# A MEMORY EFFICIENT ALGORITHM FOR STRUCTURAL ALIGNMENT OF RNAS WITH EMBEDDED SIMPLE PSEUDOKNOTS

THOMAS WONG, Y. S. CHIU, T. W. LAM, S. M. YIU

*Department of Computer Science, The University of Hong Kong*
*Pokfulam Road, Hong Kong*

In this paper, we consider the problem of structural alignment of a target RNA sequence of length n and a query RNA sequence of length m with known secondary structure that may contain embedded simple pseduoknots. The best known algorithm for solving this problem (Dost et al. [13]) runs in O(mn4) time with space complexity of O(mn3), which requires too much memory making it infeasible for comparing ncRNAs (non-coding RNAs) with length several hundreds or more. We propose a memory efficient algorithm to solve the same problem. We reduce the space complexity to O(mn2 + n3) while maintaining the same time complexity of Dost et al.'s algorithm. Experimental reslts show that our algorithm is feasible for comparing ncRNAs of length more than 500.
Availability: The source code of our program is available upon request.

## 1 Introduction

A non-coding RNA (ncRNA) is a RNA molecule which is not translated into a protein. It is a general belief that ncRNAs are involved in many cellular functions. The number of ncRNAs within the genome was underestimated before, but recently some databases reveal over 30,000 ncRNAs [1] and more than 500 ncRNA families[2]. Large discoveries of ncRNAs and families show the possibilities that ncRNAs may be as diverse as protein molecules [3]. Identifying these ncRNAs becomes an important problem.

It is known that the secondary structure of an ncRNA molecule usually plays an important role in its biological functions. Some researches attempted to identify ncRNAs by considering the stability of secondary structures formed by the substrings of a given genome [15]. However, this method is not effective because a random sequence with high GC composition also allows an energetically favorable secondary structure [8]. A more promising direction is comparative approach which makes use of the idea that if a substring of genome from which a RNA is transcribed with similar sequence and structure to a known ncRNA, then this genome region is likely to be an ncRNA gene whose corresponding ncRNA is in the same family of the known ncRNA. Thus, to locate ncRNAs in a genome, we can use a known ncRNA as a query and searches along the genome for substrings with similar sequence and structure to the query. The key of this approach is to compute the structural alignment between a query sequence with known structure and a target sequence with unknown structure. The alignment score represents their sequence and structural similarity. RSEARCH [9], FASTR [10], and a recent tool INFERNAL [11] for Rfam are using this approach.

However, all of these tools do not support pseudoknots. Given two base-pairs at positions $(i, j)$ and $(i', j')$, where $i < j$ and $i' < j'$, pseudoknots are base-pairs either $i < i' < j < j'$ or $i' < i < j' < j$. In some studies, secondary structures including pseudoknots are found involved in some functions such as telomerase [5], catalytic functions [6], and self-splicing introns [7]. The presence of pseudoknots makes the problem computationally

harder, so finding ncRNA genes with secondary structure including pseudoknots are limited. Usually the large time complexity and considerable memory required for these algorithms make it impractical to search long pseudoknotted ncRNA along the genome. Among over 500 known ncRNA families in Rfam, only 24 families that are in pseudoknotted structure exist. The small number may reflect the uncommon situation of pseudoknotted ncRNA, but it may also reflect the difficulty of finding pseudoknotted ncRNAs due to the limitation of existing tools.

Matsui *et al*. [12] developed a method of computing the structural alignment to support a pseudoknot structure. They used a pseudoknot definition that a secondary structure has *m*-clossing property if and only if there exists *m* base pairs in which any two of them are crossing each other. For 2-crossing pseudoknots, their algorithm runs in $O(mn^5)$ with space complexity of $O(mn^4)$ where $m$ is the length of the query sequence and $n$ is the length of the searching sequence. The large time and space complexity makes the method infeasible for practical use. Pseudoknots can exist within another pseudoknot forming recursive pseudoknots. Since known ncRNA families are found to have a simpler structure (with only a single level of recursion), called embedded simple pseudoknots. Some focuses on this simpler structure. Dost *et al.* [13] developed a tool called PAL using dynamic programming approach that supports secondary structures with embedded simple pseudoknots. By restricting their supporting structure to be a subset of the structures having 2-crossing properties, their dynamic programming algorithm runs faster and uses less memory with time complexity of $O(mn^4)$ and space complexity of $O(mn^3)$.

However, their algorithm is still not feasible for long RNA sequences due to the extensive memory required. For example, for the pseduknotted ncRNA family RF00024 (found in the database Rfam), the average length of the members is about 548. It is estimated that performing a pair-wise structural alignment for members in this family using PAL requires at least 10GB memory. Therefore, the tool becomes impractical for ncRNA families with members of length several hundreds or more. In this paper, we proposed a memory-efficient algorithm for solving the same structural alignment problem with space complexity reduced to $O(mn^2 + n^3)$ while maintaining the same time complexity of $O(mn^4)$.

## 2. Definitions

Let $A = a_1a_2\ldots a_m$ be a RNA sequence and $M$ be the secondary structure of $A$. $M$ is represented as a set of base pairs $(a_i, a_j)$, $1 \le i < j \le m$. Let $M_{x,y} \subseteq M$ be the set of base pairs in the subsequence $a_xa_{x+1}\ldots a_y$, $1 \le x < y \le m$. $M_{x,y} = \{(a_i, a_j) \in M \mid x \le i < j \le y\}$.

$M_{x,y}$ is a regular structure if there does not exist two pairs $(i, j)$, $(k, l) \in M_{x,y}$ such that $I < k < j < l$ or $k < i < l < j$. Note that an empty set is considered as a regular structure.

$M_{x,y}$ is a simple pseudoknot if $\exists\ x < x_1, x_2 < y$ such that

1. each $(i, j) \in M_{x,y}$ satisfies either $x \le i < x_1 \le j < x_2$ or $x_1 \le i < x_2 \le j \le y$; and,
2. $M_L$ and $M_R$ are both regular where $M_L = \{(i, j) \in M_{x,y} \mid x \le i < x_1 \le j < x_2\}$ and $M_R = \{(i, j) \in M_{x,y} \mid x_1 \le i < x_2 \le j \le y\}$.

An embedded simple pseudoknot structure is defined as follows [13].

M is an embedded simple pseudoknot structure if $\exists\ 1 \le x_1 < y_1 < \ldots < x_s < y_s \le m$ such that

1. $M_{xi,yi}$, for $1 \le i \le s$, is a simple pseudoknot structure; and,

2. $\left(M - \bigcup_{1 \le i \le s} M_{xi,yi}\right)$ is a regular structure

Note that simple pseudoknot structure is a subset of embedded simple pseudoknot structure. In this paper, our method is designed for ncRNAs with embedded simple pseudoknot structures.

## 3. Algorithm

### *3.1 Structural alignment*

Let $S[1\ldots m]$ be a query sequence with known secondary structure $M$, and $T[1\ldots n]$ be a target sequence with unknown secondary structure. $S$ and $T$ are sequences from the character set {A, C, G, U}. A structural alignment between $S$ and $T$ can be represented by a pair of sequences $S'[1\ldots r]$ and $T'[1\ldots r]$ where $r \geq m, n$. $S'$ is from $S$ and $T'$ is from $T$ with spaces inserted in between the characters to make both sequences of the same length. A space cannot appear in the same position of $S'$ and $T'$. The score of the alignment, which determines the sequence and structure similarity between $S'$ and $T'$, is defined as follows.

$$score = \sum_{i=1}^{r} \gamma(S'[i], T'[i]) + \sum_{i,j\ s.t.(\eta(i),\eta(j)) \in M \text{ and } S'[i],S'[j],T'[i],T'[j] \neq '\_'} \delta(S'[i], S'[j], T'[i], T'[j])$$

where $\eta(i)$ is the corresponding position in $S$ according to the position $i$ in $S'$; $\gamma(t_1, t_2)$ and $\delta(x_1, y_1, x_2, y_2)$, where $t_1,t_2 \in$ {A,C,G,U, '_'} and $x_1,x_2,y_1,y_2 \in$ {A,C,G,U}, are the score for sequence similarity and the score for structural similarity respectively. The calculation of structural alignment score is not restricted to any kind of secondary structure. It works in the same way for pseudoknot structure. The objective is to find an alignment such that the corresponding score is maximized. Higher score represents higher similarity between the two sequences according to their sequences and structures. Also, if the score is high, then the alignment can reasonably reveal the secondary structure of the target sequence.

### *3.2 Structural alignment for simple pseudoknot*

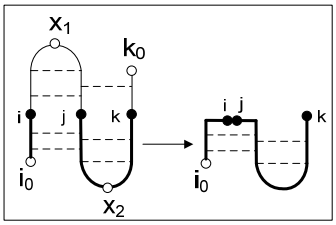


Figure 1: Subpseudoknot P(i,j,k) = [$i_0$..i][j..k], where $i_0 \leq i < x_1$, $x_1 \leq j < x_2$, $x_2 \leq k \leq k_0$

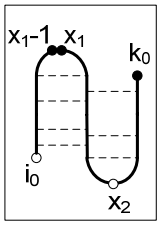


Figure 2: P($x_1$-1,$x_1$,$k_0$) represents the whole pseudoknot structure of S[$i_0$…$k_0$]

Consider a length-m subsequence $S[i_0\ldots k_0]$ with a simple pseudoknot structure, there exists $i_0 \leq x_1$, $x_2 \leq k_0$ such that the simple pseudoknot structure $M$ can be divided into two

regular structures $M_L$ and $M_R$ as mentioned in Section 2. A subpseudoknot $P(i,j,k)$ of $S$ is defined as the union of subinterval $[i_0..i]$ and $[j..k]$, where $i_0 \le i < x_1$, $x_1 \le j < x_2$, $x_2 \le k \le k_0$, as shown in Figure 1. Let $B[p,q,r,i,j,k]$ be the optimal alignment score between $P(i,j,k)$ of $S$ and $P(p,q,r)$ of another length-$n$ subsequence $T[p_0 \ldots r_0]$ whose structure is unknown. Consider the case of $(i, j) \in M_L$, the following recurrence equation [13] gives the value for $B$. The case for $(i, j) \in M_R$ is similar.

$$B[p,q,r,i,j,k] = \max \begin{cases} // \textit{ match} \\ B[p-1,q+1,r,i-1,j+1,k] + \delta(T[p],T[q],S[i],S[j]) \\ \qquad + \gamma(T[p],S[i]) + \gamma(T[q],S[j]) \\ // \textit{ delete} \\ B[p-1,q,r,i-1,j+1,k] + \gamma(T[p],S[i]) + \gamma('\_',S[j]) \\ B[p,q+1,r,i-1,j+1,k] + \gamma('\_',S[i]) + \gamma(T[q],S[j]) \\ B[p,q,r,i-1,j+1,k] + \gamma('\_',S[i]) + \gamma('\_',S[j]) \\ // \textit{ insert} \\ B[p-1,q,r,i,j,k] + \gamma(T[p],'\_') \\ B[p,q-1,r,i,j,k] + \gamma(T[q],'\_') \\ B[p,q,r-1,i,j,k] + \gamma(T[r],'\_') \end{cases}$$

As shown in Figure 2, $P(x_1\text{-}1,x_1,k_0)$ represents the whole pseudoknot structure of $S[i_0 \ldots k_0]$. Therefore, the score of an optimal alignment between $S[i_0,k_0]$ and $T[p_0,r_0]$ is $\max_{p_0 \le p \le r_0} B[p-1,p,r_0,x_1-1,x_1,k_0]$. Note the changes of indices of $(i,j,k)$ in the recursive calculation. The value of $i$ decreases from $x_1$-1 to $i_0$, the value of $j$ increases from $x_1$ to $x_2$-1 and the value of $k$ deceases from $k_0$ to $x_2$. That sequence of triples can be first built from a simple-pseudoknot structure in linear time [13]. The triple $(x_1\text{-}1,x_1,k_0)$ is chosen as the first item (called *root*). The sequence of triples ensures each nucleotide is touched by at least one triple, and every base pair is reached by one and only one triple (i.e. for each base pair $B(x,y)$, there must exist one and only one triple $(i,j,k)$ such that $i=x$ and $j=y$ if $B \in M_L$ or $j=x$ and $k=y$ if $B \in M_R$). The number of triples in the sequences is $O(m)$, where $m$ is the length of the query sequence $S$.

By using this sequence of triples, the function $B$ can be rewritten as $B[p,q,r,v]$ where $v$ is a triple in the sequence. The recurrence relationship for the case of $v \in M_L$ can be modified as follows.

$$B[p,q,r,v] = \max \begin{cases} // \textit{ match} \\ B[p-1,q+1,r,next(v)] + \delta(T[p],T[q],S[v_i],S[v_j]) \\ \qquad + \gamma(T[p],S[v_i]) + \gamma(T[q],S[v_j]) \\ // \textit{ delete} \\ B[p-1,q,r,next(v)] + \gamma(T[p],S[v_i]) + \gamma('\_',S[v_j]) \\ B[p,q+1,r,next(v)] + \gamma('\_',S[v_i]) + \gamma(T[q],S[v_j]) \\ B[p,q,r,next(v)] + \gamma('\_',S[v_i]) + \gamma('\_',S[v_j]) \\ // \textit{ insert} \\ B[p-1,q,r,v] + \gamma(T[p],'\_') \\ B[p,q-1,r,v] + \gamma(T[q],'\_') \\ B[p,q,r-1,v] + \gamma(T[r],'\_') \end{cases}$$

where $v_i, v_j$ is the $i$ value and $j$ value of the triple $v$ respectively. Next($v$) represents the next triple after $v$ in the sequence of triples. The cases for $v \in M_R$, $v \notin M_L$ or $M_R$ are similar [13]. Since $p_0 \le p < q < r \le r_0$ and the number of $v = O(m)$, then both time and space

complexities are $O(mn^3)$. The score of an optimal alignment between $S[i_0,k_0]$ and $T[p_0,\mathbf{r}_0]$ is $\max_{p_0 \le p \le r_0} B[p-1, p, r_0, root]$.

### *3.3 Our memory-efficient algorithm*

A simple-pseudoknot has two interesting features. Firstly, as shown in the Figure 3a, the reversal of a simple pseudoknot is also a simple pseudoknot. The subpseudoknot $P(i,j,k)$ becomes the upper region including $(i,j,k)$ in the reverse structure. If we consider the alignment between a reverse query sequence and a reverse target sequence, the previous algorithm should also work, but the order of $B$ calculation will be in reversed order according to the reverse sequence of triples in which the root becomes $(x_2+1,x_2,\ i_0)$ instead of $(x_1-1,x_1,k_0)$.

Secondly, as shown in the Figure 3b, a simple pseudoknot can be separated into two simple-pseudoknots according to a triple $(i,j,k)$: the upper region including $(i,j,k)$ and the lower region excluding $(i,j,k)$. This indicates that the alignment problem between a pair of sequences can be divided into two alignment problems between two pairs of shorter sequences. Based on these two features and inspired by Hirschberg's algorithm [14], we derive a method which can reduce the memory consumption of structural alignment algorithm for simple pseudoknots from $O(mn^3)$ to $O(n^3)$, while maintaining the same time complexity of $O(mn^3)$.

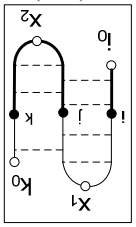


Figure 3a: Reverse of a simple-pseudoknot

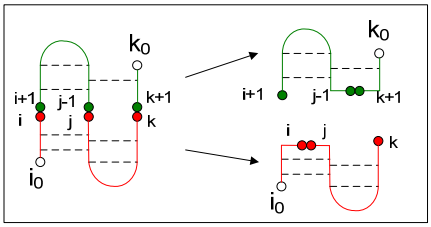


Figure 3b: a simple-pseudoknot can be separated into two simple-pseudoknots

For the sake of simplicity, we rename the indices of $S[i_0,k_0]$ as $S[1\ldots m]$ and $T[p_0,r_0]$ as $T[1\ldots n]$. We first show a *score-only* algorithm is to compute $B[p,q,r,root]$ where $1 \le p < q < r \le n$. $B$ is calculated triple by triple, from the last triple in the triple sequence up to root. At the end of every iteration, we discard all values of $B$ calculated in the previous iterations and only keep those calculated in this iteration for the $B$ calculation in the next iteration.

score-only ($V$, $T$)
1. Initialize $B_{prev}$
2. for $v$ = last-item to root in the sequence of triples $V$
3. for all $p,q,r$ where $1 \le p < q < r \le n$
4. compute $B[p,q,r]$ with respect to current value of $v$ and $B_{prev}$
5. Swap the pointer $B_{prev}$ with $B$
6. return $B_{prev}$

The score-only algorithm can compute $B[p,q,r,root]$ where $1\le p<q<r\le n$ in $O(mn^3)$ time and requires $O(n^3)$ memory space. The score-only algorithm can only give the alignment scores.

Let $S^R[1\ldots m]$ be the reversal of a query sequence $S$ (i.e. $S^R[i] = S[m-i+1]$ for $1\le i\le m$). Let $v^r$ be the corresponding triple $v$ of $S$ for $S^R$ and let $v_i,v_j,v_k$ be the positions $i,j,k$ of triple $v$, respectively. $v^r=((v^r)_i,(v^r)_j,(v^r)_k)=(m-v_k+1,m-v_j+1,m-v_i+1)$. As illustrated in Figure 4, the subpseudoknot $P(v^r)$ of $S^R$ is the upper region including the triple $v$ of $S$. The union of the subpseudoknot $P(v_i-1,v_j+1,v_k-1)$ of $S$ and the subpseudoknot $P(v^r)$ of $S^R$ is the whole pseudoknot of $S$. If a triple $(p,q,r)$ of $T$ is mapped to $v$ of $S$ in the optimal alignment between $S$ and $T$, then the optimal alignment score between $S$ and $T$ = the optimal alignment score between $P(v_i-1,v_j+1,v_k-1)$ of $S$ and $P(p-1,q+1,r-1)$ of $T$ + the optimal alignment score between $P(v^r)$ of $S^R$ and $P((p,q,r)^r)$ of $T^R$.

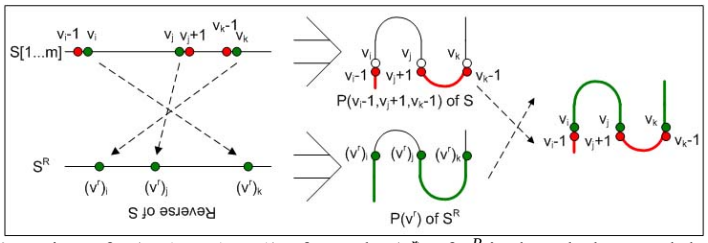


Figure 4: Union of $P(v_i-1,v_j+1,v_k-1)$ of $S$ and $P(v^r)$ of $S^R$ is the whole pseudoknot of $S$.

The following is a divide-and-conquer approach to get the alignment between $S[1..m]$ and $T[1..n]$.

1. Select the middle triple $w$ of $S$ and prepare the triple sequences for the reverse of the upper region $V_{upper}^R$ and for the lower region $V_{lower}$
   a. Build the sequence of triples $V[1\ldots m_v]$ for $S$.
   b. Let $w = V\left[\frac{m_v}{2}\right]$, the middle triple in the sequence.
   c. As in Figure 3b, partition $S$ into two subseudoknots: upper region including $w$ and lower region excluding $w$ (i.e. $S_{upper} = [w_i,w_j]\cup[w_k,m]$, $S_{lower}=[1,w_i-1]\cup[w_i+1,w_k-1]$)
   d. Reverse sequence $S_{upper}$ to obtain $S_{upper}^R$
   e. Build the triple sequence $V_{upper}^R$ and $V_{lower}$ for $S_{upper}^R$ and $S_{lower}$ respectively. Note that the root of $V_{upper}^R$ is corresponding to $w^r$ on $S$ and the root of $V_{lower}$ is $(w_i-1,w_j+1,w_k-1)$.
2. Find $p_h,q_h,r_h$ such that triple $(p_h,q_h,r_h)$ of $T$ is mapped to triple $w$ of $S$ in the optimal alignment between $S$ and $T$.
   a. Use score-only($V_{lower}$,$T$) to compute $B[p,q,r]$ $\forall$ $1\le p<q<r\le n$ for alignment between $T$ and $V_{lower}$.
   b. Use score-only($V_{upper}^R$, $T^R$) to compute $B^R[p,q,r]$ $\forall$ $1\le p<q<r\le n$ for alignment between $T^R$ and $V_{upper}^R$.
   c. Find out $p_h,q_h,r_h$ such that $B[p_h-1,q_h+1,r_h-1]+ B^R[p',q',r']$ is maximum where $p'= n-r_h+1,q'=n-q_h+1,r'=n-p_h+1$.

3. As in Figure 3b, partition $T$ into two subseudoknots. $T_{\text{upper}}$ : upper region including $(p_h,q_h,r_h)$ and $T_{\text{lower}}$ : lower region excluding $(p_h,q_h,r_h)$.
4. Recursively find out the optimal alignment of $S_{\text{upper}}$ and $T_{\text{upper}}$.
5. Recursively find out the optimal alignment of $S_{\text{lower}}$ and $T_{\text{lower}}$.
6. Combine results from Steps 4 and 5 to obtain the optimal alignment.

Lemma 1. The above procedure runs in $O(mn^3)$ time with space complexity of $O(n^3)$.

*Proof*.

To analyze the total time complexity required for the whole procedure, let $K[m,n]$ be the total time required to align $S$ of length $m$ and $T$ of length $n$. Both Steps 1 and 6 require $O(m)$ time. Step 2 takes $O(mn^3)$ time. For Step 4, it takes $K[m/2,n_1]$ time where, $n_1$=length of $T_{\text{upper}}$ and for Step 5, it takes $K[m/2,n_2]$ time, $n_2$=length of $T_{\text{lower}}$.

$$K[m,n]$$
$$= O(mn^3) + K\left[\frac{m}{2}, n_1\right] + K\left[\frac{m}{2}, n_2\right]$$
$$= O(mn^3) + O(\frac{m}{2}{n_1}^3) + O(\frac{m}{2}{n_2}^3) + K\left[\frac{m}{4}, n_{11}\right] + K\left[\frac{m}{4}, n_{12}\right] + K\left[\frac{m}{4}, n_{21}\right] + K\left[\frac{m}{4}, n_{22}\right]$$

Since $(n_1+n_2)=n$, $(n_{11}+n_{12}+n_{21}+n_{22})=n$, and so on, and the fact that $\left(\sum_i n_i\right)^3 \geq \sum_i (n_i)^3$,

therefore, ${n_1}^3+{n_2}^3 \leq n^3$ , ${n_{11}}^3+{n_{12}}^3+{n_{21}}^3+{n_{22}}^3 \leq n^3$ and so on

$$K[m,n] = O(mn^3) + O(\frac{m}{2}n^3) + O(\frac{m}{4}n^3) + ..... = O(mn^3)$$

Thus, the whole procedure requires $O(mn^3)$ time. Since the memory space for the table in the score-only algorithm can be reused during the recursion, the total space required is $O(n^3)$.

### *3.4 Structural alignment for embedded simple pseudoknot*

When considering the structural alignment for embedded-simple-pseudoknot, we extend the algorithm in [13] in order to apply the above procedure. We first binarize the query RNA converting it into a binary tree structure [13]. Each node represents a pair of nucleotides (which may not be a base-pair). Node $A$ is a descendant of node $B$ if pair $B$ is inside pair $A$. A node would have two children if the region bounded by the pair can be partitioned into two embedded simple pseudoknot regions. If the pair bounds a simple pseudoknot region, then the node will be indicated as a simple pseudoknot and has no child but the corresponding triple sequence is formed and attached under the node. The tree ensures that each nucleotide is touched by at least one node, and every base pair is reached by one and only one node. No two pairs represented by two nodes are crossing each other. Let $A[i,j,v]$ be the score of optimal alignment between a target sequence $T[i…j]$ and the subtree rooted at the node $v$ $(v_i,v_j)$, which is also the subinterval $(v_i, v_j)$ of the query sequence. The following shows the algorithm for embedded simple pseudoknot alignment.

ALIGN($S$[1..$m$],$T$[1..$n$])
1. Binarizing the query $S$ to obtain the binary tree $M$'.
2. for $i = n$-1 downto 1
3. for all nodes $v = (v_i,v_j)$ in $M$' (from leaves to root)

4. if $v$ is a simple pseudoknot
5. let $V$ be the triple sequence of $v$
   // return a set of optimal simple pseudoknot alignment scores
   // between $V$ and $T[i,j]$ where $i+1 \le j \le n$
6. $C[i+1 \ldots n]$ = score-only-SP($V$,$T$,$i$)
7. for $j = i+1$ to $n$
8. if $v$ is NIL
9. then $A[i,j,\text{NIL}]=A[i,j\text{-}1,\text{NIL}]+\gamma(T[j],'\_')$
10. if $v$ is a pseudoknot
11. then $A[i,j,v]=C[j]$
12. if $v \in M'$
13. $$A[i,j,v] = \max \begin{cases} \text{// match} \\ A[i+1, j-1, child(v)] + \delta(S[v_i], S[v_j], T[i], T[j]) \\ \qquad + \gamma(S[v_i], T[i]) + \gamma(S[v_j], T[j]) \\ \text{// insertion} \\ A[i, j-1, v] + \gamma('\_', T[j]) \\ A[i+1, j, v] + \gamma('\_', T[j]) \\ \text{// deletion} \\ A[i+1, j, child(v)] + \gamma(S[v_i], T[i]) + \gamma(S[v_j], '\_') \\ A[i, j-1, child(v)] + \gamma(S[v_i], '\_') + \gamma(S[v_j], T[j]) \\ A[i, j, child(v)] + \gamma(S[v_i], '\_') + \gamma(S[v_j], '\_') \end{cases}$$
14. if $v \notin M'$ and has less than 2 children, A can be computed
    similarily as in Step 13 [10].
15. if $v \notin M'$ and has 2 children
16. $$A[i,j,v] = \max_{i \le k \le j+1} \{A[i,k-1,left\_child(v)] + A[k,j,right\_child(v)]\}$$

The function score-only-SP($V$, $T[1 \ldots n]$, $p_0$) is to compute a set of optimal simple pseudoknot alignment scores between $V$ and $T[p_0,j]$ where $p_0+1 \le j \le n$.

score-only-SP($V$, $T[1 \ldots n]$, $p_0$)

1. Initialize $B_{\text{prev}}$
2. for $v$ = last-item to root in the triple sequence $V$
3. for all $p,q,r$ such that $p_0 \le p < q < r \le n$
4. compute $B[p,q,r]$ with respect to current $v$ and $B_{\text{prev}}$
5. $B_{\text{prev}} \leftrightarrow B$
6. Initialize $C$
7. for $k = p_0+2$ to $n$
8. for $j = p_0+1$ to $k$-1
9. if $C[k] < B_{\text{prev}}[j\text{-}1,j,k]$
10. $C[k]=B_{\text{prev}}[j\text{-}1,j,k]$
11. return $C[p_0+1 \ldots n]$

The total time and space complexity for ALIGN() procedure (with score-only-SP()) is $O(mn^4)$ and $O(mn^2+n^3)$ respectively. After running the ALIGN() procedure, although the optimal alignments between $S$ and $T$ for the pseudoknotted regions are still unknown, we know the locations of the regions on $T$. Then, for each pseudoknotted region of $T$

mapped to a pseudoknotted region of $S$, the previous divide-and-conquer procedure can be used to obtain the corresponding alignments. The time and space complexities for this are $O(mn^4)$ and $O(n^3)$, respectively. Therefore, we have the following lemma.

Lemma 2. The overall time and space complexities required for aligning $S[1..m]$ and $T[1..n]$ with embedded simple pseudoknots are $O(mn^4)$ and $O(mn^2 + n^3)$, respectively.

## 4. Experimental Results

We implemented the memory efficient algorithm in C++. Since PAL [13] program is not available, we also implement their method for comparison on the performance. We selected ten RNA families which have embedded simple pseudoknot structures for the experiment. The sequence and structural information of all seed members in the family were downloaded from Rfam database [2]. For each family, we randomly picked one of the seed members as a query sequence and aligned it with the other members one by one.

In the experiment, we found that the time required for both algorithms are almost the same, however the difference between the memory consumption is large, especially for the families with long sequences. Our memory consumption is less than theirs in all ten families. For the families with short sequence length, say less than 70, their algorithm does not need much memory. However, their memory consumption increases dramatically for the families with long sequence length compared with ours. Table 1 shows the comparison of memory usage between our space-efficient algorithm and their algorithm for the families with sequence length greater than 70. For the family Telomerase-vert, their algorithm cannot be executed in our server because it consumes more than 4G memory. We estimated that the actual memory consumption would be more than 10G.

Table 1: Comparison on memory usage between our space-efficient algorithm and their algorithm for the families with sequence length greater than 70. *Mp* is the number of triples for the pseudoknotted region and *N* is the number of members.

| RNA Family | Rfam Id | N | Ave. length | Mp | Ave. memory usage in our algorithm (*M*) | Ave. memory usage in their algorithm (*M*) |
|---|---|---|---|---|---|---|
| Corona_FSE | RF00507 | 23 | 82 | 53 | 5.2 | 30 |
| Tymo_tRNA-like | RF00233 | 28 | 83 | 12 | 6.4 | 10 |
| Parecho CRE | RF00499 | 5 | 112 | 21 | 11.0 | 31 |
| IFN-gamma | RF00259 | 5 | 168 | 73 | 23.0 | 282 |
| Telomerase-vert | RF00024 | 37 | 548 | 100 | 839.0 | >4,000 |

## 5. Discussion and Conclusions

Since we are mainly interested in the RNA sequences (or substrings in the given genome) that have a high score with the given query sequence, we expect that the distances between the bases in two mapped base pairs will not differ a lot. In practice, we can impose an upper bound $\Delta$ on the difference of the lengths between the mapped base pairs in order to decrease the time complexity of the algorithm.

The algorithms in this paper are only designed for the embedded simple pseudoknot structures. We have scanned through the existing families and found that there exist other

pseudoknot structures. Developing time and memory efficient algorithms for other pseudoknot structures would be essential and helpful for discovery of new members for these pseudoknotted ncRNA families.

## 6. Acknowledgments

The project is partially supported by Seed Funding Programme for Basic Research of the University of Hong Kong (200611159001).

## 7. References

1. Noncoding RNA database http://biobases.ibch.poznan.pl/ncRNA/
2. Griffiths-Jones S, Bateman A, Marshall M, Khanna A, Eddy SR (2003) Rfam: an RNA family database. NucleicAcid Research, 31(1):439–441, Jan. 2003. http://www.sanger.ac.uk/Software/Rfam/
3. Eddy S (2001) Non-coding RNA genes and the modern RNA world. Nature Reviews in Genetics 2, 919-929
4. Rietveld K, Van Poelgeest R, Pleij CW, Van Boom JH, Bosch L (1082) The tRNA-like structure at the 3'terminus of turnip yellow mosaic virus RNA. Differences and similarities with canonical tRNA. Nucleic Acids Res 10: 1929-1946
5. Hen J, Greider CW (2005) Functional analysis of the pseudoknot structure in human telomerase RNA. PNAS, 102(23):8080-8085
6. Dam E, Pleij K, Draper D (1992) Structural and functional aspects of RNA pseudoknots. Biochemistry, 31(47):11665-11676
7. Adams PL, Stahley MR, Kosek AB, Wang J, Strobel SA (2004) Crystal structure of a self-splicing group I intron with both exons. Nature 430: 45-50
8. Rivas E and Eddy S (2000) Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. Bioinformatics, 16(7):583-605
9. Klein R and Eddy S (2003) Rsearch: Finding homologs of single structured rna sequences. BMC Bioinformatics, 4(1):44
10. Zhang S, B Hass, E Eskin, V Bafna. (2005) Searching genomes for noncoding RNA using FastR, IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2(4) October-December 2005.
11. Nawrocki EP, Eddy SR (2007) Query-Dependent Banding (QDB) for Faster RNA Similarity Searches, PLoS Comput. Biol., 3:e56
12. Matsui H, Sato K, Sakakibara Y (2005) Pair Stochastic Tree Adjoining Grammars for Aligning and Predicting Pseudoknot RNA Structures. Bioinformatics 21 2611-2617
13. Dost B, Han B, Zhang S, Bafna V (2006) Structural Alignment of Pseudoknotted RNA. RECOMB 2006, LNBI 3909, 43-158
14. Hirschberg DS (1975) *A linear space algorithm for computing maximal common subsequences.* Comm. A.C.M. **18**(6) 341-343
15. Le SY, Chen JH, Maizel J (1990) *Structure and Methods: Human Genome Initiative and DNA Recombination*, volume 1, chapter Efficient searches for unusual folding regions in RNA sequences, 127-130