A Memory Efficient Algorithm for Structural Alignment of RNAs with Simple Pseudoknots

S.M. Yiu
Department of Computer Science
The University of Hong Kong

Joint work with Thomas Wong, Y.S. Chiu, T.W. Lam

Outline

- Motivation
- Background
  - Secondary structure of RNA
  - Simple pseudoknots
- Problem definition
- PAL Algorithm
- Idea of our Space-Efficient Algorithm
- Result and Conclusion

Motivation

Traditional thinking (central dogma):
DNA -> RNA -> Protein

- Non-coding RNAs - RNA molecule which is not translated into a protein but may involve in many cellular functions.
- Recently over 30,000 ncRNAs and more than 500 ncRNA families have been discovered. It is believed that there are still many undiscovered yet.

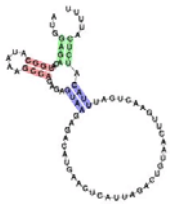
Ribozyme RNA

tRNA
(clover-like)

Finding ncRNA genes

- More difficult to detect
- Computational approaches – Given known ncRNAs, can we predict more?
  - ncRNAs may not conserve in sequence
  - It seems the secondary structure of members in the same ncRNA family is highly conserved.
  - A possible approach: given a known ncRNA, scan through the genome and locate the substrings which has similar sequence and can fold into a similar structure.

e.g. ncRNA family
CAESAR (RF00172)
Consensus structure:



Information from Rfam

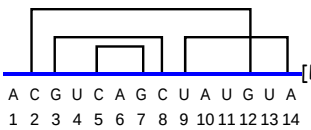
Background

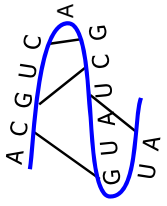
RNA is single stranded, can be regarded as a sequence of A, C, G, U.

Bases may bind (C-G; A-U; each base can pair with at most one other base) forming the secondary structure.

e.g. ACGUCAGCUAUGUA

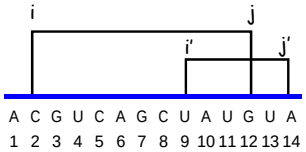
{(2,12), (3,8), (5,7), (9,14)}





Pseudoknot

- Two pairs of base pairs (i, j) and (i', j') cross each other
  - i < i' < j < j'
  - e.g. S: {(2,12), (3,8), (5,7), (9,14)}

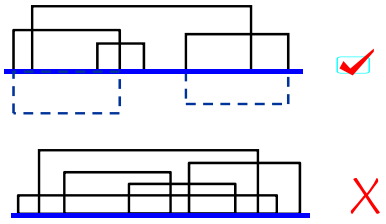


With the presence of pseudoknots, problems related to 2nd structure of RNAs become more difficult (e.g. 2nd structure prediction prob is NP-hard).

Good news: Some pseudoknots have simpler structure (e.g simple pseudoknots).

Simple Pseudoknot

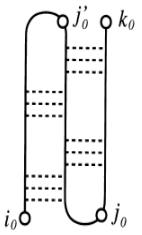
Intuitively, a simple pseudoknot is one such that if you are able to draw the pseudoknots with some above the black bone, some under the black bone such that none of them cross each other.

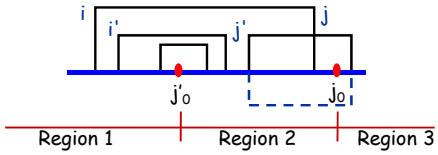


Simple Pseudoknot

The structure of the region from i_0 to k_0 is a simple pseudoknot if:

1) no base pair shares the same point, and;
2) there exist points j_0' and j_0 s.t. every base pair (i,j) inside the region satisfies either $i_0 \le i < j_0 \le j < j_0$ or $j_0' \le i < j_0 \le j \le k_0$, and;
3) if two base pairs (i,j) and (i',j') satisfy either $i < i' < j_0$ or $j_0' \le i < i'$, then $j > j'$ holds.

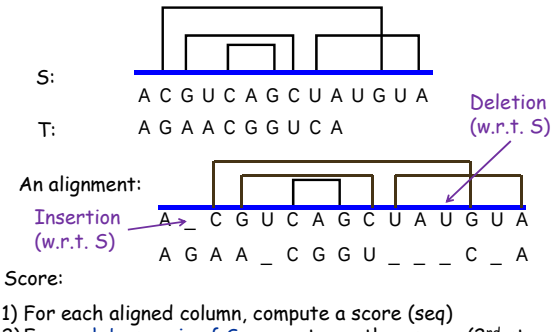




Problem Definition

- Input:
  1. A query sequence and its secondary structure,
  2. A target sequence without secondary structure
- Output:
  1. A score representing the similarity between their primary sequence and secondary structure.
  2. The best alignment between them according to their primary sequence and secondary structure.

(Structural) Alignment



Score:

1) For each aligned column, compute a score (seq)
2) For each base pair of S, compute another score (2nd struct)

Alignment score

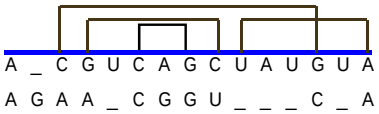
- Query sequence: S[1..m] and target sequence: T[1..n]
- Structural alignment is represented by:
  - S'[1…r] and T'[1…r] where r ≥ m, n.
- S' is from S and T' is from T with some spaces inserted.
- The score of the alignment is defined as follows.

$$score = \sum_{i=1}^{r} \gamma(S'[i], T'[i]) + \sum_{i,j\ s.t.(\eta(i),\eta(j))\ is\ base\ pair} \delta(S'[i], S'[j], T'[i], T'[j])$$

- where $\eta(i)$ is the corresponding position in S
- $\gamma(t_1, t_2)$ is the score for sequence similarity
- $\delta(x_1, y_1, x_2, y_2)$ is the structural similarity score

Example

An alignment:



Seq. score: 4(2) + 11(-1) = -3
Struct. Score: 1(3) + 1(2) = 5

Overall score: 2

λ Table (seq. score):

	A	C	G	U	_
A	2	-1	-1	-1	-1
C		2	-1	-1	-1
G			2	-1	-1
U				2	
_					0

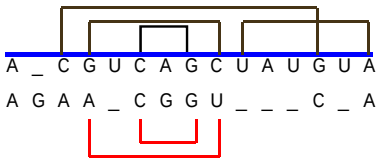
δ Table (2nd Struct. score):

	(A, U)	(C, G)	others
(A, U)	3	2	0
(C, G)		3	0
others			0

Remark

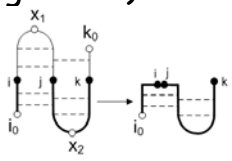
The alignment implies a secondary structure on T in which the base pairs form a subset of those in S.

An alignment:

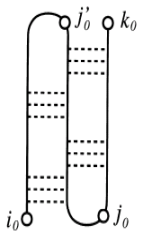


DP Solution (PAL Algorithm)

- Let P(i,j,k) be the sub-pseudoknot structure of the union of subinterval $[i_0 \ldots i]$ and $[j \ldots k]$.



- Let B[p,q,r,i,j,k] be the optimal alignment score between P(p,q,r) of target sequence and P(i,j,k) of query sequence with known secondary structure.



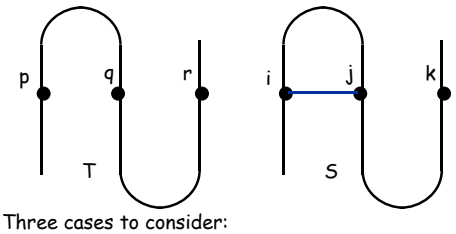
For S, we know j_0', so P[j_0'-1, j_0',k_0] represent the whole structure.

To align S[i_0, k_0] with T[p_0, r_0], we try to compute
$\text{Max}_{\{p_0 \le p \le r_0\}}$ B[p-1, p, r_0, j_0'-1, j_0', k_0]

A straight-forward implementation:
Both time and space complexity: $O(m^3n^3)$ where m is the length of S and n is the length of T.

Recurrence

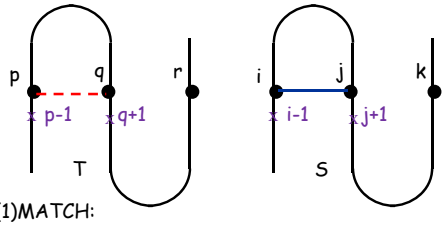
If (i, j) is a base pair in S



Three cases to consider:
(1) Match: Align p with i; and align q with j
(2) Delete in S: Align i with _ or j with _ or both
(3) Insert in S: Align p with _ or q with _ or r with _

Recurrence

If (i, j) is a base pair in S



(1)MATCH:
We align p to i and q to j.

B[p,q,r,i,j,k] = B[p-1,q+1,r,i-1,j+1,k]+ δ(S[i],S[j],T[p],T[q])
+ γ(S[i],T[p]) + γ(S[j],T[q])

(2) DELETE in S:
then either S[i] aligns with a space or S[j] aligns with a space or both.

B[p,q,r,i,j,k] = max of the followings

S[i] aligns with space
- B[p,q,r,i-1,j,k] + γ(S[i], _)

S[j] aligns with space
- B[p,q,r,i,j+1,k] + γ(S[j], _)

BothS[i] and S[j] align with space
- B[p,q,r,i-1,j+1,k] + γ(S[i], _]) + γ(S[j], _)

(3) INSERT in S:
then either T[p] aligns with a space or T[q] aligns with a space; or T[r] aligns with a space.

B[p,q,r,i,j,k] = max of the followings

T[p] aligns with space
- B[p-1,q,r,i,j,k] + γ(T[p], _)

T[q] aligns with space
- B[p,q+1,r,i,j,k] + γ(T[q], _)

T[r] align with space
- B[p,q,r-1,i,j,k] + γ(T[r], _])

B[p,q,r,i,j,k] = max of the followings
[Match]
B[p-1,q+1,r,i-1,j+1,k]+ δ(S[i],S[j],T[p],T[q])
+ γ(S[i],T[p]) + γ(S[j],T[q])
[Delete]
B[p,q,r,i-1,j,k] + γ(S[i], _)
B[p,q,r,i,j+1,k] + γ(S[j], _)
B[p,q,r,i-1,j+1,k] + γ(S[i], _]) + γ(S[j], _) Same as subcase (a)

[Insert]
B[p-1,q,r,i,j,k] + γ(T[p], _)
B[p,q+1,r,i,j,k] + γ(T[q], _)
B[p,q,r-1,i,j,k] + γ(T[r], _])

3 subcases:
a) S[j] aligns with space
b) S[j] aligns with T[q]
c) T[q] aligns with space

b) B[p,q+1,r,i-1,j+1,k] + γ(S[i], _) + γ(S[j], T[q])
c) B[p,q+1,r,i-1,j,k] + γ(S[i], _) + γ(T[q], _)

Note: c) < B[p,q+1,r,i,j,k] + γ(T[q], _) [Case 2 of Insert]

B[p,q,r,i,j,k] = max of the followings
[Match]
B[p-1,q+1,r,i-1,j+1,k]+ δ(S[i],S[j],T[p],T[q])
+ γ(S[i],T[p]) + γ(S[j],T[q])
[Delete]
B[p,q+1,r,i-1,j+1,k] + γ(S[i], _) + γ(S[j], T[q])
~~B[p,q,r,i,j+1,k] + γ(S[j], _)~~ <= symmetric case!
B[p,q,r,i-1,j+1,k] + γ(S[i], _]) + γ(S[j], _)

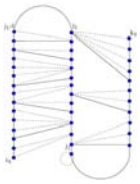
B[p-1,q,r,i-1,j+1,k] + γ(S[j], _) + γ(S[i], T[p])

[Insert]
B[p-1,q,r,i,j,k] + γ(T[p], _)
B[p,q+1,r,i,j,k] + γ(T[q], _)
B[p,q,r-1,i,j,k] + γ(T[r], _])

Observation:
To compute the case for (i, j, k), we only need the result for (i-1, j+1, k)
=> Not all triplets are needed!

For the case when (j,k) is base pair or both are not base pairs,
……

Results from [Recomb06]:
To compute B[p,q,r,i,j,k], we have the followings.
1) We only need to consider O(m) triplets (i,j,k)
2) We can give a total order to these triplets for the purpose of computing B.



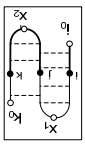
The (total ordered) sequence V of triple (i,j,k) can be defined according to the pseudoknotted structure in linear time.

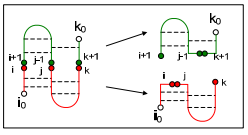
Redefine the DP:
B[p,q,r,i,j,k]=>B[p,q,r,v] where v is a node in the triple sequence V.
So, time and space complexity for score calculation = $O(mn^3)$

Space still an issue: ncRNA famililes with members of length > 500, > 10G memory is required!

Idea for memory efficient version

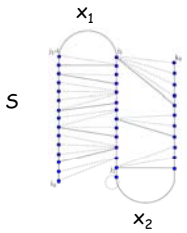
- Reverse of a simple pseudoknot is also a simple pseudoknot.
  - For alignment between S^R and T^R, the previous algorithm still works.
- A simple pseudoknot can be separated into two simple-pseudoknots.
  - Alignment problem between a pair of sequences can be divided into two alignment problems between two pairs of shorter sequences.





Inspired by Hirschberg's algorithm, a memory-efficient algorithm is developed.

(1) Score-only function



⇔

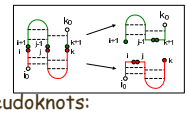
Sequence of triples V:
$v_1 = \langle x_1-1, x_1, k_0 \rangle$
v_2
v_3
.....

Remark: to compute B for v_i, we only need the results for v_{i+1}.

Starting from the last triple in the sequence V, to compute B[p,q,r,v_i], only keep the set of B values for B[p,q,r,v_{i+1}].

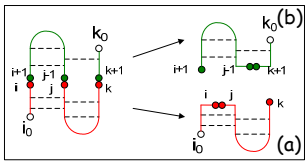
Time: $O(mn^3)$; Space: $O(n^3)$.

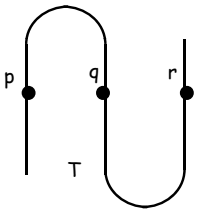
(2) Divide-and-conquer approach



Alignment between S[1..m] and T[1..n]

- Select the middle triple w of S
- Divide the whole S into two subpseudoknots: upper region excluding w and lower region including w.
- Use score-only function to compute B[p,q,r] ∀ p<q<r for alignment between S and T w.r.t. triple w (for lower region of S).
  - Values for B[p,q,r,w] are available ∀ p<q<r
- Use score-only function to compute B'[p,q,r] ∀ p<q<r for alignment between reverse of S and reverse of T w.r.t. triple w' which is the triple above w (for upper region of S).
  - Values for B'[p,q,r,w'] are available ∀ p<q<r





That is, for any p,q,r, we know the score of aligning the lower part of (p,q,r) of T to (a) and the upper part of (p,q,r) of T to (b)

Find p_h,q_h,r_h such that $B[p_h,q_h,r_h]+B'[p',q',r']$ is maximum where $p'=n-r_h+1$, $q'=n-q_h+1$, $r'=n-p_h+1$ [It takes $O(n^3)$ time.]

For this step: Time: $O(mn^3)$; Space: $O(n^3)$

Alignment between S[1..m] and T[1..n]

- Select the middle triple w of S
- Divide the whole S into two subpseudoknots: upper region excluding w and lower region including w.
- Use score-only function to compute B[p,q,r] ∀ p<q<r for alignment between S and T w.r.t. triple w (for lower region of S).
- Use score-only function to compute B'[p,q,r] ∀ p<q<r for alignment between reverse of S and reverse of T w.r.t. triple w' which is the triple above w (for upper region of S).
- Find q_h,q_h,r_h such that $B[p_h,q_h,r_h]+B'[p',q',r']$ is maximum where $p'=n-r_h+1$, $q'=n-q_h+1$, $r'=n-p_h+1$
- Partition T into two subpseudoknots: upper region excluding (p_h,q_h,r_h) and lower region including (p_h,q_h,r_h)
- Recursively find out optimal alignment between upper region of S and that of T.
- Recursively find out optimal alignment between lower region of S and that of T.

Result

- For simple pseudoknot,
  - Time and space complexity of PAL algorithm is $O(mn^3)$.
  - For our algorithm, space required=$O(n^3)$; same time complexity $O(mn^3)$.
- The algorithm can be extended to handle the case for embedded-simple pseudoknot [secondary structure with regions of simple pseudoknot and regions without pseudoknot],
  - Time and space complexity of PAL algorithm is $O(mn^4)$.
  - For our algorithm, space required=$O(mn^2+n^3)$; time required also $O(mn^4)$.

Experimental Results

RNA Family	Mp	Ave. memory usage in OUR algorithm (M)	Ave. memory usage in PAL algorithm (M)
Corona_FSE	53	5.2	30
Tymo_tRNA-like	12	6.4	10
Parecho CRE	21	11	31
IFN-gamma	73	23	282
Telomerase-vert	100	839	>4,000

Mp – length of triple sequence in pseudoknot region

Future Directions

- Consider other types of pseudoknots
- Consider other scoring functions such as energy-based functions

<Thank you>