Inter*net* Electronic Journal of Molecular Design

May 2004, Volume 3, Number 5, Pages 361–367

Editor: Ovidiu Ivanciuc

Special issue dedicated to Professor Nenad Trinajstić on the occasion of the 65th birthday
Part 11

Guest Editor: Douglas J. Klein

Identifying the Largest Common Substructure of RNA Structures

Bo Liao[1] and Tian–ming Wang[2]

[1] Department of Applied Mathematics Dalian University of Technology, Dalian 116024, China
[2] Hai Nan Normal Unervisity, Hai Kou 571158, China

Received: July 8, 2003; Revised: March 29, 2004; Accepted: April 25, 2003; Published: May 31, 2004

Citation of the article:
B. Liao and T. Wang, Identifying the Largest Common Substructure of RNA Structures, *Internet Electron. J. Mol. Des.* **2004**, *3*, 361–367, http://www.biochempress.com.

Copyright © 2004 BioChem Press

Internet Electronic Journal of Molecular Design

BioChem Press
http://www.biochempress.com

Identifying the Largest Common Substructure of RNA Structures[#]

Bo Liao[1,]* and Tian–ming Wang[2]

[1] Department of Applied Mathematics Dalian University of Technology, Dalian 116024, China
[2] Hai Nan Normal Unervisity, Hai Kou 571158, China

Received: July 8, 2003; Revised: March 29, 2004; Accepted: April 25, 2003; Published: May 31, 2004

Internet Electron. J. Mol. Des. **2004**, *3* (5), 361–367

Abstract

The primary structure of a ribonucleic acid (RNA) molecule is a sequence of nucleotides (bases) over the four–letter alphabet {A, C, G, U}. The secondary structure of an RNA is a set of free bases and base pairs formed bonds between A–U and C–G. For secondary structures, these bonds have been traditionally assumed to be one–to–one and non–crossing. We consider the largest common substructure (LCS) between two RNA molecule structures taking into account the primary and the secondary structures. We present a dynamic programming algorithm for identifying the largest common substructure of two RNA structures. The proposed algorithm solve the LCS problem in time $O(mn)$.

Keywords. RNA structures; computational biology; dynamic programming; molecular biology.

1 INTRODUCTION

Ribonucleic acid (RNA) is an important molecule which performs a wide range of functions in the biological system. RNA has recently become the center of much attention because of its catalytic properties, leading to an increased interest in obtaining structural information. More and more people show interest in computing the similarity between RNA structures [8,9,10].

Almost all comparisons of primary RNA structures are based on the comparison of strings. As is well–known, string comparisons are computer intensive, and despite the fact that practical schemes for sequence comparison have been outlined, there are a number of steeps in such approaches that involve arbitrary decisions, *e.g.*, decisions on the relative weights of different elementary string operations: deletion, insertions, substitution, and penalties for unacceptable alignments. The similarity between two structures have been formulated as problems of exact and approximate structure matching, finding a largest common substructure of the structures and computing optimal

[#] Dedicated to Professor Nenad Trinajstić on the occasion of the 65th birthday.
* Correspondence author; phone: 86–411–4706103; fax: 86–411–4706100; E–mail: dragonbw@163.com.

alignments under general scoring functions [1,3–7]. Consequently, much of the work on comparing the secondary structures of two RNAs have been modeled as problems of comparing two trees. [2,3,11,12]. The idea is to view an alignment of two strings a and b or trees a and b, which has the minimum number of edit operations. Obviously, comparison of two sequences/structures should also produce a set of largest common subsequences or substructures(LCS), and a correction between two sequences/structures and each LCS.

In this paper, we shall focus on the LCS problems of RNA structures taking into consideration both the primary structure and secondary structure provided with the string representation. We present a dynamic programming algorithm for identifying the largest common substructure of two RNA structures directly, instead of edit distance over strings and trees.

2 METHODS AND RESULTS

2.1 Primary Structure

The single stranded RNA is view as a linear sequence $\alpha = a_1 a_2 \cdots a_n$ of ribonucleotides. The sequence α is called the primary structure. Each a_i is identified with one of four bases or nucleotides: A, C, G, U.

An RNA sequence can be described as a string over the alphabet $\Omega = \{A, C, G, U\}$. Given two RNA sequences $S_1 = a_1 a_2 \cdots a_m, S_2 = b_1 b_2 \cdots b_n, a_i, b_j \in \Omega$, the LCS problem is to find a largest substructure $C_l = c_1 c_2 \cdots c_l, c_k \in \Omega, 1 < k < l$ satisfies $c_k = a_{i_k} = b_{j_k}$ and

$$i_1 < i_2 < \cdots < i_l, j_1 < j_2 < \cdots < j_l$$

Let $LCS\{A_i, B_j\}$ denote the largest common substructures of S_1 and S_2. we suppose

$$A_0 = \Phi, A_1 = a_1, A_2 = a_1 a_2, \cdots, A_m = a_1 a_2 \cdots a_m$$
$$B_0 = \Phi, B_1 = b_1, B_2 = b_1 b_2, \cdots, B_n = b_1 b_2 \cdots b_n$$

then

$$LCS\{A_i, B_j\} = \begin{cases} \Phi & \text{if } i, j = 0 \\ LCS\{A_{i-1}, B_{j-1}\} \cup \{a\}, a \in \{A, U, C, G\} & \text{if } a_i = b_j = a, i, j > 0 \\ \max\{LCS\{A_{i-1}, B_j\}, LCS\{A_i, B_{j-1}\}\} & \text{if others} \end{cases}$$

$$LCS\{S_1, S_2\} = LCS\{A_m, B_n\}$$

Algorithm 1: Computing $LCS\{A_m, B_n\}$

Step 1: Let

$$LCS\{A_i,\Phi\} \leftarrow \Phi, i = 1,\cdots,m$$
$$LCS\{\Phi,B_j\} \leftarrow \Phi, j = 1,\cdots,n$$
$$i \leftarrow 1$$

Step 2: If $i \le m$, then let $j \leftarrow 1$ go **Step 3**, else go **Step 7**.

Step 3: If $j \le n$, then go **Step 4**, else $i \leftarrow i+1$, go **Step 2**.

Step 4: If $a_i = b_j = a, a \in \{A,C,G,U\}$, then

$LCS\{A_i,B_j\} \leftarrow LCS\{A_{i-1},B_{j-1}\} \cup \{a\}$, go **Step 3**; else go **Step 5**.

Step 5: If $LCS\{A_{i-1},B_j\} \supseteq LCS\{A_i,B_{j-1}\}$ then

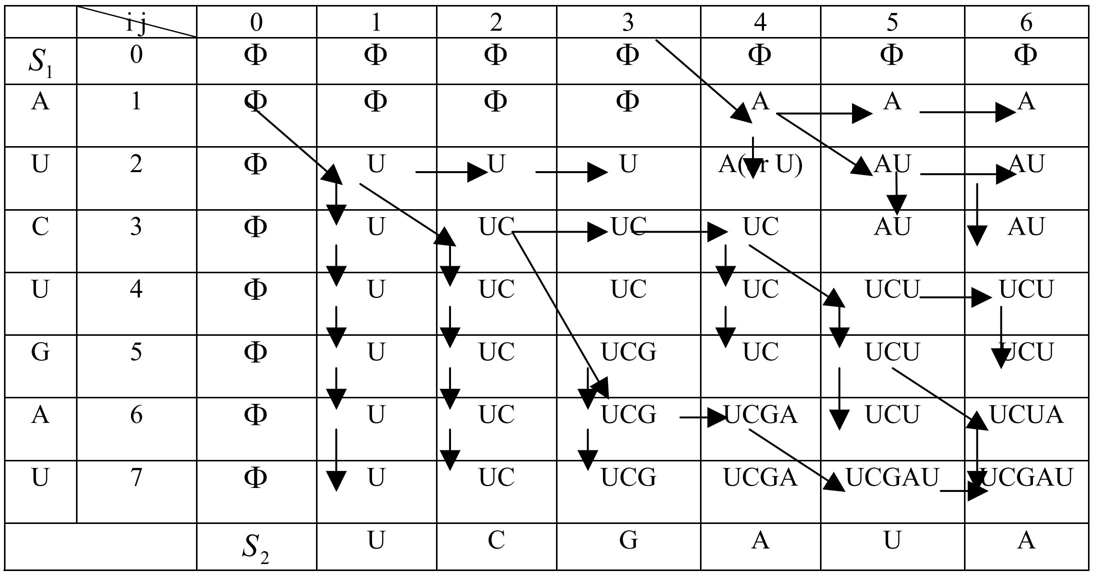
$LCS\{A_i,B_j\} \leftarrow LCS\{A_{i-1},B_j\}, j \leftarrow j+1$, go **Step 3**, else go **Step 6**.

Step 6: $LCS\{A_i,B_j\} \leftarrow LCS\{A_i,B_{j-1}\}, j \leftarrow j+1$, go **Step 3**.

Step 7: Print $LCS\{A_m,B_n\}$

End

For example, we suppose $S_1' = AUCUGAU, S_2' = UCGAUA$



	i j	0	1	2	3	4	5	6
S_1	0	Φ	Φ	Φ	Φ	Φ	Φ	Φ
A	1	Φ	Φ	Φ	Φ	A	A	A
U	2	Φ	U	U	U	A(or U)	AU	AU
C	3	Φ	U	UC	UC	UC	AU	AU
U	4	Φ	U	UC	UC	UC	UCU	UCU
G	5	Φ	U	UC	UCG	UC	UCU	UCU
A	6	Φ	U	UC	UCG	UCGA	UCU	UCUA
U	7	Φ	U	UC	UCG	UCGA	UCGAU	UCGAU
		S_2	U	C	G	A	U	A

From the above table we can obtain

$$LCS\{A_{i_1},B_1\} = U, i_1 = 2,3,4,5,6,7; LCS\{A_{i_2},B_2\} = UC, i_2 = 3,4,5,6,7$$
$$LCS\{A_{i_3},B_3\} = UCG, i_3 = 5,6,7; LCS\{A_{i_4},B_3\} = UC, i_4 = 3,4$$
$$LCS\{A_{i_5},B_4\} = UCGA, i_5 = 6,7; LCS\{A_7,B_j\} = UCGAU, j = 5,6$$
$$LCS\{A_1,B_{j_1}\} = A, j_1 == 4,5,6; LCS\{A_{i_6},B_5\} = UCU, i_6 = 4,5,6$$
$$LCS\{A_2,B_2\} = U; LCS\{A_{i_7},B_5\} = AU, i_7 = 2,3$$
$$LCS\{A_6,B_6\} = UCAU; LCS\{A_{i_8},B_6\}, i_8 = 4,5$$

2.1 Secondary Structure

The secondary structure of an RNA is a set of free bases and base pairs formed bonds between A–U and C–G. For secondary structures, these bonds have been traditionally assumed to be one–to–one and non–crossing.

Definition (Waterman [9]): A secondary structure is a vertex–labeled graph on n vertices with an adjacency matrix A fulfilling

(i) $a_{i,i+1} = 1 \quad for\ 1 \le i \le n-1$

(ii) For each i there is at most a single $k \ne i-1, i+1$ such that $a_{i,k} = 1$

(iii) If $a_{i,j} = a_{k,l} = 1$ and $i < k < j$ then $i < l < j$

We will call an edge $(i,k), |i-k| \ne 1$ a base pair. A vertex i connected only to $i-1$ and $i+1$ will be called unpaired. A vertex i is said to be interior the base pair (k,l), if $k < i < l$. If, in addition, there is no base pair (p,q) such that $k < p < i < q$, we will say that i is immediately interior to the base pair (k,l).

A string representation S of RNA secondary structure can be obtained by the following rules:

(i) If vertex i is unpaired, we suppose it is base $a \in \Omega$ then $S_i = "a"$.

(ii) (ii) If (p,q) is a base pair and $p < q$ then $S_p = "("$ and $S_q = ")"$.

These rules yield a sequence of matching brackets and free bases, which is different the string representation in paper[9]. Obviously, a secondary structure Γ can be described as a string S over the alphabet $\Omega' = \{A, C, G, U, (,)\}$.Without losing generality ,we suppose that $(\,)_1$ denote the base pair "A–U" and $(\,)_2$ denote the base pair "G–C".

Let $a = a'_1 a'_2 \cdots a'_m, b = b'_1 b'_2 \cdots b'_n, a'_i, b'_j \in \Omega', i = 1,2,\cdots,m, j = 1,\cdots,n$, let $LCS\{a,b\}$ denote the largest common substructures of a and b. we suppose

$$A'_0 = \Phi, A'_1 = a'_1, A'_2 = a'_1 a'_2, \cdots, A'_m = a'_1 a'_2 \cdots a'_m$$
$$B'_0 = \Phi, B'_1 = b'_1, B'_2 = b'_1 b'_2, \cdots, B'_n = b'_1 b'_2 \cdots b'_n$$

Theorem 1

$$LCS\{A'_i, B'_j\} = \begin{cases} \Phi & if\ i,j = 0 \\ LCS\{A'_{i-1}, B'_{j-1}\} \cup \{a\}, a \in \Omega & if\ a'_i = b'_j = a, i, j > 0 \\ LCS\{A'_{k-1}, B'_{k'-1}\} \cup \{\mathrm{S}'\} & \text{where } A'_i = A'_{k-1}(S'), B'_j = B'_{k'}(S') \\ LCS\{A'_{k-1}, B'_{k'-1}\} & where\ A'_i = A'_{k-1}(S''), B'_j = B'_{k'}(S^*), S'' \ne S^* \\ \max\{LCS\{A'_{i-1}, B'_j\}, LCS\{A'_i, B'_{j-1}\}\} & others \end{cases}$$

$$LCS\{a,b\} = LCS\{A'_m, B'_n\}$$

Proof: Obviously, $LCS\{A'_i, B'_j\} = \Phi$, if $i, j = 0$

If $a_i = b_j = a, a \in \Omega$, $LCS\{A'_i, B'_j\} \leftarrow LCS\{A'_{i-1}, B'_{j-1}\} \cup \{a\}$,

If $a'_i = b'_j = ")_t", t = 1,2$ there must be exit $a'_k =_t ($, $b'_{k'} =_t ($

We suppose $A'_i = A'_{k-1}(S'), B'_j = B'_{k'-1}(S^*), 1 \le k \le i-2, 1 \le k' \le j-2$;

if $S' = S^*$, then $LCS\{A'_i, B'_j\} \leftarrow LCS\{A'_{i-1}, B'_{j-1}\} \cup \{(S')_t\}$

if $S' \ne S^*$, then $LCS\{A'_i, B'_j\} \leftarrow LCS\{A'_{i-1}, B'_{j-1}\}$,

If $a'_i \ne b'_j$, then $LCS\{A'_i, B'_j\} = \max\{LCS\{A'_i, B'_{j-1}\}, LCS\{A'_{i-1}, B'_j\}\}$

Algorithm 2: Computing $LCS\{A'_m, B'_n\}$

Step 1: Let

$$LCS\{A'_i, \Phi\} \leftarrow \Phi, i = 1, \cdots, m$$
$$LCS\{\Phi, B'_j\} \leftarrow \Phi, j = 1, \cdots, n$$
$$i \leftarrow 1$$

Step 2: If $i \le m$, then let $j \leftarrow 1$ go **Step 3**, else go **Step 7**.

Step 3: If $j \le n$, then go **Step 4**, else $i \leftarrow i+1$, go **Step 2**.

Step 4: If $a'_i = b'_j = a, a \in \{A, C, G, U\}$, then

$$LCS\{A'_i, B'_j\} \leftarrow LCS\{A'_{i-1}, B'_{j-1}\} \cup \{a\}, \text{ go } \textbf{Step 3};$$

else if $a'_i = b'_j = ")_t", t = 1,2$ suppose $A'_i = A'_{k-1}(S'), B'_j = B'_{k'-1}(S^*), 1 \le k \le i-2, 1 \le k' \le j-2$

if $S' = S^*$, then $LCS\{A'_i, B'_j\} \leftarrow LCS\{A'_{i-1}, B'_{j-1}\} \cup \{(S')_t\}$, go **Step 3**;

if $S' \ne S^*$, then $LCS\{A'_i, B'_j\} \leftarrow LCS\{A'_{i-1}, B'_{j-1}\}$, go **Step 3**;

else go **step 5**;

Step 5: If $LCS\{A'_{i-1}, B'_j\} \supseteq LCS\{A'_i, B'_{j-1}\}$ then

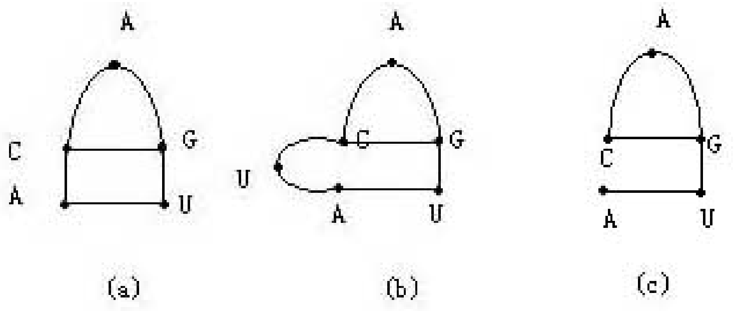
$LCS\{A'_i, B'_j\} \leftarrow LCS\{A'_{i-1}, B'_j\}, j \leftarrow j+1$, go **Step 3**, else go **Step 6**.

Step 6: $LCS\{A'_i, B'_j\} \leftarrow LCS\{A'_i, B'_{j-1}\}, j \leftarrow j+1$, go **Step 3**.

Step 7: Print $LCS\{A'_m, B'_n\}$

End

For example, we suppose α, β be the following structures (a) and (b), respectively.



From Algorithm 2, we can obtain the following table.

		0	1	2	3	4	5	6
α	0	Φ	Φ	Φ	Φ	Φ	Φ	Φ
$_1($	1	Φ	$_1($	$_1($	$_1($	$_1($	$_1($	$_1($
$_2($	2	Φ	$_1($	$_1($	$_1(_2($	$_1(_2($	$_1(_2($	$_1(_2($
A	3	Φ	$_1($	$_1($	$_1(_2($	$_1(_2(A$	$_1(_2(A$	$_1(_2(A$
$)_2$	4	Φ	$_1($	$_1($	$_1(_2($	$_1(_2(A$	$_1(_2(A)_2$	$_1(_2(A)_2$
$)_1$	5	Φ	$_1($	$_1($	$_1(_2($	$_1(_2(A$	$_1(_2(A)_2$	$_1(_2(A)_2)_1$
		β	$_1($	U	$_2($	A	$)_2$	$)_1$

From the above table we can obtain $LCS\{\alpha,\beta\} = {}_1(_2(A)_2)_1$ corresponding to substructure (c).

4 CONCLUSIONS

We present a dynamic programming algorithm for identifying the largest common substructure of two RNA structures, which needs no any score functions and computations of edit distance over strings and trees. Using our approach, one can obtain the largest common substructure directly instead of the minimum number of edit operations– insertion, deletion or substitution of one symbol–to transform one string or tree into another.

Acknowledgment
The authors would like to thank the anonymous referees for many valuable suggestions that have improved this manuscript.

5 REFERENCES

[1] M. S. Waterman, Introduction to Computational Biology: Maps, Sequences and Genomes, Chapman & Hall, London, 1995.
[2] J. T. L. Wang and K. Z. Zhang, Identifying approximately common substructures in tree based on a restricted edit distance, *Information Sci.* **2000**, *126*, 165–189.
[3] D. Angluin, Finding patterns common a set of strings, *J. Comput. System Sci*. **1980**, *21*, 46–62.
[4] W. J. Masek, A Faster Algorithm Computing string Edit Distances, *J. Comput. System Sci*. **1980**, *20*, 18–31.
[5] V. Chratal and D. Sankoff, Longest Common subsequences of two random sequences, *J. Appl. Probab*. **1975**, *12*, 306–315.
[6] D. S. Hirschberg, A linear space algorithm for computing maximal common subsequences, *Commun*. *ACM 18*, 341–343.
[7] C. N. S. Pedersen, Algorithms in Computational Biology, in: BRICS Dissertation Series 2000.
[8] X. G .Vienmt and M. v. de Chaumont, Enumeration of RNA's secondary structures by complexity, in: V. Capasso, E. Grosso, S. L. Paver–Fontana (Eds), Mathematics in Medicine and Biology, Lect. Notes in Biomath, Vol. 57 Springer, Berlin, 1985, 360–365.
[9] I. L. Hofacker, P. Schuster, and P. F. Stadler, Combinatorics of RNA secondary structures, *Discr. Appl. Math*. **1998**, *88*, 207–237.
[10] P. Hogeweg and B. Hesper, Energy directed folding of RNA sequences, *Nucl. Acids Res*. **1984**, *12*, 67–74.
[11] K. Zhang and D. Shasha, Simple fast algorithms for the editing distance between trees and related problems, *SIAM J. Comput*. **1989**, *18*, 1245–1262.
[12] K. Zhang, R. Statman, and D. Shasha, On the editing distance between unordered labeled trees, *Inform. Proc. Lett.* **1998**, *42*, 133–139.