ABSTRACT

A NEW APPROACH TO FEATURE EXTRACTION FOR RNA STRUCTURE COMPARISON

by
Girish Prakash Walawalkar

In recent years, RNA structural comparison becomes a crucial problem in bioinformatics research. Generally, it is a popular approach for representing the RNA secondary structures with arc-annotation sets. Several methods can be used to compare two RNA structures, such as tree edit distance, longest arc-preserving common subsequence (LAPCS) and stem based alignment. However, these methods may be helpful only for small RNA structures because of their high time complexity. In this thesis, we propose a simplified method to compare two RNA structures in O(mn) time, where m and n are the lengths of the two RNA sequences, respectively. The method transforms the RNA structures into specific sequences called object sequences, then compare these object sequences to find their common substructures. The comparison method is tested with 118 RNA structures obtained from RNase P Database. For any two structures, it is important to identify whether they are in the same family by both structure comparison and sequence comparison. In the experiment, it is found that the method for comparing RNA structures can yield better hit rates and is faster than the traditional method to compare the RNA sequences. Therefore, the approach to extract and compare the RNA secondary structures is more sensitive in biology and more efficient in time complexity.

A NEW APPROACH TO FEATURE EXTRACTION FOR RNA STRUCTURE COMPARISON

by
Girish Prakash Walawalkar

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer Science

January 2006

APPROVAL PAGE

A NEW APPROACH TO FEATURE EXTRACTION FOR RNA STRUCTURE COMPARISON

Girish Prakash Walawalkar

Dr. Jason T. Wang, Dissertation Advisor — Date
Professor of Computer Science, NJIT

Dr. Chengjun Liu, Committee Member — Date
Assistant Professor of Computer Science, NJIT

Dr. Qun Ma, Committee Member — Date
Assistant Professor of Computer Science, NJIT

BIOGRAPHICAL SKETCH

Author: Girish Prakash Walawalkar

Degree: Master of Science

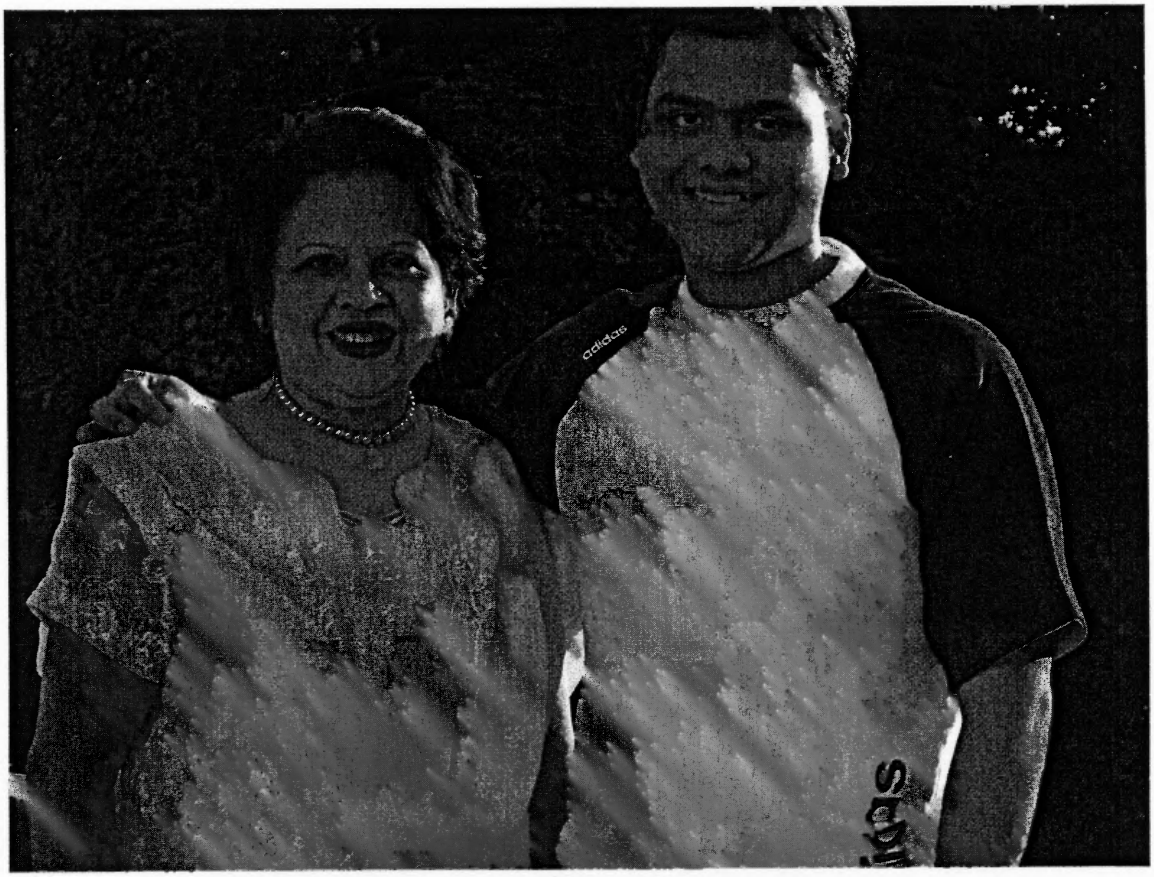
Date: January 2006

Undergraduate and Graduate Education:

- Master of Science in Computer Science,
  New Jersey Institute of Technology University, Newark, NJ 2006

- Bachelor of Science in Computer Science
  Vidyalankar Institute of Technology, Mumbai, India 2004

Major: Computer Science



Dedicated to my family for their support, guidance, love and blessings throughout my life

Father:	Prakash	Dhondu	Walawalkar
Mother:	Shobha	Prakash	Walawalkar
Brother:	Prasad	Prakash	Walawalkar
Brother:	Milind	Prakash	Walawalkar

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Jason Wang, who not only served as my research supervisor, providing valuable and countless resources, insight, and intuition, but also constantly gave me support, encouragement, and reassurance. Special thanks are given to Dr. Chengjun Liu and Dr. Qun Ma for actively participating in my committee.

Many of my fellow graduate students in the Data and Knowledge Engineering Laboratory are deserving of recognition for their support. I also wish to thank Clarissa Gonzalez-Lenahan for her assistance.

I would like to thank, Dr. Ronald Kane, , my friend Sunkara, Prathy Usha and my family members Shobha Prakash Walawalkar, Prakash Walawalkar, Prasad Walawalkar and Milind Walawalkar for their guidance and for playing very important role in my success.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

CHAPTER 1

INTRODUCTION

1.1 Background

In recent years, bioinformatics is a new research area, which grows up quickly. As its name suggests, it uses the technology of computer science to help biologists dealing with the huge amounts of information that obtain from the experiments. Generally, most of the bioinformatics works can be described as analyzing the biological data, such as sequence alignment and structure comparison. To understand what problems one are going to deal with, one would like to introduce some basic biological knowledge in this chapter, such as DNA and RNA. Afterwards, one would introduce how to represent an RNA secondary structure

1.2 DNA

There are various kinds of living things in the world, each has its own characteristics and looks. They are different because they have different DNAs (deoxyribonucleic acids). DNA exists in the cells and plays the role to define the characteristics of creatures. As a result, sometimes one also call DNA the gene code and some people believes that DNA sis the code designed from the god.

In 1953, James Watson and Francis Crick discovered the structure of DNA, the double helix. Each single helix in DNA is a long sequence composed of nucleotides. There are four types of nucleotides A, G, C, T to form the DNA helix. In order to form the double helix from two single helix, there are two types of pairs A-T and C-G to connect these two helix.

1.3 RNA

From the view of the nucleotides, the main difference in nucleotide between DNA and RNA is the replace of 'T' by 'U'. RNA sequence is composed of four types of nucleotides A, C, G, U. Generally speaking, RNA is similar to DNA. However, it is single stranded and shorter that DNA sequences. In addition, DNA only records the inheritable message, while there are three main types of RNA and each has its own function. One would briefly introduce these three types of RNA in the following.

1. mRNA (messenger RNA): mRNA plays an important role in translation, because it copies the necessary message from DNA in order to produce protein.

2. rRNA (ribosomal RNA): rRNA is one of the material of a large molecule, ribosome. Ribose will cover the specific mRNA section and some tRNA in order to translate mRNA into proteins.

3. tRNA (transfer RNA): tRNA is a small RNA which always brings three nucleotides and one amino acid with it. These three nucleotides are also called the anti-codon, because it will bind with the complementary site(called codon) in mRNA to make tRNA release its amino acid which is used to form the amino acid chain. The produced amino acid chain is also called the primary structure of the protein.

1.4 RNA Structure Representation

The sequence composed of the nucleotides A, G, C and U, is called the primary structure of RNA, or simply the RNA sequence. Nonetheless, due to the hydrogen bonds, an RNA sequence will fold into a secondary structure. These three types of base pairs (hydrogen bond) are as follows:

1. G-C base pair with a triple-hydrogen bond.
2. A-U base pair with a double-hydrogen bond.
3. G-U base pair with a single-hydrogen bond.

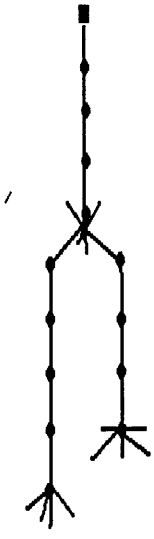
Given the RNA sequence and its base pair set, of course one would know what this RNA structure is. However, there are at least six kinds of methods to represent RNA structures.

1. Full representation
2. Circle representation
3. Tree representation
4. Arc-annotated representation
5. Mountain representation
6. Bracket representation

For the above six representations, one would like to show a simple example with a short RNA sequence and its secondary structure representations. Except for the above mentioned representations, RNA structure can also be represented by Stochastic Context Free Grammar (SCFG), which is implemented to represent the backbone for predicting the structure of another homologous sequence.

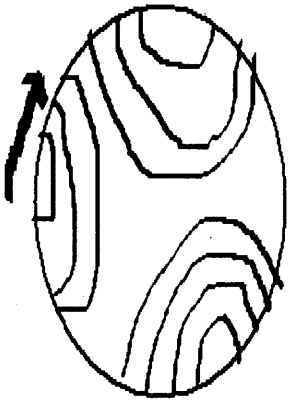
Among all representations, one choose the arc-annotated representation to implement the algorithm, because it can easily reveal the nested loops and one can use it to describe the method very fluently. One will discuss the representation in detail in the latter chapter.

Full Representation



Tree Representation

Arc annotated



Circle Representation

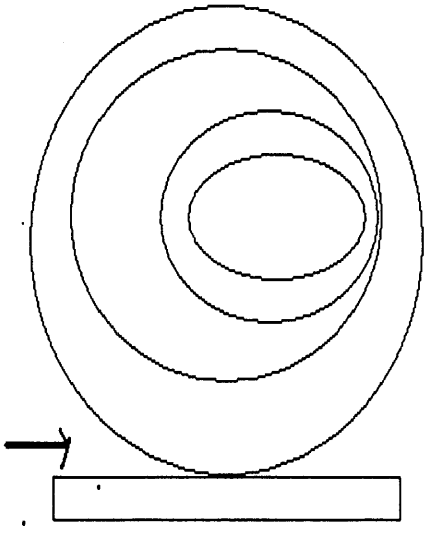


Figure 1.1 Various kinds of representations of RNA Secondary Structures.

1.5 Pseudoknots on RNA Secondary Structures

Pseudoknot means the base pairs in the RNA structure could be crossing. If there exists pseudoknot in the RNA structure, some representations like arc-annotated representation will also be crossing. Pseudoknot is important because it affects the structure, in biology one say it will affect the function of the RNA. A website named Pseudobase[2] on http://wwwbio.leideuniv.nl/~Batenburg/PKB.html provides useful information about the pseudoknots found in various kinds of RNAs.

However, considering the pseudoknot usually makes the structural prediction or comparison problems much more difficult. In order to simplify the problems, one will only discuss the RNA structure without pseudoknots.

In Chapter 2, one will introduce some related study on RNA structures and give an illustration on the field one focus on. Also one will introduce an important technique called dynamic programming, which is crucial to the research. In Chapter 3, one will take a look at some related algorithms for comparing RNA structures. Afterwards, in Chapter 4 one will give a detailed illustration and analysis for the method. In Chapter 5, one record the experimental result and some interesting discoveries. At last in Chapter 6 one will come to the conclusions.

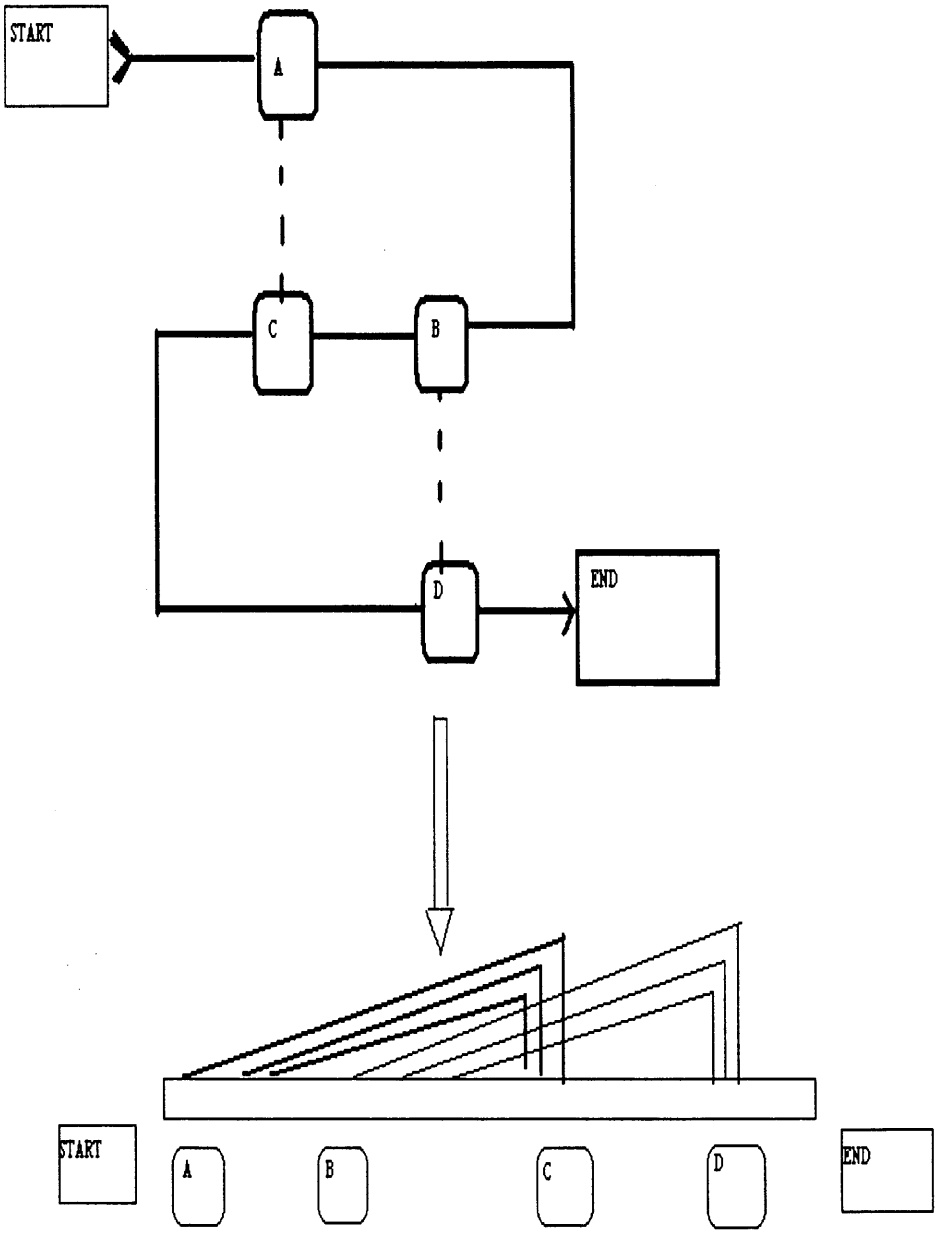


Figure 1.2 The sample of psuedoknots.

CHAPTER 2

PRELIMINARIES

2.1 Background

In this chapter, one would present some research that is related to RNA structure. By doing so, one may not only give some examples for what could be done with the RNA structure, but also give an explanation to the research one are working on. Afterwards, one would introduce an important technique called dynamic programming for solving algorithmic problems. Finally, one will introduce a typical problem in bioinformatics called LCS (Longest Common Subsequence), which can be solved by means of dynamic programming and would be very useful to explain the algorithm in the latter chapters.

2.2 Related Research on RNA structures

For RNA structure, there are two main kinds of research. one is the structure prediction and the other is the structure comparison. one will give a brief description in the following.

1. Structure prediction:

This type of research tries to predict the secondary structures of a given RNA sequence. The method for prediction could be on thermodynamics to find the structure with lowest energy, based on the given structure (backbone) to predict the structure of another given sequence, or based on sequence similarity to predict local structure. Additional biological and chemical knowledge may be useful to improve the accuracy.

2. Structure comparison:

This type of research tries to find the common substructure between two given RNA structures, each has its own RNA sequence and base pair set. Common substructure may be defined as the longest arc-preserving common subsequence of the two given arc-annotated sequences, the common sub-tree in the tree representation, or the stem-based alignment. As the definitions are different from the view, the computational algorithms are also different from their difficulty and time complexity.

The research focuses on the feature extraction and comparison, not structure prediction. In addition, one only consider the structure without pseudoknots. Therefore, please keep in mind that two RNA secondary structures without pseudoknot should be given in advance.

2.3 Dynamic Programming

Dynamic programming is am important technique for solving algorithmic problems. Briefly speaking, it is a technique which can avoid the repeated computation and obtain the optimal solution in a way more efficient that exhaustive search. In fact, a lot of examples show that replacing exhaustive search by dynamic programming can greatly reduce the time complexity.

In order to understand the characteristics of dynamic programming, consider an easy example. Consider the definition of the Fibonacci sequence in the following.

$$Fib(1) = 1$$

$$Fib(2) = 1$$

$$Fib(n) = Fib(n-1) + Fib(n-2)$$

From the definition above One can get the sequence 1, 1, 2, 3, 5, 8, 13, 21.... and the recursion tree of Fib(6) can be represented in Figure 2.1 if One implement the recursive formula directly. In figure 2.1 One can obtain that One compute Fib(3) three times and One

```
                    Fib(6)
                  /       \
            Fib(4)         Fib(5)
              / \            / \
        Fib(2)   Fib(3)   Fib(3)   Fib(4)
                  / \      / \     /   \
            Fib(1)  Fib(2)  Fib(1) Fib(2) Fib(2) Fib(3)
                                                  / \
                                              Fib(1) Fib(2)
```

Recursion tree for Fib(6).

also compute Fib(4) twice, which implies that there is a lot of redundant work. In addition, if One try to analyze the time complexity, One can see that T(n) = T(n-1) + T(n-2), which means that T(n) grows in exponential rate.

However, one can use the concept of dynamic programming to reduce the required time. In the case of Fibonacci sequence, a better method is to record the

computed Fib(x) in a table. Whenever one need Fib(x), one can read it in the table instantly, rather than spend time to recomputed Fib(x). In fact, one only need to record the last two values. Figure 2.2 shows an illustration for finding the Fib(6) by using the dynamic programming. In Figure 2.2, one can see that the time complexity can be reduced to O(n), which is obviously a great improvement. Therefore, one can say that dynamic programming is an important technique when one design algorithms.

STEP 1:

Fib(1) = 1	Fib(2) = 1	Fib(3) = ?	Fib(4) = ?	Fib(5) = ?	Fib(6) = ?

STEP 2:

Fib(1) = 1	Fib(2) = 1	Fib(3) = 2	Fib(4) = ?	Fib(5) = ?	Fib(6) = ?

Fib(1) = 1 + Fib(2) = 1 == Fib(3) = 2

STEP 3:

Fib(1) = 1	Fib(2) = 1	Fib(3) = 2	Fib(4) = 3	Fib(5) = ?	Fib(6) = ?

Fib(2) = 1 + Fib(3) = 1 == Fib(4) = 3

STEP 4:

Fib(1) = 1	Fib(2) = 1	Fib(3) = 2	Fib(4) = 3	Fib(5) = 5	Fib(6) = ?

Fib(3) = 2 + Fib(4) = 3 == Fib(5) = 5

STEP 5:

Fib(1) = 1	Fib(2) = 1	Fib(3) = 2	Fib(4) = 3	Fib(5) = 5	Fib(6) = 8

Fib(4) = 3 + Fib(5) = 5 == Fib(6) = 8

Figure 2.2 Calculation of Fib(6) using Dynamic Programming.

2.4 Longest Common Subsequence Problem

Given two biological sequence S1 and S2, a fundamental problem in bioinformatics is to find the similarity between them. Although one do not know the exact meaning of similarity, finding the longest common subsequence between S1 and S2 is still a good approach since what one find is the common primary structures between S1 and S2. Therefore, the longest common subsequence problem (LCS) becomes important when one analyze primary structures. A simple example is that S1 = ACGGUAG and S2 = GUCAGA, then one of the LCS between S1 and S2 is GCGA.

As described before, LCS problem can be solved using dynamic programming. Suppose S1= a1a2a3.....am, S2=b1b2b3.....bn. Let L(i, j) denote the score of the best LCS score of a1a2a3.....ai and b1b2b3.....bj where ai and bj are two characters. The recursive formula for calculating L(i.j) is given as follows.

$$
\begin{aligned}
L(i, j) = \max\{ & L(i-1, j) \\
& L(i, j-1) \\
& L(i-1, j-1) + \text{match}(ai, bj) \} \\
\text{match}(ai, bj) = \{ & 1 \text{ ; if } ai \text{ is equal to } bj \\
& 0 \text{ ; if } ai \text{ is not equal to } bj \}
\end{aligned}
$$

From the formula one can solve this problem by means of a two-dimensional lattice. Consider S1 = AGCGUAG and S2 = GUCAGA. One now construct a lattice to obtain the length of the longest common subsequence. In this case, One can obtain that one of the longest common subsequences between S1 and S2 is GCGA.

As for the time complexity of solving the LCS using the dynamic programming, it is obvious that building the lattice costs O(mn) time and also the tracing for solution path

costs O(m + n) time. Therefore the time complexity is O(mn) + O(m + n) = O(mn). In addition, the dynamic programming used to solve LCS can be easily modified to solve other biological problems such as sequence alignment and edit distance.

	- ------	A	C	C	C	U	A	C
- -------	0	0	0	0	0	0	0	0
C	0	0	1	1	1	1	1	1
U	0	0	1	1	1	2	2	2
C	0	0	1	2	2	2	2	2
A	0	1	1	2	2	2	3	3
C	0	1	2	2	3	3	3	4
A	0	1	2	2	3	3	4	4

Figure 2.3 The scoring Lattice for LCS.

	- --------		G	C	G		A	
- -------	0	0	0	0	0	0	0	0
G	0	0	1	1	1	1	1	1
	0	0	1	1	1	2	2	2
C	0	0	1	2	2	2	2	2
	0	1	1	2	2	2	3	3
G	0	1	2	2	3	3	3	4
A	0	1	2	2	3	3	4	4

Figure 2.4 Tracing the solution path in LCS.

CHAPTER 3

RELATED ALGORITHMS

3.1 Background

In this chapter, one would introduce some related algorithms for RNA secondary structure comparison. Generally, all of the methods can be classified into four types. They are Hamming distance, tree edit distance, longest arc preserving common subsequence and stem-based alignment. One will give a brief introduction to these methods in the following sections.

3.2 Hamming Distance

Given two sequences, the time complexity required for computing the Hamming distance between them is O(n). Therefore, one can also find the Hamming distance between two RNA secondary structures if one can reduce the structure to a numerical sequence. Here one would provide an example.

Let G=0, C=1, A=2, U=3, GC=0, CG=1, AU=2, UA=3, GU=4 AND UG=5, Figure 3.1 shows an illustration how to reduce RNA secondary structures to numerical sequences.

After one obtain the numerical sequences NS1 and NS2, one can compute the Hamming distance between NS1 and NS2 and treat this result as the score for similarity. The time complexity of this method is only O(n), therefore it could be a choice if one out emphasis on the computing time. In addition, the way to reduce structures to sequences can also depend on the users. However, the comparison is too rough to be used.

RNA A	RNA B
.(((((((...)))) ...)))) . CGCCGGCGGGCCCUGGGGCA	.(((((((...)))) ...)))). GUGGGUCUCCGACAGUCCGA
Base pair reduced C011401GGC---CGG---A	Base pair reduced G500031UCC---ACU---A
Unpair bases reduced 1011401001---100---1	Unpair bases reduced 0511131311---203---2
Obtain numerical sequences 10114010011002	Obtain numerical sequences 05000313112032

Figure 3.1 Reduction from structures to sequences.

3.3 Tree Edit Distance

Given two trees, to find their edit distance can also be an approach to compare two RNAs since one can represent an RNA secondary structure as an ordered tree. The tree edit distance problem allows three kinds of operation, which are relabeling, insertion and deletion. Figure 3.2 is an illustration for these three kinds of operation.

Given two ordered trees T1 with and t2, the tree edit distance between T1 and T2 cal also be solved using dynamic programming, which is delivered by K. Zhang and D.Shasha in 1989 with O(| T1|2 |T2|2) time complexity in worst case, where |T1| and |T2| denote the number of nodes in T1 and T2, respectively. Up to now, Zhang-Shasha tree edit algorithm is still famous reference when dealing with tree edit problems. In 2003, S. Dulucq and L. Tichit delivered an exact analysis and proved that the time complexity of Zhang Shasha tree edit distance algorithm in average case is O(|T1|3/2 |T2|3/2|).

For tree edit distance, there also exists many related problems, such as finding the largest common subtree, finding the smallest common super-tree, giving some constraint to the editing rules or expanding the editing rules etc. As the problem differs, the time complexity is also different. Some of related problems may be NP-hard such as tree conclusion problem between two unordered tree. However, one shall not introduce them one by one in detail since the research does not focus on tree edit distance.

The main advantage of dealing with tree edit distance is that it provides a more detailed view for structure comparison. However, one still cannot apply it directly on the structures with pseudoknots since this kind of structures cannot be easily represented by trees.

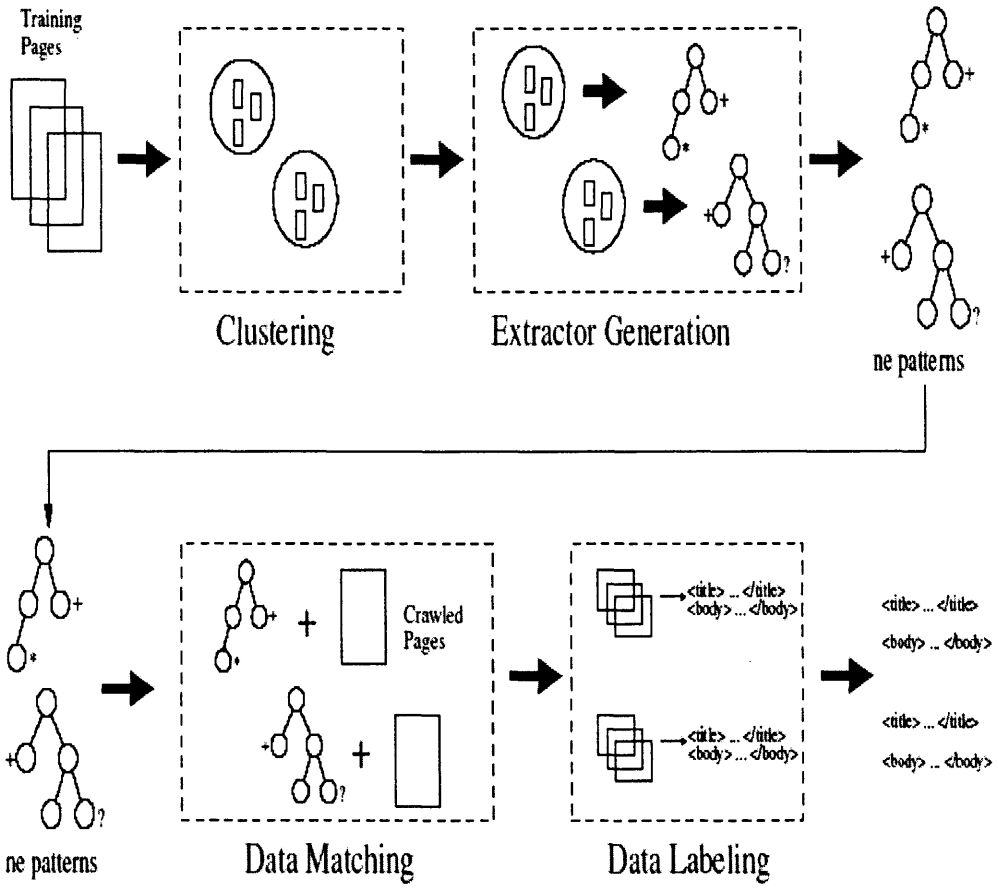


Figure 3.2 Operations in the tree edit distance problem.

3.4 Longest Arc-Preserving Common Subsequence Problem

In this section, One would introduce another LCS-like problem called longest arc-preserving common subsequence (LAPCS). In LAPCS problem, the given sequences are arc-annotated, which is also a representation of RNA secondary structures. A formal definition for LAPCS can be described as follows.

Given arc-annotated sequences (S1, A1) and (S2, A2), where S1 and S2 are the sequences and A1 and A2 are the arc annotation sets for S1 and S2, respectively, the longest arc-preserving common subsequence between S1 and S2 is a mapping MS. MS is a subset of $\{1 \ldots |S1|\} * \{1 \ldots \{S2\}\}$ between the positions of S1 and S2 satisfying that.

1. The mapping is one-to-one and preserves the order of the subsequence:

   For all $(i1, j1), (i2, j2) \in MS$

   $i1 = i2 \leftrightarrow j1 = j2$ and

   $i1 < i2 \leftrightarrow j1 < j2$

2. The arcs induced by the mapping are preserved:

   for all $(i1, j1), (i2, j2) \in MS$

   $(i1, i2) \in A1 \leftrightarrow (j1, j2) \in A2$

3. The mapping produces common subsequence:

   for all $(i, j) \in MS$, $S[i] = S[j]$

To understand the meaning of the above definition for LAPCS, here One take Figure 3.3 as example. In Figure 3.3, one can see that the common subsequence C1 is not an arc-preserving subsequence while the common subsequence C2 is. Note that C1 cannot preserve the A-U arc both in sequences S1 and S2, hence it does not satisfy the above second constraint. However, C2 satisfies all of the constraints and in this case it is

the longest arc-preserving common subsequence between sequences S1 and S2. For LAPCS problems, the arc annotation can be classified into five levels, which are unlimited, crossing, nested, chain and plain. As the annotated level changes, the time complexities of the LAPCS algorithms also change. It was first proved in 1999 that LAPCS (crossing, crossing) is NP-hard and if one of the input sequence has the annotated level higher than nested, then this type of LAPCS problem is also NP-hard. Afterwards, LAPCS(nested, nested) was also proved to be NP-hard in 2001.

For those sequences with arc annotated level loOner than nested, there exists polynomial time algorithms to find LAPCS. However, most of the RNA secondary structures are still nested in arc annotation, even there is no pseudoknots. Therefore, LAPCS (nested, nested) needs to be properly approximated. Recently, much research focuses on how to design a good approximation algorithm to find LAPCS between two nested RNA secondary structures.

3.5 Stem-Based Alignment

In stem-based alignment, the input structures are two sequences and their arc annotation sets. The arcs in the sets cannot be crossing. Stem-based alignment tries to find an alignment that maximizes the score from arc match and base match by breaking some arcs. If one of the given structures in plain (no arcs), then stem-based alignment will only compute the sequence alignment score.

Stem-based alignment is different from LAPCS since the alignment result may not be a subsequence. Moreover, it is not the same as tree edit distance problem because breaking an arc does not mean to relabel or to delete a node in a tree. In the tree edit

distance, relabeling and deletion will change or remove the bases. However, in stem-based alignment, breaking an arc only removes the arc from the arc annotation set, but it does not remove the bases associated with the removed arc. Given two nested arc-annotated sequences (S1, A1) and (S2, A2). Where S1 and S2 are the sequences, A1 and A2 are the arc annotation for S1 and S2, respectively, let seqA (i1, j1,i2, j2) be the sequence alignment score and SBA(i1, j1, i2, j2) be the stem-based alignment between two structures S1[i1] ~ S2[j1] and S2[i2] ~ S2[j2], i1<j1 and i2<j2. Then the recursive formula can be given as follows.

If there exist (a1, b1) Є A1, (a2, b2)) Є A2, a1>i1 and a2>i2 such that (a1, b1) is an arc in A1 which is closest to i1 and (a2, b2) is an arc in A2 that is closest to i2, then

$$
\begin{aligned}
SBA(i1, j1, i2, j2) = \max \quad & | \; SBA\ (i1, j1, i2, j2) \text{ with } (a1, b1) \text{ removed}, \\
& | \; SBA\ (i1, j1, i2, j2) \text{ with } (a2, b2) \text{ removed}, \\
& | \; SBA\ (a1+1, b1-1, a2+1, b2-1) \\
& \quad + SBA(i1, a1-1, a2-1) \\
& \quad + SBA\ (b1+1, j1, b2+1, j2) + \text{match}
\end{aligned}
$$

If one of the given structures is a plain structure, in other words the arc annotation set is empty, then SBA(i1, j1, i2, j2) = SeqA (i1, j1, i2, j2). In stem-based alignment, the arcs can also be merged into stem blocks to reduce the number of stems in A1 and A2, which is also a good idea in the method. Stem-based alignment can be Solved O(R1*R1*R2*R2) where R1 and R2 are the numbers of stems in the given structures. If

one modify the input parameters properly, stem-based alignment can maximize the number of arcs which are matched. Figure 3.4 is an illustration for stem-based alignment that maximize the matched arcs by breaking two arcs in structure A and one arc in structure B. Stem-based alignment can also be extended to edit distance problem by adding other operation rules. In 2002, a general edit distance between two RNA secondary structures was delivered. Up to now, one know that for some editing rules, the optimal solution for the edit distance between a crossing structure and a nested structure can solve in O(R1R2mn), where R1 and R2 denote the numbers of stems, m and n denote the lengths of the sequences. For different editing rules, the difficulty of the edit problems will also be different. For example, if one adopt the general editing rule, then EDIT(crossing, nested) and EDIT(nested, nested) can only be approximated under a reasonable scoring scheme. How to choose the feasible editing rules based on the biological knowledge is another field of research, since the time complexity required for solving EDIT(nested, plain) adopting the general editing rules in O(n*n*n*n), which is not applicable to long sequences. Good approximation can also be helpful.

CHAPTER 4

THE METHOD

4.1 Background

In this chapter, one will introduce the algorithm for finding the longest common substructure. The algorithm is based on the concept of LCS and it merges the base pairs into objects. It is not the same as tree alignment because one only consider the object sequence alignment not the insertion and deletion of every node in the tree. Also, it is different from stem-based alignment because each object can only be compared to another object from outside to inside. Due to these differences, One can prove that the time complexity of the method is O(nm) where m and n denote the lengths of the two RNA sequences. The space complexity of he algorithm is also O(nm). One will introduce the method clearly in the following.

4.1 Merging Arcs Into Objects

In the method an object is defined as a group of base pairs that are nested are consecutive, or very close. Here one use the integers to represent an object, which are the start of the head, the end of the head, the start of the tail and the end of the tail, respectively. From these the integers, one can easily obtain the thicknesses and capacity. Figure 4.1 shows a simple example. By means of merging close arcs into a single object, one can reduce the number of objects found. This idea is similar to stem, however, here one can set different merging size to fit other situation. If the merging size is set to k, then one merge the nested arcs into an object, satisfying that the distance between any two consecutive arcs is not greater that k. For example, when the merging size is set to 1, every object will

actually be a stem. Note that the thickness of the head and the tail of the object may be different since the merging size may not be 1.

4.3 The Object Sequence

Given an arc-annotated RNA secondary structure, one can scan the RNA sequence and find all object sequences. Figure 4.2 shows an example that an RNA secondary structure contains two object sequences. Note that object sequences may appear inside another object because the structure is nested. In Figure 4.2, one can see that an object from 19 to 43 in A1 contains another object sequence from 20 to 42. Here one name these object sequences according to the start point of the sequence. Therefore, the object sequence scanned from 1 to 60 is named A1 and that scanned from 20 to 42 is named A20. If the length of the RNA sequence is n, then one will cost O(n) to find all object sequences by scanning the sequence along the base pairs.

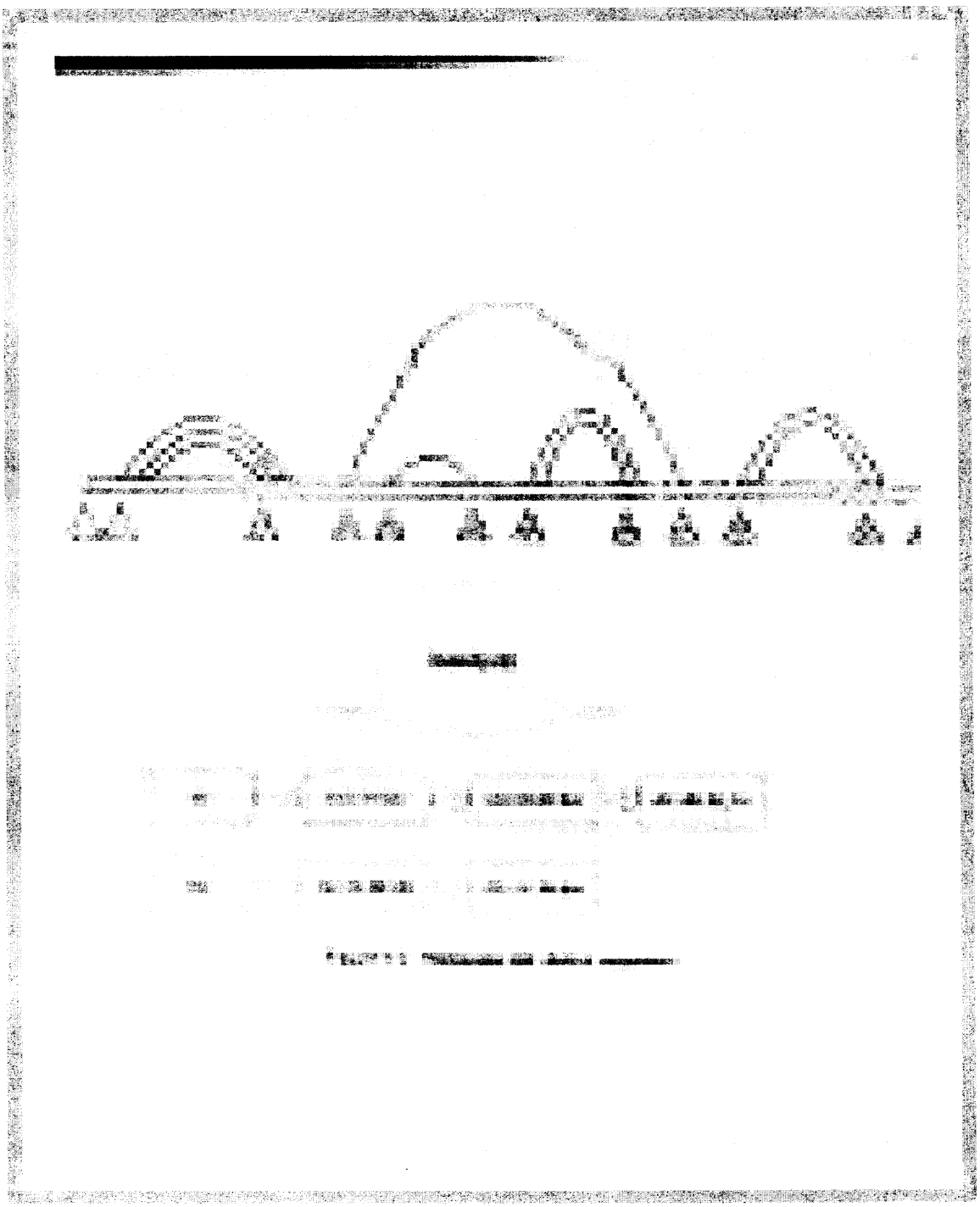


Figure 4.2 Obtaining the object sequence.

4.4 LCS in Object Sequences

Finding the best match between two object sequences is very similar to LCS problem. It can also be solved by the concept of dynamic programming, which one call object based LCS (OLCS).

Given two object sequences A = a1a2a3.....am, B = b1b2b3....bn, let S(i,j) denote the best match score of the longest common subsequence between a1a2....ai and b1b2b2...bj where ai and bj represent two objects. Then the recursive formula for calculating S(i,j) is given as follows.

$$S(i, j) = \max \begin{cases} S(i-1, j) \\ S(i, j-1) \\ S(i-1, j-1) + \text{match}(ai, bj) \end{cases}$$

In the above formula, the math function computes the similarity score between ai and bj. This function may have more than one definition, which means this function could be flexible. For the method to trace back the OLCS path, one build another tracing table to record the way of tracing back since the math function may be more complicated than it is used in the original LCS problem. With this additional tracing table, one can spare the time for recomputing the match score of two objects and keep the time complexity for tracing in O(m + n).

4.5 Math Function

Given two objects, the math function is the scoring function that determines the similarity between the two objects. In traditional LCS, it is easy to judge if the two objects are the same, because the object here is only a single letter. However, in the case, an object is a structure that may be further nested. Therefore, one modify the scoring function from the traditional LCS, and try to make it fit into the object comparison, called math function. one define the math function as follows.

Input: two objects a and b

Output: the match score between a and b

Step 1. Compute the outside score Mo from outside comparison, which compares the size, including the thickness and capacity.

Step 2. If both objects contain inside object sequences (say X in ai and Y in bj), then the inside score Mi is determined by OLCS(X, Y). Otherwise, Mi = 0, which means there is no inside match score.

Step 3. M = pMo + qMi, where M is the similarity score of the two objects, p and q are variable parameters.

In the above procedure, one provide a math function that compares two objects from outside to inside. Besides, the function is also flexible in each step. Which means one can easily modify each step to deal with other demands in the future.

In order to clarify the relationship between the OLCS and math function in the method, one use Figure 4.3 as an example. Note that the math function may trigger off another OLCS(in step 2 above). Nonetheless, when a new OLCS is triggered off, its problem size is actually smaller than the previous one, hence there are some smaller

blocks in the scoring table. One shall also use this table to analyze the time complexity of the algorithm later.

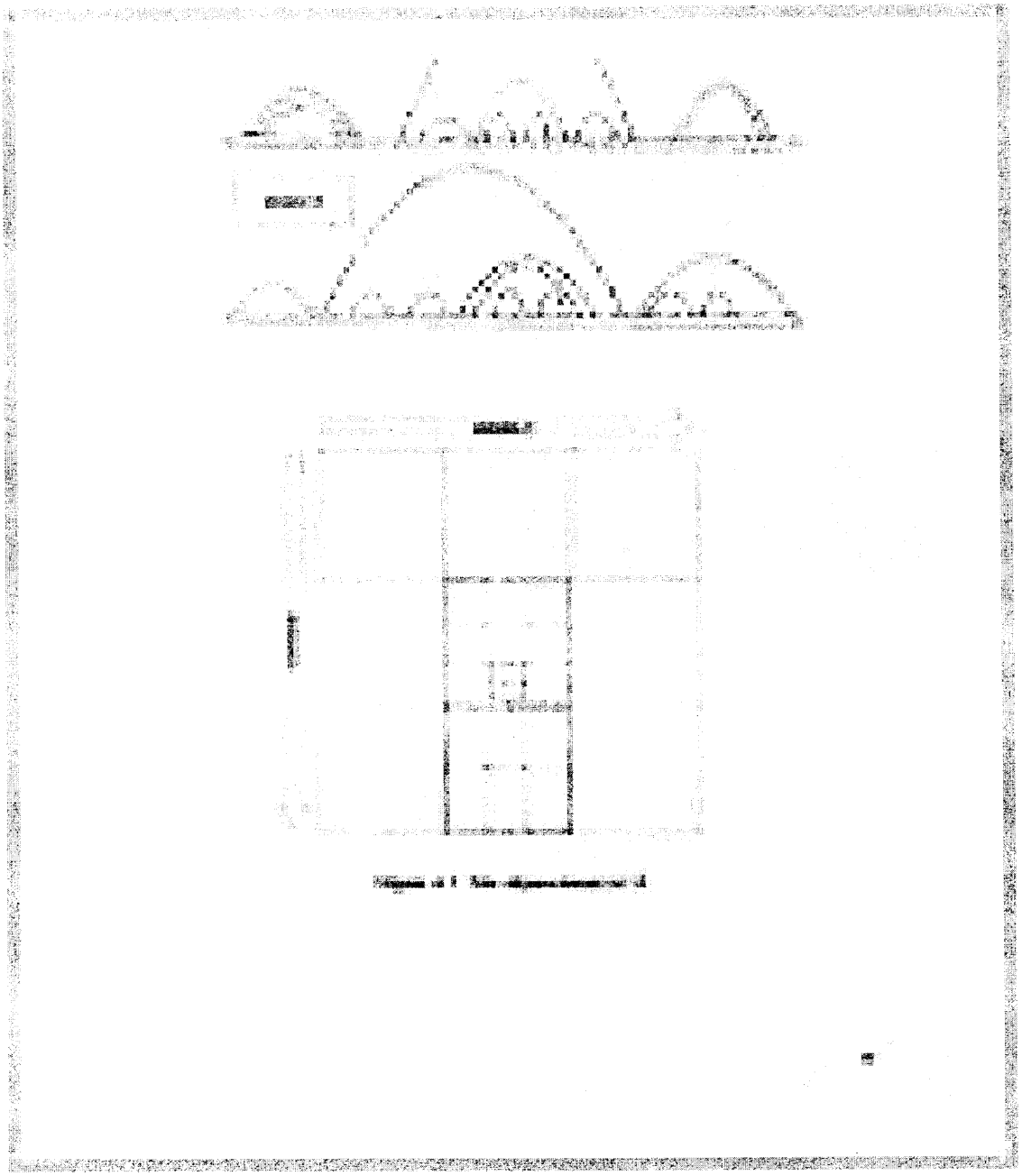


Figure 4.3 The object-based LCS.

4.6 OLCS With Different Depths

In the above example, one can see that in depth in structure A and structure B are both 3. However, in several cases the depths would be different, which implies that A may contain B or covered by B. If One always perform the OLCS for Ao and Bo, it means that one will never investigate whether A and B could contain each other, which may lead to bad solutions. Therefore, when the depth of the two structures are not the same (let A be larger than B), one will not only perform OLCS on Ao and Bo, but also find each object sequence Ai in structure A with depth equal to the structure B, then perform the OLCS on Ai and Bo. Among these solutions, one obtain the solution with the highest OLCS score and treat this solution as the longest common substructure between structure A and structure B. It is a heuristic search since the object sequence Ai must have the same depth as Bo. However, in this way the time complexity would be kept in O(mn). One will give one more detailed illustration for this idea. In figure 4.4 one can see that the depth of Ao is 3, the depth of A20 is 2, the depth of Bo is 2 and in this example the OLCS for A20 and Bo would be better than the OLCS for Ao and Bo.

4.7 The Additional Sequence Alignment Score

After one use the OLCS algorithm to find the longest common substructure, the aligned stems have been found. One can treat these aligned stems as the dividing points in the sequence. By doing so, one can split the original two sequences SA and SB into two groups of smaller pieces. Figure 4.5 shows an example of the two structures in Figure 4.3. Theoretically speaking, alignment of these smaller pieces could still reveal some

message from primary structure comparison. Therefore, one also pay attention to this score, called seq_score.

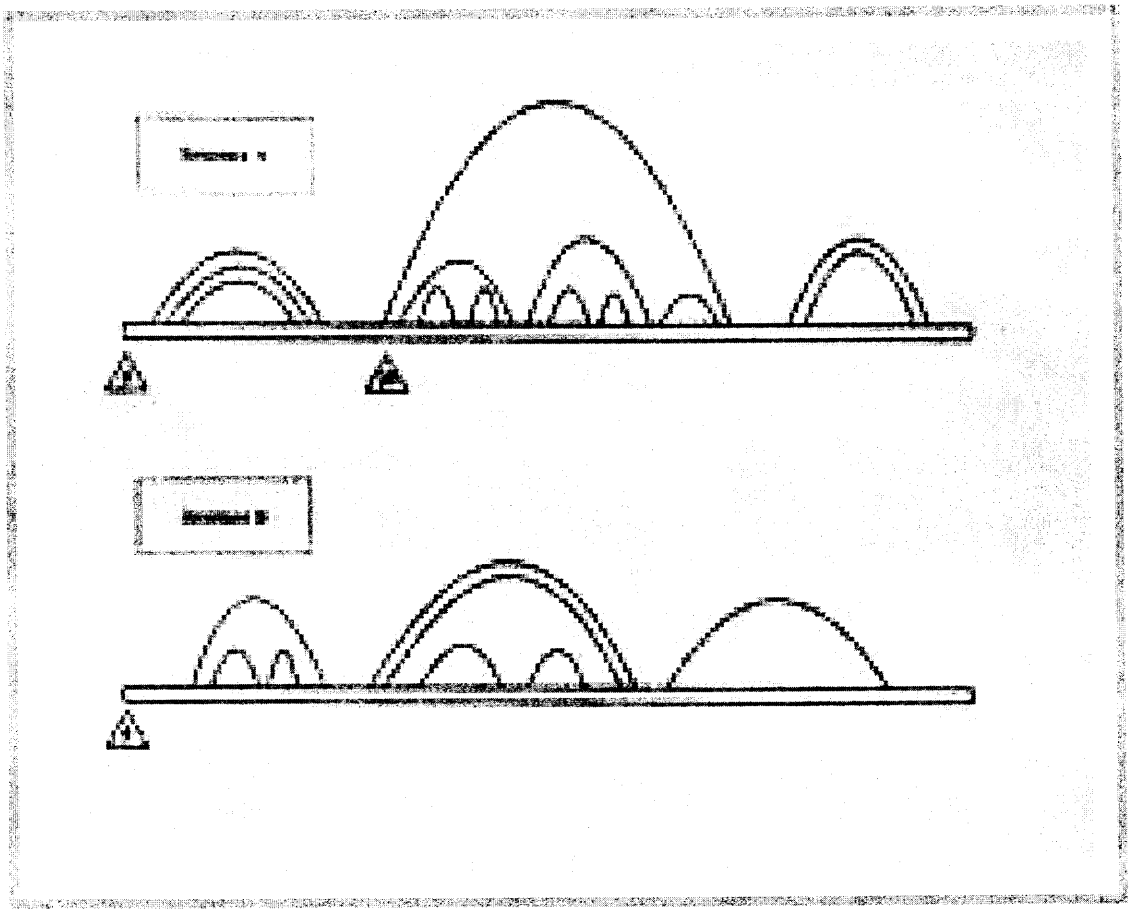


Figure 4.4 OLCS with different score.

In the method, OLCS_score could be obtained by OLCS algorithm and seq_score can be retrieved by the traditional sequence alignment. Depending in this seq_score, one can observe the structure similarity and sequence similarity individually.

4.8 Time Complexity and Space Complexity

To analyze the time complexity of the algorithm, one breaks the whole algorithm into three main steps.

Step 1. Finding the object sequences:

As mentioned before, the time complexity for this step is O(n) where n is length of the sequence. For two sequences with lengths m and n, the time complexity in this step will be O(m + n).

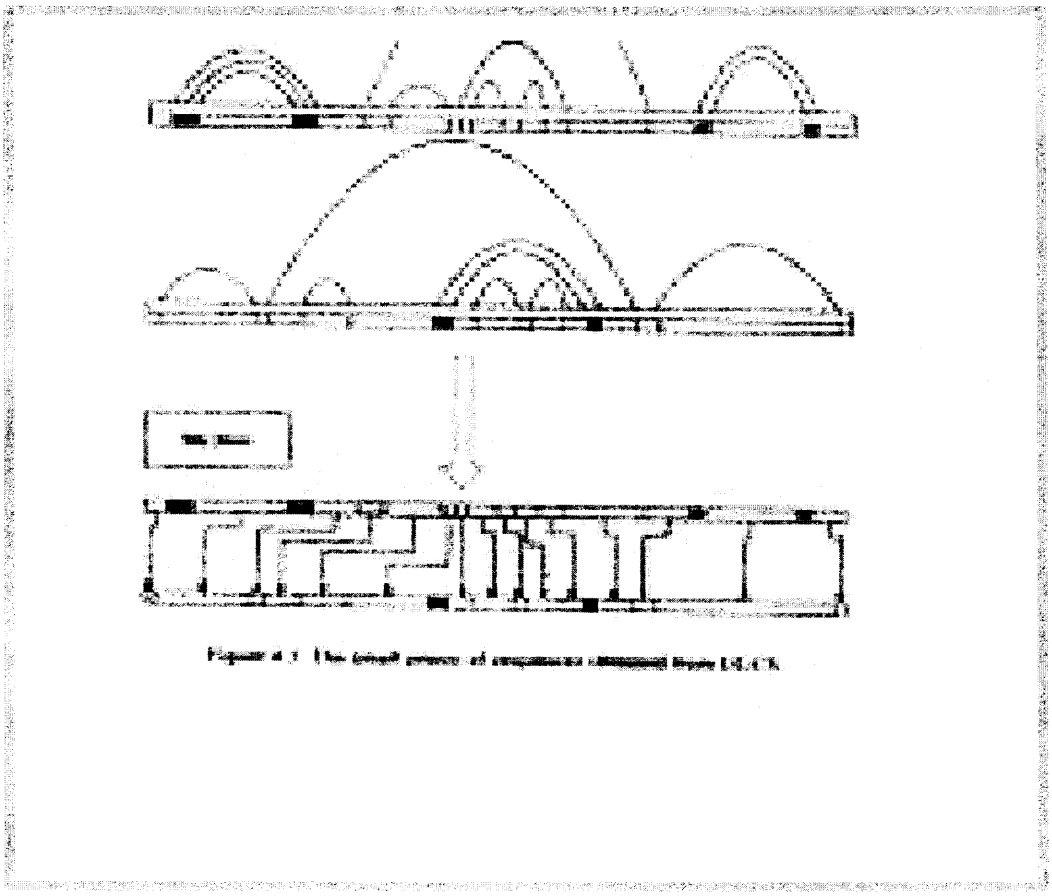


Figure 4.5 The small pieces of sequences obtained from OLCS.

Step 2. Object-based LCS:

In the previous example, one can see the scoring table for OLCS very clearly. Each block in the table may be divided into smaller blocks, which means the match function here invokes a new OLCS. If there is no new OLCS invoked, then the block will not be divided and the time complexity of computation for this block is constant. The total block numbers would be at most be mn/4 where m and n are the lengths of the two sequences. It is because that the number of rows and number of columns represent the objects in the two sequences. For any sequences of length L, it will contain no more than L/2 objects, since each object will occupy at least two letters in the sequence if the structure is nested. As a result, one can make a conclusion that there are at most mn/4 blocks and the time complexity required for computing this scoring table is O(mn) in the worst case.

For the time complexity of tracking back the solution path in this table, it is obviously O(m + n) since there are at most mn blocks in the table and one perform this trace only once after one build the whole OLCS table.

Step 3. OLCS with different depth:

If structure A is larger that structure B and B has depth d, then one need to find each object sequence Ai in A with depth d and then perform OLCS on Ai and Bo. In this case, it is clear that all Ai with depth d cannot overlap. In other words, these Ai's are independent object sequences in A. In fact, these Ai's can also be represented as subtrees with height d in A when one use the tree representation. Therefore, performing OLCS on every Ai and Bo will cost O(∑mi n) where mi is the sequence length of Ai. If all Ai's are independent to each other. Hence , one can make conclusion that O(∑mi n) will not be

greater than O(mn). By above analysis, the time complexity for building the OLCS scoring table in case with different depth is also O(mn).

Step 4: Additional sequence alignment:

Once the whole OLCS finishes, one then perform the sequence alignment for those smaller sequence pieces produced by OLCS. If one want to align two sequences with lengths m and n, the time complexity would be O(nm). However, if the sequences are breaked into smaller pieces for performing alignment individually, the computation time would be shorter that O(mn). For the worst case, the time complexity in this step is as O(mn).

From step 1 through step 4, the time complexity of the algorithm is O(m + n) + O(mn) + O(mn) + O(mn) = O(mn). For the space complexity, in order to store the whole table One need at most mn blocks, therefore the space complexity can be easily verified to be O(mn). The space complexity may be reduced to linear space by implementing the FastLSA algorithm since the algorithm is very similar to the traditional LCS algorithm. Due to the above advantages, the algorithm can use time and memory in an effective way.

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 Background

In this chapter, one would introduce the definition of similarity. By means of this definition, one can easily know how similar two RNAs are, once after one obtains the OLCS score and LCS score. Afterwards, one will show some results about comparing the RNA secondary structures. Here one provide two types of methods for comparison. On one hand, OLCS method will only compare the structural similarity. On the other hand, the traditional LCS will only compute the LCS score for the whole sequences (primary structure) without paying attention to the secondary structure. In this experiment, One can observe the relationship between the sequence and the structure.

Both OLCS and LCS are implemented by C++ and visual C++ 6.0 is the compiler. One perform the experiment on a PC with AMD Athlon XP 1800+ as processor and 256 MB DDR RAM. All the test data can be obtained from RNase P Database on the website http://www.mbio.ncsu.edu/RNaseP/. Among all test data, there exists some pseudoknots. In order to make all test data feasible, one delete the pseudoknot which can be detected in the scanning stage. The method for deleting the pseudoknot is to ignore the second object if two objects are crossing. By doing so, One can make sure that the input structures are truly nested.

5.2 Definition of Similarity

Similarity is very important because it can tell you how similar two structures are when they are compared. Here one provide a simple formula to define the similarity both in OLCS and LCS.

This simple formula is given as follows.

$$Sim = \frac{2 * Func(A, B)}{Func(A, A) + Func(B, B)}$$

Where Func in the formula can be OLCS if one want to find the structural similarity. One can also find sequence similarity if One replace the Func by LCS. A and B in the formula means two input RNA secondary structures. In this formula, if two structures are the same then Sim will be 1. Besides, if A contain B or B contains A, Sim will not be 1 because Func(A, A) + Func(B, B) > Func(A, B). According to the above reasons, this formula can reflect the similarity in an feasible way.

5.3 Family Relationship

For the convenience, one shortens some family names in the database and here one provide the mapping table and the relationship of the test data in Table 5.1 and Figure 5.1.

5.4 Distribution of Similarity

In this section, one would like to show the distribution of similarity computed by object-based LCS and traditional LCS. In Table 5.2 through Table 5.6, the in put sequences are in different families. In Table 5.2, one can see that there are 9 sequences in bacteroide.

Therefore, the total number of cases for intra family comparison in Table 5.2 is 36. At the same time in Table 5.7, there are 9 sequences in Bacteroide and 8 sequences in Crena. Therefore the total number of case for inter family comparison in Table 5.7 is 9*8 = 72. In the experiments, One set the merging size to 5, p=1 and q=1 when One are running the OLCS algorithm.

5.5 Family Identification

In this experiment, one try to identify if two given RNAs are in the same family. At first, one want to find a good bound for both OLCS and LCS. If the similarity obtained is higher than the bound, then these two RNAs will be classifying into different families. Otherwise, these two RNAs will be classified into the same family. From the results, one can see the distribution of similarity computed by two kinds of views. In these tables, one find an interesting trend. For LCS, the bound of family identification seems to be very close to 0.65 in general. Therefore, one searches the bound from 0.63 to 0.67 individually and finally one find that 0.64 is the best bound for LCS. As for OLCS, one also searches for the bound from 0.73 to 0.77 and finally one set the bound as 0.74. Here one has 6903 cases since one have 118 sequences in the sample. One list the hit rates of OLCS in the Table 5.11 and 5.15, and the hit rates of LCS in Table 5.16 through Table 5.20.

OLCS completed these 6903 comparisons in 54 seconds while LCS completes these comparisons in 275 seconds. It is because OLCS merges the arcs into objects and makes the problem size smaller.

At last, one would like to show six figures for the experiment in family identification. The bound of OLCS is set to 0.74 and that of LCS is set to 0.64. Figure 5.2

and Figure 5.3 are the successful cases for both OLCS and LCS. Figure 5.4 and Figure 5.5 are the successful cases for OLCS but not for LCS. Finally, Figure 5.6 and Figure 5.7 are the failed cases for both the OLCS and LCS. Here the successful case means the method can identify the two RNAs correctly.

For the falied cases for both OLCS and LCS, it implies that one may need other biological rules to determine the identification, or maybe these two structures were placed in the wrong family. Anyway, there are still many works to do in the future.

CHAPTER 6

CONCLUSION

In this chapter, one will make some conclusions about the method and experimental results. From algorithmic view, OLCS is an algorithm for finding common structures between two RNA secondary structures in O(mn). Although OLCS may not be very accurate as tree alignment or LAPCS due to its low complexity, it still provides another good choice when one want to compare two RNA structures. In addition , OLCS is easy to be implemented because it is modified from LCS and it is applicable because of its low complexity.

In the experimental results, one showed the structural similarity computed by OLCS and the sequence similarities computed by LCS. Generally, the vague range of structural similarity is larger than that of sequence similarity, which makes the structural similarity more sensitive. In the tables of hit rates, One can see that in some inter family identification, both OLCS and LCS has very low hit rates (Bacteroide vs Plancto, Bacteroide vs Spiroch). In these cases, one find that these two families belong to the same parent family. As a result, it is reasonable that these two families may have some substructures or subsequence in common. That is, the similarity between these two families could be a little bit high and it makes the identification more difficult.

One also find that for the intra identification of Nuclear, OLCS and LCS both has very low hit rates. In this case, one find that the number of sequences in the Nuclear is much more than that in other families, which implies this family, may contain some child families. As for the family classification (searching for the child families), one leave this work for future because at this time one have no idea about how to split the parent

correctly. If one set the bounds of OLCS and LCS to their own best bound, one can see that in Table 5.12 and Table 5.17, OLCS only loses 4 hit rates among all 21 hit rates. It implies that comparing the RNA secondary structures is more sensitive than comparing the primary structures (sequences) directly. In addition, in the experiments one find that OLCS is even faster than LCS. Therefore, one can make a conclusion that the method one defined to compare RNA secondary structures is more sensitive in biology and more efficient in time complexity.

APPENDIX

Distribution of Similarity Between RNA Sequences

Tables describe the distribution of similarity between different RNA sequences and also measure the hit rate for family identification. The Figures shows the similarity between different RNA structures.

Table 5.2 Distribution of similarity in Bacteroide(9 sequences)

		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.9 ~ 1.0	9	0
2		0.8 ~ 0.9	15	0
3		0.7 ~ 0.8	12	23
4		0.6 ~ 0.7	0	12
5		0.5 ~ 0.6	0	1
6		0.4 ~ 0.5	0	0
7		0.3 ~ 0.4	0	0
8		0.2 ~ 0.3	0	0
9		0.1 ~ 0.2	0	0
10		0.0 ~ 0.1	0	0
		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.79 ~ 0.80	1	0
2		0.78 ~ 0.79	1	2
3		0.77 ~ 0.78	2	2
4		0.76 ~ 0.77	0	1
5		0.75 ~ 0.76	1	1
6		0.74 ~ 0.75	4	3
7		0.73 ~ 0.74	1	5
8		0.72 ~ 0.73	1	2
9		0.71 ~ 0.72	1	7
10		0.70 ~ 0.71	0	0
11		0.69 ~ 0.70	0	0
12		0.68 ~ 0.69	0	1
13		0.67 ~ 0.68	0	1
14		0.66 ~ 0.67	0	1

15		0.65 ~ 0.66	0	1
16		0.64 ~ 0.65	0	1
17		0.63 ~ 0.64	0	3
18		0.62 ~ 0.63	0	3
19		0.61 ~ 0.62	0	1
20		0.60 ~ 0.61	0	0

Table 5.3 Distribution of similarity in Crena(9 sequences)

		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.9 ~ 1.0	2	1
2		0.8 ~ 0.9	12	0
3		0.7 ~ 0.8	5	8
4		0.6 ~ 0.7	1	15
5		0.5 ~ 0.6	4	4
6		0.4 ~ 0.5	4	0
7		0.3 ~ 0.4	0	0
8		0.2 ~ 0.3	0	0
9		0.1 ~ 0.2	0	0
10		0.0 ~ 0.1	0	0
		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.79 ~ 0.80	0	2
2		0.78 ~ 0.79	0	0
3		0.77 ~ 0.78	0	1
4		0.76 ~ 0.77	0	2
5		0.75 ~ 0.76	1	1
6		0.74 ~ 0.75	4	0
7		0.73 ~ 0.74	0	2
8		0.72 ~ 0.73	0	0
9		0.71 ~ 0.72	0	0
10		0.70 ~ 0.71	0	0
11		0.69 ~ 0.70	1	0
12		0.68 ~ 0.69	0	0
13		0.67 ~ 0.68	0	1

14		0.66 ~ 0.67	0	2
15		0.65 ~ 0.66	0	0
16		0.64 ~ 0.65	0	7
17		0.63 ~ 0.64	0	2
18		0.62 ~ 0.63	0	1
19		0.61 ~ 0.62	0	2
20		0.60 ~ 0.61	0	0

Table 5.4 Distribution of similarity in Plancto(9 sequences)

		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.9 ~ 1.0	20	2
2		0.8 ~ 0.9	12	14
3		0.7 ~ 0.8	4	20
4		0.6 ~ 0.7	0	0
5		0.5 ~ 0.6	0	0
6		0.4 ~ 0.5	0	0
7		0.3 ~ 0.4	0	0
8		0.2 ~ 0.3	0	0
9		0.1 ~ 0.2	0	0
10		0.0 ~ 0.1	0	0
		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.79 ~ 0.80	0	4
2		0.78 ~ 0.79	2	2
3		0.77 ~ 0.78	0	0
4		0.76 ~ 0.77	0	9
5		0.75 ~ 0.76	2	2
6		0.74 ~ 0.75	0	0
7		0.73 ~ 0.74	0	1
8		0.72 ~ 0.73	0	0
9		0.71 ~ 0.72	0	2
10		0.70 ~ 0.71	0	0
11		0.69 ~ 0.70	0	0
12		0.68 ~ 0.69	0	0
13		0.67 ~ 0.68	0	0

14		0.66 ~ 0.67	0	0
15		0.65 ~ 0.66	0	0
16		0.64 ~ 0.65	0	0
17		0.63 ~ 0.64	0	0
18		0.62 ~ 0.63	0	0
19		0.61 ~ 0.62	0	0
20		0.60 ~ 0.61	0	0

Table 5.5 Distribution of similarity in EuryTA(28 sequences)

		Similarity	\Structure (OLCS)	Sequence(LCS)
1		0.9 ~ 1.0	79	17
2		0.8 ~ 0.9	100	20
3		0.7 ~ 0.8	75	52
4		0.6 ~ 0.7	44	229
5		0.5 ~ 0.6	20	59
6		0.4 ~ 0.5	39	1
7		0.3 ~ 0.4	20	0
8		0.2 ~ 0.3	1	0
9		0.1 ~ 0.2	0	0
10		0.0 ~ 0.1	0	0
		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.79 ~ 0.80	5	2
2		0.78 ~ 0.79	5	1
3		0.77 ~ 0.78	21	2
4		0.76 ~ 0.77	21	3
5		0.75 ~ 0.76	4	7
6		0.74 ~ 0.75	2	3
7		0.73 ~ 0.74	2	2
8		0.72 ~ 0.73	10	5
9		0.71 ~ 0.72	4	7
10		0.70 ~ 0.71	1	20
11		0.69 ~ 0.70	6	14
12		0.68 ~ 0.69	7	20
13		0.67 ~ 0.68	2	20

14		0.66 ~ 0.67	10	19
15		0.65 ~ 0.66	2	32
16		0.64 ~ 0.65	0	43
17		0.63 ~ 0.64	7	18
18		0.62 ~ 0.63	3	17
19		0.61 ~ 0.62	5	23
20		0.60 ~ 0.61	2	23

Table 5.6 Distribution of similarity in Nuclear(57 sequences)

		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.9 ~ 1.0	128	23
2		0.8 ~ 0.9	255	36
3		0.7 ~ 0.8	288	136
4		0.6 ~ 0.7	348	778
5		0.5 ~ 0.6	279	623
6		0.4 ~ 0.5	256	0
7		0.3 ~ 0.4	42	0
8		0.2 ~ 0.3	0	0
9		0.1 ~ 0.2	0	0
10		0.0 ~ 0.1	0	0
		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.79 ~ 0.80	24	3
2		0.78 ~ 0.79	21	4
3		0.77 ~ 0.78	15	4
4		0.76 ~ 0.77	33	5
5		0.75 ~ 0.76	33	15
6		0.74 ~ 0.75	28	23
7		0.73 ~ 0.74	37	19
8		0.72 ~ 0.73	30	17
9		0.71 ~ 0.72	27	18
10		0.70 ~ 0.71	40	28
11		0.69 ~ 0.70	34	39
12		0.68 ~ 0.69	37	21
13		0.67 ~ 0.68	33	38

14		0.66 ~ 0.67	38	46
15		0.65 ~ 0.66	38	60
16		0.64 ~ 0.65	43	107
17		0.63 ~ 0.64	29	145
18		0.62 ~ 0.63	44	122
19		0.61 ~ 0.62	15	101
20		0.60 ~ 0.61	37	99

Table 5.7 Distribution of similarity between Bacteroide(9 sequences) and Crena(9 sequences)

		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.9 ~ 1.0	0	0
2		0.8 ~ 0.9	0	0
3		0.7 ~ 0.8	1	0
4		0.6 ~ 0.7	28	51
5		0.5 ~ 0.6	9	21
6		0.4 ~ 0.5	26	0
7		0.3 ~ 0.4	8	0
8		0.2 ~ 0.3	0	0
9		0.1 ~ 0.2	0	0
10		0.0 ~ 0.1	0	0
		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.79 ~ 0.80	0	0
2		0.78 ~ 0.79	0	0
3		0.77 ~ 0.78	0	0
4		0.76 ~ 0.77	0	0
5		0.75 ~ 0.76	0	0
6		0.74 ~ 0.75	0	0
7		0.73 ~ 0.74	0	0
8		0.72 ~ 0.73	1	0
9		0.71 ~ 0.72	0	0
10		0.70 ~ 0.71	0	0
11		0.69 ~ 0.70	1	0
12		0.68 ~ 0.69	3	0

13		0.67 ~ 0.68	1	0
14		0.66 ~ 0.67	2	5
15		0.65 ~ 0.66	1	11
16		0.64 ~ 0.65	6	12
17		0.63 ~ 0.64	7	7
18		0.62 ~ 0.63	1	2
19		0.61 ~ 0.62	1	5
20		0.60 ~ 0.61	5	9

Table 5.8 Distribution of similarity between Crena(8 sequences) and Plancto(9 sequences)

		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.9 ~ 1.0	0	0
2		0.8 ~ 0.9	0	0
3		0.7 ~ 0.8	0	0
4		0.6 ~ 0.7	17	63
5		0.5 ~ 0.6	19	9
6		0.4 ~ 0.5	18	0
7		0.3 ~ 0.4	18	0
8		0.2 ~ 0.3	0	0
9		0.1 ~ 0.2	0	0
10		0.0 ~ 0.1	0	0
		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.79 ~ 0.80	0	0
2		0.78 ~ 0.79	0	0
3		0.77 ~ 0.78	0	0
4		0.76 ~ 0.77	0	0
5		0.75 ~ 0.76	0	0
6		0.74 ~ 0.75	0	0
7		0.73 ~ 0.74	0	0
8		0.72 ~ 0.73	0	0
9		0.71 ~ 0.72	0	0
10		0.70 ~ 0.71	0	0
11		0.69 ~ 0.70	0	0
12		0.68 ~ 0.69	0	0

13		0.67 ~ 0.68	0	0
14		0.66 ~ 0.67	0	0
15		0.65 ~ 0.66	0	0
16		0.64 ~ 0.65	0	8
17		0.63 ~ 0.64	3	16
18		0.62 ~ 0.63	2	14
19		0.61 ~ 0.62	4	15
20		0.60 ~ 0.61	8	10

Table 5.9 Distribution of similarity between Bacteroide(9 sequences) and Plancto(9 sequences).

		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.9 ~ 1.0	15	0
2		0.8 ~ 0.9	33	0
3		0.7 ~ 0.8	29	14
4		0.6 ~ 0.7	4	67
5		0.5 ~ 0.6	0	0
6		0.4 ~ 0.5	0	0
7		0.3 ~ 0.4	0	0
8		0.2 ~ 0.3	0	0
9		0.1 ~ 0.2	0	0
10		0.0 ~ 0.1	0	0
		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.79 ~ 0.80	7	0
2		0.78 ~ 0.79	2	0
3		0.77 ~ 0.78	4	0
4		0.76 ~ 0.77	2	0
5		0.75 ~ 0.76	5	0
6		0.74 ~ 0.75	3	0
7		0.73 ~ 0.74	1	0
8		0.72 ~ 0.73	2	1
9		0.71 ~ 0.72	0	5
10		0.70 ~ 0.71	3	8
11		0.69 ~ 0.70	0	14
12		0.68 ~ 0.69	0	19

13		0.67 ~ 0.68	2	8
14		0.66 ~ 0.67	0	8
15		0.65 ~ 0.66	3	0
16		0.64 ~ 0.65	0	2
17		0.63 ~ 0.64	0	6
18		0.62 ~ 0.63	0	7
19		0.61 ~ 0.62	0	2
20		0.60 ~ 0.61	0	1

Table 5.10 Distribution of similarity between Nuclear(57 sequences) and EuryTA (28 sequences)

		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.9 ~ 1.0	0	0
2		0.8 ~ 0.9	0	0
3		0.7 ~ 0.8	15	0
4		0.6 ~ 0.7	167	860
5		0.5 ~ 0.6	193	733
6		0.4 ~ 0.5	424	3
7		0.3 ~ 0.4	563	0
8		0.2 ~ 0.3	232	0
9		0.1 ~ 0.2	2	0
10		0.0 ~ 0.1	0	0
		Similarity	Structure (OLCS)	Sequence(LCS)
1		0.79 ~ 0.80	0	0
2		0.78 ~ 0.79	0	0
3		0.77 ~ 0.78	0	0
4		0.76 ~ 0.77	2	0
5		0.75 ~ 0.76	1	0
6		0.74 ~ 0.75	1	0
7		0.73 ~ 0.74	0	0
8		0.72 ~ 0.73	0	0
9		0.71 ~ 0.72	6	0
10		0.70 ~ 0.71	5	0
11		0.69 ~ 0.70	2	0
12		0.68 ~ 0.69	2	0

13		0.67 ~ 0.68	27	1
14		0.66 ~ 0.67	24	3
15		0.65 ~ 0.66	23	9
16		0.64 ~ 0.65	21	53
17		0.63 ~ 0.64	11	103
18		0.62 ~ 0.63	20	255
19		0.61 ~ 0.62	20	277
20		0.60 ~ 0.61	17	208

Table 5.11 Hit rate of family identification for OLCS with 0.73 as bound (5376 hits)

	Bacteroide	Crena	Plancto	Spiroch	EuryTA	Nuclear
Bacteroide	0.944	1.000	0.111	0.143	0.905	1.000
Crena	----------	0.679	1.000	0.946	0.527	0.980
Plancto	----------	---------	--------	0.444	0.984	1.000
Spiroch	----------	----------	--------	1.000	0.776	1.000
EuryTA	----------	----------	---------	----------	0.627	0.997
Nuclear	----------	----------	---------	----------	--------	0.360

Table 5.12 Hit rate of family identification for OLCS with 0.74 as bound (5363 hits)

	Bacteroide	Crena	Plancto	Spiroch	EuryTA	Nuclear
Bacteroide	0.917	1.000	0.123	0.190	0.937	1.000
Crena	----------	0.679	1.000	1.000	0.549	0.980
Plancto	----------	---------	1.000	0.524	0.992	1.000
Spiroch	----------	----------	--------	1.000	0.776	1.000
EuryTA	----------	----------	---------	----------	0.632	0.997
Nuclear	----------	----------	---------	----------	--------	0.360

Table 5.13 Hit rate of family identification for OLCS with 0.75 as bound (5358 hits)

	Bacteroide	Crena	Plancto	Spiroch	EuryTA	Nuclear
Bacteroide	0.806	1.000	0.160	0.222	0.952	1.000
Crena	----------	0.536	1.000	1.000	0.567	0.980
Plancto	----------	---------	1.000	0.571	1.000	1.000
Spiroch	----------	----------	--------	0.810	0.867	1.000
EuryTA	----------	----------	---------	----------	0.622	0.998
Nuclear	----------	----------	---------	----------	--------	0.319

Table 5.14 Hit rate of family identification for OLCS with 0.76 as bound (5345 hits)

	Bacteroide	Crena	Plancto	Spiroch	EuryTA	Nuclear
Bacteroide	0.788	1.000	0.222	0.254	0.980	1.000
Crena	----------	0.500	1.000	1.000	0.585	0.993
Plancto	----------	---------	0.944	0.571	1.000	1.000
Spiroch	----------	----------	--------	0.810	0.878	1.000
EuryTA	----------	----------	---------	----------	0.611	0.999
Nuclear	----------	----------	---------	----------	--------	0.298

Table 5.15 Hit rate of family identification for OLCS with 0.77 as bound (5336 hits)

	Bacteroide	Crena	Plancto	Spiroch	EuryTA	Nuclear
Bacteroide	0.778	1.000	0.111	0.302	0.984	1.000
Crena	----------	0.500	1.000	1.000	0.688	1.000
Plancto	----------	---------	0.944	0.635	1.000	1.000
Spiroch	----------	----------	--------	0.714	0.929	1.000
EuryTA	----------	----------	---------	----------	0.556	1.000
Nuclear	----------	----------	---------	----------	--------	0.278

Table 5.16 Hit rate of family identification for LCS with 0.63 as bound (4747 hits)

	Bacteroide	Crena	Plancto	Spiroch	EuryTA	Nuclear
Bacteroide	0.861	0.514	0.123	0.302	0.595	0.856
Crena	----------	0.750	0.667	0.625	0.567	0.923
Plancto	----------	---------	1.000	0.429	0.448	0.789
Spiroch	----------	----------	--------	0.762	0.694	0.862
EuryTA	----------	----------	---------	----------	0.675	0.893
Nuclear	----------	----------	---------	----------	--------	0.409

Table 5.17 Hit rate of family identification for LCS with 0.64 as bound (4996 hits)

	Bacteroide	Crena	Plancto	Spiroch	EuryTA	Nuclear
Bacteroide	0.806	0.611	0.198	0.413	0.714	0.965
Crena	----------	0.679	0.889	0.714	0.638	0.971
Plancto	----------	---------	1.000	0.429	0.615	0.875
Spiroch	----------	----------	--------	0.667	0.837	0.952
EuryTA	----------	----------	---------	----------	0.627	0.958
Nuclear	----------	----------	---------	----------	--------	0.318

Table 5.18 Hit rate of family identification for LCS with 0.65 as bound (5134 hits)

	Bacteroide	Crena	Plancto	Spiroch	EuryTA	Nuclear
Bacteroide	0.750	0.778	0.222	0.460	0.865	0.998
Crena	----------	0.429	1.000	0.821	0.732	0.998
Plancto	----------	---------	1.000	0.524	0.802	0.969
Spiroch	----------	----------	--------	0.571	0.918	0.985
EuryTA	----------	----------	---------	----------	0.513	0.992
Nuclear	----------	----------	---------	----------	--------	0.251

Table 5.19 Hit rate of family identification for LCS with 0.66 as bound (5163 hits)

	Bacteroide	Crena	Plancto	Spiroch	EuryTA	Nuclear
Bacteroide	0.722	0.931	0.222	0.492	0.952	1.000
Crena	----------	0.429	1.000	0.911	0.781	0.998
Plancto	----------	---------	1.000	0.540	0.897	0.998
Spiroch	----------	----------	--------	0.571	0.974	1.000
EuryTA	----------	----------	---------	----------	0.429	0.997
Nuclear	----------	----------	---------	----------	--------	0.212

Table 5.20 Hit rate of family identification for LCS with 0.67 as bound (5157 hits)

	Bacteroide	Crena	Plancto	Spiroch	EuryTA	Nuclear
Bacteroide	0.694	1.000	0.321	0.508	0.984	1.000
Crena	----------	0.357	1.000	0.982	0.817	1.000
Plancto	----------	---------	1.000	0.603	0.956	1.000
Spiroch	----------	----------	--------	0.524	1.000	1.000
EuryTA	----------	----------	---------	----------	0.378	0.999
Nuclear	----------	----------	---------	----------	--------	0.184

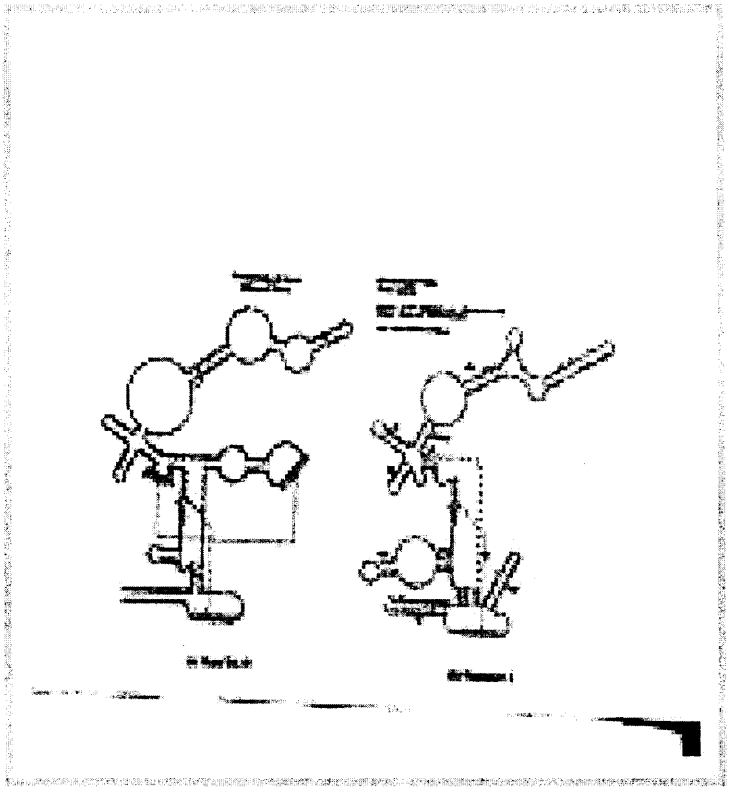


Figure 5.2 Two structures with OLCS similarity 0.9301 and LCS similarity 0.7098

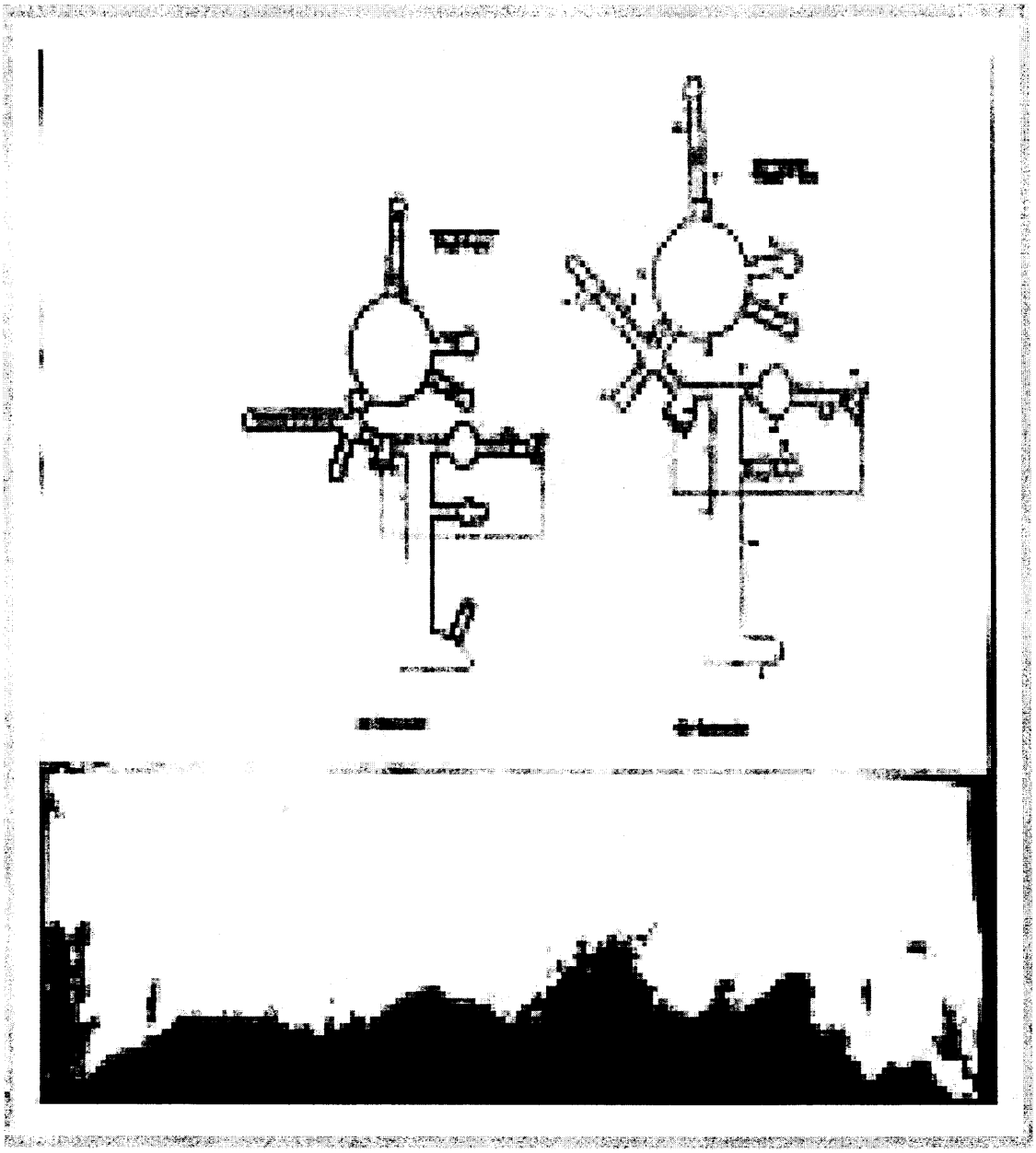


Figure 5.3 Two structures with OLCS similarity 0.9301 and LCS similarity 0.7098

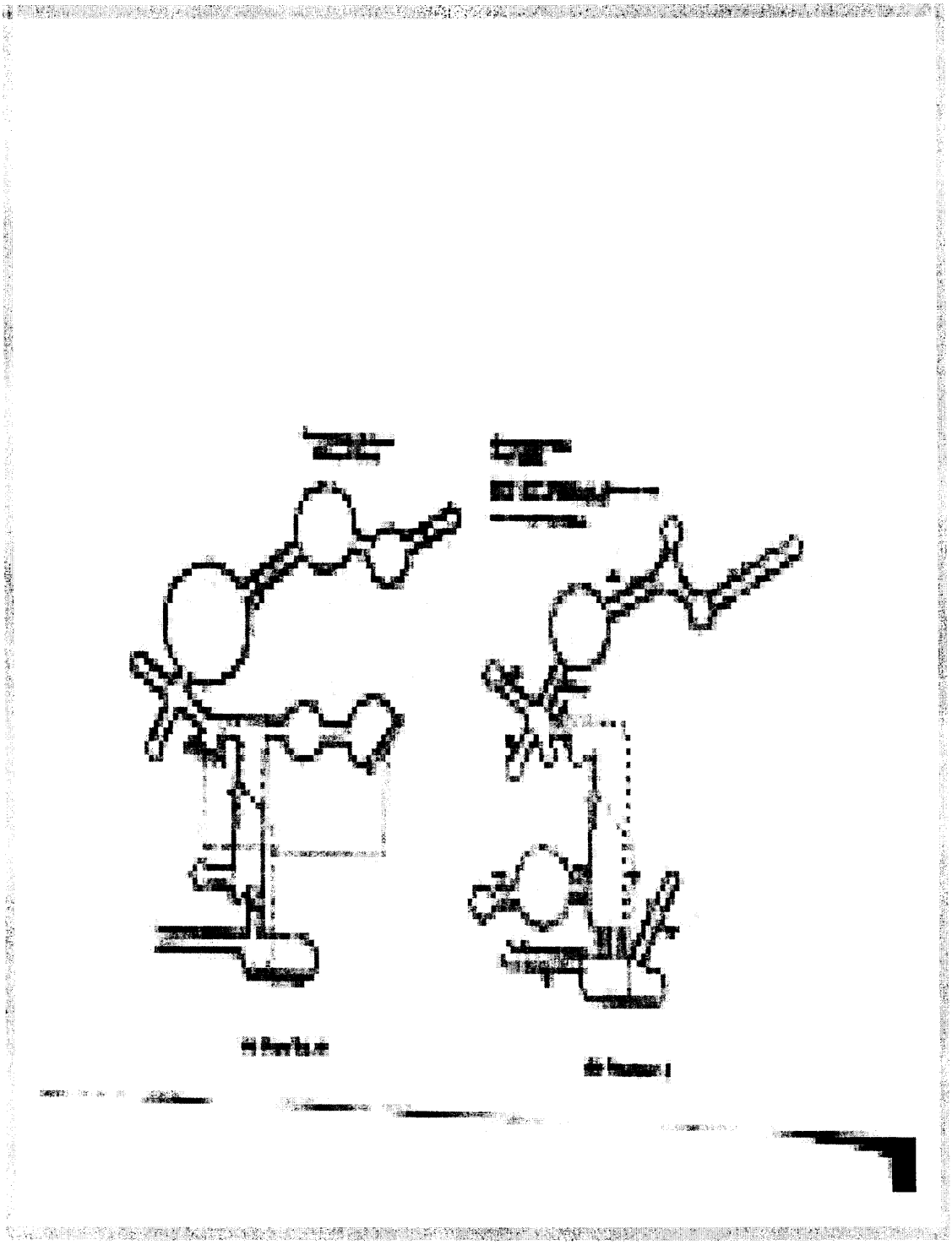


Figure 5.4 Two structures with OLCS similarity 0.3535 and LCS similarity 0.5545

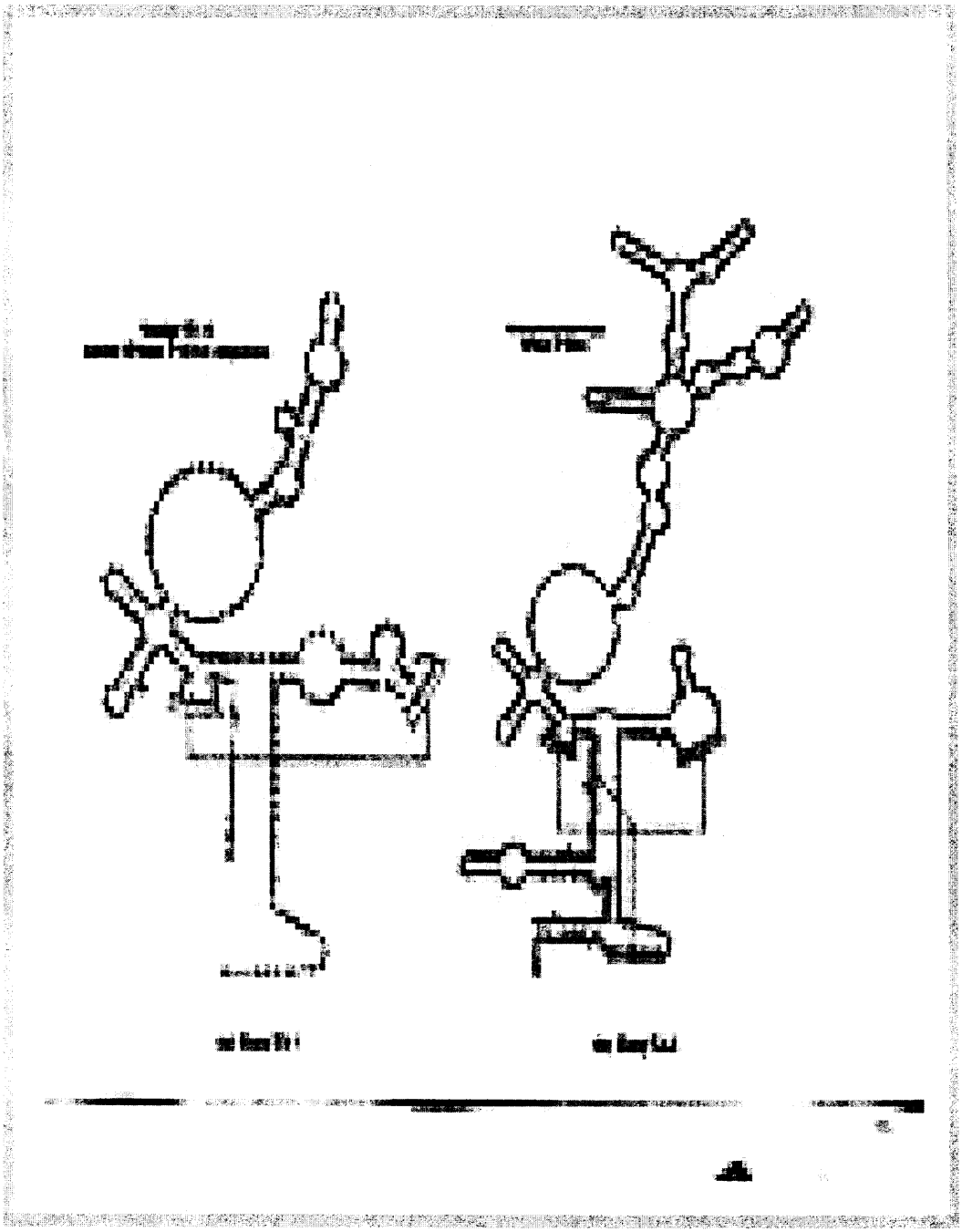


Figure 5.4 Two structures with OLCS similarity 0.8947 and LCS similarity 0.6195

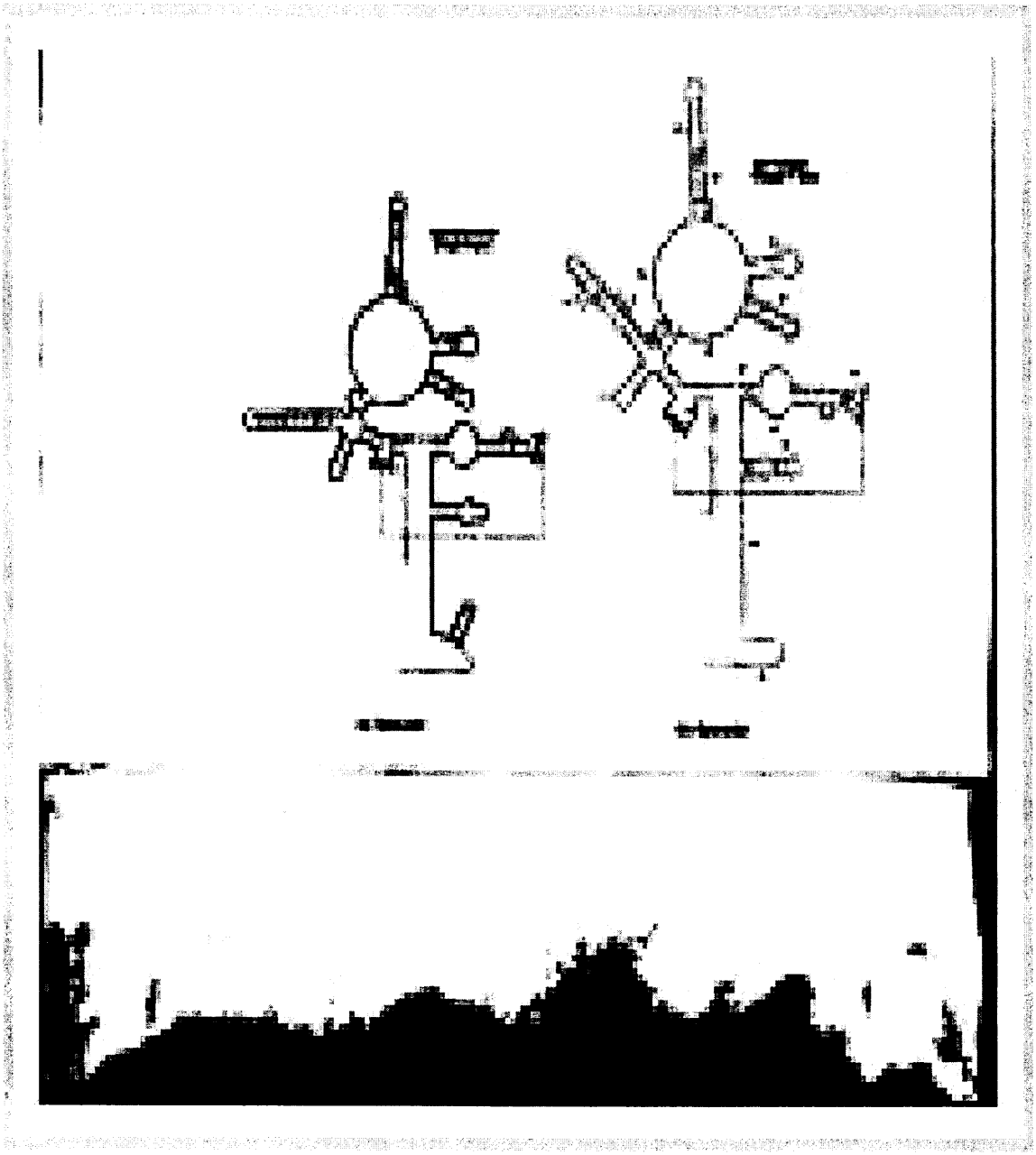


Figure 5.5 Two structures with OLCS similarity 0.4430 and LCS similarity 0.67

REFERENCES

1. Brown, JW. 1998. The Ribonuclease P Database. Nucleic Acids Res, 26, 351-52.

2. Gutell, RR. 1994. Collection of small subunit (16S- and 16S-like) ribosomal RNA structures. .Nucleic Acids Res, 22, 3502-3507.

3. Hofacker, IL, Fontana, W, Stadler, PF, Bonhöffer, S, Tacker, M, Schuster, P. 1994. Fast Folding and Comparison of RNA Secondary Structures, 125(2), 167-188.

4. Huynen, M, Gutell, RR, Konings, DA. 1997. Assessing the reliability of RNA folding using statistical mechanics. *J Mol Biol*, *267*, 1104-1112.

5. Jacobson, AB, Zuker, M. 1993. Structural analysis by energy dot plot of a large mRNA. *J Mol Biol*, *233*, 261-269.

6. Jacobson, AB, Arora, R, Priano, C, Lin, CH, Zuker, M, Mills, D. 1998. coliphage Q^{β}. *J Mol Biol*, *274*, 589-600.

7. Jaeger, JA, Turner, DH, Zuker, M. 1989. Improved predictions of secondary structures for RNA. *Proc Natl Acad Sci USA*, *86*, 7706-7710.

8. Jaeger, JA, Turner, DH, Zuker, M. 1990. Predicting optimal and suboptimal secondary structure for RNA. *Meth Enzymol*, *183*, 281-306.

9. Konings, DA, Gutell, RR. 1995. A comparison of thermodynamic foldings with comparatively derived structures of 16S and 16S-like rRNAs. *RNA*, *1*, 559-574.

10. LaGrandeur, TE, Darr, SC, Haas, ES, Pace, NR. 1993. Characterization of the RNase P of *S ulfolobus acidocaldarius*. *J Bacteriol*, *175*, 5043-8.

11. McCaskill, JS. 1990. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, *29*, 1105-19.

12. Murzina, NV, Vorozheykina, DP, Matvienko, NI. 1988. Nucleotide sequence of *T hermus thermophilus* HB8 gene coding 16S rRNA. *Nucleic Acids Res*, *16*, 8172.

13. Noller, HF. 1984. The structure of ribosomal RNA. *Ann Rev Biochem*, *53*, 119-62.

14. Palmenberg, AC, Sgro, J-Y. 1998. Topological Organization of Picornaviral Genomes: Statistical Prediction of RNA Structural Signals. *Seminars in Virology*, *8*, 231-241.

15. Reed, RE, Baer, MF, Guerrier-Takada, C, Donis-Keller, H, Altman, S. . 1982. Nucleotide sequence of the gene encoding the RNA subunit (M1 RNA) of Ribonuclease P from *E scherichia coli*. *Cell*, *30*, 627-36.

16. Walter, AE, Turner, DH, Kim, J, Lyttle, MH, Muller, P, Mathews, DH, Zuker, M. 1994. Coaxial stacking of helixes enhances binding of oligoribonucleotides and improves predictions of RNA folding. *Proc Natl Acad Sci USA*, *91*, 9218-9222.

17. Weiser, B, Noller, HF. 1995.XRNA: Auto interactive program of RNA. The Center for Molecular Biology of RNA, University of California, Santa Cruz. Internet: ftp://fangio.ucsc.edu/pub/XRNA.

18. Williams, AL, Tinoco, IJr. 1986. A dynamic programming algorithm for finding alternate RNA secondary structures. Nucleic Acids Res , 14, 219-354.

19. Zuker, M. 1989a. On finding all suboptimal foldings of an RNA molecule. Science, 244, 48-52.

20 Zuker, M. 1989b. The use of Dynamic Programming Algorithms for RNA Structure Prediction Mathematical methods for DNA sequences, Waterman MS, Ed. Boca Raton, Florida: CRC Press, Inc. Chapter 7, pages 159-184.

21. Zuker, M. 1994. Prediction of RNA Secondary Structure by Energy Minimization. Computer Analysis of Sequence Data, Part II, AM Griffin & HG Griffin, Eds, vol 25. Totowa, NJ: CRC Press, Inc. Chapter 23, pages 267-294.