Aligning Two RNA Secondary Structures with l-Block

Zhuozhi Wang* and Elisabeth R. M. Tillier
Ontario Cancer Institute, University Heath Network
620 University Health Network, Suite 703
Toronto, Ontario, Canada
zwang@uhnres.utoronto.ca, e.tillier@utoronto.ca

1 Introduction

Ribonucleic Acid (RNA) sequences play numerous important roles in biology. To perform their functions, RNA sequences fold into stable higher-order structures by forming intra-strand base pairings. Generally, an RNA sequence can be viewed as a sequence of characters over the alphabet set {A, C, G, U}, and the base pairing occurs between the bases A and U, G and C, or G and U by hydrogen bonding. An RNA secondary structure consists of both paired bases and single bases.

Although RNA sequences can change throughout evolution, the higher-order structures that determine its function are maintained by selection. Therefore, to determine the similarity between two RNA sequences, it is better to compare their structures. The question of aligning two RNA secondary structures has been studied for some time [1, 3, 4, 5, 6, 8, 9, 10, 11]. Some of these methods require the specification of a gap opening penalty, but it is not clear how much we should assign to it for the RNA secondary structure alignment.

In this paper, we would like to propose an algorithm to align two RNA secondary structures, in which we get around the question of assigning gap opening penalty. We adopt the idea first proposed in [2] by David Sankoff, for any two given RNA secondary structures A and B, we would like to find an optimal alignment of them with exactly $l(l > 0)$ blocks which only consist of substitutions (no gaps are allowed in the blocks). Thus we transform the question of assigning gap penalty to the question of finding exact number of blocks.

*To whom correspondence should be addressed.

2 Basic Definitions

Before proceeding we first introduce some basic concepts and the computation model. For our convenience, we use A and B to represent two RNA secondary structures, A_i to represent the ith base in structure A, (A_i, A_j) to represent a base pair (if the context is clear, we only use (i, j)), $A_{[i,j]}$ to represent the bases in A indexed from i to j inclusively.

Concept 1: *(RNA Secondary Structure)* An RNA *secondary structure* of an RNA sequence A is a set of base pairs, say S, which should satisfy the following conditions:

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

In an RNA secondary structure, if A_i pairs with A_j, we define $p(i) = j$ and $p(j) = i$. The smaller of the two values i and j is called the $5'$ end of a base pair, the larger one is called the $3'$ end. If A_i is a single base (it does not pair with any other bases), then we define $p(i) = i$.

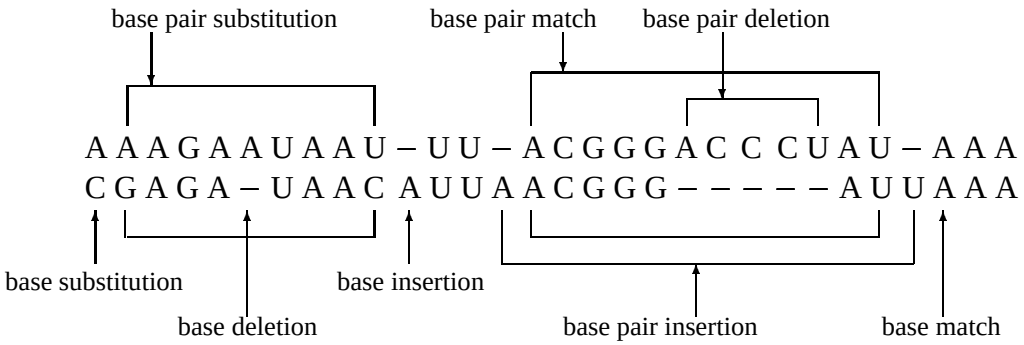

Figure 1. Pairwise RNA structure alignment and the edit operations

We adopt the computation model introduced in [10], in which the edit operations are extended to be operated upon RNA base pairs. An RNA base pair is treated as an entity. We have *single base insertion, deletion* and *substitution*, and *base pair insertion, deletion* and *substitution*. When a deletion or an insertion is applied to a base pair, both of the bases consisting of the base pair are deleted or inserted. A substitution of a base pair results in substituting both of the bases. Figure 1 shows the possible operations applied to an RNA secondary structure alignment.

Concept 2: *(Global RNA Secondary Structure Alignment)* Given two RNA secondary structures A and B, we define the alignment (A', B') of A and B as following:

1. A' is A and B' is B except some spaces ('$-$') are inserted and $|A'| = |B'|$.
2. If A'_i is a single base in A', then either B'_i is a single base or a space ('$-$'); and vice versa.
3. If (A'_i, A'_j) is a base pair in A', then either (B'_i, B'_j) is a base pair or both of them are spaces ('$-$'); and vice versa.
4. There are no opposing spaces in the alignment, e.g. $\frac{-}{-}$.

The optimal alignment of A and B is the one which has the maximum score over all possible alignments of A and B. Let $S(A', B')$ be the score of alignment (A', B'), we define $S(A', B')$ as:

$$\begin{aligned} S(A', B') = & del\_single(A', B') + ins\_single(A', B') + \\ & sub\_single(A', B') + del\_pair(A', B') + \\ & ins\_pair(A', B') + sub\_pair(A', B') \end{aligned}$$

where $del\_single(A', B')$, $ins\_single(A', B')$ and $sub\_single(A', B')$ are the total scores of deletion, insertion and substitution of the single bases in the alignment (A', B') respectively; and $del\_pair(A', B')$, $ins\_pair(A', B')$ and $sub\_pair(A', B')$ are the total scores of deletion, insertion and substitution of the base pairs in the alignment (A', B') respectively.

Concept 3: *(Aligned block)* An *aligned block* (*block* for short) is a maximum run of substitutions in an alignment (A', B').

In other words, there are no insertions and deletions in the aligned block and aligned blocks are always flanked by insertions and/or deletions.

3 Algorithm

By adopting the computation model in [10], we can design a simple algorithm for the question of aligning two RNA secondary structures, which follows the sequence alignment paradigm, such as Needleman-Wunsch algorithm [7]. We define a scoring function δ as following:

$$\delta(i,j) = \begin{cases} \gamma \\ \quad i = 0 \; or \; j = 0 \; but \; not \; both \\ A_i \rightarrow B_j \\ \quad i \neq 0 \; and \; j \neq 0 \; and \; p(i) = i \; and \; p(j) = j \\ (A_{p(i)}, A_i) \rightarrow (B_{p(j)}, B_j) \\ \quad i \neq 0 \; and \; j \neq 0 \; and \; p(i) < i \; and \; p(j) < j, or \\ \quad i \neq 0 \; and \; j \neq 0 \; and \; p(i) > i \; and \; p(j) > j \end{cases}$$

where γ is the score of inserting/deleting a base, $A_i \to B_j$ is the score of substituting single base A_i with single base B_j, and $(A_{p(i)}, A_i) \to (B_{p(j)}, B_j)$ is score of substituting a base pair $(A_{p(i)}, A_i)$ with $(B_{p(j)}, B_j)$.

Let $D_k(i_1, i; j_1, j)$ be the score of the alignment between $A_{[i_1,i]}$ and $B_{[j_1,j]}$, which has k blocks and A_{i_1} is aligned with B_{j_1}, and the deletion of A_i is the last operation in the alignment; $I_k(i_1, i; j_1, j)$ be the score of the alignment, which has k blocks and A_{i_1} is aligned with B_{j_1}, and the insertion of B_j is the last operation in the alignment; $M_k(i_1, i; j_1, j)$ be the score of the alignment, which has k blocks and A_{i_1} is aligned with B_{j_1}, and the substitution of A_j and B_j is the last operation.

Note that we require that A_{i_1} be aligned with B_{j_1}. Thus the alignment of $A_{[i_1,i]}$ and $B_{[j_1,j]}$ contains at least one block. We first describe how to compute the alignment between $A_{[i_1,i]}$ and $B_{[j_1,j]}$, *where A_{i_1} and B_{j_1} are $5'$ ends of base pairs*. The following are recurrence equations necessary to compute the alignment. Definitions 1 to 4 are boundary equations.

Definition 1 *(Boundary conditions)*

$$D_1(i_1, i_1; j_1, j_1) = 0$$
$$I_1(i_1, i_1; j_1, j_1) = 0$$
$$M_1(i_1, i_1; j_1, j_1) = \delta(p(i_1), p(j_1))/2$$

At a block boundary it is assumed that both A_{i_1} and B_{j_1} are $5'$ ends of a base pair and they are aligned, only $M_1(i_1, i_1; j_1, j_1)$ is defined. For convenience, we set 0 to $D_1(i_1, i_1; j_1, j_1)$ and $I_1(i_1, i_1; j_1, j_1)$.

In the alignment of $A_{[i_1,i_2]}$ and B_{j_1}, if A_{i_1} is aligned with B_{j_1}, then the rest of the operations in the alignment are deletions, thus only $D_1(i_1, i; j_1, j_1)$ are defined. For convenience, $I_1(i_1, i; j_1, j_1)$ and $M_1(i_1, i; j_1, j_1)$ are set to a relatively small number. Therefore we have:

Definition 2 *(Boundary conditions) For any $i_1 < i \le i_2$,*

$$D_1(i_1, i; j_1, j_1) = D_1(i_1, i-1; j_1, j_1) + \delta(i, 0)$$
$$I_1(i_1, i; j_1, j_1) = D_1(i_1, i; j_1, j_1) + \delta(0, j_1 + 1)$$
$$M_1(i_1, i; j_1, j_1) = I_1(i_1, i; j_1, j_1)$$

and similarly:

Definition 3 *(Boundary conditions) For any $j_1 < j \le j_2$,*

$$I_1(i_1, i_1; j_1, j) = I_1(i_1, i_1; j_1, j-1) + \delta(0, j)$$
$$D_1(i_1, i_1; j_1, j) = I_1(i_1, i_1; j_1, j) + \delta(i_1 + 1, 0)$$
$$M_1(i_1, i_1; j_1, j) = D_1(i_1, i_1; j_1, j)$$

When $k = 0$ or $k > 1$, there are some cases which are not defined.

Definition 4 *(Boundary conditions) For $0 < k \le l$, $k \ne 1$,*

$$D_k(i_1, i_1; j_1, j_1) = I_k(i_1, i_1; j_1, j_1) =$$
$$M_k(i_1, i_1; j_1, j_1) = \quad -\infty$$

for $1 < k \le l$, $i_1 < i \le i_2$ and $j_1 < j \le j_2$,

$$D_k(i_1, i; j_1, j_1) = I_k(i_1, i; j_1, j_1) = M_k(i_1, i; j_1, j_1) = -\infty$$
$$D_k(i_1, i_1; j_1, j) = I_k(i_1, i_1; j_1, j) = M_k(i_1, i_1; j_1, j) = -\infty$$

for $i_1 < i \le i_2$ and $j_1 < j \le j_2$,

$$D_0(i_1, i; j_1, j) = I_0(i_1, i; j_1, j) = M_0(i_1, i; j_1, j) = -\infty$$

We may then proceed with the processes of substitution, insertion and deletion which we define next.

Definition 5 *Substitutions: When a substitution occurs at A_i and B_j, there are three cases to consider:*
case A: both A_i and B_j are single bases,

$$M_k(i_1, i; j_1, j) = \delta(i, j) + \max \begin{cases} D_{k-1}(i_1, i-1; j_1, j-1) \\ I_{k-1}(i_1, i-1; j_1, j-1) \\ M_k(i_1, i-1; j_1, j-1) \end{cases}$$

case B: both A_i and B_j are base paired and at $3'$ end,

$$M_k(i_1, i; j_1, j) = \max\{M^1, M^2\}$$

where

$$M^1 = \delta(i, j)/2 +$$
$$\max_{k'=1}^{k} \left\{ \begin{array}{l} \max \left\{ \begin{array}{l} D_{k'}(i_1, p(i)-1; j_1, p(j)-1) \\ I_{k'}(i_1, p(i)-1; j_1, p(j)-1) \end{array} \right. + \\ \max \left\{ \begin{array}{l} D_{k-k'-1}(p(i), i-1; p(j), j-1) \\ I_{k-k'-1}(p(i), i-1; p(j), j-1) \\ M_{k-k'}(p(i), i-1; p(j), j-1) \end{array} \right. \end{array} \right\}$$

$$M^2 = \delta(i, j)/2 +$$
$$\max_{k'=1}^{k} \left\{ \begin{array}{l} M_{k'}(i_1, p(i)-1; j_1, p(j)-1) + \\ \max \left\{ \begin{array}{l} D_{k-k'}(p(i), i-1; p(j), j-1) \\ I_{k-k'}(p(i), i-1; p(j), j-1) \\ M_{k-k'+1}(p(i), i-1; p(j), j-1) \end{array} \right. \end{array} \right\}$$

case C: otherwise undefined,

$$M_k(i_1, i; j_1, j) = -\infty$$

In case A, when A_i aligns with B_j after an insertion or deletion then we initiate a new block, otherwise we extend a block.

In case B, the condition means $p(i) < i$ and $p(j) < j$, and base pairs $(A_{p(i)}, A_i)$ and $(B_{p(j)}B_j)$ align. This breaks the alignment between $A_{[i_1,i]}$ and $B_{[j_1,j]}$ into three parts: 1. alignment of $A_{[i_1,p(i)-1]}$ and $B_{[j_1,p(j)-1]}$; 2. alignment of $A_{[p(i),i-1]}$ and $B_{[p(j),j-1]}$; 3. A_i and B_j. To calculate the number of blocks correctly, we have nine cases to consider, since there are three possibilities at $A_{p(i)-1}$ and $B_{p(j)-1}$ and another three at A_{i-1} and B_{j-1}.

When there is an insertion or a deletion at $A_{p(i)-1}$ and $B_{p(j)-1}$, the alignment of $A_{p(i)}$ and $B_{p(j)}$ initiates a new aligned block, therefore if there are k' blocks in $A_{[i_1,p(i)-1]}$ and $B_{[j_1,p(j)-1]}$, we need $k-k'$ blocks from the second and the third part alignment. If there is an insertion or deletion at A_{i-1} and B_{j-1}, part 3 is an individual aligned block, so in this case we need $k-k'-1$ aligned blocks from part 2; otherwise we need $k-k'$ blocks. This is handled by equation M^1.

When there is a substitution at $A_{p(i)-1}$ and $B_{p(j)-1}$, then the aligned block starts from $A_{p(i)}$ and $B_{p(j)}$ is merged into the last aligned block in part 1. Furthermore, if A_{i-1} and B_{j-1} are aligned, part 3 is not an individual aligned block either. Thus we need $k-k'+1$ aligned blocks from part 2, otherwise, we need $k-k'$. This is handled by equation M^2.

For case C, there are three possibilities to consider: 1. If A_i is a single base and B_j is a base pair (or vice versa), we do not allow single base to be aligned with any part of base pair; 2. If A_i is a $3'$ end base and B_j is a $5'$ end base (or vice versa), this violates condition of case 2; 3. If both A_i and B_j are $5'$ end base of a base pair. In this situation, the same as the third described in definition 6, also violates the condition of case 2. For these cases, no substitution operation is allowed.

Definition 6 *Deletions: For deleting A_i, whether A_i is a single base or is part of a base pair,*

$$D_k(i_1,i;j_1,j) = \delta(i,0) + \max\begin{cases} D_k(i_1,i-1;j_1,j) \\ I_k(i_1,i-1;j_1,j) \\ M_k(i_1,i-1;j_1,j) \end{cases}$$

A deletion neither initiates nor extends a block, so that the number of blocks is not changed. For A_i, we need to consider three cases, where A_i is a single base, $5'$ or $3'$ end of a base pair. Firstly in the trivial case where A_i is a single base we take the maximum of the three possible operations before the deletion. Secondly, if A_i is at $5'$ end of a base pair, the $3'$ of A_i, namely $A_{p(i)}$ is out of our range now, so deletion of A_i also does not cause a problem. Thirdly, deletion of A_i if it is a $3'$ end base is also not problematic since we already have deleted the $5'$ end $A_{p(i)}$. That deletion occurred during the consideration for substitution of a base pair handled in case B of definition 5.

Definition 7 *Insertions: The definition for insertions is similar to that of deletion. For inserting B_j, whether B_j is a single base or part of a base pair,*

$$I_k(i_1,i;j_1,j) = \delta(0,j) + \max\begin{cases} D_k(i_1,i;j_1,j-1) \\ I_k(i_1,i;j_1,j-1) \\ M_k(i_1,i;j_1,j-1) \end{cases}$$

Definition 8 *For the very first cell of matrices D, I and M,*

$$D_0(0,0;0,0) = I_0(0,0;0,0) = 0 \quad (1)$$
$$M_0(0,0;0,0) = -\infty \quad (2)$$

for $0 < i \le m$, $j = 0$ and $k = 0$,

$$D_0(0,i;0,0) = \delta(i,0) + D_0(0,i-1;0,0) \quad (3)$$
$$I_0(0,i;0,0) = D_0(0,i;0,0) + \delta(0,j) \quad (4)$$
$$M_0(0,i;0,0) = I_0(0,i;0,0) \quad (5)$$

for $0 < j \le n$, $i = 0$ and $k = 0$,

$$I_0(0,0;0,j) = \delta(0,j) + I_0(0,0;0,j-1) \quad (6)$$
$$D_0(0,0;0,j) = I_0(0,0;0,j) + \delta(1,0) \quad (7)$$
$$M_0(0,0;0,j) = D_0(0,0;0,j) \quad (8)$$

for $1 \le i \le m$ and $1 \le j \le n$,

$$I_0(0,i;0,j) = I_0(0,0;0,j) + D_0(0,i;0,j) \quad (9)$$
$$D_0(0,i;0,j) = I_0(0,0;0,j) + D_0(0,i;0,j) \quad (10)$$
$$M_0(0,i;0,j) = -\infty \quad (11)$$

for $0 < k \le l$, $0 \le i \le m$, and $0 \le j \le n$,

$$I_k(0,0;0,j) = D_k(0,0;0,j) = M_k(0,0;0,j) = -\infty \quad (12)$$
$$I_k(0,i;0,0) = D_k(0,i;0,0) = M_k(0,i;0,0) = -\infty \quad (13)$$

and,

$$M_1(0,1;0,1) = \delta(1,1) (or\ \delta(1,1)/2\ or\ -\infty) \quad (14)$$

To start the alignment of $A_{[1,m]}$ and $B_{[1,n]}$, we therefore need to create two aligned artificial bases A_0 and B_0. These do not contribute to the aligned blocks, so we set $M_0(0,0;0,0)$ to be $-\infty$. We define $M_1(0,1;0,1)$ explicitly which is the earliest starting point of a aligned block, where A_1 and B_1 could be single bases or $5'$ end of base pairs. As A_0 and B_0 is only for definition purpose, we only need to define D_0, I_0 and M_0.

Given these definitions we can now describe a bottom-up algorithm to compute the alignment of two RNA secondary structures. The algorithm first calculates the sub-alignment between each pair of (i_1, i_2) and (j_1, j_2) and stores the necessary values, then uses those stored values to compute the desired alignment.

Algorithm for computing RNA structure alignment with l aligned blocks

1. for k = 1 to l
   for each pair (i_1, i_2) in A
   for each pair (j_1, j_2) in B
   Use definitions 1 to 5 to compute matrices D_k, I_k and M_k and store the values.
   Use definitions 6 to 8 to compute the values of $D_l(0,m;0,n)$, $I_l(0,m;0,n)$, $M_l(0,m;0,n)$.

2. The optimal score is:
   $\max\{M_l(0,m;0,n), D_l(0,m;0,n), I_l(0,m;0,n)\}$

3. Trace back to find the alignment itself.

This algorithm finds the optimal alignment of A and B which has exactly l aligned blocks. Suppose that there are p_1 base pairs in A and p_2 base pairs in B, then it is easy to see that the algorithm needs time $O(l^2mnp_1p_2)$ and space $O(lmn)$.

4 Discussion

The advantage of this algorithm is that although it requires the specification of the maximum number of blocks, it circumvents the need for the definition of a gap opening penalty in the scoring function. Given modern fast computers, when l is not too large, the increasing time complexity is tolerable. A possible extension of this work would be to adopt the computation model proposed in [8] to allow operations that break base pairs so that conservation of secondary structure absolute. Another possible direction would be in the alignment of two RNA tertiary structures with l-block.

References

[1] B. A. Shapiro and K. Zhang. Comparing Multiple RNA Secondary Structures Using Tree Comparisons. *CABIO*, 6:309–318, 1990.

[2] D. Sankoff. Matching Sequences under Deletion/Insertion Constraints. *PNAS*, 69:4–6, 1972.

[3] 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.

[4] H. Lenhof, K. Reinert and M. Vingron. A polyhedral approach to RNA sequence structure alignment. *Proceedings of the 2nd Annual International Conference on Computational Molecular Biology*, pages 153–159, 1998.

[5] I. Holmes and G. M. Rubin. Pairwise RNA Structures Comparison with Stochastic Context-Free Grammars. In *Proceedings of Pacifi c Symposium on Biocomputing*, pages 163–174, 2002.

[6] 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.

[7] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48(3):443–453, 1970.

[8] 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.

[9] 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.

[10] 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.

[11] K. Zhang. Computing Similarity between RNA Secondary Structures. In *Proceedings of IEEE International Joint Symposia on Intelligence and Systems*, pages 126–132, 1998.