Spring 5-31-2015

Data mining in computational proteomics and genomics

Yang Song
New Jersey Institute of Technology

ABSTRACT

DATA MINING IN COMPUTATIONAL PROTEOMICS AND GENOMICS

by
Yang Song

This dissertation addresses data mining in bioinformatics by investigating two important problems, namely peak detection and structure matching. Peak detection is useful for biological pattern discovery while structure matching finds many applications in clustering and classification.

The first part of this dissertation focuses on elastic peak detection in 2D liquid chromatographic mass spectrometry (LC-MS) data used in proteomics research. These data can be modeled as a time series, in which the X-axis represents time points and the Y-axis represents intensity values. A peak occurs in a set of 2D LC-MS data when the sum of the intensity values in a sliding time window exceeds a user-determined threshold. The elastic peak detection problem is to locate all peaks across multiple window sizes of interest in the dataset. A new method, called PeakID, is proposed in this dissertation, which solves the elastic peak detection problem in 2D LC-MS data without yielding any false negative. PeakID employs a novel data structure, called a Shifted Aggregation Tree or AggTree for short, to find the different peaks in the dataset. This method works by first constructing an AggTree in a bottom-up manner from the dataset, and then searching the AggTree for the peaks in a top-down manner. PeakID uses a state-space algorithm to find the topology and structure of an efficient AggTree. Experimental results demonstrate the superiority of the proposed method over other methods on both synthetic and real-world data.

The second part of this dissertation focuses on RNA pseudoknot structure matching and alignment. RNA pseudoknot structures play important roles in many genomic processes. Previous methods for comparative pseudoknot analysis mainly focus on simultaneous folding and alignment of RNA sequences. Little work has been done to align two known RNA secondary structures with pseudoknots taking into account both sequence and structure information of the two RNAs. A new method, called RKalign, is proposed in this dissertation for aligning two known RNA secondary structures with pseudoknots. RKalign adopts the partition function methodology to calculate the posterior log-odds scores of the alignments between bases or base pairs of the two RNAs with a dynamic programming algorithm. The posterior log-odds scores are then used to calculate the expected accuracy of an alignment between the RNAs. The goal is to find an optimal alignment with the maximum expected accuracy. RKalign employs a greedy algorithm to achieve this goal. The performance of RKalign is investigated and compared with existing tools for RNA structure alignment. An extension of the proposed method to multiple alignment of pseudoknot structures is also discussed. RKalign is implemented in Java and freely accessible on the Internet. As more and more pseudoknots are revealed, collected and stored in public databases, it is anticipated that a tool like RKalign will play a significant role in data comparison, annotation, analysis, and retrieval in these databases.

DATA MINING IN COMPUTATIONAL PROTEOMICS AND GENOMICS

by
Yang Song

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

Department of Computer Science

May 2015

APPROVAL PAGE

DATA MINING IN COMPUTATIONAL PROTEOMICS AND GENOMICS

Yang Song

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

Dr. James McHugh, Committee Member — Date
Professor, Department of Computer Science, NJIT

Dr. David Nassimi, Committee Member — Date
Associate Professor, Department of Computer Science, NJIT

Dr. Dimitrios Theodoratos, Committee Member — Date
Associate Professor, Department of Computer Science, NJIT

Dr. Yi Chen, Committee Member — Date
Associate Professor, School of Management, NJIT

BIOGRAPHICAL SKETCH

Author: Yang Song

Degree: Doctor of Philosophy

Date: May 2015

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2015
- Master of Science in Computer Science,
  State University of New York at New Paltz, New Paltz, NY, 2003
- Master of Business Administration,
  Baker University, Overland Park, KS, 2000
- Bachelor of Engeering in Computer Science,
  University of Science and Technology of China, China, 1998

Major: Computer Science

Publications:

L. Hua, Y. Song, D. Wen, J.T.L. Wang and C. Laing, "Pairwise Alignment of RNA Secondary Structures in the Presence of Coaxial Helical Stacking," *PLoS ONE*, Submitted.

Y. Song, L. Hua, B.A. Shapiro and J.T.L. Wang, "Effective Alignment of RNA Pseudoknot Structures Using Partition Function Posterior Log-odds Scores," *BMC Bioinformatics,* Vol. 16, No. 39, 2015.

M. Vasavada, K. Byron, Y. Song and J.T.L. Wang, "Genome-Wide Search for Pseudoknotted Non-Coding RNAs: A Comparative Study," *Pattern Recognition in Computational Molecular Biology: Techniques and Approaches,* (eds. Mourad Elloumi, Costas S. Iliopoulos, Jason T. L. Wang and Albert Y. Zomaya), Chapter 12, Hoboken, NJ: John Wiley & Sons, 2015.

X. Zhang, D. Shasha, Y. Song and J.T.L. Wang, "Fast Elastic Peak Detection for Mass Spectrometry Data Mining," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 24, No. 2, pp 114-121, 2012.

S.J. Griesmer, M. Cervantes-Cervantes, Y. Song and J.T.L. Wang, "In Silico Prediction of Noncoding RNAs Using Supervised Learning and Feature Ranking Methods," *International Journal of Bioinformatics Research and Applications*, Vol. 7, No. 4, pp 355-357, 2011.

ACKNOWLEDGMENT

I would like to thank my adviser, Dr. Jason T.L. Wang, his guidance and passion at research has led me through my Ph.D. study. I am also deeply grateful to Dr. David Nassimi, his consistent support that has helped me overcome many difficult situations.

Secondly, I offer my regards and blessings to my other committee members, Dr. James McHugh, Dr. Dimitrios Theodoratos and Dr. Yi Chen. Their supervision and advice have improved this work into a new level.

I am thankful to Dr. Hong Li, Dr. Tong Liu, Dr. Jun Hu and Dr. Cexiong Fu from the Center for Advanced Proteomics Research (CAPR) at the New Jersey Medical School (NJMS) of the Rutgers Biomedical and Health Sciences. Dr. Hong Li has provided me resources to do the research at proteomics, Dr. Jun Hu and Dr. Cexiong Fu have given me many useful ideas about the data analysis. I have special thanks to Dr. Tong Liu, without her warm help and excellent domain knowledge, this interdisciplinary research would never be accomplished.

Here I want to thank all the previous and current members at the Data and Knowledge Engineering Laboratory of NJIT. Ms. Lei Hua has been a great partner in research, without her significant contribution the project in genomics would not happen. And I am so lucky to have great friends here, in particular, Dr. Mugdha Khaladkar, Dr. Dongrong Wen and Dr. Tao Wu. The friendship and the time we worked and studied together always take a sweet spot in my memory.

Last but certainly not the least; I do not know how to express my appreciation to my family. The patience and trust from my wife, the sacrifice and unconditional love from my parents, they are my most solid support. I am blessed to have such a wonderful family.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

This dissertation is composed of two parts. The first part is concerned with elastic peak detection in 2D liquid chromatographic mass spectrometry (LC-MS) data. These data can be modeled as a time series, in which the X-axis represents time points and the Y-axis represents intensity values. A peak occurs in a set of 2D LC-MS data when the sum of the intensity values in a sliding time window exceeds a user-determined threshold. The elastic peak detection problem is to locate all peaks across multiple window sizes of interest in the dataset. We present a data structure, called a Shifted Aggregation Tree or AggTree for short, and use the data structure to find the different peaks. A new method, called PeakID, is proposed, which solves the elastic peak detection problem in 2D LC-MS data without yielding any false negative. This method works by first constructing an AggTree in a bottom-up manner from the given dataset, and then searching the AggTree for the peaks in a top-down manner. We describe a state-space algorithm for finding the topology and structure of an efficient AggTree to be used by PeakID. Our experimental results demonstrate the superiority of the proposed method over other methods on both synthetic and real-world data.

The rest of the first part is organized as follows. Chapter 2 surveys related work and contrasts our approach with existing techniques. The chapter then describes the PeakID method in detail, introducing the concept of an Aggregation Pyramid that acts as a host data structure into which a Shifted Aggregation Tree can be embedded, and explaining how to find an efficient Shifted Aggregation Tree given input data. Chapter 3 evaluates the performance of PeakID and presents experimental results on both synthetic and real-world data.

The second part of this dissertation is concerned with RNA pseudoknot structure alignment. RNA pseudoknot structures play important roles in many genomic processes. Previous methods for comparative pseudoknot analysis mainly focus on simultaneous folding and alignment of RNA sequences. Little work has been done to align two known RNA secondary structures with pseudoknots taking into account both sequence and structure information of the two RNAs. We present a novel method for aligning two known RNA secondary structures with pseudoknots. We adopt the partition function methodology to calculate the posterior log-odds scores of the alignments between bases or base pairs of the two RNAs with a dynamic programming algorithm. The posterior log-odds scores are then used to calculate the expected accuracy of an alignment between the RNAs. The goal is to find an optimal alignment with the maximum expected accuracy. We present a greedy algorithm to achieve this goal. The performance of our method is investigated and compared with existing tools for RNA structure alignment. An extension of the method to multiple alignment of pseudoknot structures is also discussed. The method described here has been implemented in a tool named RKalign, which is freely accessible on the Internet. As more and more pseudoknots are revealed, collected and stored in public databases, we anticipate a tool like RKalign will play a significant role in data comparison, annotation, analysis, and retrieval in these databases.

The rest of the second part is organized as follows. Chapter 4 describes related work and compares our approach with existing techniques. The chapter then presents the RKalign method in detail, introducing the partition function methodology used to calculate the posterior log-odds scores, and describing how to find a (sub)optimal alignment between two RNAs. Then, the analysis of the time and space complexity of RKalign is presented. Chapter 5 evaluates the performance of RKalign and presents experimental results on the RNA pseudoknot structures obtained from both the Protein Data Bank (PDB) and PseudoBase. Finally, Chapter 6 concludes the dissertation and points out some future research directions.

CHAPTER 2

ALGORITHMS FOR PEAK DETECTION

2.1 Background for Peak Detection

2.1.1 Motivation

Recently, mass spectrometry data mining has drawn much attention in the computational proteomics community [1, 2, 3, 4]. Typical mining processes include peak detection [5, 6, 7, 8], spectrum alignment [9], data correlation [10], biomarker discovery [11, 12], among others. In this thesis proposal, we present a new approach, called PeakID, for identifying peaks in liquid chromatography-mass spectrometry (LC-MS) data. LC-MS data has three dimensions, namely retention time, mass-to-charge ratio (m/z) and intensity [13, 2]. One important step in mass spectrometry data mining is to detect peaks in the three-dimensional (3D) LC-MS data [13, 1, 5, 2, 8, 14]. Due to the complex nature of the 3D data with different peak shapes, this is known to be a difficult problem [5, 2, 15, 8]. One approach is to convert the 3D data to lower dimensional data as explained below.

For each given m/z value, 3D LC-MS data can be expressed as a two-dimensional (2D) map (see Figure 2.1). In Figure 2.1, the X-axis represents time points and the Y-axis represents intensities. A peak in the 2D map, M_t, is a collection of intensity values occurring within a certain time window where the sum of the intensity values is greater than or equal to a user-specified threshold. Suppose a peak occurs in a time window $[t_l, t_r]$ in which the largest intensity value occurs at t_{top} in M_t. For the given t_{top} value, one can check the corresponding 2D map, M_{mz}, whose X-axis has m/z values and Y-axis has intensities. Find a small range $[mz_l, mz_r]$ that surrounds each mz_{pos} whose intensity is a sufficiently large positive value in M_{mz}. Then the intensities in the cube constructed based on $[t_l, t_r]$ and $[mz_l, mz_r]$ form a

3D peak. Thus, by detecting peaks among the 2D data points shown in Figure 2.1, one is able to derive peaks in the 3D LC-MS data. Finding the 2D peaks, or simply peaks when the context is clear, in Figure 2.1 in an efficient way is the subject of this proposal.

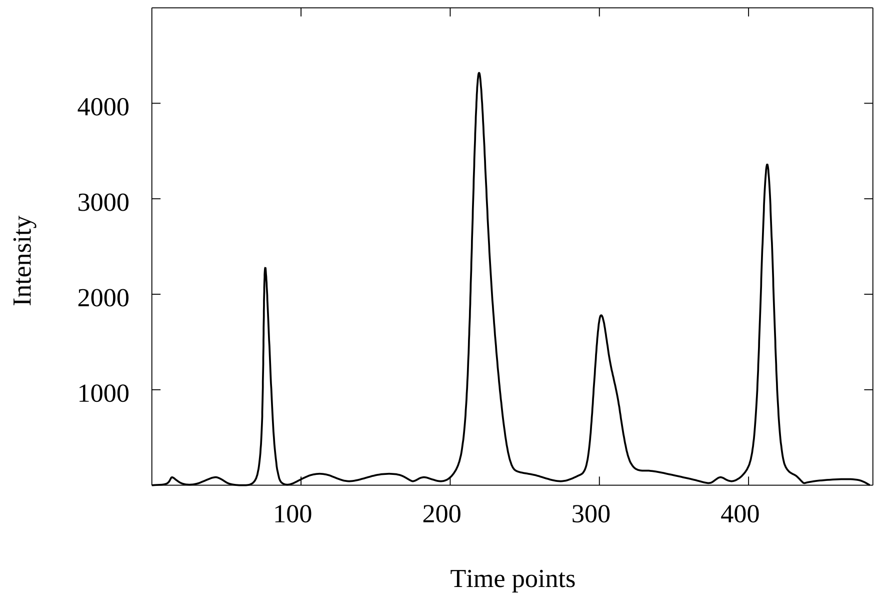


Figure 2.1 An example of 2D LC-MS data.

If the size of a sliding time window in which a 2D peak occurs is known *a priori*, then peak detection can easily be done in linear time by summing up the intensity values within each time window of the known size. However, in practice, the window size is unknown *a priori*. The size itself may be an interesting subject to be discovered. Also, in many cases, it is required to detect peaks across a variety of window sizes [13, 2].

2.1.2 Problem Formulation

elastic peak detection problem is to detect peaks across multiple window sizes. Formally, given a set of non-negative intensity values $x_1, x_2, \ldots, x_N$, a set $\mathcal{W}$ of window sizes $w_1, w_2, \ldots, w_m$, where $w_i < w_j$, $1 \leq i < j \leq m$, and a threshold associated with each window size, $f(w_j)$, $j = 1, 2, \ldots, m$, the elastic peak detection is the problem of finding all pairs (t, w) such that t is a time point, w is a window size in $\mathcal{W}$ and

$$\sum_{p=t}^{t+w-1} x_p \geq f(w)$$

A brute-force algorithm is to check each window size of interest one at a time. To detect peaks across m window sizes in a sequence of intensity values over N time points, the brute-force algorithm requires $O(mN)$ time.

In [16, 17], we showed that a simple data structure called a *Shifted Binary Tree*, abbreviated as a *BinaryTree*, could be the basis of a filter that can be used to detect all peaks in time independent of the number of window sizes when the probability of peaks is low. This tree is a hierarchical data structure, inspired by the Haar wavelet tree [18]. Each leaf node at level 1 of the Shifted Binary Tree corresponds to a time point in the input data; a node at level 2 aggregates two adjacent nodes at level 1. In general, a node v at level i aggregates two nodes v_1, v_2 at level $i - 1$; v is the parent of v_1, v_2 and v_1, v_2 are the children of v. Thus, v contains or corresponds to 2^{i-1} time points. There are $\log_2 N + 1$ levels in the Shifted Binary Tree where N is the number of time points in the input 2D LC-MS data. Except level 1 and the top level, each level i has two sublevels, namely base sublevel i and shifted sublevel i. Each node at shifted sublevel i is shifted by 2^{i-2} time points with respect to base sublevel i. Figure 2.2 shows an example of Shifted Binary Trees.

The overlap between the base sublevels and the shifted sublevels guarantees that every time window of size w, $0 < w \leq 2^{i-2} + 1$, is contained in either a node at base sublevel i or a node at shifted sublevel i, or both. In [16, 19, 17], we exploited this property, developing an algorithm that uses Shifted Binary Trees to search for peaks in time sequence data. The algorithm works well when there are few peaks, but performs poorly if there are many near-peaks [16].

From [19], we have a new data structure called a *Shifted Aggregation Tree*, abbreviated as an *AggTree*, which improves the performance of a Shifted Binary Tree,

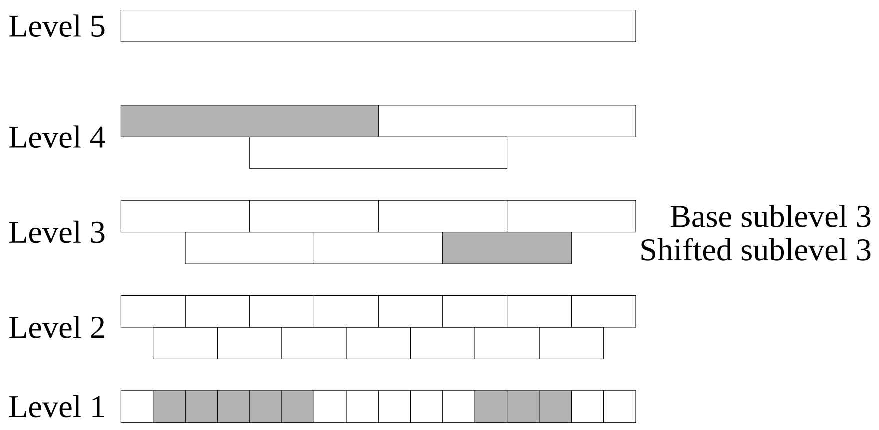


Figure 2.2 An example of Shifted Binary Trees. Each cell at level 1 is a leaf node; each leaf node corresponds to a time point in the input LC-MS data. Each cell at base sublevel i (shifted sublevel i, respectively) represents a node at base sublevel i (shifted sublevel i, respectively), and corresponds to or contains 2^{i-1} time points in the input data. In the figure, the first highlighted sequence at level 1 is contained in the highlighted node at base sublevel 4; the second highlighted sequence at level 1 is contained in the highlighted node at shifted sublevel 3.

and sketched the use of AggTrees in a general setting of time series mining. Here, we extend the work in [19] by presenting:

(1) The algorithmic details and theoretical foundation of AggTrees.

(2) The PeakID method that adapts AggTrees to elastic peak detection in 2D LC-MS data.

(3) Experimental results showing the superiority of PeakID over other methods.

2.1.3 Related Work

There are two groups of work that are closely related to ours. The first group is concerned with 3D peak detection in mass spectrometry data. Most 3D peak detection algorithms are based on either statistical distributions or a variety of smoothing functions [15, 20]. To reduce the number of false positives (i.e., those that are non-peaks but are detected as peaks), these algorithms often assume a minimum peak width. The algorithms focus on dealing with 2D data where the X-axis has m/z values and the Y-axis has intensity values [21, 5, 22]. An alternative approach, as described in Section 2.1.1, is to examine 2D data where the X-axis has time points and the Y-axis has intensity values. Stolt *et al.* [8], for example, developed a second-order peak detection algorithm capable of finding peaks of different widths in such 2D data. The authors set the minimum peak width to 3. One disadvantage of Stolt *et al.*'s algorithm is that it may produce false negatives (i.e., those that are real peaks but are predicted as non-peaks). To reduce the number of false negatives, the brute-force algorithm checking different window sizes could be used. In contrast to Stolt *et al.*'s work, we develop a new data structure for identifying all peaks quickly without using the brute-force approach.

The second group of related work is concerned with burst modeling and detection in time series. Wang *et al.* [23] used a one-parameter model, b-model, to model the bursty behavior in self-similar time series and to synthesize realistic trace data. This type of time series occurs in a large number of real world applications, such as Ethernet, file systems, web, video and disk traffic. Kleinberg [24] studied the bursty and hierarchical structure in temporal text streams, with a focus on finding how high frequency words change over time. Vlachos *et al.* [25] mined the bursty behavior in the query logs of the MSN search engine. They used moving averages to detect time regions having high numbers of queries. Only two window sizes were considered, short term and long term. The detected bursts were further compacted and stored in a database to support burst-based queries. Other methods for finding surprising and periodic patterns in time series have also been developed [26, 27, 28]. The 2D LC-MS data we deal with here are time series in nature. However, in contrast to the above time series mining methods, our work mainly focuses on detecting bursts (peaks) across multiple window sizes.

2.2 The PeakID Method

In this section, we describe the PeakID method in detail, introducing the concept of an Aggregation Pyramid that acts as a host data structure into which a Shifted Aggregation Tree can be embedded; the concept of Detailed Search Region and the advantages of Shifted Aggregation Tree over Binary Aggregation Tree. We also explain the details of algorithms for the Shifted Aggregation Tree construction and the peak detection. Lastly, we introduce a heuristic state-space algorithm to find an efficient Shifted Aggregation Tree given input data.

2.2.1 The Aggregation Pyramid

An *Aggregation Pyramid* is an N-level isosceles triangular-shaped data structure built over the input 2D LC-MS data with N time points, satisfying the following properties:

- Level 1 has N cells where each cell stores the intensity value associated with each time point in the input 2D LC-MS data.

- Level 2 has $N-1$ cells where the first cell stores the sum of the first two intensity values (i.e., the intensity value at time point 1 and the intensity value at time point 2) in the 2D LC-MS data; the second cell stores the sum of the second two intensity values (i.e., the intensity value at time point 2 and the intensity value at time point 3), and so on.

- Level h has $N - h + 1$ cells where the ith cell, c, stores the sum of the h consecutive intensity values starting at time point i and ending at time point $i + h - 1$ in the 2D LC-MS data. The time window starting at time point i and ending at time point $i + h - 1$ is called the *shadow window*, or simply the *shadow*, of cell c. When the context is clear, we also refer to the set of the h consecutive cells at level 1 starting with the ith cell as the shadow window, or the shadow, of cell c.

- The top level has one cell only, storing the sum of all intensity values in the input 2D LC-MS data.

Notice that each time window of size w is the shadow of some cell at level w. Conversely, each cell at level w has a shadow window of size w; the cell stores the sum of the intensity values within its shadow window. Figure 2.3 shows an example of an 8-level Aggregation Pyramid. In Figure 2.3(b), for example, the highlighted cell, c, at level 4 stores a value of 106, which is the sum of the intensity values 27, 11, 18 and 50 at level 1 that are within the shadow window of the cell c.

By construction, an Aggregation Pyramid has the following properties:

- All the shadows of the cells along the 45° diagonal have the same starting time point. All the shadows of the cells along the 135° diagonal have the same ending time point.

- A cell at level h with the shadow ending at time point t is denoted as $cell(h, t)$, which stores the sum of the intensity values in $cell(1, t - h + 1)$ to $cell(1, t)$. When the context is clear, we also use $cell(h, t)$ to represent the sum value stored in this cell.

- The shadow of any cell c in the subpyramid rooted at cell r is a subset of the shadow of cell r. We say c is *shaded* by r. By monotonicity, the value in cell c is guaranteed to be less than or equal to the value in cell r.

- The *overlap* of two cells c_1 and c_2 in that order at the same level is the cell c at the intersection of the 135° diagonal touching cell c_1 and the 45° diagonal touching cell c_2 (see Figure 2.4). The shadow of the cell c is the intersection of the shadow of c_1 and the shadow of c_2. For example, in Figure 2.4, the shadow of cell c_1 contains time points 4, 5, . . . , 10, the shadow of cell c_2 contains time points 8, 9, . . . , 14, and the shadow of the overlap c contains time points 8, 9 and 10.

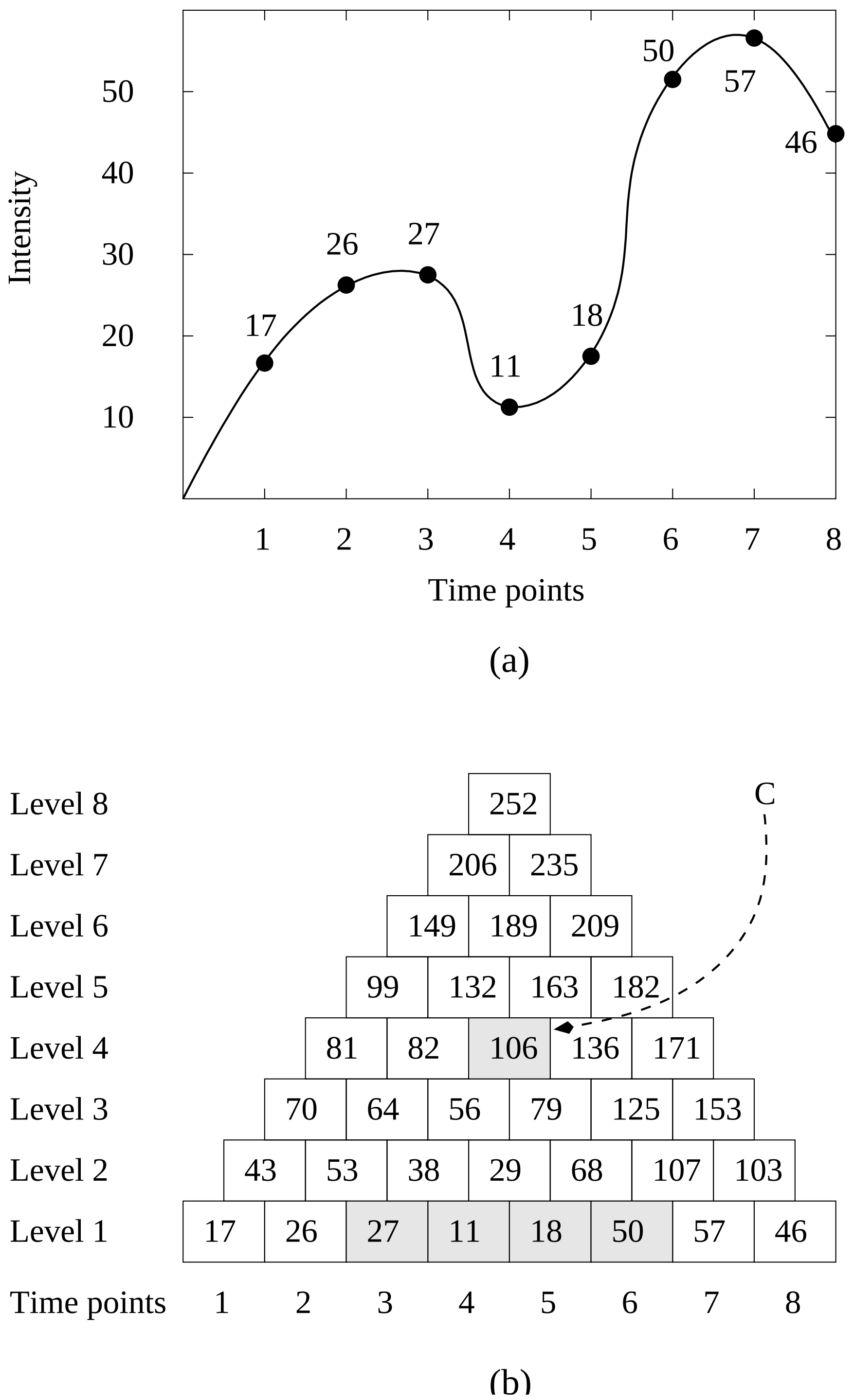


Figure 2.3 An example of an 8-level Aggregation Pyramid. (a) An example of 2D LC-MS data. (b) The Aggregation Pyramid built over the 2D LC-MS data in (a).

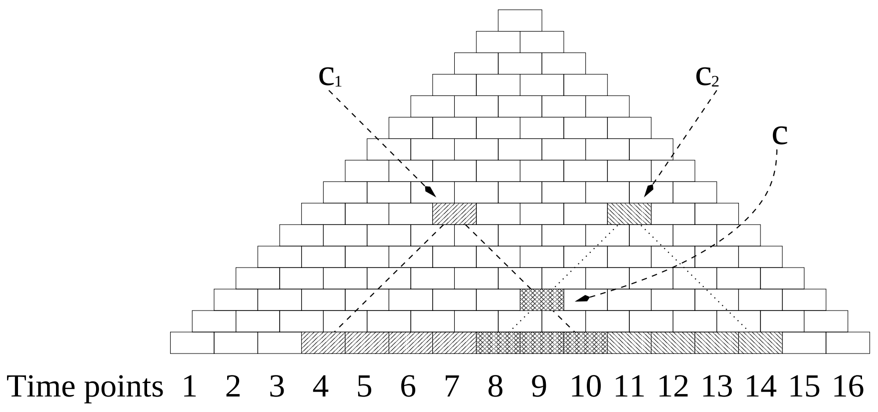


Figure 2.4 Illustration of the overlap of two cells in an example Aggregation Pyramid.

The values stored in the cells of the Aggregation Pyramid can be calculated in a bottom-up manner using Equation (2.1) below: for $1 < h, t \leq N$,

$$cell(h,t) \quad = \quad cell(h-1,t-1) + cell(1,t) \tag{2.1}$$

If $cell(h,t)$ exceeds the threshold $f(h)$ for the time window size h, then there exists a peak starting at time point $t - h + 1$ and ending at time point t.

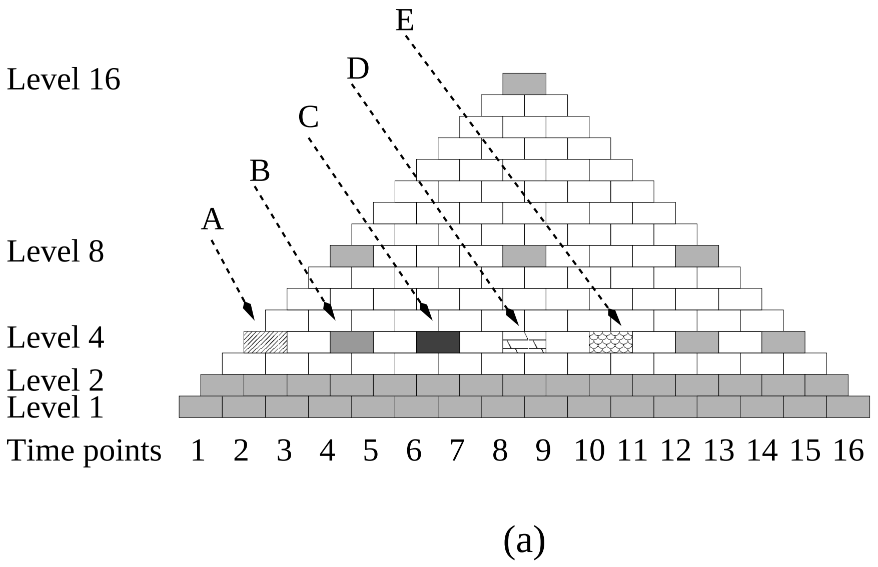
2.2.2 Embedding a BinaryTree into an Aggregation Pyramid

Recall that in a Shifted Binary Tree, each node at level 1 corresponds to a time point in the input LC-MS data, and each node at level i corresponds to or contains 2^{i-1} time points in the input data. Observe that each node in a Shifted Binary Tree with N time points corresponds to a cell in an Aggregation Pyramid with the same number of time points. Figure 2.5 shows the correspondence between the nodes in a Shifted Binary Tree with 16 time points and the cells in an Aggregation Pyramid with 16 time points. Each cell in the Aggregation Pyramid that corresponds to a node in the Shifted Binary Tree is highlighted in Figure 2.5(a). Specifically, the cells labeled A, B, C, D and E, respectively in the Aggregation Pyramid in Figure 2.5(a) correspond to the nodes labeled A, B, C, D and E, respectively in the Shifted Binary Tree in

Figure 2.5(b). Notice that level i in the Shifted Binary Tree corresponds to level 2^{i-1} in the Aggregation Pyramid. This correspondence shows how to embed the Shifted Binary Tree in Figure 2.5(b) into the Aggregation Pyramid in Figure 2.5(a).

An important property of the Shifted Binary Tree is that any given time window of size w, $w \leq 2^{i-2} + 1$, is contained in at least one of the nodes at level i of the BinaryTree. By induction, any time window of size w, $w \leq 2^{i-3} + 1$, is contained in at least one of the nodes at level $i - 1$ of the BinaryTree. After the BinaryTree is constructed, we search the BinaryTree for peaks in a top-down manner. If the value stored in a node $v = cell(2^{i-1}, t)$ at level i of the BinaryTree exceeds the threshold $f(2^{i-3} + 2)$ associated with the time window size $2^{i-3} + 2$, then an alarm is raised, indicating the possible occurrence of a peak. A detailed search has to be performed to check the cells of the Aggregation Pyramid whose shadow sizes are in the range $[2^{i-3} + 2,\, 2^{i-2} + 1]$.

Note that we need to search and check only the cells in the Aggregation Pyramid whose shadows end after time point $t - 2^{i-2}$, because the cells whose shadows end at or before time point $t - 2^{i-2}$ are shaded by one of v's preceding nodes at level i of the BinaryTree. We refer to the region in which the search is performed as the *detailed search region* of v, denoted $DSR(v)$. The detailed search region consists of cells from level $2^{i-3} + 2$ to level $2^{i-2} + 1$ in the Aggregation Pyramid, where the shadows of these cells end at time points in $[t - 2^{i-2} + 1, t]$. The purpose of checking the cells in $DSR(v)$ is to find peaks occurring in time windows whose sizes are in the range $[2^{i-3} + 2,\, 2^{i-2} + 1]$ and that end at time points in $[t - 2^{i-2} + 1, t]$. Notice that, if v is the leftmost node at level i, since no node precedes v, $DSR(v)$ contains cells whose shadows end at time points in $[1, 2^{i-1}]$. It was proved [16] that searching the cells in $DSR(v)$ guarantees that all peaks are detected. Figure 2.6 illustrates the detailed search region $DSR(v)$ for a node v in the BinaryTree in Figure 2.5(b).



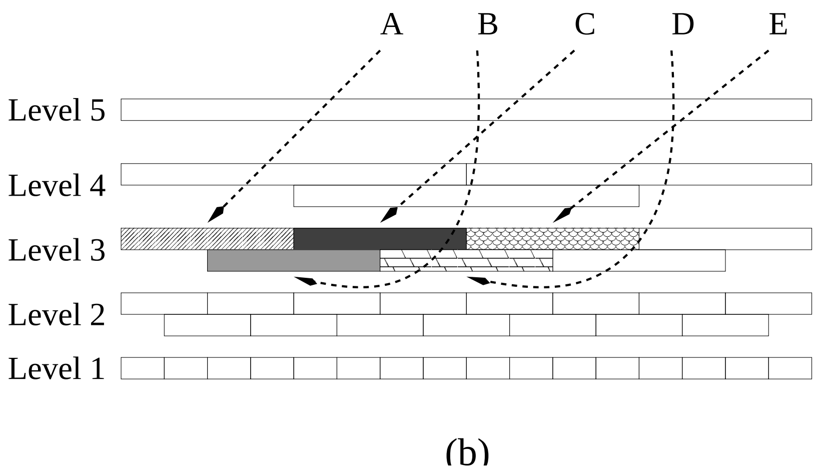


Figure 2.5 Illustration of the correspondence between the Aggregation Pyramid in (a) and the Shifted Binary Tree in (b). The highlighted cells at level 1, 2, 4, 8, 16, respectively in the Aggregation Pyramid in (a) correspond to the nodes at level 1, 2, 3, 4, 5, respectively in the BinaryTree in (b).

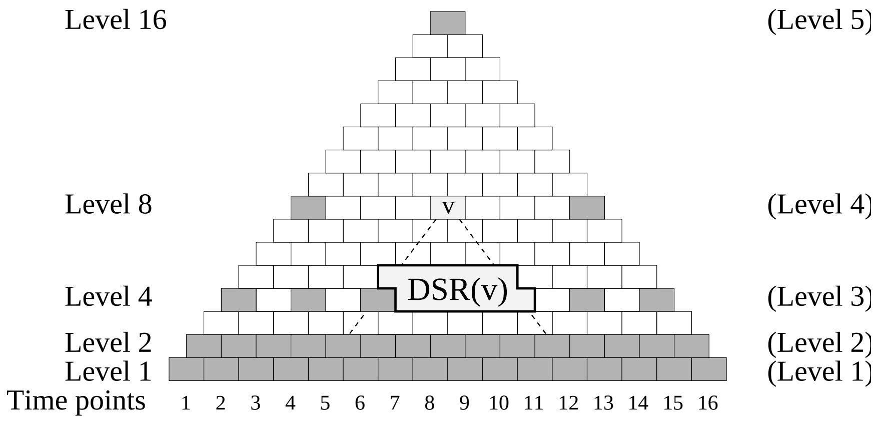


Figure 2.6 Illustration of the detailed search region $DSR(v)$ in the Aggregation Pyramid in Figure 2.5(a) for a node v at level 4 in the BinaryTree in Figure 2.5(b). The level numbers on the left represent the level numbers of the Aggregation Pyramid in Figure 2.5(a), and the level numbers in the parentheses on the right represent the level numbers of the BinaryTree in Figure 2.5(b). The cells of the Aggregation Pyramid that correspond to the nodes of the BinaryTree are highlighted.

2.2.3 Generalizing a BinaryTree to an AggTree

A detailed search in $DSR(v)$ may turn out to be fruitless (i.e., no peak is found in $DSR(v)$). It has been observed that [16, 19, 17]

(1) When peaks are rare but not very rare, the number of fruitless detailed searches grows, suggesting that we may want more levels than a Shifted Binary Tree provides.

(2) Conversely, when peaks are exceedingly rare we may need fewer levels than a Shifted Binary Tree provides.

In other words, we want a data structure that adapts to the input data. For this reason, we generalize Shifted Binary Trees to Shifted Aggregation Trees. Like a BinaryTree, an AggTree is a hierarchical data structure defined on a subset of the cells of an Aggregation Pyramid. It has several levels, each of which contains several nodes. The nodes at level 1 are in one-to-one correspondence with the time points in the input 2D LC-MS data. The value stored in a node at level i is obtained by aggregating the values stored in some nodes below level i. The shadows of two neighboring nodes at the same level overlap.

A Shifted Aggregation Tree differs from a Shifted Binary Tree in two ways:

(1) The parent-child structure

This defines the topological relationship between a node and its children, i.e., how many children the node has and their placements.

(2) The shifting pattern

This defines how many time points apart there are between two neighboring nodes v_1 and v_2 at the same level i. Formally, letting the shadow of v_1 end at time point t_1 and the shadow of v_2 end at time point t_2, where $t_1 < t_2$, we call $(t_2 - t_1)$ the *shift* between v_1 and v_2, or the shift of level i.

In a BinaryTree, the parent-child structure for each node is always the same: one node at level i aggregates two nodes at level $i-1$. The shifting pattern is also fixed: the shadows of two neighboring nodes in the same level always half-overlap. In an AggTree, a node could have 3 children and be 2 time points away from its preceding neighbor, or could have 64 children and be 128 time points away from its preceding neighbor. We define the shadow size of level i, denoted a_i, to be the size of the shadow of a node at level i. Define the overlapping shadow size of level i, denoted o_i, to be the size of the intersection of the shadows of two neighboring nodes at level i. Define the degree of level i, denoted d_i, to be the degree of a node at level i, i.e., the number of children the node has. Let s_i denote the shift of level i. Table 2.1 gives a side-by-side comparison between AggTrees and BinaryTrees. Clearly, BinaryTrees are a special case of AggTrees.

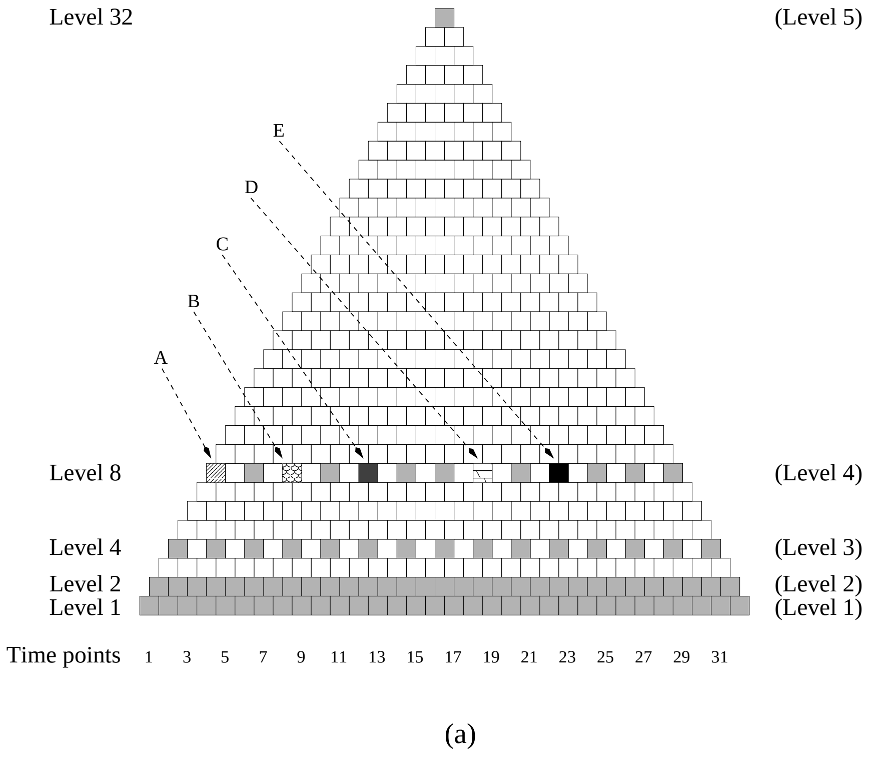
Figure 2.7 shows an example of a Shifted Aggregation Tree with 32 time points; the figure also shows how the AggTree is embedded into an Aggregation Pyramid

Table 2.1 Comparison between BinaryTrees and AggTrees

BinaryTree	AggTree
$d_i = 2$	$d_i \geq 2$
$s_i = 2 \times s_{i-1}$	$s_i = k \times s_{i-1}$, $k \geq 1$
$o_i = a_{i-1}$	$o_i \geq a_{i-1}$

with 32 time points. In the AggTree, the shift of level 2 is 1; the shift of level 3 is the same as the shift of level 4, which equals 2. Each node at level 2, 3 and 4, respectively has 2 children, whereas the node at level 5 has 4 children. The overlapping shadow size of level 2 is 1, which equals the shadow size of level 1. The overlapping shadow size of level 3 is 2, which equals the shadow size of level 2. The overlapping shadow size of level 4 is 6, which is larger than the shadow size of level 3. This example shows the difference between an AggTree and a BinaryTree (cf. Table 2.1 and Figure 2.5).

Recall that in the elastic peak detection problem whose goal is to find peaks across multiple window sizes, we are given the 2D LC-MS data, LM[N], with N time points, a set of non-negative intensity values $x_1, x_2, \ldots, x_N$, a set $\mathcal{W}$ of window sizes $w_1, w_2, \ldots, w_m$ where $w_i < w_j$, $1 \leq i < j \leq m$, and a threshold associated with each window size, $f(w_j)$, $j = 1, 2, \ldots, m$. Our approach to solving this problem, called PeakID, is to first construct an AggTree on the given 2D LC-MS data and then search the AggTree for peaks, where each peak is represented by a pair (t, w) such that t is a time point, w is a window size in $\mathcal{W}$ and $\sum_{p=t}^{t+w-1} x_p \geq f(w)$. The algorithm for constructing the AggTree, called BuildTree, takes as input the 2D LC-MS data LM[N] where LM[j], $1 \leq j \leq N$, contains the intensity value x_j associated with the jth time point. In addition, the input data of the algorithm include the shift s_i, shadow size a_i and degree d_i of each level i in the AggTree to be constructed. We will use a state-space algorithm to find appropriate values for s_i, a_i and d_i, as explained later in this section. The algorithm builds the AggTree in a bottom-up manner. In the first



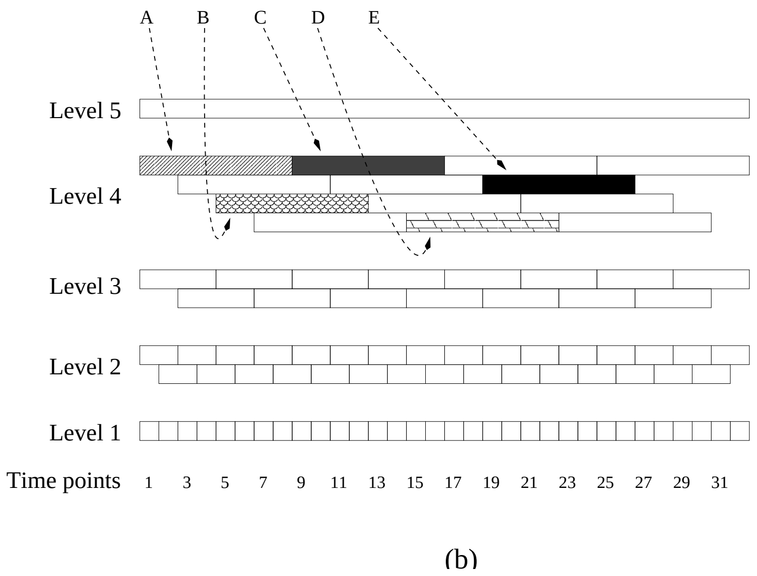


Figure 2.7 Illustration of the correspondence between the Aggregation Pyramid in (a) and the Shifted Aggregation Tree in (b). In (a), the level numbers on the left represent the level numbers of the Aggregation Pyramid, and the level numbers in the parentheses on the right represent the level numbers of the AggTree in (b). The cells of the Aggregation Pyramid that correspond to the nodes of the AggTree are highlighted in (a).

level, each node corresponds to a time point in the input LC-MS data, and stores the intensity value associated with that time point. The top level has one node only, whose shadow contains all the N time points.

In practice, we do not need to build the entire AggTree. It suffices for the BuildTree algorithm to construct nodes from level 1 to level I where a_{I-1} - s_{I-1} + 1 $< w_m \leq a_I - s_I + 1$. Here, w_m is the largest window size of interest, a_I is the shadow size of level I and s_I is the shift of level I in the AggTree. Let AggTree$[i][j]$, $1 < i \leq I$ and $1 \leq j \leq \lfloor \frac{N-a_i}{s_i} \rfloor + 1$, represent the value stored in the jth node at level i. We have

$$\text{AggTree}[i][j] = \sum_{p=1}^{d_i} \text{AggTree}[i-1][\lfloor \frac{(j-1) \times s_i + (p-1) \times a_{i-1}}{s_{i-1}} \rfloor + 1] \tag{2.2}$$

Figure 2.8 summarizes the algorithm for building the AggTree. Notice that for efficiency reasons, we do not actually build the Aggregation Pyramid into which the AggTree is embedded. As explained below, the partially constructed AggTree is sufficient for detecting all peaks without yielding false negatives.

2.2.4 Detecting Peaks Using an AggTree

We search the AggTree constructed in the previous subsection to detect all peaks in a top-down manner. To detect peaks in the time windows of size w_m, we examine the nodes at level I where $a_{I-1} - s_{I-1} + 1 < w_m \leq a_I - s_I + 1$. In general, to detect peaks in the time windows of size w_j, $1 \leq j \leq m$, we examine the nodes at level i where $a_{i-1} - s_{i-1} + 1 < w_j \leq a_i - s_i + 1$. We check each node $v = cell(a_i, t)$ at level i to see if the value stored in v exceeds the threshold $f(w_j)$ associated with the window size w_j. If so, an alarm is raised, indicating the possible occurrence of a peak, and a check in the detailed search region $DSR(v)$ is performed. The shadow of the node v ends at time point t. $DSR(v)$ comprises cells in the Aggregation Pyramid into which the AggTree is embedded, where the sizes of the shadows of the cells are in the range

```
Procedure: BuildTree(LM[$N$], $s_i$, $a_i$, $d_i$)
Input: The LC-MS data LM[$N$], shift $s_i$, shadow size $a_i$ and degree $d_i$.
Output: The shifted aggregation tree AggTree[$I$][$N$].
        /* Initialize the first level of the AggTree. */
1.      for $j = 1$ to $N$ do
2.          AggTree[1][$j$] $\leftarrow$ LM[$j$];
        /* Construct the AggTree in a bottom-up manner. */
3.      for $i = 2$ to $I$ do
4.          for $j = 1$ to $\lfloor \frac{N-a_i}{s_i} \rfloor + 1$ do
5.              calculate AggTree[$i$][$j$] using Equation (2.2);
6.      return AggTree[$I$][$N$];
```

Figure 2.8 Algorithm for constructing the AggTree from the LC-MS data LM[N].

$[a_{i-1} - s_{i-1} + 2,\ a_i - s_i + 1]$, and the shadows end at time points in $[t - s_i + 1, t]$. Notice that, if v is the leftmost node at level i, since no node precedes v, $DSR(v)$ contains cells whose shadows end at time points in $[1, a_i]$. Figure 2.9 illustrates the $DSR(v)$ for a node v at level 3 in an AggTree with 16 time points. In the figure, the sizes of the shadows of the cells in the $DSR(v)$ are in the range [4, 9], and the shadows end at time points in [13, 16].

In searching $DSR(v)$ where $v = cell(a_i, t)$, if $cell(w_j, t') \geq f(w_j)$, $t - s_i + 1 \leq t' \leq t$, then a peak is detected, which is represented and output as a pair $(t' - w_j + 1, w_j)$. That is, the peak occurs in the time window $[t' - w_j + 1,\ t']$ where $\sum_{p=t'-w_j+1}^{t'} x_p \geq f(w_j)$. Since we do not actually build the Aggregation Pyramid containing $DSR(v)$, the values of the cells in $DSR(v)$ are computed "on-the-fly", as explained below. Observing that the shadows of two neighboring cells overlap, to avoid duplicate computation, we start from one seed node in the AggTree, and then by adding or subtracting the difference between two neighboring cells, we can get the

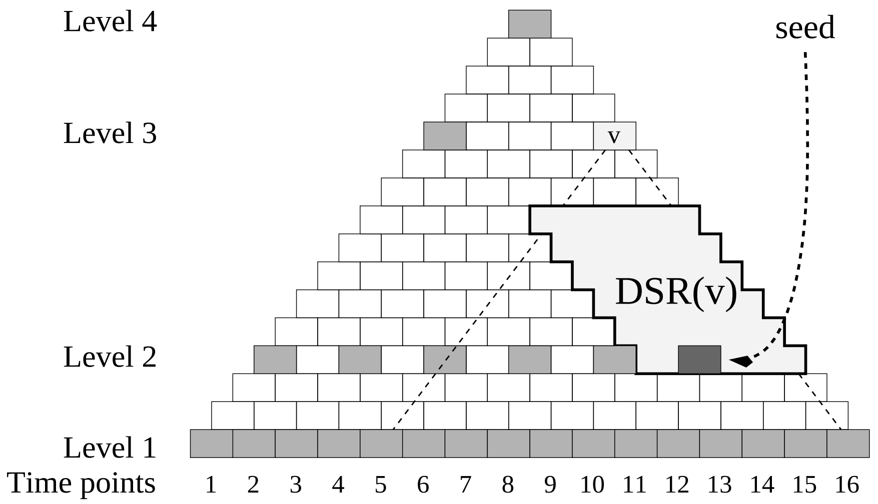


Figure 2.9 Illustration of the detailed search region $DSR(v)$ for a node v at level 3 in an AggTree with 16 time points.

values of the cells in $DSR(v)$. Specifically, assuming the seed node is $cell(h, t_s)$, we have

$$cell(h, t_s + 1) = cell(h, t_s) - cell(1, t_s - h + 2) + cell(1, t_s + 1) \tag{2.3}$$

In general, for all $1 < t_s + k \leq N$,

$$cell(h, t_s + k) = cell(h, t_s + k - 1) - cell(1, t_s + k - h + 1) + cell(1, t_s + k) \tag{2.4}$$

Furthermore,

$$cell(h + 1, t_s) = cell(h, t_s) + cell(1, t_s - h) \tag{2.5}$$

In general, for all $1 < h + k \leq I$,

$$cell(h + k, t_s) = cell(h + k - 1, t_s) + cell(1, t_s - h - k + 1) \tag{2.6}$$

Due to the properties of Shifted Aggregation Trees, it's guaranteed to find a seed node in or near a detailed search region. Here is how. Notice $s_{i-1} \leq s_i$ (cf. Table 2.1). The shadows of the cells in $DSR(v)$ end at time points in $[t - s_i + 1, t]$, i.e., the time span of $DSR(v)$ is s_i. Thus, there exists a node at level $i-1$ whose shadow ends at some time point in $[t-s_i+1,\ t]$; call it the seed node (Figure 2.9). Notice $a_{i-1} \leq a_i - s_i + 1$. If $s_{i-1} > 1$, then $a_{i-1} - s_{i-1} + 2 \leq a_{i-1} \leq a_i - s_i + 1$. The sizes of the shadows of the cells in $DSR(v)$ are in the range $[a_{i-1} - s_{i-1} + 2,$ $a_i - s_i + 1]$. Therefore, the seed node lies in $DSR(v)$. If $s_{i-1} = 1$, then the seed node is immediately below $DSR(v)$.

Figure 2.10 summarizes the algorithm, called SearchTree, for finding peaks in the AggTree$[I][N]$ constructed by the BuildTree algorithm in Figure 2.8. The SearchTree algorithm takes as input the AggTree$[I][N]$, a set $\mathcal{W}$ of window sizes $w_1, w_2, \ldots, w_m$ where $w_i < w_j$, $1 \leq i < j \leq m$, and a threshold associated with each window size, $f(w_j)$, $j = 1, 2, \ldots, m$. The algorithm outputs all pairs (t, w) such that t is a time point in AggTree$[I][N]$, w is a window size in $\mathcal{W}$ and $\sum_{p=t}^{t+w-1} x_p \geq f(w)$. Each pair (t, w) represents a peak occurring in the time window $[t, t+w-1]$. Notice that in step 6 of SearchTree, if v is the leftmost node at level i, the shadows of the cells in $DSR(v)$ end at time points in $[1,\ t]$, which equals $[1,\ a_i]$. Under this circumstance, we need to check each $cell(w_j, t')$, where $1 \leq t' \leq t = a_i$ in $DSR(v)$.

```
Procedure: SearchTree(AggTree[$I$][$N$], $w_j$, $f(w_j)$)
Input: The AggTree[$I$][$N$], window size $w_j$ and its associated threshold $f(w_j)$.
Output: The set $\mathcal{P}$ of pairs $(t, w)$ representing peaks.
1.   $i \leftarrow I$, $j \leftarrow m$, $\mathcal{P} \leftarrow \emptyset$;
     /* Search AggTree[$I$][$N$] in a top-down manner for peaks. */
2.   while $j \geq 1$ do
3.     if $(a_{i-1} - s_{i-1} + 1 < w_j \leq a_i - s_i + 1)$ then
4.       for each node $v = cell(a_i, t)$ at level $i$ of AggTree[$I$][$N$] do
5.         if ($v$'s value $\geq f(w_j)$) then
             /* Search $DSR(v)$. */
6.           for each $cell(w_j, t')$, $t - s_i + 1 \leq t' \leq t$, in $DSR(v)$ do
7.             if $cell(w_j, t') \geq f(w_j)$ then
8.               $\mathcal{P} \leftarrow \mathcal{P} \cup \{(t' - w_j + 1, w_j)\}$;
9.       $j \leftarrow j - 1$;
10.    else $i \leftarrow i - 1$;
11.  return $\mathcal{P}$;
```

Figure 2.10 Algorithm for searching AggTree[I][N] for peaks.

Below we show that the proposed PeakID method, composed of the two algorithms BuildTree and SearchTree, finds all peaks without yielding false negatives. Let T be the Shifted Aggregation Tree constructed by the BuildTree algorithm and let AP be the Aggregation Pyramid into which T is embedded.

Lemma 2.2.4.1. Let W be a time window of size w where $a_{i-1} - s_{i-1} + 1 < w \leq a_i - s_i + 1$. Let $cell(w, t)$ be the cell at level w of AP whose shadow is W that ends at time point t. There exists a node v at level i of T such that $cell(w, t)$ is shaded by v. Furthermore, $cell(w, t)$ lies in $DSR(v)$.

Proof. We use mathematical induction to show that the lemma holds for any positive integer k, where $t \leq a_i + (k - 1) \times s_i \leq N$.

Base step. When $k = 1$, i.e., $t \leq a_i$, since the shadow of the first node, i.e., the leftmost node, v at level i of T ends at time point a_i, $cell(w, t)$ is shaded by v. $DSR(v)$ comprises cells in AP where the sizes of the shadows of the cells are in the range $[a_{i-1} - s_{i-1} + 2,\ a_i - s_i + 1]$. Since $a_{i-1} - s_{i-1} + 1 < w \leq a_i - s_i + 1$, $cell(w, t)$ lies in $DSR(v)$.

Hypothesis step. Assume the lemma holds when $k = p$. That is, when $t \leq a_i + (p-1) \times s_i$ and $a_{i-1} - s_{i-1} + 1 < w \leq a_i - s_i + 1$, there exists a node v at level i of T such that $cell(w, t)$ is shaded by v and $cell(w, t)$ lies in $DSR(v)$.

Induction step. We want to show that the lemma holds when $k = p + 1$. Based on the properties of T, the shadows of the cells in the detailed search regions of the first p nodes at level i of T end at time points in $[1, a_i + (p-1) \times s_i]$. The shadows of the cells in the detailed search region of the $(p + 1)$th node at level i of T end at time points in $[a_i + (p-1) \times s_i + 1, a_i + p \times s_i]$. Now, consider the time window W of size w where $a_{i-1} - s_{i-1} + 1 < w \leq a_i - s_i + 1$ and the $cell(w, t)$ whose shadow is W. If $1 \leq t \leq a_i + (p-1) \times s_i$, by the induction hypothesis, there exists a node v at level i of T such that $cell(w, t)$ is shaded by v and $cell(w, t)$ lies in $DSR(v)$.

If $a_i + (p-1) \times s_i + 1 \leq t \leq a_i + p \times s_i$, which means W ends at some time point in $[a_i + (p-1) \times s_i + 1,\ a_i + p \times s_i]$, then W must start at some time point in $[a_i + (p-1) \times s_i - w + 2, a_i + p \times s_i - w + 1]$. Notice that, the shadow of the $(p+1)$th node at level i of T starts at time point $p \times s_i + 1$. Since $w \leq a_i - s_i + 1$, we have $a_i - s_i + 1 - w \geq 0$. Thus, $(a_i + (p-1) \times s_i - w + 2)$ - $(p \times s_i + 1) = a_i - s_i + 1 - w \geq 0$, which means $(a_i + (p-1) \times s_i - w + 2) \geq (p \times s_i + 1)$. Therefore, all time points in W are in $[p \times s_i + 1, a_i + p \times s_i]$, which is the shadow of the $(p+1)$th node at level i of T. Hence, $cell(w, t)$ is shaded by the $(p + 1)$th node at level i of T.

Furthermore, because $a_{i-1} - s_{i-1} + 1 < w \leq a_i - s_i + 1$, and $DSR(v)$ comprises cells in AP where the sizes of the shadows of the cells are in the range $[a_{i-1} - s_{i-1} + 2,$ $a_i - s_i + 1]$, we know that $cell(w, t)$ lies in $DSR(v)$. This completes the proof.

`Theorem 1.` The SearchTree algorithm finds all peaks without yielding false negatives.

`Proof.` Assume, for contradiction, that SearchTree yields a false negative. That is, there exists a peak $P = (t, w_j)$ for some j, $1 \leq j \leq m$, that can not be detected by SearchTree.

The peak P occurs in the time window starting at time point t and having a size of w_j. This time window is the shadow of $cell(w_j, t+w_j-1)$, and $cell(w_j, t+w_j-1) \geq f(w_j)$. From `Lemma 2.2.4.1`, there exists a node v at level i of the AggTree T where $a_{i-1} - s_{i-1} + 1 < w_j \leq a_i - s_i + 1$, such that $cell(w_j, t + w_j - 1)$ is shaded by the node v. Thus, w_j satisfies the condition in step 3 of the SearchTree algorithm. Since SearchTree checks each node at level i in step 4, the algorithm must be able to find the node v. Since $cell(w_j, t + w_j - 1) \geq f(w_j)$ and $cell(w_j, t + w_j - 1)$ is shaded by the node v, by monotonicity, the value stored in v must be greater than or equal to $f(w_j)$. Thus, the condition in step 5 is satisfied and $DSR(v)$ is searched.

By `Lemma 2.2.4.1`, $cell(w_j, t + w_j - 1)$ lies in $DSR(v)$. Since the algorithm checks each cell at level w_j in $DSR(v)$ in step 6, and $cell(w_j, t + w_j - 1) \geq f(w_j)$, the algorithm is able to detect and output (t, w_j) in step 8, which contradicts the assumption.

2.2.5 A State-Space Algorithm

PeakID employs a heuristic state-space algorithm to search for an efficient AggTree from a training dataset. The algorithm treats each (partially constructed) AggTree as a state and considers the growth from one partially constructed AggTree to another as a transformation. The training process is done only once, and the shift s_i, shadow size a_i and degree d_i of each level i in a final state will be used subsequently as the input values of the BuildTree algorithm in Figure 2.8 to construct AggTrees for different sets of 2D LC-MS data when detecting peaks in those datasets.

In a state-space algorithm, the problem to be solved is represented by a set of states and a set of transformation rules mapping states to states. The solutions to the problem are represented by final states that satisfy some conditions and have no outgoing transformations. The search algorithm starts from one initial state, and then repeatedly applies the transformation rules to the set of states currently being explored to generate new states. When at least one final state is reached, the algorithm stops. There are different strategies to choose the order to traverse the state space. Depth-first search, breadth-first search, best-first search, and A^* search are commonly used ones [29]. Below, we describe the main components of our best-first search algorithm.

- Initial state

  Each partially constructed AggTree must contain the input 2D LC-MS data. Thus, the initial state is the partially constructed AggTree consisting of level one only, in which each leaf node corresponds to a time point in the LC-MS data (Figure 2.11).

- Transformation rule

  If by adding a level of nodes to the top of a partially constructed AggTree A, we can get another (partially constructed) AggTree B, we say AggTree A or state A can be transformed to AggTree B or state B. Recall that there are some constraints that nodes of the top level of AggTree B must satisfy. Each node at the top level of AggTree B must aggregate several children below the top level. Each node in AggTree A is shaded by a node at the top level of AggTree B. The shift of the newly added top level of AggTree B must be an integral multiple of the shift of the level below the top level (cf. Table 2.1). The transformation rule defines how to grow a more complicated AggTree from a simpler AggTree.

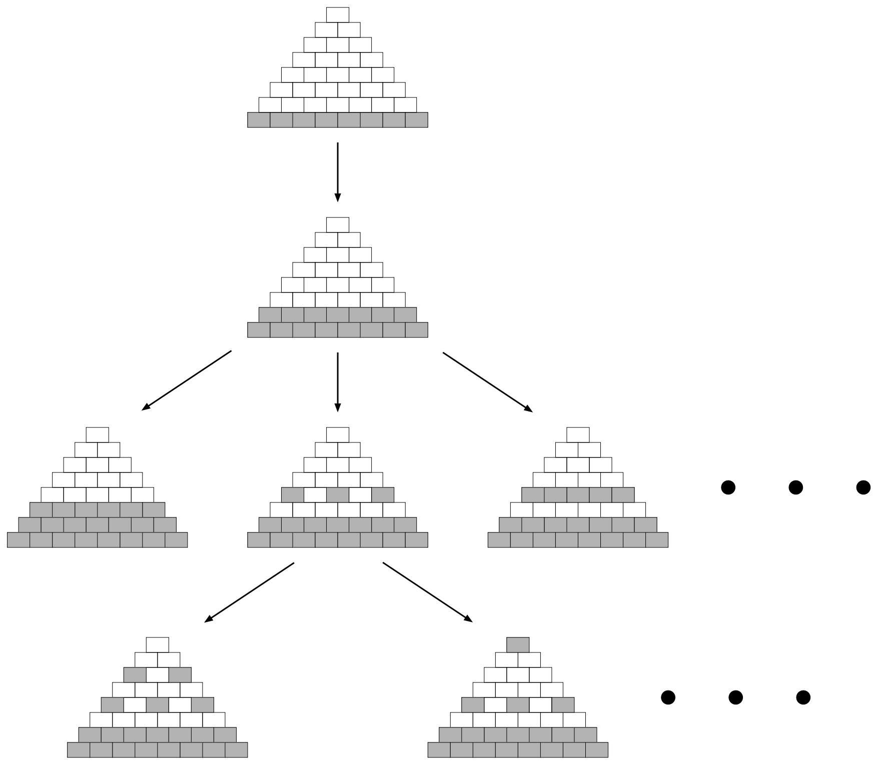


Figure 2.11 Illustration of the state-space growth process.

- Final states

  Final states are those (partially constructed) AggTrees that can be used to detect peaks across all window sizes of interest. Let h be the shadow size of the top level of an AggTree T and let s be the shift of the top level of T. For any time window W of size w, $w \leq h - s + 1$, W is shaded by a node at the top level of T. Thus, T is a final state if $h - s + 1 \geq w_m$ where w_m is the largest window size of interest.

- Traversing strategy

  In order to find an efficient AggTree, we use the best-first strategy to explore the state space. Each state (or AggTree) T is associated with a cost; the state with the minimum cost is picked as the next state to be explored. One can calculate this cost empirically by measuring the CPU time needed to build the

AggTree T and search T for peaks based on the training dataset or calculate the cost based on a theoretical model, as we will explain below.

- Desired tree

  The final state (or AggTree) T with the minimum cost is picked as the desired tree, where the shift s_i, shadow size a_i and degree d_i of each level i in T are used as input values for the BuildTree algorithm in Figure 2.8.

In summary, the heuristic state-space algorithm starts with a partially constructed AggTree having level one only, and then continues growing the candidate set of AggTrees, until a set of final AggTrees is obtained. Figure 2.11 illustrates how the state space grows.

Given an AggTree T, there are an exponential number of ways to grow T. We develop some constraints to reduce the complexity of our state-space algorithm. Let L be the maximum of the shadow sizes of the top levels of all the explored AggTrees (or states). Let S be the current state to be explored or visited. Instead of generating all possible next states from S, we generate only partially constructed AggTrees (or states) from S where the shadow sizes of the top levels of the AggTrees do not exceed $2L$. In addition, we introduce a parameter, $N = Max\_num\_states$, to govern the traversal of the state space. If we have visited N states in which the shadow sizes of the top levels are the same, we don't explore any more states whose top levels have this same shadow size. Likewise, if we have visited N final states, the algorithm stops and the final state we have visited with the minimum cost is returned as the output of the algorithm.

The cost associated with each state is used to decide which state is the next one to be explored. We develop a theoretical model to calculate the cost of an AggTree (or state) T. With this model, the cost of T equals the number of node construction operations needed to build T plus the number of cells to be checked in performing detailed searches in T when alarms are raised. The number of node construction operations equals the number of nodes in T. Let AP be the Aggregation Pyramid into which T is embedded. Let $P_a(w_j \mid a_i)$ be the probability that an alarm is raised, i.e., the probability that we check the cells at level w_j in AP given a node at level i of T with shadow size a_i (cf. step 5 of the SearchTree algorithm in Figure 2.10). Let s_i be the shift of level i and n_i be the number of nodes at level i of T. The expected number of cells to be checked is

$$\sum_{i=1}^{I}\sum_{j=1}^{m} P_a(w_j \mid a_i) \times s_i \times n_i$$

where the alarm probability $P_a(w_j \mid a_i)$ can be calculated from the training dataset as explained in the next section.

CHAPTER 3

EXPERIMENTS FOR PEAK DETECTION

In this section, we conduct a series of experiments to evaluate the performance of the PeakID method using both synthetic and real-world data. All the experiments were performed on a 2GHz Pentium 4 PC having a memory of 2G bytes. The operating system was Windows XP and the method was implemented in C++.

3.1 Experimental Results on Synthetic Data

We generated synthetic data using a random number generator with the Gaussian or normal distribution. This distribution is often used in the theoretical analysis of peaks in 2D LC-MS data [15, 8]; the synthetic data have the same statistics as the real-world LC-MS data. Twenty sets of 2D LC-MS data were generated, with each set containing one million time points. These datasets were used as test data. Our algorithms were run on the twenty test datasets, and the mean was plotted. Error bars, representing one standard error of the mean, were also plotted where the standard error was calculated by dividing the standard deviation by the square root of number of runs in the experiments. The error bars represent a description of how confident one is that the mean represents the true value. The smaller the error bars, the more reliable the plotted mean values are. In addition, 20,000 time points were generated and used as training data by the state-space algorithm to find an efficient AggTree.

Table 3.1 lists parameters and their default values used in the experiments. The parameter Max_num_states is used by the state-space algorithm to reduce the time spent in traversing the state space. The window sizes of interest comprised consecutive integers in the range $[Min\_window\_size, Max\_window\_size]$. The default value of Min_window_size was set to 3, as suggested in [15, 8, 20]. The peak probability is

Table 3.1 Parameters and Default Values Used in Experiments

Parameter	Value
Max_num_states	200
Min_window_size	3
Max_window_size	500
Peak probability	10^{-5}

the probability that a peak occurs in a time window of some size w, i.e., it is the probability that the sum of the intensity values within the time window exceeds the threshold $f(w)$ associated with w. The peak probability is inversely proportional to the threshold—the smaller the peak probability, the larger the threshold is. We assumed that the peak probability was the same for each window size of interest.

Assume that each time point in the generated 2D LC-MS data has an intensity value characterized by a mean μ and a standard deviation σ. Then the sum of the intensity values within a time window of size w has a mean $w\mu$ and standard deviation $\sqrt{w}\sigma$. Let p be the peak probability for the window size w. We can characterize this situation by saying that $Pr[S_o(w) \geq f(w)] \leq p$, where $S_o(w)$ is the observed sum of the intensity values within the time window of size w. Let $\Phi(x)$ be the normal cumulative distribution function for a normal random variable X,

$$Pr[X \geq -\Phi^{-1}(p)] \leq p$$

We have

$$Pr[\frac{S_o(w) - w\mu}{\sqrt{w}\sigma} \geq -\Phi^{-1}(p)] \leq p$$

Therefore, $f(w)$ should be set to $w\mu - \sqrt{w}\sigma\Phi^{-1}(p)$.

In Section 2.2.5, we introduced the alarm probability $P_a(w_j \mid a_i)$. This probability can be rewritten more generally as $P_a(w \mid W)$, $w \leq W$, which is the probability that the sum of the intensity values within a time window of size W exceeds the threshold $f(w)$ associated with the window size w. Referring to Figure 2.10, W is the shadow size of the node v in step 4 and w is the window size w_j in step 5 in the figure. Thus, the alarm probability is $Pr[S_o(W) \geq f(w)]$. Therefore,

$$\begin{aligned}
P_a(w \mid W) &= Pr[S_o(W) \geq f(w)] \\
&= Pr[\frac{S_o(W) - W\mu}{\sqrt{W}\sigma} \geq \frac{f(w) - W\mu}{\sqrt{W}\sigma}] \\
&= \Phi(-\frac{f(w) - W\mu}{\sqrt{W}\sigma}) \\
&= \Phi(\frac{(W-w)\mu}{\sqrt{W}\sigma} + \frac{\sqrt{w}\sigma\Phi^{-1}(p)}{\sqrt{W}\sigma}) \\
&= \Phi((\sqrt{B} - \frac{1}{\sqrt{B}})\sqrt{w}\frac{\mu}{\sigma} + \frac{\Phi^{-1}(p)}{\sqrt{B}}) \qquad (3.1)
\end{aligned}$$

where $B = \frac{W}{w}$ denotes the *bounding ratio* with respect to W, w.

So, $P_a(w \mid W)$ is determined by the distribution parameters μ and σ, the peak probability p, the bounding ratio B, and the cell level w in the Aggregation Pyramid into which the AggTree in Figure 2.10 is embedded. It can be seen from Equation (3.1) that the larger the ratio $\frac{\mu}{\sigma}$, the larger the alarm probability $P_a(w \mid W)$ is. As μ increases, there are more chances to raise an alarm. On the other hand, as σ increases, there are fewer chances to raise an alarm.

Figure 3.1 compares the theoretical cost model with the empirical cost model used in the state-space algorithm described in Section 2.2.5. We first used the theoretical cost model to find an efficient AggTree T_1 from the training dataset. The shift s_i, shadow size a_i and degree d_i of each level i in T_1 were then used by the PeakID method to build an AggTree for each test dataset and to search for peaks in that test dataset. The CPU time used by PeakID on each test dataset was recorded; the mean and error bars were plotted. We then used the empirical cost model to find an efficient AggTree T_2 from the same training dataset. The shift, shadow size and degree values of T_2 were then used by PeakID for peak detection in each test dataset.

The CPU time used by PeakID on each test dataset was recorded; the mean and error bars were also plotted. Figure 3.1 shows that the theoretical cost model is better than the empirical cost model; the AggTree T_1 produced from the theoretical cost model leads to a more efficient PeakID than that based on the AggTree T_2 produced from the empirical cost model. The X-axis in Figure 3.1 shows different k values where each k corresponds to a peak probability 10^{-k}. As k becomes larger, the peak probability becomes smaller and consequently less time is required by PeakID to detect the fewer peaks.

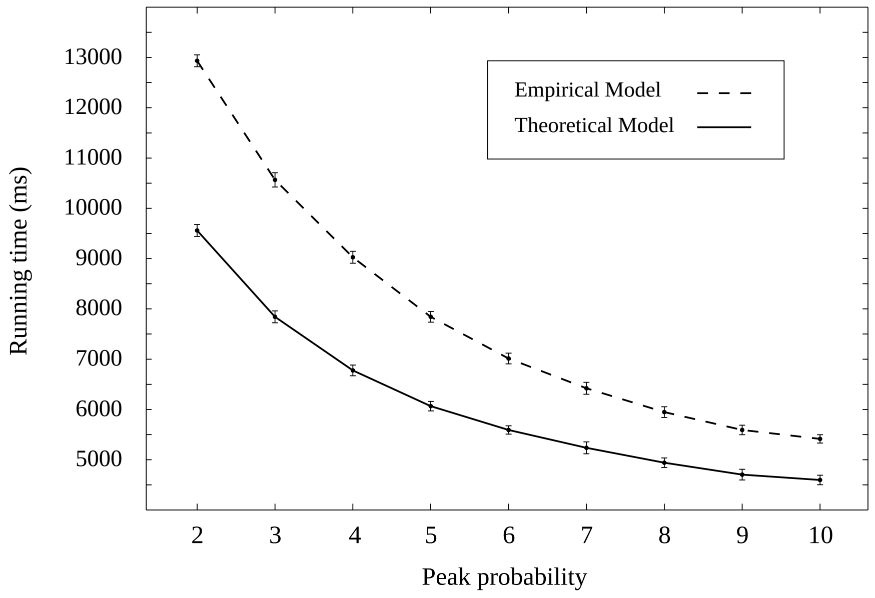


Figure 3.1 Comparison of the theoretical cost model and the empirical cost model used in the state-space algorithm.

In subsequent experiments, we used the theoretical cost model to generate the efficient AggTree employed by PeakID. Figure 3.2 shows the impact of Max_num_states on the running time of PeakID. There is not much difference among the efficient AggTrees produced by the state space algorithm using different values of Max_num_states, provided that this value is sufficiently large (e.g., $Max\_num\_states \geq 10$). Since the method using BinaryTrees does not employ a state space algorithm to generate a desired tree from the training dataset, the running time for the method using

BinaryTrees is a constant, independent of the Max_num_states values on the X-axis. By spending some time in the training phase to obtain an efficient AggTree, PeakID can run significantly faster than the method using BinaryTrees, as demonstrated in Figure 3.2.

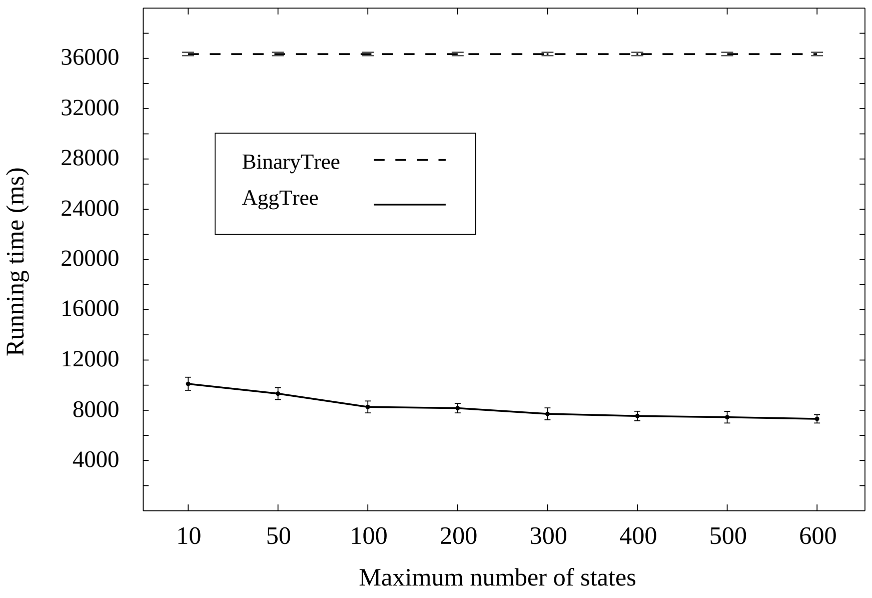


Figure 3.2 The effect of Max_num_states on the running time of PeakID.

Figure 3.3 shows the density of an AggTree (BinaryTree, respectively) as a function of peak probabilities. The X-axis in the figure shows different k values where each k corresponds to a peak probability 10^{-k}. The Y-axis shows different densities where the density of a tree is defined as N_n/N_c; N_n is the number of nodes in the tree and N_c is the number of cells in the Aggregation Pyramid into which the tree is embedded. The figure shows that an AggTree becomes sparser or less dense when the peak probability becomes smaller. This happens because when peaks are rare, we only need few nodes in the AggTree to find those peaks. By contrast, the density of a BinaryTree is almost a constant, since the parent-child structure and the shifting pattern of the BinaryTree is fixed (cf. Table 2.1). Figure 3.3 shows that an AggTree is able to adapt to the input data. Notice that AggTrees have a higher density than BinaryTrees. This implies that AggTrees generally have more levels and hence have smaller bounding ratios than BinaryTrees.

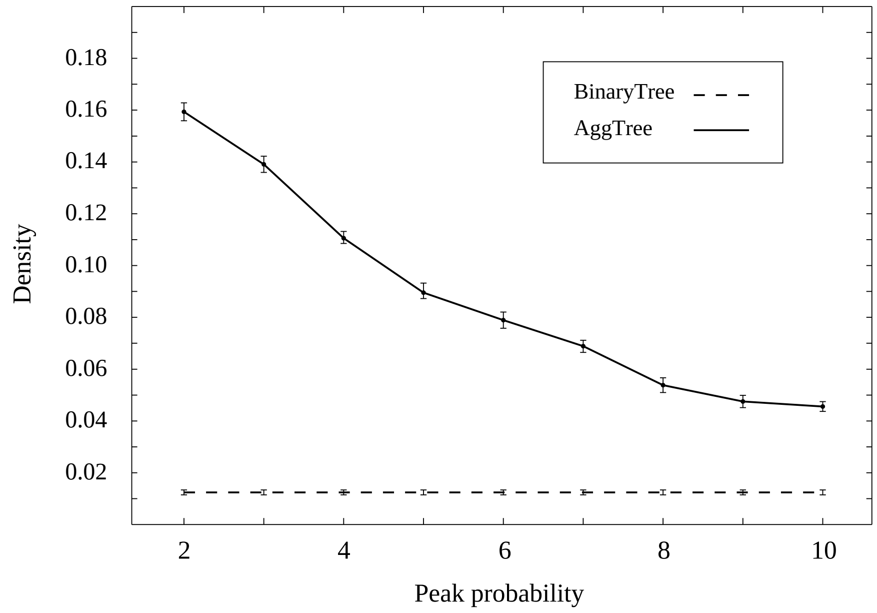


Figure 3.3 The impact of peak probabilities on the density of a tree.

Figure 3.4 shows the bounding ratio with respect to each level in an AggTree (BinaryTree, respectively). In an AggTree, the bounding ratio B is large at a low level where the shadow window size w is small; B is small at a high level where the shadow window size w is large. These changes of bounding ratios do not occur in BinaryTrees. The reason is that in a BinaryTree, for a node $v = cell(2^{i-1}, t)$, the shadow sizes of the cells in the detailed search region of v are in the range $[2^{i-3}+2, 2^{i-2}+1]$, cf. Section 2.2.2. We check the value stored in v; if the value exceeds the threshold for some window size in the range $[2^{i-3}+2, 2^{i-2}+1]$, an alarm is raised. Thus, the bounding ratio in a BinaryTree is always in the range $[2^{i-1}/(2^{i-3}+2)$, $2^{i-1}/(2^{i-2}+1)]$, which is approximately [2, 4].

Figure 3.5 shows the alarm probability with respect to each level in an AggTree (BinaryTree, respectively). As explained above, in an AggTree, when the node level becomes larger, the bounding ratio becomes smaller. Referring to Equation (3.1), as

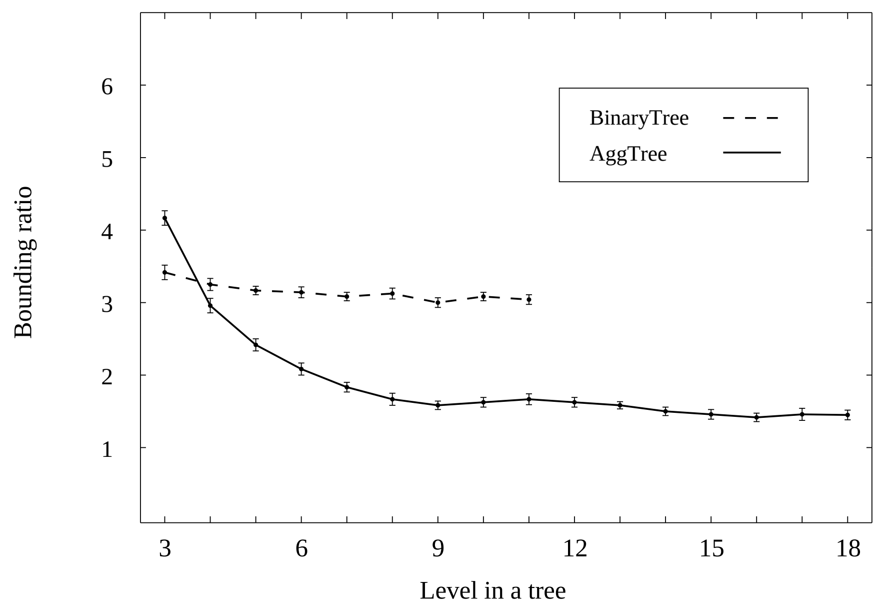


Figure 3.4 The bounding ratio with respect to each level in a tree.

the bounding ratio decreases, so does the alarm probability, and consequently fewer detailed searches are performed. On the other hand, in a BinaryTree, the alarm probability is high and hence many detailed searches are needed when the node level is large.

The above analyses lead to the conclusion that searching for peaks using AggTrees would require less time and hence be more efficient than using BinaryTrees. Figure 3.6 confirms this conclusion, showing the relative performance of these two data structures for varying peak probabilities. For comparison purposes, we also include the brute-force method in Figure 3.6. Clearly, PeakID using AggTrees outperforms the other two methods. Notice that as the peak probability decreases, so does the alarm probability, cf. Equation (3.1). Consequently, the running time of PeakID decreases.

Figure 3.7 compares the relative performance of the three studied methods for varying window sizes of interest. The window sizes of interest comprised consecutive integers in the range [3, Max_window_size]. The X-axis shows different

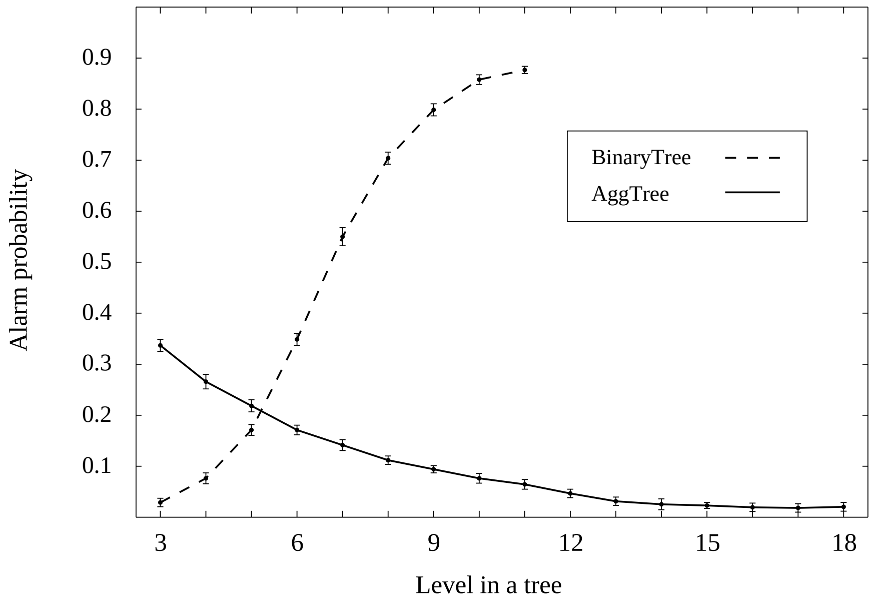


Figure 3.5 The alarm probability with respect to each level in a tree.

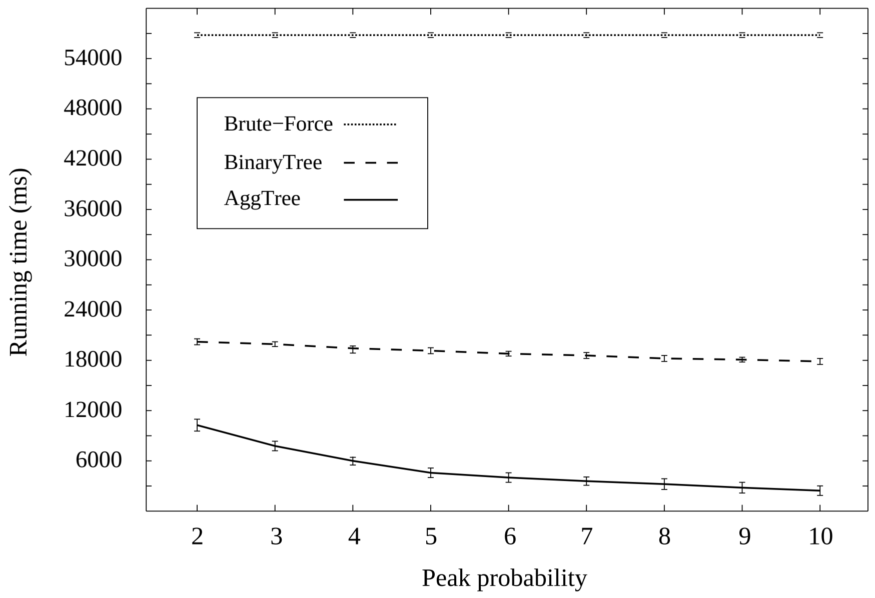


Figure 3.6 Comparison of the three studied methods for varying peak probabilities.

values of *Max_window_size*. Again, PeakID using AggTrees is the best. When *Max_window_size* becomes large, the superiority of PeakID becomes more obvious. This happens because with a large *Max_window_size*, there are more node levels in an AggTree where the bounding ratios can be adjusted, thereby speeding up the search for peaks in the AggTree.

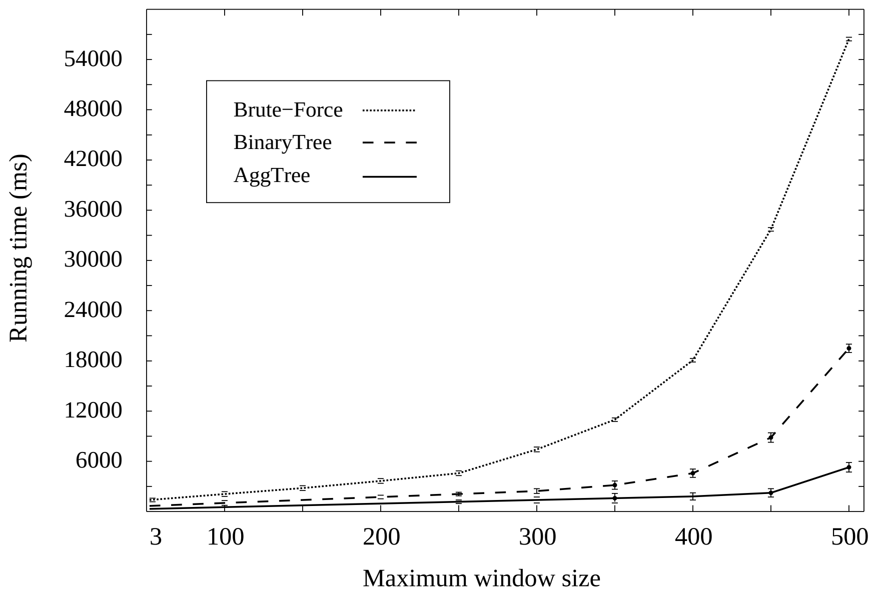


Figure 3.7 Comparison of the three studied methods for varying window sizes of interest.

In summary, an AggTree can adapt to the input data, adjusting its topology and structure through the training process to reduce alarm probabilities. By contrast, a BinaryTree does not employ the training process, and its parent-child structure and shifting pattern is always fixed (cf. Table 2.1). As a consequence, AggTrees are far more efficient than BinaryTrees when used in detecting peaks across different window sizes of interest.

3.2 Experimental Results on Real-World Data

We obtained 2D LC-MS data from bovine serum albumin (BSA), which is a protein commonly used to test biochemical characters [30]. There were six datasets in total.

Table 3.2 Description of Real-World Data

Data	Dataset Size	Max Intensity	Min Intensity	Mean	StdDev
Dataset 1	5168	1822.5	457.3	1022.5	516.4
Dataset 2	5263	1992.3	456.2	1025.1	519.6
Dataset 3	5143	1992.1	464.9	1029.5	516.1
Dataset 4	5155	2013.5	463.5	1044.6	518.5
Dataset 5	5261	1785.4	467.9	1043.7	519.8
Dataset 6	5284	1883.2	463.3	1038.3	515.5

Table 3.2 gives details of these datasets, showing the size of each dataset, i.e., the number of time points in each dataset. In addition, the table lists the maximum intensity value, the minimum intensity value, as well as the mean and standard deviation of the intensity values associated with the time points in each dataset. The experiments were performed in six phases. In phase i, $1 \leq i \leq 6$, dataset i was used as the training data and five runs of experiments were performed where in run j, $1 \leq j \leq 6$, $j \neq i$, dataset j was used as the test data. This led to thirty runs in total; the mean and error bars obtained from the thirty runs were plotted. The window sizes of interest in the experiments comprised consecutive integers in the range [3, Max_window_size]. The peak probability was fixed at 1/200.

Figure 3.8 compares the relative performance of the three studied methods on the protein data for varying window sizes of interest. The trend observed here is consistent with that from the synthetic data (cf. Figure 3.7). PeakID using AggTrees outperforms the method using BinaryTrees, which in turn is better than the brute-force method. We also tested the three methods by varying peak probabilities; the results were similar to those presented here.

Finally, we compared PeakID with the techniques developed by Stolt *et al.* [8] on the protein data. To avoid false negatives, we combined Stolt *et al.*'s techniques with

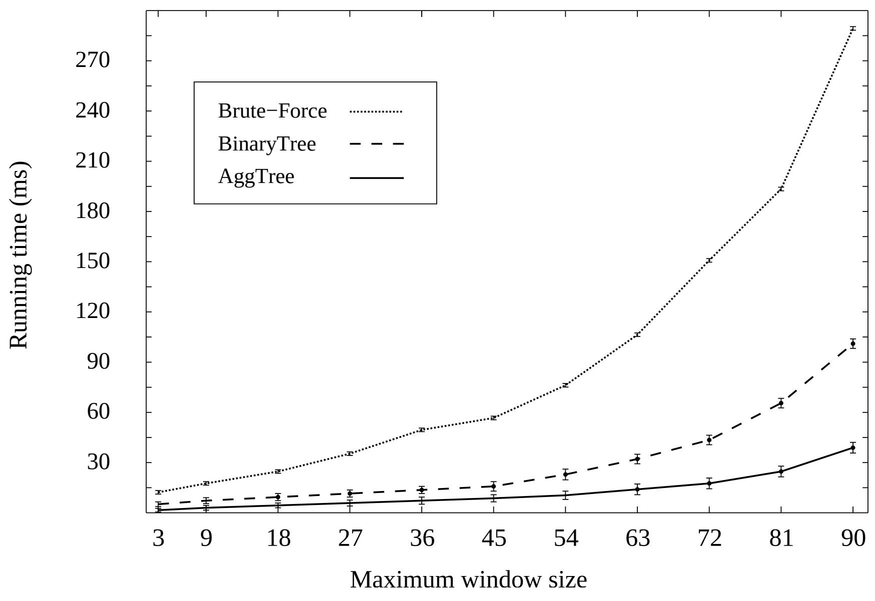


Figure 3.8 Comparison of the three studied methods on real-world data.

the brute-force algorithm for checking different window sizes. Table 3.3 summarizes the threshold, denoted by θ, and the corresponding number of peaks, denoted by n, with respect to each dataset and window size found by Stolt *et al.*'s method. With the window sizes and threshold values in Table 3.3, PeakID obtained the same results while speeding up Stolt *et al.*'s method by a factor of 10.

Table 3.3 Experimental Results Obtained from Real-World Data

Window Size	Dataset 1		Dataset 2		Dataset 3		Dataset 4		Dataset 5		Dataset 6	
	θ	n	θ	n	θ	n	θ	n	θ	n	θ	n
3	5123	6	5201	9	5166	5	5353	7	5230	6	5501	7
4	7019	2	7312	1	7189	2	7033	3	7225	4	7306	2
5	8636	3	7991	5	8822	2	8617	4	8781	2	8430	3
6	11137	1	10355	1	11678	2	12309	2	9978	4	10225	1
7	12998	1	12782	1	13350	2	12001	1	13130	1	12644	2
8	14203	2	14552	1	15687	1	13884	2	14232	1	16008	1
11	18727	3	18556	1	19207	1	17770	2	21644	1	20399	1
12	22810	1	21943	1	23157	1	22632	1	20246	3	24338	1
14	26172	2	27403	2	26189	1	26636	1	25887	2	28001	1
17	31827	1	30228	2	32774	2	31555	1	32077	1	32222	1
20	34134	2	34413	1	33318	1	35256	1	36021	1	34188	1
21	39261	1	41195	1	41552	1	42397	1	40275	1	39565	1

CHAPTER 4

ALGORITHMS FOR PSEUDOKNOT ALIGNMENT

4.1 Background for Pseudoknot Alignment

RNA pseudoknots are formed by pairing bases on single-stranded loops, such as hairpin and internal loops, with bases outside the loops [31, 32]. They are often mingled with other RNA tertiary motifs [33], and are also found in non-coding RNAs [34, 35]. RNA pseudoknots, with diverse functions [36, 37], play important roles in many biological processes [38, 39]; for example, they are required for telomerase activity [37], and have been shown to regulate the efficiency of ribosomal frameshifting in viruses [40].

Analysis and detection of RNA pseudoknots has been an active area of research. Many published articles in this area were focused on pseudoknot alignment [41, 42, 43, 44]. In this paper, we present a new approach, called RKalign, for RNA pseudoknot alignment. RKalign accepts as input two pseudoknotted RNAs where each RNA has both sequence data (i.e., nucleotides or bases) and structure data (i.e., base pairs), and produces as output an alignment between the two pseudoknotted RNAs. The structure data of a pseudoknotted RNA can be obtained from the literature or public databases [45, 46, 47, 48].

RKalign adopts the partition function methodology to calculate the posterior probabilities or log-odds scores of structural alignments. The idea of using posterior probabilities to align biomolecules originated from [49, 50] where the partition function methodology was employed to calculate the posterior probabilities of protein sequence alignments. Similar techniques were proposed by Do *et al.* [51] where the authors used hidden Markov models (HMMs) to calculate the posterior probabilities. Will *et al.* [52] extended the idea of [49, 50, 51] to structure-based multiple RNA alignment where the authors calculated partition functions inside and outside of subsequence pairs on two pseudoknot-free RNAs. Here, we further extend this idea to pseudoknot alignment.

Several tools are available for RNA sequence-structure alignment [53, 54, 55]. These tools do not deal with pseudoknots. Mohl *et al.* [56] proposed a method to perform sequence-structure alignment for RNA pseudoknots. The authors set up a pipeline for combining alignment and prediction of pseudoknots, and showed experimentally the effectiveness of this pipeline in pseudoknot structure annotation. Han *et al.* [57] decomposed embedded pseudoknots into simple pseudoknots and aligned them recursively. Yoon [58] used a profile-HMM to establish sequence alignment constraints, and incorporated these constraints into an algorithm for aligning RNAs with pseudoknots. Wong *et al.* [59] identified the pseudoknot type of a given structure and developed dynamic programming algorithms for structural alignments of different pseudoknot types. Huang *et al.* [34] applied a tree decomposition algorithm to search for non-coding RNA pseudoknot structures in genomes.

The above methods were concerned with aligning a pseudoknot structure with a sequence or genome. Through the alignment, the sequence is folded and its structure is predicted. Xu *et al.* [41] presented a different method, called RNA Sampler, which can simultaneously fold and align two or multiple RNA sequences considering pseudoknots without known structures. Similar techniques were implemented in DAFS [42] and SimulFold [43]. Additional methods can be found in the CompaRNA web server [60]. In contrast to these methods, which perform alignment and folding at the same time, RKalign aims to align two known RNA pseudoknot structures where the structures are obtained from existing databases [45, 46, 47]. As more pseudoknot structures become available in these databases, a tool like RKalign will be useful in performing data analysis in the repositories.

There are two groups of algorithms which are also capable of aligning two known RNA structures. The first group is concerned with aligning two RNA three-dimensional (3D) structures, possibly containing pseudoknots. Ferre *et al.* [61] presented a dynamic programming algorithm by taking into account nucleotide, dihedral angle and base-pairing similarities. Capriotti and Marti-Renom [62] developed a program to align two RNA 3D structures based on a unit-vector root-mean-square approach. Chang *et al.* [63] and Wang *et al.* [64] employed a structural alphabet of different nucleotide conformations to align RNA 3D structures. Hoksza and Svozil [65] developed a pairwise comparison method based on 3D similarity of generalized secondary structure units. Rahrig *et al.* [66] presented the R3D Align tool for performing global pairwise alignment of RNA 3D structures using local superpositions. He *et al.* [67] developed the RASS web server for comparing RNA 3D structures using both sequence and 3D structure information. The above methods and tools were mainly designed for aligning two RNA tertiary structures by considering their geometric properties and torsion angles. In contrast, RKalign is used to align two RNA secondary structures with pseudoknots.

The second group of algorithms is concerned with aligning two RNA secondary structures without pseudoknots. These algorithms employed general edit-distance alignment [68] or tree matching techniques [69, 70, 71]. Jiang *et al.* [72] developed an approximation algorithm for aligning a pseudoknot-free structure with a pseudoknotted structure. Our work differs from Jiang *et al.*'s work in that we focus on the alignment of two pseudoknotted structures. Furthermore, we use the partition function methodology whereas Jiang *et al.* adopted a general edit-distance approach to the structural alignment.

The method that is most closely related to ours is an option offered by the CARNA tool [44]. Like RKalign, this option is able to accept two known RNA secondary structures with pseudoknots, and produce an alignment between the two

RNA structures. This option employs constraint programming techniques with a branch and bound scheme. It gradually refines solutions until the best solution is found. To understand the relative performance of the two tools, we perform extensive experiments to compare RKalign with CARNA using different datasets.

4.2 The RKalign Method

In this section, we present algorithmic details of RKalign. To align two RNA pseudoknot structures A and B, we adopt the partition function methodology to calculate the posterior probabilities or log-odds scores of the alignments between bases or base pairs in A and B, respectively. After calculating the posterior log-odds scores, we then compute the expected accuracy of an alignment between structure A and structure B. The goal is to find an optimal alignment between A and B where the alignment has the maximum expected accuracy. We will present a heuristic to achieve this goal.

4.2.1 Definitions and Notation

Suppose (i, j) is a base pair of pseudoknot structure A and (p, q) is a base pair of pseudoknot structure B. We use $score((i, j), (p, q))$ to represent the score of aligning (i, j) with (p, q) where the score is obtained from the log-odds RIBOSUM matrix [73]. The use of this scoring matrix permits RKalign to determine the similarity between pseudoknot structures that contain compensatory base changes. With this scoring matrix, RKalign is able to handle non-canonical base pairs. Aligning a single base with a base pair is prohibited by RKalign.

Suppose structure A has m nucleotides, i.e., the length of A is m, and structure B has n nucleotides, i.e., the length of B is n. We use $A[c_1, c_1]$ where $1 \leq c_1 \leq c_2 \leq m$ to represent the portion of A that begins at position c_1 and ends at position c_2 inclusively. We use $B[d_1, d_2]$ where $1 \leq d_1 \leq d_2 \leq n$ to represent the portion of B

that begins at position d_1 and ends at position d_2 inclusively. We use $A[c]$ to represent the nucleotide and secondary structure at position c of A, and $B[d]$ to represent the nucleotide and secondary structure at position d of B.

4.2.2 Partition Function Computation

Suppose $(i, j) \in A$ is aligned with $(p, q) \in B$. Let $Z_{c,d}$ ($Z'_{c,d}$, respectively) represent the partition function of all alignments between $A[1, c]$ ($A[c, m]$, respectively) and $B[1, d]$ ($B[d, n]$, respectively). Let $Z''_{c,d}$ represent the partition function of all alignments between $A[i+1, c]$ and $B[p+1, d]$. We focus on the case in which both i, j and (p, q) are base pairs. The case for aligning single bases is simpler, and thus omitted.

First, we show how to calculate $Z_{c,d}$ where $1 \leq c < i$ and $1 \leq d < p$. There are three cases to be considered:

(1) $A[c]$ is aligned with $B[d]$.

(2) $B[d]$ is aligned to a gap.

(3) $A[c]$ is aligned to a gap.

Let $Z^M_{c,d}$ represent the partition function of all alignments between $A[1, c]$ and $B[1, d]$ where $A[c]$ is aligned with $B[d]$. Let $Z^E_{c,d}$ represent the partition function of all alignments between $A[1, c]$ and $B[1, d]$ where $B[d]$ is aligned to a gap. Let $Z^F_{c,d}$ represent the partition function of all alignments between $A[1, c]$ and $B[1, d]$ where $A[c]$ is aligned to a gap. Then $Z_{c,d}$ can be calculated by Equation (4.1).

$$Z_{c,d} = Z^M_{c,d} + Z^E_{c,d} + Z^F_{c,d} \tag{4.1}$$

We ignore and skip the computation of $Z_{c,d}$ when $A[c]$ or $B[d]$ is the left base of some base pair. If $A[c]$ ($B[d]$, respectively) is a single base and $B[d]$ ($A[c]$, respectively) is the right base of some base pair, $Z^M_{c,d} = 0$. Otherwise, let $A[c]$ be the right base of some base pair (x, c) and let $B[d]$ be the right base of some base pair (y, d). Following [50], $Z^M_{c,d}$ can be calculated by Equation (4.2).

$$Z^M_{c,d} = Z_{c-1,d-1} e^{\frac{score((x,c),(y,d))}{T}} \tag{4.2}$$

Here T is a constant, and $score((x, c), (y, d))$ is obtained from the RIBOSUM85-60 matrix [73]. Thus, the partition function $Z^M_{c,d}$ can be computed recursively by dynamic programming as follows:

$$Z^M_{c,d} = (Z^M_{c-1,d-1} + Z^E_{c-1,d-1} + Z^F_{c-1,d-1}) e^{\frac{score((x,c),(y,d))}{T}} \tag{4.3}$$

When calculating $Z^E_{c,d}$, since $B[d]$ is aligned to a gap, we know that $A[c]$ must be aligned with $B[d - 1]$. Therefore,

$$Z^E_{c,d} = Z_{c,d-1} e^{\frac{score(-,(y,d))}{T}} \tag{4.4}$$

where $score(-, (y, d))$ is the gap penalty value obtained by aligning base pair (y, d) to gaps. Thus,

$$Z^E_{c,d} = (Z^M_{c,d-1} + Z^E_{c,d-1} + Z^F_{c,d-1}) e^{\frac{score(-,(y,d))}{T}} \tag{4.5}$$

When calculating $Z^F_{c,d}$, since $A[c]$ is aligned to a gap, $B[d]$ must be aligned with $A[c - 1]$. Therefore,

$$Z^F_{c,d} = Z_{c-1,d} e^{\frac{score((x,c),-)}{T}} \tag{4.6}$$

where $score((x, c), -)$ is the gap penalty value obtained by aligning base pair (x, c) to gaps. Thus,

$$Z^F_{c,d} = (Z^M_{c-1,d} + Z^E_{c-1,d} + Z^F_{c-1,d}) e^{\frac{score((x,c),-)}{T}} \tag{4.7}$$

Next, we show how to calculate $Z'_{c,d}$ where $j < c \leq m$ and $q < d \leq n$. There are three cases to be considered:

(1) $A[c]$ is aligned with $B[d]$.

(2) $B[d]$ is aligned to a gap.

(3) $A[c]$ is aligned to a gap.

Let $Z'^{M}_{c,d}$ represent the partition function of all alignments between $A[c,m]$ and $B[d,n]$ where $A[c]$ is aligned with $B[d]$. Let $Z'^{E}_{c,d}$ represent the partition function of all alignments between $A[c,m]$ and $B[d,n]$ where $B[d]$ is aligned to a gap. Let $Z'^{F}_{c,d}$ represent the partition function of all alignments between $A[c,m]$ and $B[d,n]$ where $A[c]$ is aligned to a gap. Then $Z'_{c,d}$ can be calculated by Equation (4.8).

$$Z'_{c,d} = Z'^{M}_{c,d} + Z'^{E}_{c,d} + Z'^{F}_{c,d} \tag{4.8}$$

We ignore the computation of $Z'_{c,d}$ when $A[c]$ or $B[d]$ is the right base of some base pair. If $A[c]$ ($B[d]$, respectively) is a single base and $B[d]$ ($A[c]$, respectively) is the left base of some base pair, $Z'^{M}_{c,d} = 0$. Otherwise, let $A[c]$ be the left base of some base pair (c,x) and let $B[d]$ be the left base of some base pair (d,y). Following [50], $Z'^{M}_{c,d}$ can be calculated by Equation (4.9).

$$\begin{aligned} Z'^{M}_{c,d} &= Z'_{c+1,d+1} e^{\frac{score((c,x),(d,y))}{T}} \\ &= (Z'^{M}_{c+1,d+1} + Z'^{E}_{c+1,d+1} + Z'^{F}_{c+1,d+1}) e^{\frac{score((c,x),(d,y))}{T}} \end{aligned} \tag{4.9}$$

When calculating $Z'^{E}_{c,d}$, since $B[d]$ is aligned to a gap, $A[c]$ must be aligned with $B[d+1]$. Therefore,

$$\begin{aligned} Z'^{E}_{c,d} &= Z'_{c,d+1} e^{\frac{score(-,(d,y))}{T}} \\ &= (Z'^{M}_{c,d+1} + Z'^{E}_{c,d+1} + Z'^{F}_{c,d+1}) e^{\frac{score(-,(d,y))}{T}} \end{aligned} \tag{4.10}$$

When calculating $Z'^{F}_{c,d}$, since $A[c]$ is aligned to a gap, $B[d]$ must be aligned with $A[c+1]$. Therefore,

$$Z'^{F}_{c,d} = Z'_{c+1,d} e^{\frac{score((c,x),-)}{T}}$$

$$= (Z'^{M}_{c+1,d} + Z'^{E}_{c+1,d} + Z'^{F}_{c+1,d})e^{\frac{score((c,x),-)}{T}} \quad (4.11)$$

Finally, we show how to calculate $Z''_{c,d}$ where $i < c < j$ and $p < d < q$. There are three cases to be considered:

(1) $A[c]$ is aligned with $B[d]$.

(2) $B[d]$ is aligned to a gap.

(3) $A[c]$ is aligned to a gap.

Let $Z''^{M}_{c,d}$ represent the partition function of all alignments between $A[i+1,c]$ and $B[p+1,d]$ where $A[c]$ is aligned with $B[d]$. Let $Z''^{E}_{c,d}$ represent the partition function of all alignments between $A[i+1,c]$ and $B[p+1,d]$ where $B[d]$ is aligned to a gap. Let $Z''^{F}_{c,d}$ represent the partition function of all alignments between $A[i+1,c]$ and $B[p+1,d]$ where $A[c]$ is aligned to a gap. Then $Z''_{c,d}$ can be calculated by Equation (4.12).

$$Z''_{c,d} = Z''^{M}_{c,d} + Z''^{E}_{c,d} + Z''^{F}_{c,d} \quad (4.12)$$

We ignore the computation of $Z''_{c,d}$ when $A[c]$ or $B[d]$ is the left base of some base pair. If $A[c]$ ($B[d]$, respectively) is a single base and $B[d]$ ($A[c]$, respectively) is the right base of some base pair, $Z''^{M}_{c,d} = 0$. Otherwise, let $A[c]$ be the right base of some base pair (x, c) and let $B[d]$ be the right base of some base pair (y, d). If $x < i+1$ or $y < p+1$ we ignore and skip the computation of $Z''_{c,d}$. We consider only the case where $x \geq i+1$ and $y \geq p+1$. Following [50], $Z''^{M}_{c,d}$ can be calculated by Equation (4.13).

$$\begin{aligned} Z''^{M}_{c,d} &= Z''_{c-1,d-1}e^{\frac{score((x,c),(y,d))}{T}} \\ &= (Z''^{M}_{c-1,d-1} + Z''^{E}_{c-1,d-1} + Z''^{F}_{c-1,d-1})e^{\frac{score((x,c),(y,d))}{T}} \quad (4.13) \end{aligned}$$

When calculating $Z''^{E}_{c,d}$, since $B[d]$ is aligned to a gap, $A[c]$ must be aligned with $B[d-1]$. Therefore,

$$\begin{aligned} Z''^{E}_{c,d} &= Z''_{c,d-1} e^{\frac{score(-,(y,d))}{T}} \\ &= (Z''^{M}_{c,d-1} + Z''^{E}_{c,d-1} + Z''^{F}_{c,d-1}) e^{\frac{score(-,(y,d))}{T}} \end{aligned} \tag{4.14}$$

When calculating $Z''^{F}_{c,d}$, since $A[c]$ is aligned to a gap, $B[d]$ must be aligned with $A[c-1]$. Therefore,

$$\begin{aligned} Z''^{F}_{c,d} &= Z''_{c-1,d} e^{\frac{score((x,c),-)}{T}} \\ &= (Z''^{M}_{c-1,d} + Z''^{E}_{c-1,d} + Z''^{F}_{c-1,d}) e^{\frac{score((x,c),-)}{T}} \end{aligned} \tag{4.15}$$

4.2.3 Calculation of Posterior Log-odds Scores

There are four cases to be considered when calculating the posterior probability or log-odds score of aligning base pair (i, j) of structure A with base pair (p, q) of structure B, denoted by $Prob((i, j), (p, q))$.

`Case 1.` Base pair (i, j) doesn't cross another base pair and (p, q) doesn't cross another base pair. That is, for any base pair (u, v), $i < v < j$ iff $i < u < j$. Furthermore, for any base pair (x, y), $p < y < q$ iff $p < x < q$. Consequently, the alignment between structure A and structure B can be divided into the following three parts:

(1) The alignment between $A[1, i-1]$ and $B[1, p-1]$.

(2) The alignment between $A[i+1, j-1]$ and $B[p+1, q-1]$.

(3) The alignment between $A[j+1, m]$ and $B[q+1, n]$.

Following [50] we get

$$Prob((i, j) \sim (p, q)) = \frac{Z_{i-1,p-1} Z''_{j-1,q-1} Z'_{j+1,q+1}}{Z_{m,n}} e^{\frac{score((i,j),(p,q))}{T}} \tag{4.16}$$

`Case 2.` Base pair (i, j) crosses another base pair whereas (p, q) doesn't cross another base pair. That is, there exists a base pair (u, v) in A falls into one of the following conditions:

(1) $i < v < j$ and $u < i$.

(2) $i < u < j$ and $v > j$.

Furthermore, for any base pair (x, y), $p < y < q$ iff $p < x < q$. In this case, (i, j) crosses (u, v), which forms a pseudoknot in structure A, while (p, q) doesn't form a pseudoknot in structure B.

When (1) is true, since $u < i$, we have $1 \leq u \leq i - 1$. Furthermore, since $v > i > i - 1$, (u, v) is ignored when calculating $Z_{i-1,p-1}$ in Equation (4.16). In addition, since $u < i < i + 1$, (u, v) is ignored when calculating $Z''_{j-1,q-1}$ in Equation (4.16). Base pair (u, v) will be considered when calculating $Prob((u, v) \sim (p, q))$. Thus, our algorithm doesnt miss the calculation of the posterior log-odds score of aligning any two base pairs from structure A and structure B, respectively.

When (2) is true, since $v > j$, we have $j + 1 \leq v \leq m$. Furthermore, since $u < j < j + 1$, (u, v) is ignored when calculating $Z'_{j+1,q+1}$ in Equation (4.16). Base pair (u, v) will be considered when calculating $Prob((u, v) \sim (p, q))$.

`Case 3.` Base pair (p, q) crosses another base pair whereas (i, j) doesn't cross another base pair. This case is similar to `Case 2` above.

`Case 4.` Base pair (i, j) crosses another base pair and (p, q) also crosses another base pair. That is, there exists a base pair (u, v) in A falls into one of the following conditions:

(1) $i < v < j$ and $u < i$.

(2) $i < u < j$ and $v > j$.

Furthermore, there exists a base pair (x, y) in B falls into one of the following conditions:

(3) $p < y < q$ and $x < p$.

(4) $p < x < q$ and $y > q$.

In this case, (i, j) crosses (u, v), which forms a pseudoknot in structure A. Furthermore (p, q) crosses (x, y), which also forms a pseudoknot in structure B.

When (1) and (3) are true, (u, v) is ignored when calculating $Z_{i-1,p-1}$ and $Z''_{j-1,q-1}$ as discussed in `Case 2 (1)`. Moreover, (x, y) is also ignored when calculating $Z_{i-1,p-1}$ and $Z''_{j-1,q-1}$ (`Case 3`). When (1) and (4) are true, (u, v) is ignored when calculating $Z_{i-1,p-1}$ and $Z''_{j-1,q-1}$ (`Case 2 (1)`); (x, y) is also ignored when calculating $Z'_{j+1,q+1}$ (`Case 3`). When (2) and (3) are true, (u, v) is ignored when calculating $Z'_{j+1,q+1}$ (`Case 2 (2)`); (x, y) is also ignored when calculating $Z_{i-1,p-1}$ and $Z''_{j-1,q-1}$ (`Case 3`). When (2) and (4) are true, (u, v) is ignored when calculating $Z'_{j+1,q+1}$ (`Case 2 (2)`); (x, y) is also ignored when calculating $Z'_{j+1,q+1}$ (`Case 3`).

When both (i, j) and (p, q) are single bases, i.e., $i = j$ and $p = q$, the value of $Z''_{j-1,q-1}$ in Equation (4.16) is defined as 1, and we use the same formula in Equation (4.16) to calculate $Prob((i, j) \sim (p, q))$.

From the above discussions, Equation (4.16) can be used to calculate the posterior log-odds score of aligning two bases or base pairs with a dynamic programming algorithm. Furthermore, the algorithm doesn't miss the calculation of the posterior log-odds score of aligning any two bases or base pairs from structure A and structure B, respectively.

4.2.4 Pairwise Alignment

Let $a_{A,B}$ be an alignment between structure A and structure B. The expected accuracy of $a_{A,B}$, denoted $\overline{Accu}(a_{A,B})$, is defined as follows [51]:

$$\overline{Accu}(a_{A,B}) = \frac{\sum_{((i,j)\sim(p,q)\in a_{A,B})} Prob((i, j) \sim (p, q))}{max\{h, k\}} \tag{4.17}$$

where $((i,j) \sim (p,q) \in a_{A,B})$ means $(i,j) \in A$ is aligned with $(p,q) \in B$ in $a_{A,B}$, $Prob((i,j) \sim (p,q))$ is the posterior log-odds score of aligning $(i,j) \in A$ with $(p,q) \in B$ as defined in Equation (4.16), and h (k, respectively) is the number of single bases plus the number of base pairs in A (B, respectively).

An optimal alignment between structure A and structure B is an alignment with the maximum expected accuracy. We present here a heuristic to find a (sub)optimal alignment. From the previous subsection, we are able to construct the posterior log-odds score matrix for aligning structure A with structure B where the matrix contains $Prob((i,j) \sim (p,q))$ for all $(i,j) \in A$ and $(p,q) \in B$. Our heuristic is an iterative procedure. In the first step, we select two bases or base pairs with the largest score from this matrix to build the first alignment line between A and B where the alignment line connects the selected bases or base pairs. Then, we select the second largest score from the matrix to construct the next alignment line provided that the newly constructed alignment line satisfies the following two constraints:

(1) A base (base pair, respectively) can be aligned with at most one base (base pair, respectively).

(2) The newly constructed alignment lines do not cross the alignment lines built in the previous steps. Specifically, suppose (i,j) is aligned with (p,q) and (i',j') is aligned with (p',q'). The alignment lines between (i,j) and (p,q) do not cross the alignment lines between (i',j') and (p',q') if and only if the following conditions hold:

   (i) $i' < i$ iff $p' < p$.

   (ii) $i < i' < j$ iff $p < p' < q$.

   (iii) $i' > j$ iff $p' > q$.

   (iv) $j' < i$ iff $q' < p$.

   (v) $i < j' < j$ iff $p < q' < q$.

(vi) $j' > j$ iff $q' > q$.

If the newly constructed alignment line violates the above constraints, it is discarded. We repeat the above steps until the smallest posterior log-odds score in the matrix is considered. If there are still bases or base pairs that are not aligned yet, these remaining bases or base pairs are aligned to gaps.

4.2.5 Time and Space Complexity

In calculating $Prob((i,j) \sim (p,q))$, we need to compute $Z_{i-1,p-1}$, $Z''_{j-1,q-1}$ and $Z'_{j+1,q+1}$; cf. Equation (4.16). Computing $Z_{i-1,p-1}$, $Z''_{j-1,q-1}$ and $Z'_{j+1,q+1}$ requires $O(mn)$ time. Since we need to calculate $Prob((i,j) \sim (p,q))$ for all $(i,j) \in A$ and $(p,q) \in B$, the time complexity of the pairwise alignment algorithm is $O(m^2n^2)$. At any moment, we maintain a two-dimensional matrix for storing $Z_{i-1,p-1}$, $Z''_{j-1,q-1}$ and $Z'_{j+1,q+1}$, which requires $O(mn)$ space. Since the total number of bases and base pairs in structure A (B, respectively) is at most m (n, respectively), we use a two-dimensional matrix to store $Prob((i,j) \sim (p,q))$ which also requires $O(mn)$ space. Thus, the space complexity of the algorithm is $O(mn)$. Notice that the time complexity derived here is a very pessimistic upper bound since in calculating the partition functions, some base pairs are ignored as described in the previous subsections. During our experiments, we tested over 200 alignments and the running times of our algorithm ranged from 16 ms to roughly 7 minutes, where the lengths of the aligned structures ranged from 22 nt to 1,553 nt.

4.2.6 Extension to Multiple Alignment

Our pairwise alignment method can be extended to align multiple RNA pseudoknot structures by utilizing a guide tree. Specifically, we treat each structure as a cluster and use the expected accuracy defined in Equation (4.17) as the measure to determine the similarity of two structures or clusters. Initially, we merge two RNA structures that are most similar into one cluster. Subsequently, we merge two clusters that are most similar into a larger cluster using the agglomerative hierarchical clustering algorithm [83], where the similarity of two clusters is calculated by the average linkage algorithm [83].

An alignment of two clusters is actually an alignment of two profiles, where each cluster is treated as a profile. Initially, each profile contains a single RNA pseudoknot structure. As the guide tree grows, a profile may contain multiple RNA pseudoknot structures; more precisely, the profile is a multiple alignment of these RNA structures. A single base of a profile is a column of the profile where the column contains single bases or gaps; a base pair of a profile includes two columns of the profile where the left column contains left bases or gaps and the right column contains corresponding right bases or gaps, and left bases and corresponding right bases form base pairs.

Suppose we want to align profile A' and profile B', which amounts to aligning two multiple alignments. Let R (S, respectively) be an RNA pseudoknot structure in profile A' (B', respectively) and let (i, j) ((p, q), respectively) be a base pair of R (S, respectively). Let (i', j') represent a base pair of profile A' and let (p', q') represent a base pair of profile B'. We use $(i, j) \in (i', j')$ ($(p, q) \in (p', q')$, respectively) to represent that (i, j) ((p, q), respectively) occurs in the column(s) of base pair (i', j') ((p', q'), respectively) of profile A' (B', respectively). Equation (4.18) shows how to calculate $Prob'((i', j') \sim (p', q'))$, which represents the transformed probability of aligning base pair (i', j') of profile A' with base pair (p', q') of profile B'.

$$Prob'((i', j') \sim (p', q')) = \frac{\sum_{(i,j)\in(i',j'),(p,q)\in(p',q')} Prob((i, j) \sim (p, q))}{|A'||B'|} \tag{4.18}$$

Here, $Prob((i, j) \lim (p, q))$ is defined in Equation (4.16), $|A'|$represents the number of RNA pseudoknot structures in profile or cluster A', and $|B'|$ represents the number of RNA pseudoknot structures in profile or cluster B'.

The multiple alignment algorithm can now be summarized as follows. The input of the algorithm is a set SS of RNA pseudoknot structures. For every two structures A and B in SS, we calculate their posterior log-odds score matrix as described in the Calculation of posterior log-odds scores subsection. After all the posterior log-odds score matrices are calculated, we compute the expected accuracy $\overline{Accu}(a_{A,B})$ as defined in Equation (4.17) where $a_{A,B}$ is a (sub)optimal alignment, found by the heuristic described in the Pairwise alignment subsection, between structure A and structure B. We use the expected accuracy or similarity values to construct the guide tree for the set SS, to determine the order in which two structures or profiles are aligned. To align two profiles A' and B', we use the same heuristic as described in the Pairwise alignment subsection, with the transformed probabilities $Prob'((i',j') \sim (p',q'))$ defined in Equation (4.18) replacing the posterior probabilities $Prob((i,j) \sim (p,q))$ of structures A and B defined in Equation (4.16). The time complexity of this multiple alignment algorithm is $O(k^2n^4)$ where k is the number of structures in the alignment and n is the maximum of the lengths of the structures; the space complexity of the algorithm is $O(k^2n^2)$.

CHAPTER 5

EXPERIMENTS FOR PSEUDOKNOT ALIGNMENT

5.1 Experimental Design

5.1.1 Datasets

RKalign is implemented in Java. The program accepts as input two pseudoknotted RNAs where each RNA has both sequence data (i.e., nucleotides or bases) and structure data (i.e., base pairs), and produces as output an alignment between the two pseudoknotted RNAs. Popular benchmark datasets such as BRAliBase [74], RNase P [75] and Rfam [76] are not suitable for testing RKalign. The reason is that BRAliBase contains only sequence information, while RNase P and Rfam contain consensus structures of multiple sequence alignments rather than alignments of individual structures of RNAs. As a consequence, we manually created two datasets for testing RKalign and comparing it with related alignment methods.

The first dataset, denoted Dataset1, contains 38 RNA pseudoknot structures chosen from the PDB [46] and RNA STRAND [45] (see Table 5.1 - 5.5). These RNAs were selected in such a way that they have a wide range of sequence lengths. Each three-dimensional (3D) molecule in this dataset was taken from the PDB.

The secondary structure of the 3D molecule was obtained with RNAview [77], retrieved from RNA STRAND. The second dataset, denoted Dataset2, contains 36 RNA pseudoknot structures chosen from PseudoBase [47, 48] (see Table 5.6 - 5.8).

As in the first dataset, the RNA molecules in the second dataset have a wide range of sequence lengths. The pseudoknots in these datasets can be broadly classified into two types: H-type and recursive pseudoknots [38, 59]. There are 12 H-type pseudoknots and 26 recursive pseudoknots in Dataset1. There are 22 H-type pseudoknots and 14 recursive pseudoknots in Dataset2.

Table 5.1 Selected RNA Pseudoknot Structures from PDB and RNA STRAND to Perform Alignment Quality Experiments.

The RNA pseudoknot structures that are selected from the PDB and RNA STRAND are listed here. We use this dataset to evaluate the performance of RKalign, CARNA, RNA Sampler, DAFS, R3D Align and RASS. Each three-dimensional (3D) molecule (e.g. 2AW7) is retrieved from the PDB and used by R3D Align and RASS. The secondary structure of the 3D molecule is obtained with RNAview and stored in RNA STRAND (e.g. PDB_00935), which is used by RKalign and CARNA.

PDB ID	RNA STRAND ID	Molecule Name	RNA Type	Length
2AW7	PDB_00935	Crystal structure of the bacterial ribosome from Escherichia coli at 3.5 A resolution	16S rRNA	1530
2I2P	PDB_01120	Crystal structure of ribosome with messenger RNA and the anticodon stem-loop of P-site tRNA	16S rRNA	1553
2B57	PDB_00994	Guanine Riboswitch C74U mutant bound to 2,6-diaminopurine	Synthetic RNA	65
1FJG	PDB_00408	Structure of the Thermus Thermophilus 30S ribosomal subunit in complex with the Antibiotics Streptomycin, Spectinomycin, and Paromomycin	16S rRNA	1513
2FD0	PDB_01049	HIV-1 DIS kissing-loop in complex with lividomycin	Synthetic RNA	46
2D19	PDB_00988	Solution RNA structure of loop region of the HIV-1 dimerization initiation site in the kissing-loop dimer	Synthetic RNA	34

Table 5.2 Selected RNA Pseudoknot Structures from PDB and RNA STRAND to Perform Alignment Quality Experiments. *Part 2*

PDB ID	RNA STRAND ID	Molecule Name	RNA Type	Length
1XP7	PDB_00816	HIV-1 subtype F genomic RNA Dimerization Initiation Site	Synthetic RNA	46
437D	PDB_00269	Crystal structure of an RNA pseudoknot from beet western yellow virus involved in ribosomal frameshifting	Other rRNA	28
2G1W	PDB_01059	NMR structure of the Aquifex aeolicus tmRNA pseudoknot PK1	Synthetic RNA	22
1L2X	PDB_00138	Atomic resolution crystal structure of a viral RNA pseudoknot	Viral & Phage	28
3B4A	PDB_01300	T. tengcongensis glmS ribozyme with G40A mutation, bound to glucosamine-6-phosphate	Other Ribozyme	142
1FFZ	PDB_00403	Large ribosomal subunit complexed with R(CC)-Da-Puromycin	Other rRNA	500
1KAJ	PDB_00124	Conformation of an RNA pseudoknot from mouse mammary tumor virus, NMR, 1 structure	Synthetic RNA	32
1VC5	PDB_00764	Crystal structure of the Wild Type Hepatitis Delta Virus Genomic Ribozyme Precursor, in EDTA solution	Other Ribozyme	70

Table 5.3 Selected RNA Pseudoknot Structures from PDB and RNA STRAND to Perform Alignment Quality Experiments. *Part 3*

PDB ID	RNA STRAND ID	Molecule Name	RNA Type	Length
2FCY	PDB_01047	HIV-1 DIS kissing-loop in complex with Neomycin	Synthetic RNA	46
1BAU	PDB_00018	NMR structure of the dimer initiation complex of HIV-1 Genomic RNA, minimized average structure	Synthetic RNA	46
1KPZ	PDB_00135	PEMV-1 P1-P2 frameshifting pseudoknot regularized average structure	Synthetic RNA	28
1E95	PDB_00041	Solution structure of the pseudoknot of SRV-1 RNA, involved in ribosomal frameshifting	Other RNA	36
1DRZ	PDB_00346	U1A Spliceosomal Protein/Hepatitis Delta Virus Genomic Ribozyme complex	Other Ribozyme	72
1SJ3	PDB_00714	Hepatitis Delta Virus Genomic Ribozyme Precursor, with Mg2+ bound	Other Ribozyme	73
1JGO	PDB_00484	The path of messenger RNA through the ribosome	Other RNA	232
2B8S	PDB_00951	Structure of HIV-1(MAL) genomic RNA DIS	Synthetic RNA	46
1RNK	PDB_00209	The structure of an RNA pseudoknot that causes efficient frameshifting in mouse mammary tumor virus	Synthetic RNA	34

Table 5.4 Selected RNA Pseudoknot Structures from PDB and RNA STRAND to Perform Alignment Quality Experiments. *Part 4*

PDB ID	RNA STRAND ID	Molecule Name	RNA Type	Length
1SJF	PDB_00716	Crystal structure of the Hepatitis Delta Virus Genomic Ribozyme Precursor, with C75U mutation, in Cobalt Hexammine solution	Other Ribozyme	74
2AP5	PDB_00931	Solution structure of the C27A ScYLV P1-P2 frameshifting pseudoknot, average structure	Synthetic RNA	28
1YG4	PDB_00843	Solution structure of the ScYLV P1-P2 frameshifting pseudoknot, regularized average structure	Synthetic RNA	28
1F27	PDB_00053	Crystal structure of a Biotin-Binding RNA pseudoknot	Synthetic RNA	30
1KPD	PDB_00133	A mutant RNA pseudoknot that promotes ribosomal frameshifting in mouse mammary tumor virus, NMR, minimized average structure	Other rRNA	32
1IBK	PDB_00463	Structure of the Thermus Thermophilus 30S ribosomal subunit in complex with the antibiotic paromomycin	16S rRNA	1512
1E8O	PDB_00352	Core of the ALU domain of the Mammalian SRP	Synthetic RNA	50
2NUG	PDB_01165	Crystal structure of RNase III from Aquifex aeolicus complexed with ds-RNA at 1.7-Angstrom resolution	Synthetic RNA	44

Table 5.5 Selected RNA Pseudoknot Structures from PDB and RNA STRAND to Perform Alignment Quality Experiments. *Part 5*

PDB ID	RNA STRAND ID	Molecule Name	RNA Type	Length
1Y3O	PDB_00831	HIV-1 DIS RNA subtype F-Mn soaked	Synthetic RNA	46
1JGQ	PDB_00486	The path of messenger RNA through the ribosome	Other RNA	229
3B4C	PDB_01302	T. tengcongensis glmS ribozyme bound to glucosamine-6-phosphate and a substrate RNA with a 2'5'-phosphodiester linkage	Other Ribozyme	139
2TPK	PDB_00243	An investigation of the structure of the pseudoknot within gene 32 messenger RNA of Bacteriophage T2 using heteronuclear NMR methods	Viral & Phage	36
2OOM	PDB_01194	NMR structure of a kissing complex formed between the TAR RNA element of HIV-1 and a LNA/RNA aptamer	Synthetic RNA	32
1FG0	PDB_00404	Large ribosomal subunit complexed with A 13 BP Minihelix-Puromycin compound	Other rRNA	499
2F4X	PDB_01040	NMR solution of HIV-1 Lai kissing complex	Synthetic RNA	48

Table 5.6 Selected RNA Pseudoknot Structures from PseudoBase to Perform Alignment Quality Experiments.

The RNA pseudoknot structures that are selected from PseudoBase are listed below. We use this dataset to evaluate the performance of RKalign, CARNA, RNA Sampler and DAFS.

PKB Number	Abbreviation	Organism	RNA Type	Length
PKB309	IFNG_PK_B_Taurus	Bos taurus (cow)	mRNA	145
PKB106	IBV	infectious bronchitis virus	Viral frameshift	57
PKB121	STNV1_PK1	satellite tobacco necrosis virus 1	Viral 3 UTR	26
PKB122	STNV1_PK2	satellite tobacco necrosis virus 1	Viral 3 UTR	31
PKB123	STNV1_PK3	satellite tobacco necrosis virus 1	Viral 3 UTR	26
PKB124	STNV2_PK1	satellite tobacco necrosis virus 1	Viral 3 UTR	29
PKB125	STNV2_PK2	satellite tobacco necrosis virus 1	Viral 3 UTR	25
PKB126	STNV2_PK3	satellite tobacco necrosis virus 1	Viral 3 UTR	27
PKB127	EAV	equine arteritis virus	Viral frameshift	56
PKB128	BEV	Berne virus	Viral frameshift	59

Table 5.7 Selected RNA Pseudoknot Structures from PseudoBase to Perform Alignment Quality Experiments. *Part 2*

PKB Number	Abbreviation	Organism	RNA Type	Length
PKB131	NGF-H1	-	Aptamers	48
PKB132	NGF-L2	-	Aptamers	49
PKB133	NGF-L6	-	Aptamers	48
PKB144	ORSV-S1_PKbulge1	odontoglossum ringspot virus	Viral tRNA-like	71
PKB145	ORSV-S1_PKbulge2	odontoglossum ringspot virus	Viral tRNA-like	58
PKB146	ORSV-S1_PKbulge3	odontoglossum ringspot virus	Viral tRNA-like	50
PKB158	TRV-PSG2_PK1	tobacco rattle virus, strain PSG	Viral others	28
PKB159	TRV-PSG2_PK2	tobacco rattle virus, strain PSG	Viral others	25
PKB160	TRV-PSG2_PK3	tobacco rattle virus, strain PSG	Viral others	32
PKB161	TRV-PSG2_PK4	tobacco rattle virus, strain PSG	Viral others	24
PKB162	TRV-PSG2_PK5	tobacco rattle virus, strain PSG	Viral others	35
PKB2	BWYV	Beet western-yellows virus	Viral ribosomal frameshifting	50
PKB240	BChV	beet chlorosis virus	Viral frameshift	41

Table 5.8 Selected RNA Pseudoknot Structures from PseudoBase to Perform Alignment Quality Experiments. *Part 3*

PKB Number	Abbreviation	Organism	RNA Type	Length
PKB254	SARS-CoV	SARS coronavirus	Viral frameshift	82
PKB258	Hs_Ma3	Homo sapiens	Viral frameshift	60
PKB3	EIAV	Equine infectious anemic virus	Viral ribosomal frameshifting	54
PKB310	IFNG_PK_C_familiaris	Canis familiaris (dog)	mRNA	130
PKB311	IFNG_PK_C_jacchus	Callitrix jacchus (marmoset)	mRNA	120
PKB313	IFNG_PK_S_scrofa	Sus scrofa (pig)	mRNA	130
PKB346	KUNV	West Nile virus, Kunijn subtype	Viral frameshift	75
PKB347	WNV	West Nile virus	Viral frameshift	75
PKB348	JEV	Japanese encephalitis virus	Viral frameshift	77
PKB4	FIV	Feline immunodeficiency virus	Viral ribosomal frameshifting	50
PKB44	CABYV	cucurbit aphid-borne yellows virus	Viral frameshift	39
PKB46	BYDV-NY-RPV	barley yellow dwarf virus	Viral frameshift	39

5.1.2 Alignment Quality

A good structural alignment tends to align a base pair with another base pair rather than with two single bases [65, 66]. We therefore use the base_mismatch ratio to assess the quality of an alignment. A base mismatch occurs when a single base is aligned with the left or right base of a base pair or when a nucleotide is aligned to a gap. The base_mismatch ratio of an alignment $a_{A,B}$ between structure A and structure B is defined as the number of base mismatches in $a_{A,B}$ divided by the total number of alignment lines in $a_{A,B}$, multiplied by 100%. Statistically significant performance differences between alignment methods are calculated using Wilcoxon signed rank tests [78], which are commonly used for comparing alignment programs [79, 80, 81]. As in [79, 80, 81] we consider p-values below 0.05 to be statistically significant.

5.2 Experimental Results

We conducted a series of experiments to evaluate the performance of RKalign and compare it with related methods, where the performance measure used was the base_mismatch ratio. In the first experiment, we selected 106 pairs of RNA pseudoknot structures from Dataset1 and applied our method to aligning the two molecules in each pair. The two molecules in a pair belonged to the same pseudoknot type, as it is biologically meaningless to align RNA molecules that lack consensus [65, 82]. The average base_mismatch ratio calculated by RKalign for the selected 106 pairs was 34.84%, compared to the average base_mismatch ratio, 78.53%, for all pairs of molecules in Dataset1.

In addition, we also ran CARNA [44], RNA Sampler [41], DAFS [42], R3D Align [84] and RASS [67] on the 106 pairs of molecules. The CARNA tool was chosen because an option of the tool is closely related to RKalign, both of which can align known pseudoknot structures. RNA Sampler and DAFS were chosen because they are widely used tools capable of simultaneously folding and aligning RNA sequences considering pseudoknots without known structures. When running these two tools, the structure information in Dataset1 was ignored and only the sequence data was used as the input of the tools. R3D Align and RASS were chosen because they are state-of-the-art RNA 3D alignment programs; furthermore, like RKalign, R3D Align and RASS output the entire alignment of two RNA structures. Since R3D Align and RASS accept 3D structures as input whereas RKalign and CARNA accept bases and base pairs as input, we used the PDB files in Dataset1 as the input for R3D Align and RASS while using the corresponding RNA STRAND entries in Dataset1 as the input for RKalign and CARNA.

Figure 5.1 presents histograms for the base_mismatch ratios of the six tools. Figure 5.2 presents boxplots for the base_mismatch ratios of the six tools. These figures show the distribution of the base_mismatch ratios for the six tools. RKalign and CARNA were not statistically different according to a Wilcoxon signed rank test ($p > 0.05$). On the other hand, they both were significantly better than the other four tools according to the Wilcoxon signed rank test ($p < 0.05$). It was observed that the structures predicted by RNA Sampler and DAFS might not be correct. Consequently, there were many base mismatches with respect to the known structures in the alignments.

For example, consider Figure 5.3, which shows the alignment result of DAFS, R3D Align and RKalign, respectively on two pseudoknot structures with PDB IDs 1L2X and 1RNK. The base_mismatch ratio of DAFS (R3D Align, RKalign, respectively) is 57.14% (67.39%, 27.78%, respectively). Figure 5.3(a) shows the predicted common secondary structure and the alignment produced by DAFS. Figure 5.3(b) shows the known secondary structures of 1L2X and 1RNK and the alignment produced by DAFS where the known secondary structures are used to calculate the base_mismatch ratios. Figure 5.3(c) shows the alignment obtained from R3D Align and Figure 5.3(d) shows the alignment obtained from RKalign. It can be seen that

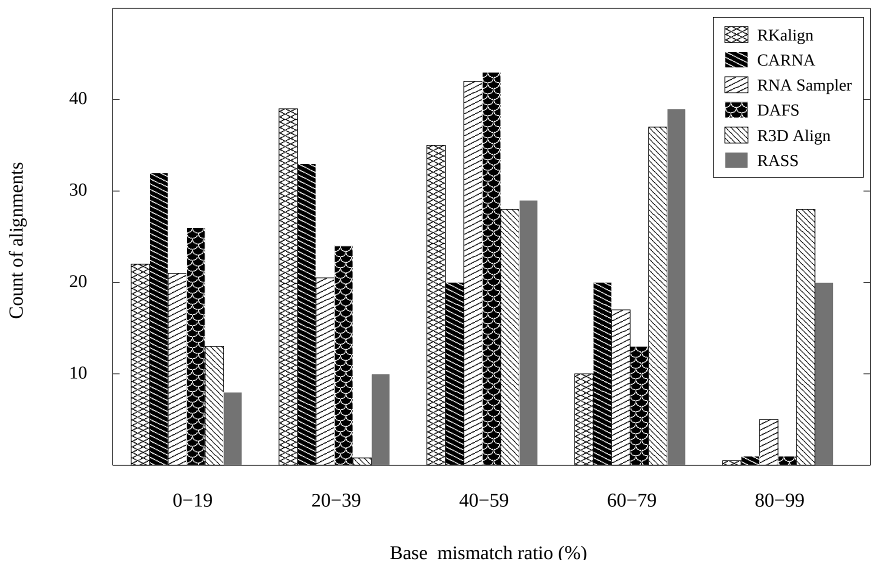


Figure 5.1 Histograms for the base_mismatch ratios of the alignments produced by RKalign, CARNA, RNA Sampler, DAFS, R3D Align and RASS, respectively on the 106 structure pairs selected from Dataset1. Buckets on the x-axis are defined by equal-width ranges 0 to19, 20 to 39, 40 to 59, 60 to 79, and 80 to 99 (rounded down to the nearest whole number). These histograms show the distribution of the base_mismatch ratios of the alignments produced by the six tools.

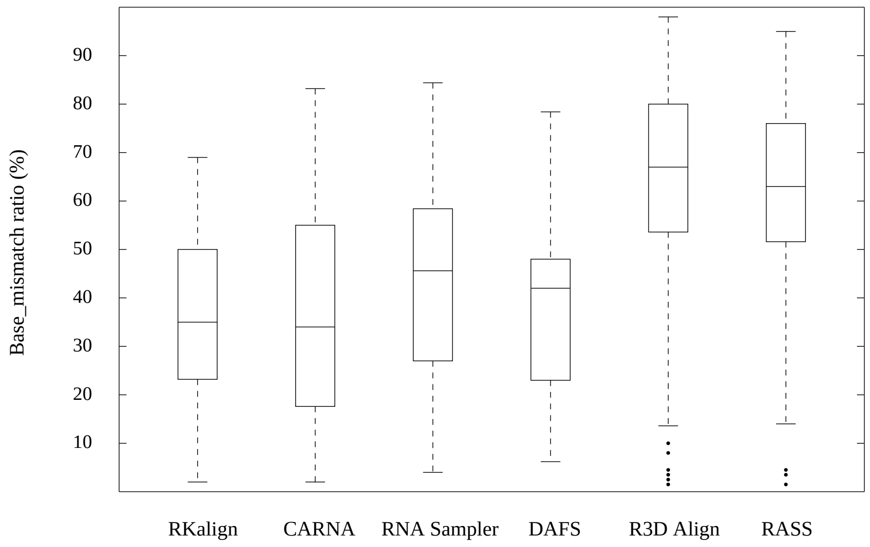


Figure 5.2 Boxplots for the base_mismatch ratios of the alignments produced by RKalign, CARNA, RNA Sampler, DAFS, R3D Align and RASS, respectively on the 106 structure pairs selected from Dataset1. The median of the base_mismatch ratios yielded by RKalign (CARNA, RNA Sampler, DAFS, R3D Align, RASS, respectively) is 35.29% (34.38%, 45.86%, 41.99%, 67.57%, 63.29%, respectively).

the predicted common secondary structure in Figure 5.3(a) is quite different from the known secondary structure of 1L2X. Refer to Figure 5.3(b). The base G (G, C, C, A, A and A, respectively) at position 1 (2, 8, 22, 23, 24 and 25, respectively) in 1L2X is a single base, which is aligned with the left or right base of some base pair in 1RNK, leading to base mismatches in the alignment. Similarly, the base G (A, C, A and U, respectively) at position 7 (20, 21, 24 and 34, respectively) in 1RNK is a single base, which is aligned with the left or right base of some base pair in 1L2X. R3D Align doesnt align the pseudoknot structures well either, due to the fact that many gaps are involved in the alignment (Figure 5.3(c)). In this example, RKalign produces the best alignment (Figure 5.3(d)). It should be pointed out, however, that 3D alignment programs such as R3D Align are general-purpose structure alignment tools capable of comparing two RNA 3D molecules with diverse tertiary motifs, whereas RKalign focuses on secondary structures with pseudoknots only.

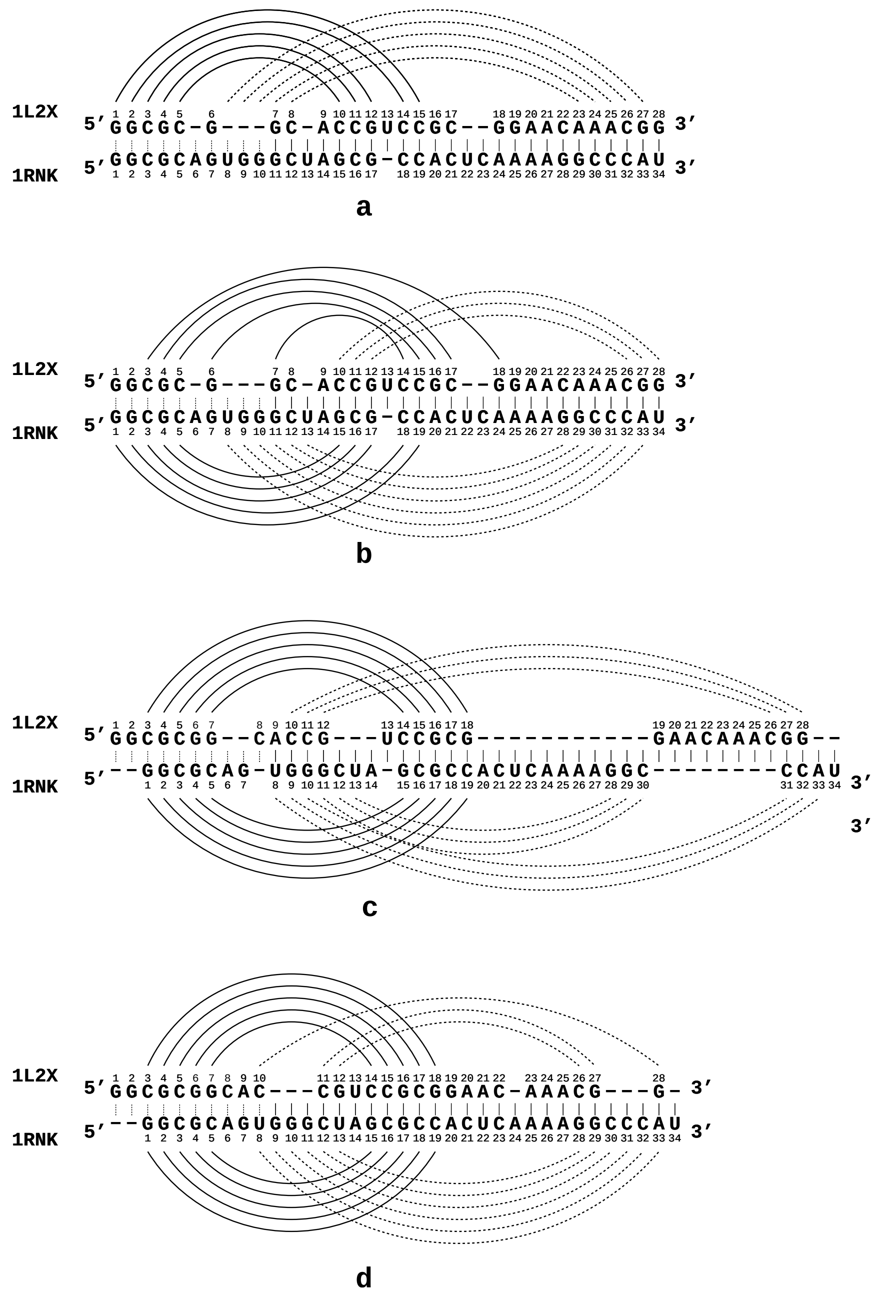

1L2X
5' GGCGC-G---GC-ACCGUCCGC--GGAACAAACGG 3'
1RNK
5' GGCGCAGUGGGCUAGCG-CCACUCAAAAGGCCCAU 3'
a
1L2X
5' GGCGC-G---GC-ACCGUCCGC--GGAACAAACGG 3'
1RNK
5' GGCGCAGUGGGCUAGCG-CCACUCAAAAGGCCCAU 3'
b
1L2X
5' GGCGCGG--CACCG---UCCGCG-----------GAACAAACGG--
1RNK
5' --GGCGCAG-UGGGCUA-GCGCCACUCAAAAGGC--------CCAU 3'
3'
c
1L2X
5' GGCGCGGCAC---CGUCCGCGGAAC-AAACG---G- 3'
1RNK
5' --GGCGCAGUGGGCUAGCGCCACUCAAAAGGCCCAU 3'
d

Figure 5.3 (a) The predicted common secondary structure and the alignment produced by DAFS between two pseudoknot structures with PDB IDs 1L2X and 1RNK, respectively. (b) The known secondary structures of 1L2X and 1RNK and the alignment produced by DAFS. (c) The known secondary structures of 1L2X and 1RNK and the alignment produced by R3D Align. (d) The known secondary structures of 1L2X and 1RNK and the alignment produced by RKalign. The base_mismatch ratio of DAFS (R3D Align, RKalign, respectively) is 57.14% (67.39%, 27.78%, respectively), where the base_mismatch ratios are calculated using the known secondary structures. RKalign produces the best alignment with respect to the known secondary structures of 1L2X and 1RNK.

In the second experiment, we compared RKalign, CARNA, RNA Sampler and DAFS using the RNA structures in Dataset2. As in the first experiment, we selected 124 pairs of molecules from Dataset2 where the two molecules in a pair belonged to the same pseudoknot type. The average base_mismatch ratio calculated by RKalign for the selected 124 pairs was 35.89%, compared to the average base_mismatch ratio, 81.56%, for all pairs of molecules in Dataset2. We applied each of the four tools to the molecules to produce 124 pairwise alignments.

Figure 5.4 presents histograms for the base_mismatch ratios of the four tools. Figure 5.5 presents boxplots for the base_mismatch ratios of the four tools. These figures show the distribution of the base_mismatch ratios for the four tools. RKalign and CARNA were not statistically different (Wilcoxon signed rank test, $p > 0.05$);both tools were significantly better than RNA Sampler and DAFS (Wilcoxon signed rank test, $p < 0.05$).

We also tested our algorithm for multiple alignment by selecting 30 groups each having 3, 4, or 5 pseudoknot structures of the same type from the datasets used in this study, and by performing multiple alignment in each group. We then compared our algorithm with three related methods: CARNA [44], RNA Sampler [41] and DAFS [42]. The base_mismatch ratio of a multiple alignment MA is defined as the sum of base_mismatch ratios of all pairs of structures in MA divided by the total number of structure pairs in MA, multiplied by 100%. The average base_mismatch ratio of RKalign (CARNA, RNA Sampler, DAFS, respectively) was 26.01% (25.79%, 32.15%, 29.23%, respectively). RKalign and CARNA were not statistically different (Wilcoxon signed rank test, $p > 0.05$); the two methods were significantly better than RNA Sampler and DAFS (Wilcoxon signed rank test, $p < 0.05$).

Based on the above experimental results, there is no statistically significant difference between RKalign and CARNA in terms of base_mismatch ratios. As described in [34, 85], a good pseudoknot alignment has many matched stems and

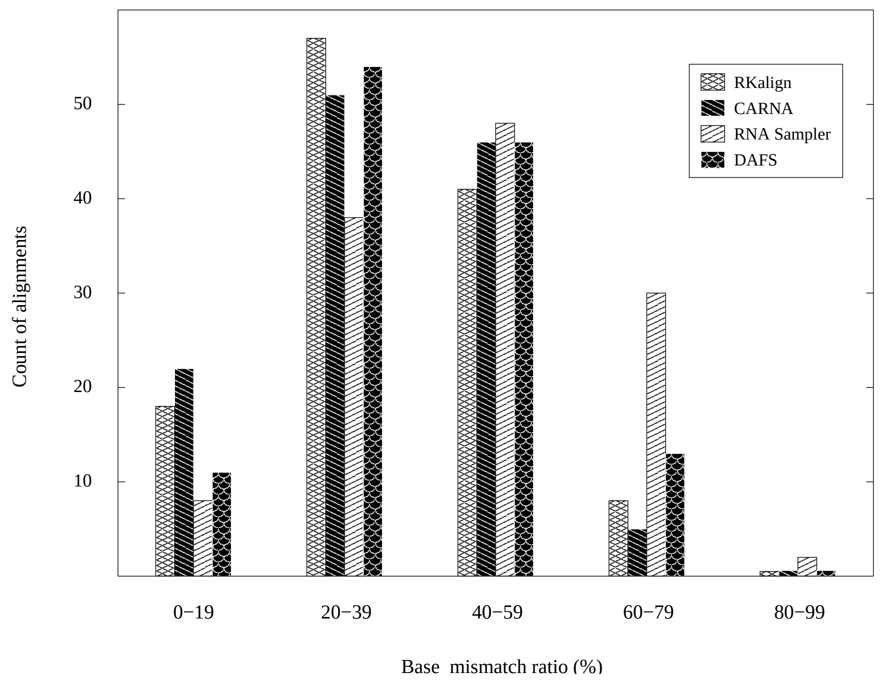


Figure 5.4 Histograms for the base_mismatch ratios of the alignments produced by RKalign, CARNA, RNA Sampler and DAFS, respectively on the 124 structure pairs selected from Dataset2. Buckets on the x-axis are defined by equal-width ranges 0 to19, 20 to 39, 40 to 59, 60 to 79, and 80 to 99 (rounded down to the nearest whole number). These histograms show the distribution of the base_mismatch ratios of the alignments produced by the four tools.

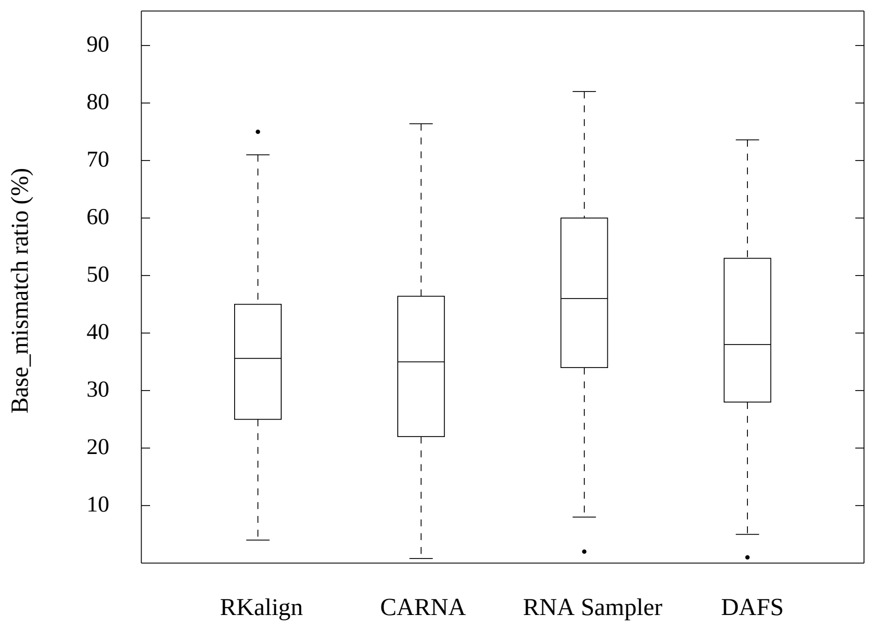


Figure 5.5 Boxplots for the base_mismatch ratios of the alignments produced by RKalign, CARNA, RNA Sampler and DAFS, respectively on the 124 structure pairs selected from Dataset2. The median of the base_mismatch ratios yielded by RKalign (CARNA, RNA Sampler, DAFS, respectively) is 36.31% (35.81%, 45.83%, 39.29%, respectively).

few mismatched stems. In the last experiment, we further compared RKalign with CARNA by examining how they match stems in two pseudoknot structures A and B. A stem $S_A \in A$ is said to match a stem $S_B \in B$ if both of the following conditions are satisfied:

(1) S_A, S_B are aligned together and they cannot be aligned with other stems.

(2) For every base $x \in S_A$ and base pair $y \in S_B$, a base of x is aligned with a base of y if and only if the other base of x is aligned with the other base of y.

If any of the conditions is violated, there is a stem mismatch between S_A and S_B. The stem_mismatch ratio of an alignment $a_{A,B}$ between structure A and structure B is defined as $(1 - M)$ where M is the number of matched stems in $a_{A,B}$ divided by the total number of stems in A and B, multiplied by 100%.

Figure 5.6 shows the average stem_mismatch ratios of RKalign and CARNA obtained by running the tools on Dataset1 and Dataset2, respectively. RKalign was significantly better than CARNA (Wilcoxon signed rank test, $p < 0.05$). A close look at the alignment results of CARNA reveals why this happens. For instance, consider Figure 5.7(a), which shows how CARNA aligns the two PDB structures, 1L2X and 1RNK, given in Figure 5.3. In Figure 5.7(a) we use the arches with solid lines to demonstrate the 1st stem of 1L2X and 1RNK, and we use the arches with dash lines to demonstrate the 2nd stem of 1L2X and 1RNK. Figure 5.7(b) illustrates mismatched stems in the alignment in Figure 5.7(a). Figure 5.7(c) shows the alignment of the same molecules, 1L2X and 1RNK, produced by RKalign where there is no stem mismatch. Refer to Figure 5.7(b). In 1L2X, the base G at position 7 and the base C at position 14 form a base pair in its 1st stem. In 1RNK, the base U at position 13 and the base G at position 28 form a base pair in its 2nd stem.

Now, observe that the base C at position 14 in 1L2X is aligned with the base U at position 13 in 1RNK, but the base G at position 7 in 1L2X is not aligned with

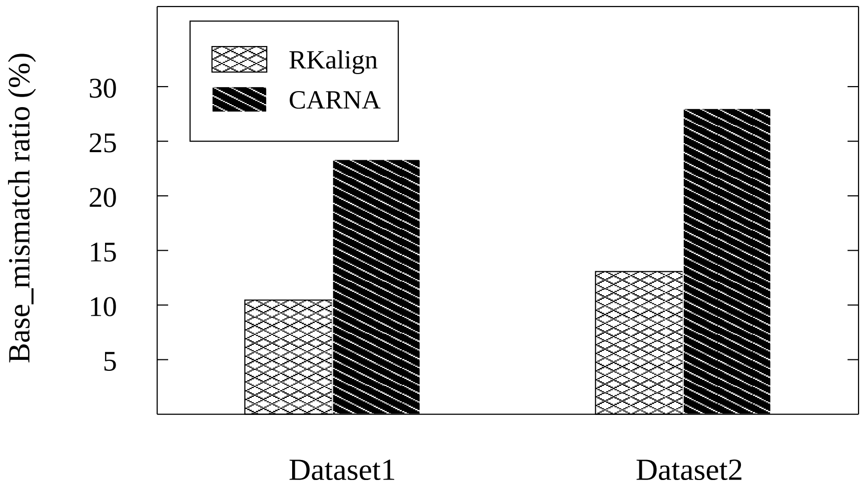
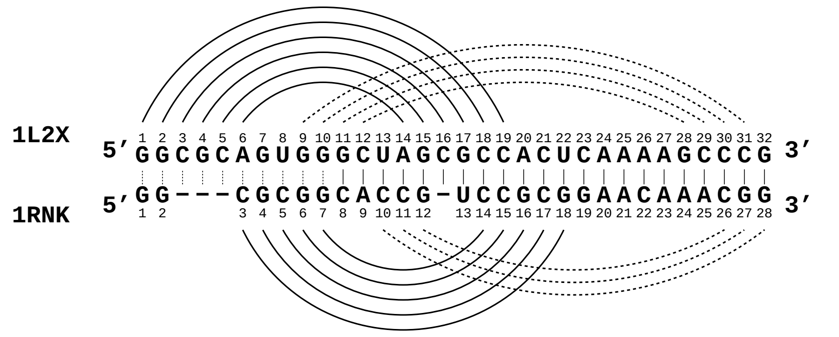
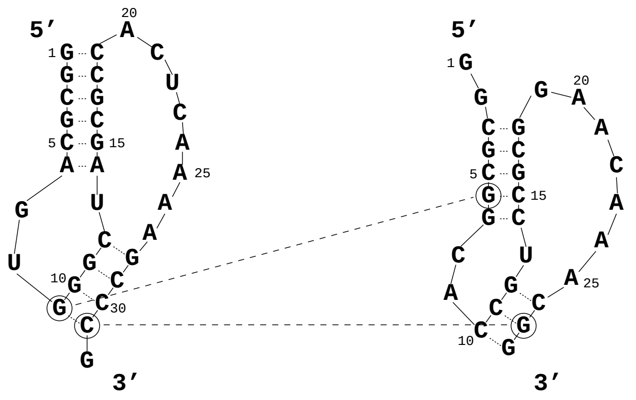


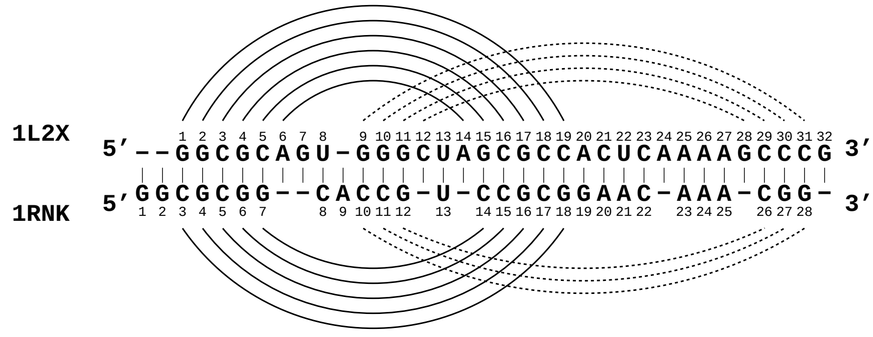
Figure 5.6 Average stem_mismatch ratios of the alignments produced by RKalign and CARNA on the 106 structure pairs selected from Dataset1 and the 124 structure pairs selected from Dataset2, respectively. Error bars are included in the figure. For Dataset1, the average stem_mismatch ratio of RKalign is 10.8% and the average stem_mismatch ratio of CARNA is 23.5%. For Dataset2, the average stem_mismatch ratio of RKalign is 13.1% and the average stem_mismatch ratio of CARNA is 28.9%. RKalign performs significantly better than CARNA in terms of stem_mismatch ratios (Wilcoxon signed rank test, $p < 0.05$).


1L2X
5′ GGCGCAGUGGGCUAGCGCCACUCAAAAGCCCG 3′
5′ GG---CGCGGCACCG-UCCGCGGAACAAACGG 3′
1RNK

a



b


1L2X
5′ --GGCGCAGU-GGGCUAGCGCCACUCAAAAGCCCG 3′
5′ GGCGCGG--CACCG-U-CCGCGGAAC-AAA-CGG- 3′
1RNK

c

Figure 5.7 (a) The alignment of two pseudoknot structures with PDB IDs 1L2X and 1RNK, respectively produced by CARNA. (b) Illustration of mismatched stems in the alignment produced by CARNA. There is a stem mismatch between the 1st stem of 1L2X and the 2nd stem of 1RNK, a situation that is not favoured when performing pseudoknot alignment. (c) The alignment of 1L2X and 1RNK produced by RKalign where there is no stem mismatch.

the base G at position 28 in 1RNK; instead the base G at position 7 in 1L2X is aligned with the single base G at position 7 in 1RNK. Thus, there is a stem mismatch between the 1st stem of 1L2X and the 2nd stem of 1RNK, a situation that is not favoured when performing pseudoknot alignment [34, 85]. This situation occurs more frequently in CARNA alignment results than in RKalign alignment results. As a consequence, CARNA has much higher stem_mismatch ratios than RKalign.

Comparing Figure 5.7(a) and Figure 5.7(c), we also note that the overall alignments produced by CARNA and RKalign are quite different. In Figure 5.7(a) in which the alignment from CARNA is shown, the base G at position 6 and the base C at position 15 form a base pair in 1L2X. The base A at position 6 in 1RNK is a single base. It can be seen that the base G at position 6 in 1L2X is aligned with the base A at position 6 in 1RNK, i.e., a base pair is aligned with a single base. In addition, in 1L2X the base C at position 14, which is the right base of a base pair, is aligned with the base U at position 13, which is the left base of a base pair in 1RNK. Aligning a base pair with a single base, and aligning the right base of a base pair with the left base of another base pair, occur in CARNA's output shown in Figure 5.7(a), but do not occur in RKalign's output shown in Figure 5.7(c). On the other hand, there are more gaps in RKalign's output than in CARNA's output; specifically there are 10 gaps in RKalign's output shown in Figure 5.7(c) compared to 8 gaps in CARNA's output shown in Figure 5.7(a).

5.3 Comparison with Related Methods

RKalign is designed to align known RNA pseudoknot structures. A different approach is to simultaneously fold and align RNA sequences without known structures, as adopted by several existing tools [41, 42, 43]. When the structure information is not available, this simultaneous folding and alignment approach is the best. However, when pseudoknot structures already exist, RKalign performs significantly better than the existing tools, as observed in our experiments. The reason is that the structures predicted by these tools may not be correct. As a consequence, there are many base mismatches with respect to the known structures in the resulting alignments.

Pseudoknots are part of RNA tertiary motifs [32]. There are 3D alignment programs that can compare RNA tertiary structures including pseudoknots [66, 67]. These programs consider the entire RNA 3D structure as a whole, and accept PDB files with 3D coordinates as input. As shown in our experiments, when considering and aligning secondary structures with pseudoknots, RKalign outperforms the 3D alignment programs. It should be noted, however, that the 3D alignment programs are general-purpose structure alignment tools capable of comparing two RNA 3D molecules with diverse tertiary motifs, whereas RKalign deals with secondary structures with pseudoknots only.

While the work reported here focuses on pseudoknot alignment, it can also be applied to RNA secondary structures without pseudoknots. We applied RKalign to 102 pairs of pseudoknot-free structures taken from RNA STARND where the pseudoknot-free structures belonged to Rfam [76] (see Table 5.9, Table 5.10 and Table 5.11).

We compared RKalign with three other tools: CARNA [44], RNAforester [71] and RSmatch [70]. RNAforester, included in the widely used Vienna RNA package [86], is a versatile RNA structure alignment tool. Like RKalign and CARNA, an option of RNAforester is able to accept as input two RNA molecules with both sequence data (nucleotides or bases) and secondary structure data (base pairs), and produce as output the global alignment of the two molecules. However, a limitation of RNAforester is that the aligned secondary structures cannot contain pseudoknots. RSmatch is similar to RNAforester, sharing the same limitation. Our experimental results showed that the average base_mismatch ratio for RKalign (CARNA, RNAforester, RSmatch, respectively) was 43.52%

Table 5.9 Selected RNA Pseudoknot-free Structures from Rfam and RNA STRAND to Perform Alignment Quality Experiments.

The RNA pseudoknot-free structures that are selected from Rfam and RNA STRAND are listed below. We use this dataset to evaluate the performance of RKalign, CARNA, RNAforester and RSmatch.

Rfam ID	RNA STRAND ID	Molecule Name	RNA Type	Length
RF00008 AJ005299.1/335-282	RFA_00396	Hammerhead ribozyme (type III), RF00008, AJ005299.1/282-335	Ham. Ribozyme	54
RF00008 AJ247113.1/53-134	RFA_00420	Hammerhead ribozyme (type III), RF00008, AJ247113.1/134-53	Ham. Ribozyme	82
RF00008 AJ247122.1/52-132	RFA_00423	Hammerhead ribozyme (type III), RF00008, AJ247122.1/132-52	Ham. Ribozyme	81
RF00008 AJ295018.1/1-58	RFA_00426	Hammerhead ribozyme (type III), RF00008, AJ295018.1/58-1	Ham. Ribozyme	58
RF00008 AJ536619.1/152-206	RFA_00430	Hammerhead ribozyme (type III), RF00008, AJ536619.1/206-152	Ham. Ribozyme	55
RF00008 Y14700.1/53-133	RFA_00450	Hammerhead ribozyme (type III), RF00008, Y14700.1/133-53	Ham. Ribozyme	81
RF00019 K01562.1/110-1	RFA_00582	Y RNA, RF00019, K01562.1/1-110	Y RNA	110
RF00019 K01564.1/77-1	RFA_00584	Y RNA, RF00019, K01564.1/1-77	Y RNA	77
RF00019 L15431.1/134-36	RFA_00585	Y RNA, RF00019, L15431.1/36-134	Y RNA	99

Table 5.10 Selected RNA Pseudoknot-free Structures from Rfam and RNA STRAND to Perform Alignment Quality Experiments. *Part 2*

Rfam ID	RNA STRAND ID	Molecule Name	RNA Type	Length
RF00025 M33461.1/346-140	RFA_00606	Ciliate telomerase RNA, RF00025, M33461.1/140-346	Cili. Telo. RNA	207
RF00025 U10565.1/238-50	RFA_00608	Ciliate telomerase RNA, RF00025, U10566.1/72-260	Cili. Telo. RNA	189
RF00025 U22353.1/206-53	RFA_00614	Ciliate telomerase RNA, RF00025, U22353.1/53-206	Cili. Telo. RNA	154
RF00025 U45435.1/283-69	RFA_00618	Ciliate telomerase RNA, RF00025, U45435.1/69-283	Cili. Telo. RNA	215
RF00109 AB169770.1/1705-1641	RFA_00640	Vimentin 3 prime UTR protein-binding region, RF00109, AB169770.1/1641-1705	Cis-reg. element	65
RF00109 AF447708.1/1826-1750	RFA_00642	Vimentin 3 prime UTR protein-binding region, RF00109, AF447708.1/1750-1826	Cis-reg. element	77
RF00109 BC053115.1/1568-1499	RFA_00645	Vimentin 3 prime UTR protein-binding region, RF00109, BC053115.1/1499-1568	Cis-reg. element	70
RF00109 M26251.1/2000-1935	RFA_00652	Vimentin 3 prime UTR protein-binding region, RF00109, M26251.1/1935-2000	Cis-reg. element	66
RF00163 X15620.1/62-19	RFA_00715	Hammerhead ribozyme (type I), RF00163, X15620.1/19-62	Ham. Ribozyme	44

Table 5.11 Selected RNA Pseudoknot-free Structures from Rfam and RNA STRAND to Perform Alignment Quality Experiments. *Part 3*

Rfam ID	RNA STRAND ID	Molecule Name	RNA Type	Length
RF00019 L15432.1/124-36	RFA_00586	Y RNA, RF00019, L15432.1/36-124	Y RNA	89
RF00019 L27537.1/96-1	RFA_00588	Y RNA, RF00019, L27537.1/1-96	Y RNA	96
RF00019 X57566.1/93-1	RFA_00595	Y RNA, RF00019, X57566.1/1-93	Y RNA	93
RF00025 AF399707.1/2345-2181	RFA_00603	Ciliate telomerase RNA, RF00025, AF399707.1/2181-2345	Cili. Telo. RNA	165
RF00025 AJ132318.1/175-1	RFA_00605	Ciliate telomerase RNA, RF00025, AJ132318.1/1-175	Cili. Telo. RNA	175
RF00163 Z69686.1/390-342	RFA_00720	Hammerhead ribozyme (type I), RF00163, Z69686.1/342-390	Ham. Ribozyme	49
RF00524 U13031.1/1644-1423	RFA_00810	R2 RNA element, RF00524, U13031.1/1423-1644	Other RNA	222
RF00524 U13033.1/1658-1423	RFA_00812	R2 RNA element, RF00524, U13033.1/1423-1658	Other RNA	236
RF00524 U81985.1/709-508	RFA_00818	R2 RNA element, RF00524, U81985.1/508-709	Other RNA	202
RF00524 X51967.1/3585-3349	RFA_00819	R2 RNA element, RF00524, X51967.1/3349-3585	Other RNA	237

(42.27%, 35.11%, 39.66%, respectively), indicating RNAforester performed the best. These results are understandable, considering that RKalign is mainly designed for comparing complex pseudoknot structures whereas RNAforester focuses on simpler pseudoknot-free structures.

The work that is most closely related to RKalign is CARNA [44]. Both methods are able to accept as input known pseudoknot structures and produce as output an alignment of the known structures. Our experimental results indicated that the two methods perform well in terms of base_mismatch ratios, though RKalign yields much lower stem_mismatch ratios. It should be pointed out, however, that the comparison with CARNA is not completely fair. The input data of RKalign are restricted to fixed structures, which are structures used in this study. Using CARNA with fixed structures is more or less a mis-use of the tool. The main purpose of CARNA is to align dot-plots, and its scoring is optimized for that data format. Thus, when dot-plots are considered, one should use CARNA. When fixed structures are considered, RKalign is recommended.

CHAPTER 6

CONCLUSIONS

6.1 Summary for Peak Detection

In the first part of this dissertation, we present a new approach, called PeakID, for elastic peak detection in 2D LC-MS data. PeakID works by first constructing a Shifted Aggregation Tree or AggTree from input data in a bottom-up manner, and then searching the AggTree for different peaks in a top-down manner. This method is able to detect multiple peaks across a variety of window sizes without yielding any false negative. Our experimental results showed that by spending some time in a training phase to find the topology and structure of an efficient AggTree, PeakID can run much faster than other methods on both synthetic and real-world data. The proposed approach lays out a framework for solving the elastic peak detection problem on time series data. While we have focused on 2D LC-MS data in this dissertation, and have shown that PeakID can speed up the analysis of such data by a factor of 10, our techniques can be easily generalized to process other time series data in an efficient way.

6.2 Summary for Pseudoknot Alignment

In the second part of this dissertation, we present a novel method, named RKalign, for comparing two known RNA pseudoknot structures. The method adopts the partition function methodology to calculate the posterior log-odds scores of the alignments between bases or base pairs of the RNAs with a dynamic programming algorithm. The posterior log-odds scores are then used to calculate the expected accuracy of an alignment between the RNAs. The goal is to find an optimal alignment with the maximum expected accuracy. We present a greedy algorithm to achieve this goal. Our experimental results demonstrated the good performance of the proposed RKalign method.

New pseudoknotted structures are found periodically, as exemplified by the recently determined ribosomal CCR5 frameshift pseudoknot [87] and the translational enhancer structures found in the 3'UTRs of plant viruses [88, 89, 90, 91]. It is therefore important to be able to compare these new structures to a database of known pseudoknots to determine the possibility of similar functionality. For example, some of the recently functionally similar pseudoknots found in the 3'UTRs of plant viruses have been shown to act as translational enhancers and have 3D structures that are similar to tRNAs. Importantly, they contain pseudoknots that produce tRNA-like 3D folds, but are not derived from the standard tRNA secondary structure cloverleaf. In addition, these elements have been shown to be important for ribosome binding. RKalign will be useful in performing this kind of database search for structure-function analysis of pseudoknots.

6.3 Future Work

Pseudoknots are important tertiary motifs in RNA. In our future work, we plan to develop new algorithms and techniques for pattern mining in biological data, particularly software for aligning RNA secondary structures including other tertiary motifs such as coaxial helical stacking [92] and A-minors [93]. These software tools will be useful in many bioinformatics applications, including RNA structure analysis, motif finding and database searching, among others.

BIBLIOGRAPHY

[1] S. Bandyopadhyay, U. Maulik, and J.T.L. Wang (eds.), *Analysis of Biological Data: A Soft Computing Approach*, Singapore: World Scientific, 2007.

[2] I. Eidhammer, K. Flikka, L. Martens, and S.-O. Mikalsen, *Computational Methods for Mass Spectrometry Proteomics*, first ed., Hoboken, NJ: Wiley-Interscience, 2008.

[3] J.T.L. Wang, M.J. Zaki, H.T.T. Toivonen, and D. Shasha (eds.), *Data Mining in Bioinformatics*, London, UK: Springer, 2005.

[4] C. Yang, Z. He, and W. Yu, "Comparison of Public Peak Detection Algorithms for MALDI Mass Spectrometry Data Analysis," *BMC Bioinformatics*, vol. 10, no. 4, 2009.

[5] M.C. Codrea, C.R. Jimenez, S. Piersma, J. Heringa, and E. Marchiori, "Robust Peak Detection and Alignment of nanoLC-FT Mass Spectrometry Data," *Proc. Fifth European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pp. 35-46, 2007.

[6] M. Katajamaa and M. Oresic, "Processing Methods for Differential Analysis of LC/MS Profile Data," *BMC Bioinformatics*, vol. 6, no. 179, 2005.

[7] B.Y. Renard, M. Kirchner, H. Steen, J.A.J. Steen, and F.A. Hamprecht, "NITPICK: Peak Identification for Mass Spectrometry Data," *BMC Bioinformatics*, vol. 9, no. 355, 2008.

[8] R. Stolt, R.J.O. Torgrip, J. Lindberg, L. Csenki, J. Kolmert, I. Schuppe-Koistinen, and S.P. Jacobsson, "Second-Order Peak Detection for Multicomponent High-Resolution LC/MS Data," *Analytical Chemistry*, vol. 78, pp. 975-983, 2006.

[9] J. Wong, G. Cagney, and H.M. Cartwright, "SpecAlign-Processing and Alignment of Mass Spectra Datasets," *Bioinformatics*, vol. 21, pp. 2088-2090, 2005.

[10] T. Hankemeier, J. Rozenbrand, M. Abhadur, J.J. Vreuls, and U.A.T. Brinkman, "Data Correlation in On-Line Solid-Phase Extraction-Gas Chromatography-Atomic Emission/Mass Spectrometric Detection of Unknown Microcontaminants," *Chromatographia*, vol. 48, pp. 273-283, 1998.

[11] T. Fushiki, H. Fujisawa, and S. Eguchi, "Identification of Biomarkers from Mass Spectrometry Data Using A Common Peak Approach," *BMC Bioinformatics*, vol. 7, no. 358, 2006.

[12] J. Listgarten, R.M. Neal, S.T. Roweis, P. Wong, and A. Emili, "Difference Detection in LC-MS Data for Protein Biomarker Discovery," *Bioinformatics*, vol. 23, no. 2, pp. 198-204, 2007.

[13] R.E. Ardrey, *Liquid Chromatography - Mass Spectrometry: An Introduction*, Hoboken, NJ: John Wiley & Sons, 2003.

[14] J.T.L. Wang, B.A. Shapiro, and D. Shasha (eds.), *Pattern Discovery in Biomolecular Data: Tools, Techniques and Applications*, New York, NY: Oxford University Press, 1999.

[15] A. Felinger, *Data Analysis and Signal Processing in Chromatography*, Amsterdam, The Netherlands: Elsevier Science, 1998.

[16] D. Shasha and Y. Zhu, *High Performance Discovery in Time Series: Techniques and Case Studies*, London, UK: Springer, 2004.

[17] Y. Zhu and D. Shasha, "Efficient Elastic Burst Detection in Data Streams," *Proc. Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 336-345, 2003.

[18] F.K.-P. Chan, A.W.-C. Fu, and C. Yu, "Haar Wavelets for Efficient Similarity Search of Time-Series: With and Without Time Warping," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 3, pp. 686-705, 2003.

[19] X. Zhang and D. Shasha, "Better Burst Detection," *Proc. Twenty-Second International Conference on Data Engineering*, pp. 146, 2006.

[20] W. Yu, B. Wu, N. Lin, K. Stone, K. Williams, and H. Zhao, "Detecting and Aligning Peaks in Mass Spectrometry Data with Applications to MALDI," *Computational Biology and Chemistry*, vol. 30, pp. 27-38, 2006.

[21] S. Chen, D. Hong, and Y. Shyr, "Wavelet-Based Procedures for Proteomic Mass Spectrometry Data Processing," *Computational Statistics & Data Analysis*, vol. 52, pp. 211-220, 2007.

[22] K.R. Coombes, S. Tsavachidis, J.S. Morris, K.A. Baggerly, M.C. Hung, and H.M. Kuerer, "Improved Peak Detection and Quantification of Mass Spectrometry Data Acquired from Surface-Enhanced Laser Desorption and Ionization by Denoising Spectra with the Undecimated Discrete Wavelet Transform," *Proteomics*, vol. 5, no. 16, pp. 4107-4117, 2005.

[23] M. Wang, T. Madhyastha, N.H. Chan, S. Papadimitriou, and C. Faloutsos, "Data Mining Meets Performance Evaluation: Fast Algorithms for Modeling Bursty Traffic," *Proc. Eighteenth International Conference on Data Engineering*, pp. 507-516, 2002.

[24] J. Kleinberg, “Bursty and Hierarchical Structure in Streams,” *Proc. Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 91-101, 2002.

[25] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos, “Identifying Similarities, Periodicities and Bursts for Online Search Queries,” *Proc. 2004 ACM SIGMOD International Conference on Management of Data*, pp. 131-142, 2004.

[26] M.G. Elfeky, W.G. Aref, and A.K. Elmagarmid, “Periodicity Detection in Time Series Databases,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 7, pp. 875-887, 2005.

[27] E. Keogh, S. Lonardi, and W. Chiu, “Finding Surprising Patterns in a Time Series Database in Linear Time and Space,” *Proc. Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[28] J. Yang, W. Wang, and P.S. Yu, “Mining Asynchronous Periodic Patterns in Time Series Data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 3, pp. 613-628, 2003.

[29] Z. Michalewicz and D.B. Fogel, *How to Solve It: Modern Heuristics*, London, UK: Springer, 2002.

[30] M.H.M. van de Meent, and G.J. de Jong, “Improvement of the Liquid-Chromatographic Analysis of Protein Tryptic Digests by the Use of Long-Capillary Monolithic Columns with UV and MS Detection,” *Analytical and Bioanalytical Chemistry*, vol. 388, pp. 195-200, 2007.

[31] C.W.A. Pleij, K. Rietveld, and L. Bosch, “A New Principle of RNA Folding Based on Pseudoknotting,” *Nucleic Acids Research*, vol. 13, no. 5, pp. 1717-1731, 1985.

[32] Y. Xin, C. Laing, N.B. Leontis, and T. Schlick, “Annotation of Tertiary Interactions in RNA Structures Reveals Variations and Correlations,” *RNA*, vol. 14, no. 12, pp. 2465-2477, 2008.

[33] C. Laing, D. Wen, J.T.L. Wang, and T. Schlick, “Predicting Coaxial Helical Stacking in RNA Junctions,” *Nucleic Acids Research*, vol. 40, no. 2, pp. 487-498, 2012.

[34] Z. Huang, Y. Wu, J. Robertson, L. Feng, R. Malmberg, and L. Cai, “Fast and Accurate Search for Non-Coding RNA Pseudoknot Structures in Genomes,” *Bioinformatics*, vol. 24, no. 20, pp. 2281-2287, 2008.

[35] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S.R. Eddy, “Rfam: an RNA Family Database,” *Nucleic Acids Research*, vol. 31, no. 1, pp. 439-441, 2003.

[36] P.L. Adams, M.R. Stahley, A.B. Kosek, J. Wang, and S.A. Strobel, "Crystal Structure of a Self-Splicing Group I Intron with Both Exons," *Nature*, vol. 430, no. 6995, pp. 45-50, 2004.

[37] C.A. Theimer, C.A. Blois, and J. Feigon, "Structure of the Human Telomerase RNA Pseudoknot Reveals Conserved Teriary Interactions Essential for Function," *Molecular Cell*, vol. 17, no. 5, pp. 671-682, 2005.

[38] D.W. Staple and S.E. Butcher, "Pseudoknots: RNA Structures with Diverse Functions," *PLoS Biology*, vol. 3, no. 6, pp. e213, 2005.

[39] C.M. Reidys, F.W.D. Huang, J.E. Andersen, R.C. Penner, P.F. Stadler, and M.E. Nebel, "Topology and Prediction of RNA Psedudoknots," *Bioinformatics*, vol. 27, no. 8, pp. 1076-1085, 2011.

[40] P.L. Nixon, A. Rangan, Y.G. Kim, A. Rich, D.W. Hoffman, M. Henning, and D.P. Giedroc, "Solution Structure of a Luteoviral P1-P2 Frameshifting mRNA Pseudoknot," *Journal of Molecular Biology*, vol. 322, no. 3, pp. 621-633, 2002.

[41] X. Xu, Y. Ji, and G.D. Stormo, "RNA Sampler: a New Sampling Based Algorithm for Common RNA Secondary Structure Prediction and Structural Alignment," *Bioinformatics*, vol. 23, no. 15, pp. 1883-1891, 2007.

[42] K. Sato, Y. Kato, T. Akutsu, K. Asai, and Y. Sakakibara, "DAFS: Simultaneous Aligning and Folding of RNA Sequences via Dual Decomposition," *Bioinformatics*, vol. 28, no. 24, pp. 3218-3224, 2012.

[43] I.M. Meyer and I. Miklos, "SimulFold: Simultaneously Inferring RNA Structures Including Pseudoknots, Alignments, and Trees Using a Bayesian MCMC Framework," *PLoS Computational Biology*, vol. 3, no. 8, pp. e149, 2007.

[44] D.A. Sorescu, M. Mohl, M. Mann, R. Backofen, and S. Will, "CARNA-Alignment of RNA Structure Ensembles," *Nucleic Acids Research*, vol. 40, no. Webserver, pp. W49-53, 2012.

[45] M. Andronescu, V. Bereg, H.H. Hoos, and A. Condon, "RNA STRAND: the RNA Secondary Structure and Statistical Analysis Database," *BMC Bioinformatics*, vol. 9, no. 340, 2008.

[46] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne, "The Protein Data Bank," *Nucleic Acids Research*, vol. 28, no. 1, pp. 235-242, 2000.

[47] F.H.D. van Batenburg, A.P. Gultyaev, C.W.A. Pleij, J. Ng, and J. Oliehoek, "PseudoBase: a Database with RNA Pseudoknots.," *Nucleic Acids Research*, vol. 28, no. 1, pp. 201-204, 2000.

[48] M. Taufer, A. Licon, R. Araiza, D. Mireles, F.H.D. van Batenburg, A.P. Gultyaev, and M.Y. Leung, "PseudoBase++: an Extension of PseudoBase for Easy Searching, Formatting and Visualization of Pseudoknots," *Nucleic Acids Research*, vol. 37, no. Database, pp. D127-135, 2009.

[49] S. Miyazawa, "A Reliable Sequence Alignment Method Based on Probabilities of Residue Correspondences," *Protein Engineering*, vol. 8, no. 10, pp. 999-1009, 1995.

[50] U. Roshan and D.R. Livesay, "Probalign: Multiple Sequence Alignment Using Partition Function Posterior Probabilities," *Bioinformatics*, vol. 22, no. 22, pp. 2715-2721, 2006.

[51] C.B. do, M.S. Mahabhashyam, M. Brudno, and S. Batzoglou, "ProbCons: Probabilistic Consistency-Based Multiple Sequence Alignment," *Genome Research*, vol. 15, no. 2, pp. 330-340, 2005.

[52] S. Will, T. Joshi, I.L. Hofacker, P.F. Stadler, and R. Backofen, "LocARNA-P: Accurate Boundary Prediction and Improved Detection of Structural RNAs," *RNA*, vol. 18, no. 5, pp. 900-914, 2012.

[53] D.H. Mathews and D.H. Turner, "Dynalign: an Algorithm for Finding the Secondary Structure Common to Two RNA Sequences," *Journal of Molecular Biology*, vol. 317, no. 2, pp. 191-203, 2002.

[54] E. Torarinsson, J.H. Havgaard, and J. Gorodkin, "Multiple Structural Alignment and Clustering of RNA Sequences," *Bioinformatics*, vol. 23, no. 8, pp. 926-932, 2007.

[55] S. Will, K. Reiche, I.L. Hofacker, P.F. Stadler, and R. Backofen, "Inferring Noncoding RNA Families and Classes by Means of Genome-Scale Structure-Based Clustering," *PLoS Computational Biology*, vol. 3, no. 4, pp. e65, 2007.

[56] M. Mohl, S. will, and R. Backofen, "Lifting Prediction to Alignment of RNA Pseudoknots," *Journal of Computational Biology*, vol. 17, no. 3, pp. 429-442, 2010.

[57] B. Han, B. Dost, V. Bafna, and S. Zhang, "Structural Alignment of Pseudoknotted RNA," *Journal of Computational Biology*, vol. 15, no. 5, pp. 489-504, 2008.

[58] B.J. Yoon, "Efficient Alignment of RNAs with Pseudoknots Using Sequence Alignment Constrains," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2009, no. 491074, 2009.

[59] T.K.F. Wong, K.L. Wan, B.Y. Hsu, B.W. Cheung, W.K. Hon, T.W. Lam, and S.M. Yiu, "RNASAlign: RNA Structural Alignment System," *Bioinformatics*, vol. 27, no. 15, pp. 2151-2152, 2011.

[60] T. Puton, L.P. Kozlowski, K.M. Rother, and J.M. Bujnicki, “CompaRNA: a Server for Continuous Benchmarking of Automated Methods for RNA Secondary Structure Prediction,” *Nucleic Acids Research*, vol. 41, no. 7, pp. 4307-4323, 2013.

[61] F. Ferre, Y. Ponty, W.A. Lorenz, and P. Clote, “DIAL: a Web Server for the Pairwise Alignment of Two RNA Three-Dimensional Structures Using Nucleotide, Dihedral Angle and Base-Pairing Similarities,” *Nucleic Acids Research*, vol. 35, no. Webserver, pp. W659-668, 2007.

[62] E. Capriotti and M.A. Marti-Renom, “SARA: a Server for Function Annotation of RNA Structures,” *Nucleic Acids Research*, vol. 37, no. Webserver, pp. W260-265, 2009.

[63] Y.F. Chang, Y.L. Huang, and C.L. Lu, “SARSA: a Web Tool for Structural Alignment of RNA Using a Structural Alphabet,” *Nucleic Acids Research*, vol. 36, no. Webserver, pp. W19-24, 2008.

[64] C.W. Wang, K.T. Chen, and C.L. Lu, “iPARTS: an Improved Tool of Pairwise Alignment of RNA Tertiary Structures,” *Nucleic Acids Research*, vol. 38, no. Webserver, pp. W340-347, 2010.

[65] D. Hoksza and D. Svozil, “Efficient RNA Pairwise Structure Comparison by SETTER Method,” *Bioinformatics*, vol. 28, no. 14, pp. 1858-1864, 2012.

[66] R.R. Rahrig, N.B. Leontis, and C.L. Zirbel, “R3D Align: Global Pairwise Alignment of RNA 3D Structures Using Local Superpositions,” *Bioinformatics*, vol. 26, no. 21, pp. 2689-2697, 2010.

[67] G. He, A. Steppi, J. Laborde, A. Srivastava, P. Zhao, and J. Zhang, “RASS: a Web Server for RNA Alignment in the Joint Sequence-Structure Space,” *Nucleic Acids Research*, vol. 42, no. Webserver, pp. W377-381, 2014.

[68] C. Zhong and S. Zhang, “Efficient Alignment of RNA Secondary Structures Using Sparse Dynamic Programming,” *BMC Bioinformatics*, vol. 14, no. 269, 2013.

[69] B.A. Shapiro and K. Zhang, “Comparing Multiple RNA Secondary Structures Using Tree Comparisons,” *Computer Applications in the Biosciences*, vol. 6, no. 4, pp. 309-318, 1990.

[70] J. Liu, J.T.L. Wang, J. Hu, and B. Tian, “A Method for Aligning RNA Secondary Structures and Its Application to RNA Motif Detection,” *BMC Bioinformatics*, vol. 6, no. 89, 2005.

[71] M. Hochsmann, B. Voss, and R. Giegerich, “Pure Multiple RNA Secondary Structure Alignments: a Progressive Profile Approach,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 1, no. 1, pp. 53-62, 2004.

[72] T. Jiang, G. Lin, B. Ma, and K. Zhang, "A General Edit Distance between RNA Structures," *Journal of Computational Biology*, vol. 9, no. 2, pp. 371-388, 2002.

[73] R.J. Klein and S.R. Eddy, "RSEARCH: Finding Homologs of Single Structured RNA Sequences," *BMC Bioinformatics*, vol. 4, no. 44, 2003.

[74] P.P. Gardner, A. Wilm, and S. Washietl, "A Benchmark of Multiple Sequence Alignment Programs upon Structural RNAs," *Nucleic Acids Research*, vol. 33, no. 8, pp. 2433-2439, 2005.

[75] J.W. Brown, "The Ribonuclease P Database," *Nucleic Acids Research*, vol. 27, no. 1, pp. 314, 1999.

[76] S.W. Burge, J. Daub, R. Eberhardt, J. Tate, L. Barquist, E.P. Nawrocki, S.R. Eddy, P.P. Gardner, and A. Bateman, "Rfam 11.0: 10 Years of RNA Families," *Nucleic Acids Research*, vol. 41, no. Database, pp. D226-232, 1995.

[77] H. Yang, F. Jossinet, N. Leontis, L. Chen, J. Westbrook, H. Berman, and E. Westhof, "Tools for the Automatic Identification and Classification of RNA Base Pairs," *Nucleic Acids Research*, vol. 31, no. 13, pp. 3450-3460, 2003.

[78] F. Wilcoxin, "Probability Table for Individual Comparisons by Ranking Methods," *Biometrics*, vol. 3, no. 3, pp. 119-122, 1947.

[79] A. Wilm, I. Mainz, and G. Steger, "An Enhanced RNA Alignment Benchmark for Sequence Alignment Programs," *Algorithms for Molecular Biology*, vol. 1, no. 19, 2006.

[80] P.A. Nuin, Z. Wang, and E.R. Tillier, "The Accuracy of Several Multiple Sequence Alignment Programs for Proteins," *BMC Bioinformatics*, vol. 7, no. 471, 2006.

[81] J.D. Thompson, F. Plewniak, and O. Poch, "A Comprehensive Comparison of Multiple Sequence Alignment Programs," *Nucleic Acids Research*, vol. 27, no. 13, pp. 2682-2690, 1999.

[82] A. Bremges, s. Schirmer, and R. Giegerich, "Fine-Tuning Structural RNA Alignments in the Twilight Zone," *BMC Bioinformatics*, vol. 11, no. 222, 2010.

[83] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, third ed., Burlington, MA: Morgan Kaufmann, 2011.

[84] R.R. Rahrig, A.I. Petrov, N.B. Leontis, and C.L. Zirbel, "R3D Align Web Server for Global Nucleotide to Nucleotide Alignments of RNA 3D Structures," *Nucleic Acids Research*, vol. 41, no. Webserver, pp. W15-21, 2013.

[85] Y. Song, C. Liu, R. Malmberg, F. Pan, and L. Cai, "Tree Decomposition Based Fast Search of RNA Structures Including Pseudoknots in Genomes," *Proc. IEEE Computational Systems Bioinformatics Conference*, pp. 223-234, 2005.

[86] I.L. Hofacker, “Vienna RNA Secondary Structure Server,” *Nucleic Acids Research*, vol. 31, no. 13, pp. 3429-3431, 2003.

[87] A.T. Belew, A. Meskauskas, S. Musalgaonkar, V.M. Advani, S.O. Sulima, W.K. Kasprzak, B.A. Shapiro, and J.D. Dinman, “Ribosomal Frameshifting in the CCR5 mRNA is Regulated by miRNAs and the NMD Pathway,” *Nature*, vol. 512, no. 7514, pp. 265-269, 2014.

[88] F. Gao, W.K. Kasprzak, C. Szarko, B.A. Shapiro, and A.E. Simon, “The 3’ Untranslated Region of Pea Enation Mosaic Virus Contains two T-Shaped, Ribosome-Binding, Cap-Independent Translation Enhancers,” *Journal of Virology*, vol. 88, no. 20, pp. 11696-11712, 2014.

[89] F. Gao, W.K. Kasprzak, V.A. Stupina, B.A. Shapiro, and A.E. Simon, “A Ribosome-Binding, 3’ Translational Enhancer Has a T-Shaped Structure and Engages in a Long-Distance RNA-RNA Interaction,” *Journal of Virology*, vol. 86, no. 18, pp. 9828-9842, 2012.

[90] V.A. Stupina, A. Meskauskas, J.C. McCormack, Y.G. Yingling, B.A. Shapiro, J.D. Dinman, and A.E. Simon, “The 3’ Proximal Translational Enhancer of Turnip Crinkle Virus Binds to 60S Ribosomal Subunits,” *RNA*, vol. 14, no. 11, pp. 2379-2393, 2008.

[91] J.C. McCormack, X. Yuan, Y.G. Yingling, W. Kasprzak, R.E. Zamora, B.A. Shapiro, and A.E. Simon, “Structural Domains within the 3’ Untranslated Region of Turnip Crinkle Virus,” *Journal of Virology*, vol. 82, no. 17, pp. 8706-8720, 2008.

[92] C. Laing, D. Wen, J.T.L Wang, and T. Schlick, “Predicting Coaxial Helical Stacking in RNA Junctions,” *Nucleic Acids Research*, vol. 40, no. 2, pp. 487-498, 2012.

[93] P. Sheth, M. Cervantes-Cervantes, A. Nagula, C. Laing, and J.T.L. Wang, “Novel Features for Identifying A-Minors in Three-Dimensional RNA Molecules,” *Computational Biology and Chemistry*, vol. 47, pp. 240-245, 2013.