]$ of \mathcal{B}' is either a single column or a space column; if $\mathcal{B}'[j]$ is a single column of \mathcal{B}' , then $\mathcal{A}'[j]$ of \mathcal{A}' is either a single column or a space column.
3. If $\mathcal{A}'[i]$ and $\mathcal{A}'[j]$ are pairing columns of \mathcal{A}' , then $(\mathcal{B}'[i], \mathcal{B}'[j])$ of \mathcal{B}' are either pairing columns or two space columns; if $\mathcal{B}'[i]$ and $\mathcal{B}'[j]$ are pairing columns of \mathcal{B}' , then $(\mathcal{A}'[i], \mathcal{A}'[j])$ of \mathcal{A}' are either pairing columns or two space columns.

Suppose that there are K structures in \mathcal{A} and L structures in \mathcal{B} , the quality of $(\mathcal{A}', \mathcal{B}')$ is defined as following:

$$Q((\mathcal{A}', \mathcal{B}')) = \sum_{i=1}^K \sum_{j=1}^L \text{ind_pair}(\mathcal{A}'_i, \mathcal{B}'_j)$$

That is the quality is the sum of the pairwise alignment score of all possible pairwise combinations between \mathcal{A}' and \mathcal{B}' . It is easy to see that if none of the RNA structures contains any base pairs, then this question is reduced to the question of aligning two sequence alignments which has been shown to be NP-Complete [3]. Thus there are no polynomial time solutions to this question at present time either.

The hardest part of this question is how to determine the gap opening exactly. Traditionally, when we are calculating the alignment, we only look at two columns, the current column and the previous one in order to determine gap openings [7]. This is enough for aligning two sequences, however, when we are considering to align two alignments,

Lemma 1 For any $i_1 \leq i \leq i_2$ and $j_1 \leq j \leq j_2$,

$$D(i_1, i; j_1, j) = \delta(i, 0) + \min \left\{ \begin{array}{l} D(i_1, i-1; j_1, j) + \\ \gamma \times \begin{cases} 0 & -opt. \\ L \times h_{01}^1(i) & -pes. \end{cases} \\ I(i_1, i-1; j_1, j) + \\ \gamma \times \begin{cases} h_1^1(i) \times h_1^2(j) & -opt. \\ L \times h_1^1(i) & -pes. \end{cases} \\ M(i_1, i-1; j_1, j) + \\ \gamma \times \begin{cases} h_1^1(i) \times h_1^2(j) & -opt. \\ h_1^1(i) \times h_1^2(j) + \\ h_{01}^1(i) \times h_0^2(j) & -pes. \end{cases} \end{array} \right.$$

Proof: Before we prove the correctness of this equation, we first prove that the gap opening function is correct. There are three kinds of possible operations preceding the current deletion.

1. Suppose that there is a deletion ending at $(i_1, i-1; j_1, j)$, then we have both column $\mathcal{A}[i-1]$ and column $\mathcal{A}[i]$ aligned with space columns. We thus have situations shown in figure 2 (suppose that the top sequence is from \mathcal{A} , the bottom sequence is from \mathcal{B}). It is easy to see that we only need to deal with case c. In pessimistic definition it is a gap opening, but in optimistic definition it is not. It is not difficult to verify that the gap opening is correct given the meaning of function h_{01}^1 .

x x	x -	- x	- -
- -	- -	- -	- -
(a)	(b)	(c)	(d)

Figure 2. A deletion precedes a deletion

2. Suppose that there is an insertion ending at $(i_1, i-1; j_1, j)$, therefore we have column $\mathcal{B}[j]$ inserted and column $\mathcal{A}[i]$ deleted, we then have the cases described in figure 3.

- x	- -	- x	- -
- -	x -	x -	- -
(a)	(b)	(c)	(d)

Figure 3. An insertion precedes a deletion

Case c is definitely a gap opening, case a is different in different definitions. It is not hard to verify that the gap opening functions are correct. Note that gap opening function for pessimistic is a combination of cases a and c.

3. Suppose that there is a substitution ending at $(i_1, i-1; j_1, j)$, then we have column $i-1$ and j matched up with each other. For any two structures, the ending cases may look like one of the cases in figure 4.

- x	- -	- x	- -
- -	x -	x -	- -
(a)	(b)	(c)	(d)
x x	x -	x x	x -
- -	x -	x -	- -
(e)	(f)	(g)	(h)

Figure 4. A substitution precedes a deletion

It is clear that cases b, d, e, f, h do not open a gap, cases c and g certainly open a gap, and case a is the ambiguous case. Cases c and g can be generalized as the number of non-space characters in i multiplies the number of non-space characters in j . Therefore the computation of gap opening is correct.

We now turn our attention to column $\mathcal{A}[i]$ itself. If column $\mathcal{A}[i]$ is a single column, there is no problem of deleting it. If column $\mathcal{A}[i]$ is a 3' end pairing column, then we claim that its 5' end pairing column $\mathcal{A}[pr(i)]$ has been deleted, since when we consider $\mathcal{A}[pr(i)]$, $\mathcal{A}[i]$ is outside interval $[i_1, pr(i)]$, which means we have deleted that column. Similarly, if $\mathcal{A}[i]$ is 5' end pairing column, its corresponding 3' end column is outside $[i_1, i]$, thus we need to delete it.

The minimum of the three possible operations preceding the current deletion gives us the best alignment when the deletion is the last operation at (i, j) . \square

Lemma 2 For $i_1 \leq i \leq i_2$ and $j_1 \leq j \leq j_2$,

$$I(i_1, i; j_1, j) = \delta(0, j) + \min \left\{ \begin{array}{l} D(i_1, i; j_1, j-1) + \\ \gamma \times \begin{cases} h_1^1(i) \times h_1^2(j) & -opt. \\ K \times h_1^2(j) & -pes. \end{cases} \\ I(i_1, i; j_1, j-1) + \\ \gamma \times \begin{cases} 0 & -opt. \\ K \times h_{01}^2(j) & -pes. \end{cases} \\ M(i_1, i; j_1, j-1) + \\ \gamma \times \begin{cases} h_1^1(i) \times h_1^2(j) & -opt. \\ h_1^1(i) \times h_1^2(j) + \\ h_0^1(i) \times h_{01}^2(j) & -pes. \end{cases} \end{array} \right.$$

Proof: Similar to lemma 1. \square

Lemma 3 For $i_1 < i < i_2$ and $j_1 < j < j_2$, if both i and j are s-columns,

$$M(i_1, i; j_1, j) = \delta(i, j) + \min \begin{cases} D(i_1, i-1; j_1, j-1) + \\ \text{gapDM}(i, j) \\ I(i_1, i-1; j_1, j-1) + \\ \text{gapIM}(i, j) \\ M(i_1, i-1; j_1, j-1) + \\ \text{gapMM}(i, j) \end{cases}$$

if both i and j are 3' end p-column and $i_1 \leq pr(i) < i$ and $j_1 \leq pr(j) < j$,

$$M(i_1, i; j_1, j) = \delta(i, j) + \min \begin{cases} D(i_1, pr(i)-1; j_1, pr(j)-1) + \\ \text{gapDM}(pr(i), pr(j)) \\ I(i_1, pr(i)-1; j_1, pr(j)-1) + \\ \text{gapIM}(pr(i), pr(j)) \\ M(i_1, pr(i)-1; j_1, pr(j)-1) + \\ \text{gapMM}(pr(i), pr(j)) \end{cases} + \min \begin{cases} D(pr(i)+1, i-1; pr(j)+1, j-1) + \\ \text{gapDM}(i, j) \\ I(pr(i)+1, i-1; pr(j)+1, j-1) + \\ \text{gapIM}(i, j) \\ M(pr(i)+1, i-1; pr(j)+1, j-1) + \\ \text{gapMM}(i, j) \end{cases}$$

otherwise, we set $M(i_1, i; j_1, j)$ to a very large number.

We calculate $\text{gapXM}(s, t)$ as following:

$$\begin{aligned} \text{gapDM}(s, t) &= \begin{cases} h_{10}^1(s) \times h_1^2(t) & -opt. \\ h_0^1(s) \times h_1^2(t) + h_{01}^1(s) \times h_0^2(t) & -pes. \end{cases} \\ \text{gapIM}(s, t) &= \begin{cases} h_1^1(s) \times h_{10}^2(t) & -opt. \\ h_1^1(s) \times h_0^2(t) + h_0^1(s) \times h_{01}^2(t) & -pes. \end{cases} \\ \text{gapMM}(s, t) &= \begin{cases} h_1^1(s) \times h_{10}^2(t) + h_{10}^1(s) \times h_1^2(t) & -opt. \\ h_1^1(s) \times h_{10}^2(t) + h_{10}^1(s) \times h_1^2(t) \\ + h_{00}^1(s) \times h_{01}^2(t) + h_{01}^1(s) \times h_{00}^2(t) & -pes. \end{cases} \end{aligned}$$

Proof: For any two structures, one from \mathcal{A} and the other from \mathcal{B} , we need to consider all sixteen combinations ending at (i, j) in order to calculate the gap openings. We omit the correctness proof of gap opening.

We now consider the optimal alignment between $\mathcal{A}[i_1, i]$ and $\mathcal{B}[j_1, j]$ where the substitution is the last operation. There are three cases that we need to consider: 1. both i and j are s-columns; 2. both i and j are 3' end p-columns and $i_1 \leq pr(i) < i$ and $j_1 \leq pr(j) < j$; 3. Other cases.

For case 1, it is easy to see, there is no problem of aligning $\mathcal{A}[i_1, i]$ with $\mathcal{B}[j_1, j]$. Other columns should not potentially affect our calculation.

For case 2, we know that $(pr(i), i)$ and $(pr(j), j)$ are base pairing columns in \mathcal{A} and \mathcal{B} respectively. The alignment of

$\mathcal{A}[i_1, i]$ and $\mathcal{B}[j_1, j]$ is broken into three parts: $(i_1, pr(i)-1; j_1, pr(j)-1)$, $(pr(i)+1, i-1; pr(j)+1, j-1)$ and pairing columns $(pr(i), i)$ and $(pr(j), j)$. We can easily verify that other base pairing columns will not affect our calculation. Since we only need two columns to determine the gap opening penalty, therefore the value of the alignment between $\mathcal{A}[pr(i)+1, i-1]$ and $\mathcal{B}[pr(j)+1, j-1]$ is correct under the assumption of *pessimistic* or *optimistic* gap opening strategy.

Case 3 contains all the other cases that we have not considered:

- Column i is a 5' end p-column, we do not allow it to be aligned with any column of \mathcal{B} .
- Column j is a 5' end p-column, we do not allow it to be aligned with any column of \mathcal{A} .
- Column i is 3' end p-column, but j is not; or j is 3' end p-column but $j_1 > pr(j)$. In this case, we need to align i with space column.
- Column j is 3' end p-column, but i is not; or i is 3' end p-column but $i_1 > pr(i)$. In this case, we need to align j with space column.

All these cases are handled by the "otherwise" statement in the lemma which prohibits them from occurring. We now draw our conclusion that $M(i_1, i_2; j_1, j_2)$ is correctly calculated. \square

Lemma 4 For $i_1 \leq i \leq i_2$, we need to consider two cases for $D(i_1, i; \phi)$, $i = i_1$ and $i_1 < i \leq i_2$,

$$D(i_1, i; \phi) = \delta(i, 0) + D(i_1, i-1; \phi) + \begin{cases} \gamma \times \begin{cases} h_1^1(i) \times h_1^2(j_1-1) & -opt. \\ h_1^1(i) \times h_1^2(j_1-1) + \\ h_{01}^1(i) \times h_0^2(j_1-1) & -pes. \end{cases} & i = i_1 \\ \gamma \times \begin{cases} 0 & -opt. \\ L \times h_{01}^1(i) & -pes. \end{cases} & i_1 < i \leq i_2 \end{cases}$$

Proof: Since we have columns $\mathcal{A}[i_1-1]$ and $\mathcal{B}[j_1-1]$ matched up with each other, and the gap opening method for the case MD is different from the case DD , thus we need two definitions. It is easy to list all the possible cases as we did in lemma 1 to verify the correctness of the equation. \square

Lemma 5 For $j_1 < j \leq j_2$, we need to consider two cases for $I(\phi; j_1, j)$, $j = j_1$ and $j_1 < j \leq j_2$,

$$I(\phi; j_1, j) = I(\phi; j_1, j-1) + \delta(0, j) + \begin{cases} \gamma \times \begin{cases} h_1^1(i_1-1) \times h_1^2(j) & -opt. \\ h_1^1(i_1-1) \times h_1^2(j) + \\ h_0^1(i_1-1) \times h_{01}^2(j) & -pes. \end{cases} & j = j_1 \\ \gamma \times \begin{cases} 0 & -opt. \\ K \times h_{01}^2(j) & -pes. \end{cases} & j_1 < j \leq j_2 \end{cases}$$

Proof: Similar to lemma 4. □

There are no definitions of $D(\phi; j_1, j)$ and $M(\phi; j_1, j)$ when $j_1 \leq j \leq j_2$, and $I(i_1, i; \phi)$ and $M(i_1, i; \phi)$ when $i_1 \leq i \leq i_2$. We assign them as the following,

$$D(\phi; j_1, j) = I(\phi; j_1, j) + K \times L \times \gamma + \delta(i_1, 0) \quad (1)$$

$$M(\phi; j_1, j) = I(\phi; j_1, j) + K \times L \times \gamma + \delta(i_1, 0) \quad (2)$$

$$M(i_1, i; \phi) = D(i_1, i; \phi) + K \times L \times \gamma + \delta(0, j_1) \quad (3)$$

$$I(i_1, i; \phi) = D(i_1, i; \phi) + K \times L \times \gamma + \delta(0, j_1) \quad (4)$$

Let R and S be the lengths of structure alignments \mathcal{A} and \mathcal{B} respectively, and P and Q be the number of pairs of pairing columns of structure alignments \mathcal{A} and \mathcal{B} respectively. We could use the following bottom-up algorithm to compute the alignment of \mathcal{A} and \mathcal{B} .

Algorithm for computing alignment of \mathcal{A} and \mathcal{B} :

Sort the pairing columns of \mathcal{A} and \mathcal{B} by 3' end in increasing order respectively;

for each pair of pairing columns (pl, pr) in \mathcal{A}
for each pair of pairing columns (ql, qr) in \mathcal{B}
Compute the alignment of $\mathcal{A}[pl + 1, pr - 1]$ and $\mathcal{B}[ql + 1, qr - 1]$ by using lemmas 1 to 5 and equations 1 to 4, and store the value in an array

Compute $\mathcal{A}[1, R]$ and $\mathcal{B}[1, S]$ and output $\min\{M(1, R; 1, S), D(1, R; 1, S), I(1, R; 1, S)\}$

Trace back arrays M, I and D to find the alignment of \mathcal{A} and \mathcal{B}

Theorem 1 We could compute the alignment of two RNA secondary structure alignments \mathcal{A} and \mathcal{B} optimally under the definitions of pessimistic and optimistic assumption in time $O(R \times S \times P \times Q)$ and space $O(R \times S + R \times K + S \times L)$.

Proof: For two RNA secondary structure alignments, there are no crossing p-columns in them, so the above algorithm can compute the alignment of two RNA secondary structure alignments \mathcal{A} and \mathcal{B} optimally under the pessimistic and optimistic gap opening definitions. Each of arrays D, I and M occupies space $O(R \times S)$. We also need some arrays to hold the values calculated by function h , we need twelve arrays, six for each alignment. The space complexity is $O(6 \times K \times R + 6 \times L \times S)$. Thus the space complexity is $O(R \times S + R \times K + S \times L)$. The pre-computation of function h needs time $O(K \times R + L \times S)$. The time complexity for the computation of all combinations of pairing columns is $O(P \times Q \times R \times S)$ since the computation of $\mathcal{A}[pl + 1, pr - 1]$ and $\mathcal{B}[ql + 1, qr - 1]$ needs at most $O(R \times S)$. □

Note that if both of \mathcal{A} and \mathcal{B} contains crossing p-columns, the algorithm could not compute the optimal answer.

4. A More Accurate Algorithm

In this section, we are to discuss another algorithm which can help us to determine the gap openings in the question of aligning alignments more accurately. The technique used in this section is similar to the one in [3]. However, some new issues have arisen, in particular, the computational time complexity could be much higher in some situations.

The main idea is that instead of looking back just two consecutive columns as we did in the last section, we look back the whole gap to determine a gap opening. To achieve this, for each alignment \mathcal{A} and \mathcal{B} , we introduce three arrays to store the lengths of the gap opened until the current column (but not including the current column). For example, suppose that we have an alignment as following:

```
AGCG---GC-----GG
AGCGG-GGCUUUU--G
          *       #
```

At positions marked by * and #, we know that the first sequence has gaps of lengths 2 and 5 respectively, and the second sequence has gaps of length 1 at each position, by comparing these values and characters at * and #, we could determine that there is no gap opening at column * and a gap opening at column #. When solving the question of aligning two sequence alignments, we used three auxiliary arrays for each alignment. The reason is that we would like to record the gap lengths of each alignment \mathcal{A} and \mathcal{B} when a deletion, insertion or substitution ends at (i, j) .

This technique works well for sequence alignment, but if we apply it to aligning two RNA structures, we will encounter difficulties. Let's consider the example in figure 5, in which (a) and (b) are two alignments.

```
U--GC-----A   U-GAA-----A   --GC-----
UCGGCUUUUUUA   UCGAAGGUUUUA   -GAA-----
**3      9**    **3      9**    3      9
(a)                (b)                (c)
```

Figure 5. An incorrect case

Suppose that the columns marked by * are base pairing columns. When we want to calculate the alignment between $\mathcal{A}[3, 9]$ and $\mathcal{B}[3, 9]$, let's consider the first sequence from \mathcal{A} and \mathcal{B} . Since we assume that the second column in each alignment should be matched up with each other, we have a situation like case c in the above figure. There is not enough information for us to determine if there is a gap opening at the third column. This is because we are using a bottom-up algorithm to compute the alignment, when we are breaking the structures into small pieces, the gaps might have been

Table 1. Some experimental results of multiple RNA structure alignment

	3	4	6	8
real	1342	2130	5172	9760
opt	1344	2147	5387	9975
pes	1357	2141	5231	9895

broken into parts too, therefore some structure information is lost. To calculate the optimal solution, when the algorithm comes across this situation, it needs to recompute the alignment of those segments having such problem. One can imagine how slow this would be in the worst case. However, if one of the RNA structure alignment contains only one structure, this algorithm is guaranteed to find the optimal solution.

One way to handle this problem is, for the first time, if the recalculation of alignment $\mathcal{A}[i_1, i_2]$ and $\mathcal{B}[j_1, j_2]$ is needed, the algorithm recalculates the alignment, and at the same time, it records gap length information before $\mathcal{A}[i_1]$ and $\mathcal{B}[j_1]$. Next time when the recalculation occurs again, the algorithm needs to check if gap opening information at $\mathcal{A}[i_1]$ and $\mathcal{B}[j_1]$ is the same as before or not, if the information recorded is the same, then we will not recompute the alignment, otherwise the algorithm recomputes it and updates the gap opening information. Another way to compromise the question is that when we find the ambiguous situations, we still use the incorrect values, but we need to adjust the alignment score. We record the last operation in the sub-alignment $\mathcal{A}[pr(i) + 1, i - 1]$ and $\mathcal{B}[pr(j) + 1, j - 1]$ that achieves the optimal, we perform a traceback operation, and recalculate the gap opening value at $\mathcal{A}[pr(i)]$ and $\mathcal{B}[pr(j)]$ by using the correct gap length information. The alignment of $\mathcal{A}[pr(i) + 1, i - 1]$ and $\mathcal{B}[pr(j) + 1, j - 1]$ might not be the best one, however, this can not only save us some time but also output a reasonable alignment. In our implementation, we used the latter method.

5. Conclusions

We have described two algorithms to compute the alignment of two RNA structures, we call the algorithms in section 3 *pessimistic* and *optimistic* algorithm respectively depending upon the gap opening strategy, and the one in section 4 *realistic* algorithm. We have adopted a computation model following the sequence alignment paradigm to compute the RNA structure alignments. We tested our algorithms on datasets containing 3, 4, 6, and 8 RNA structures respectively, which we obtained from [4]. The results are shown in table 1. For each deletion/insertion/mismatch of a single base, we charged 1; and for each deletion/insertion/mismatch of a base pair, we charged 2; and

we assigned 4 to the gap opening penalty. We used a MCST (minimum cost spanning tree)-like algorithm to align those structures. Given n sequences, we construct a complete graph as following: we consider each sequence as a vertex, and for each pair of sequences, we calculate the alignment score and assign the pairwise alignment score to the corresponding edge as weight. Then the algorithm simulates the Kruskal's Minimum Cost Spanning Tree algorithm, each time it picks up the shortest edge (which means two closest sequences or alignments), if the two vertices are in different groups (namely two alignments), the algorithm combines the two vertices or groups (namely align two alignments). The algorithm stops when all the vertices are grouped together (namely all the sequences are in the same alignment).

It can be seen from the table that each time, the *realistic* algorithm (abbreviated as *real* in the table) outputs the best alignment, the *optimistic* (abbreviated as *opt*) algorithm outputs the worst alignment, and the *pessimistic* (abbreviated as *pes*) algorithm outputs an alignment close to *realistic*'s result. However, the *realistic* algorithm is complicated and time-consuming if we want to find the optimal solution under its definition. Thus we recommend use of *pessimistic* algorithm. The reason why *pessimistic* algorithm is better than *optimistic* algorithm is that the later algorithm favors short gap openings resulting many gaps in the alignment.

It is interesting to notice that under our computational model, the triangle inequality property holds. An informal proof is as following: when there are no occurrences of base pairs, the comparison of two RNA structures is exactly the same as the comparison of two sequences; when there are base pairs, the algorithm is actually finding a way to combine the computed substructures into one structure, no special values are added. Since all the data are positive and addition keeps the triangle inequality property, thus inductively, the triangle inequality holds for entire period of computing the alignment of RNA structures. With this property, many existing multiple sequence alignment algorithms can be applied, such as *center star alignment* and the performance of the algorithm is guaranteed.

We have noticed that if one of the given alignments contains only one structure, we have the opportunity to compute the optimal alignment between \mathcal{A} and \mathcal{B} . This is for our future work.

Acknowledgement

We thank the anonymous referees for their helpful suggestions.

References

- [1] B. A. Shapiro. An Algorithm for Comparing Multiple RNA Secondary Structures. *CABIO*, 4(8):387–393, 1988.
- [2] B. A. Shapiro and K. Zhang. Comparing Multiple RNA Secondary Structures Using Tree Comparisons. *CABIO*, 6:309–318, 1990.
- [3] B. Ma, Z. Wang and K. Zhang. Alignment between Two Alignments. In *Proceedings of the 14th Symposium on Combinatorial Pattern Matching*, pages 254–265, 2003.
- [4] J. W. Brown. The Ribonuclease P Database. *Nucleic Acids Research*, 27:314, 1999.
- [5] C. Chevalet and B. Michot. An Algorithm for Comparing RNA Secondary Structures and Searching for Similar Substructures. *CABIO*, 8(3):215–225, 1992.
- [6] G. Collins, S. Le and K. Zhang. A New Method for Computing Similarity Between RNA Structures. In *Proceedings of the 2nd International Workshop on Biomolecular Informatics*, pages 761–765, 2000.
- [7] O. Gotoh. An Improved Algorithm for Matching Biological Sequences. *Journal of Molecular Biology*, 162:705–708, 1982.
- [8] O. Gotoh. Optimal Alignment Between Groups of Sequences and its Application to Multiple Alignment. *CABIOS*, 9(3):361–370, 1993.
- [9] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.
- [10] I. Holmes and G. M. Rubin. Pairwise RNA Structures Comparison with Stochastic Context-Free Grammars. In *Proceedings of Pacific Symposium on Biocomputing*, pages 163–174, 2002.
- [11] K. Zhang, L. Wang and B. Ma. Computing Similarity between RNA Structures. In *Proceedings of the 10th Symposium on Combinatorial Pattern Matching*, pages 281–293, 1999.
- [12] J. D. Kececioğlu and W. Zhang. Aligning Alignments. In *Proceedings of the 9th Symposium on Combinatorial Pattern Matching 98*, pages 180–208, 1998.
- [13] S. Le, J. Owen, R. Nussinov, J. Chen, B. A. Shapiro and J. V. Maizel. RNA Secondary Structures: Comparison and Determination of Frequently Recurring Substructures by Consensus. *CABIO*, 5(3):205–210, 1989.
- [14] D. Sankoff and J. B. Kruskal. *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison*. Addison Wesley, 1983.
- [15] T. Jiang, G. Lin, B. Ma, and K. Zhang. A General Edit Distance between Two RNA Structures. *Journal of Computational Biology*, 9(2):371–388, 2002.
- [16] V. Bafna, S. Muthukrishnan and R. Ravi. Computing Similarity between RNA Strings. In *Proceedings of the 6th Symposium on Combinatorial Pattern Matching*, pages 1–16, 1995.
- [17] L. Wang and T. Jiang. On the Complexity of Multiple Sequence Alignment. *Journal of computational Biology*, 1:337–348, 1994.
- [18] Z. Wang and K. Zhang. Alignment between Two RNA Structures. In *Proceedings of the 26th Symposium on Mathematical Foundations of Computer Science*, pages 690–702, 2001.
- [19] K. Zhang. Computing Similarity between RNA Secondary Structures. In *Proceedings of IEEE International Joint Symposium on Intelligence and Systems*, pages 126–132, 1998.
- [20] K. Zhang and D. Shasha. Simple Fast Algorithms for the Editing Distance between Trees and Related Problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.