Effective Models and Efficient Algorithms for Structural Bioinformatics

by

Yinglei Song

(Under the direction of Liming Cai)

Abstract

The major objective of bioinformatics research is to study biological systems with computational perspectives and methods. Structural bioinformatics is an important field in bioinformatics and focuses on the modeling, description and prediction of the tertiary and secondary structures of biological molecules.

Due to the inherent complexity associated with the structures of biological molecules, many problems in structural bioinformatics are computationally hard. An important challenge is thus to find computationally efficient methods (approximate or heuristic) that can provide good quality results.

This dissertation develops methods and models for noncoding RNA search and protein threading, which are two important problems in structural bioinformatics. In particular, we study statistical methods that can model RNA structures containing pseudoknots. Based on Parallel Communicating Grammar Systems (PCGSs), a memory efficient algorithm is developed to align an RNA sequence to a pseudoknot structure. In addition, we show that the sequence-structure alignment problem can be reduced to the maximum valued subgraph isomorphism problem. Based on the concept of graph tree decomposition, this problem can be solved efficiently when the tree width of the guest graph is small. Based on this technique, new methods are developed to solve the problems of noncoding RNA search and protein

tertiary structure prediction. The computational core of both problems is the sequence-structure alignment problem, and our testing results with this new algorithm demonstrate its advantage over previous methods for both problems in accuracy and computational efficiency.

INDEX WORDS: Structural bioinformatics, ncRNA search, Protein threading, Stochastic grammar models, Sequence structure alignment, Tree decomposition

Effective Models and Efficient Algorithms for Structural Bioinformatics

by

Yinglei Song

B.S., Physics, Tsinghua University, 1998

M.S., Physics, Ohio University, 2001

M.S., Computer Science, Ohio University, 2003

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2006

© 2006

Yinglei Song

All Rights Reserved

Effective Models and Efficient Algorithms for Structural Bioinformatics

by

Yinglei Song

Approved:

Major Professor: Liming Cai

Committee: E. Rodney Canfield
Russell L. Malmberg
Robert W. Robinson

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2006

Acknowledgements

I would like to gratefully thank my advisor, Dr. Liming Cai for his continuous care, guidance, patience, and his friendship during my graduate study at both University of Georgia and Ohio University. Dr. Cai led me into this new exciting field that combines knowledge and ideas from both theoretical computer science and biological research. More importantly, he encouraged me to not only focus on existing problems but also explore the field and discover new problems. Under his guidance, I have turned from a layman into a researcher.

I would also like to thank Dr. Russell Malmberg and Dr. Ying Xu for their assistance and guidance during my Ph.D. study. They introduced and taught me the fundamental knowledge in biological science that is needed to conduct research in bioinformatics. Their enlightening suggestions have helped me work on existing problems and explore the field. In addition, their generous help on biological knowledge and data made this dissertation possible.

I would like to thank Dr. E. Rodney Canfield and Dr. Robert Robinson for serving in my dissertation committee and the valuable suggestions and help from them.

I would like to thank Dr. Chunmei Liu, Jizhen Zhao, Dongsheng Che, Fangfang Pan, Ping Hu, Dr. Jun-tao Guo, Dr. Bo Yan, Kyle Ellrot and Dr. Guojun Li for all their kind help and suggestions during my Ph.D. study.

I would like to thank the Department of Computer Science, the National Institute of Health and the National Science Foundation for financial support.

Finally, I would like to thank my parents and grandparents for their faith in me. Their support and encouragement have provided me with the courage to face all the challenges and difficulty in my graduate study.

Table of Contents

Chapter 1

Introduction

Structural bioinformatics is a core research area in bioinformatics and has received significant attention in the last two decades. In general, the major goal of structural bioinformatics is to study issues related to physical structures of biological molecules using computational methods. Two commonly seen biological molecules are RNAs and proteins. The understanding of the biological functions of these molecules are largely determined by their structures. In particular, *structure prediction* and *structure comparison* are two problems of fundamental importance in structural bioinformatics.

The goal of structure prediction is to computationally determine the secondary or tertiary structure of a molecule from its primary sequence of residues. Structure determination with experimental techniques is expensive and time consuming. With the explosive growth of new biological molecules, it is desirable to develop efficient computational tools that can predict the structure of a molecule with high accuracy. In contrast, structure comparison defines a distance metric that can estimate the structural similarity between sequences. This similarity sometimes provides a basis for the recognition of new homologs for sequence families. Moreover, this similarity value can be informative as to the biological functionalities of the biological molecules.

These two problems have been under intensive study for nearly two decades. However, solutions that are completely satisfactory have not yet been available for either of them. The difficulty arises from the inherent complexity associated with the biochemical nature of these problems. In particular, the atomic interactions among sequence residues are complex and cannot be accurately described with simple computational models. The corresponding

computational problems are thus in general NP-hard, which suggests that efficient exact solutions almost certainly do not exist for these problems.

In this dissertation, we investigate possible approaches to solving these two problems. In particular, we develop a new memory efficient algorithm that can align a sequence to a pseudoknot structure with space complexity $O(N^2)$, where N is the length of the sequence to be aligned. Moreover, we show that the sequence-structure alignment used in both RNA search and protein threading can be formulated as a single problem using a method based on statistical models and graph algorithms. We notice that both problems can be reduced to optimization problems on graphs. Algorithms and theories developed in graph theory can thus provide new insights into these problems and possibly be used to develop solutions that are practically efficient. The major objective of this dissertation is thus *to develop different models and methods that can solve optimization problems in structural bioinformatics.* The development of these approaches need concepts and techniques from both statistics and graph theory.

1.1 Annotating Genomes for Noncoding RNAs

Noncoding RNAs are RNAs that do not encode proteins. Noncoding RNAs have been found to be important in many biological processes. Recent research shows that there exist a large number of different types of noncoding RNAs in a biological organism and most of them have various biological functions. In general, structure and sequence information for a noncoding RNA is contained in a genome and can be computationally identified if its sequence information is available. However, for the structure of a noncoding RNA is often conserved more than the primary sequence, models that fully describe a noncoding RNA family often needs to include information from both its primary sequence and the secondary structure.

An algorithm is thus needed to scan through a genome and compare the sequence segments in the genome against a *profile* that contains available statistical information regarding

the sequence and structure of this noncoding RNA. This procedure of comparison is also called *sequence-structure alignment*. A statistically significant alignment score suggests that the corresponding sequence segments have a high probability of containing the desired structure pattern of the given noncoding RNA.

The computational efficiency of performing the sequence-structure alignment is thus an important problem that needs to be seriously considered in the development of a searching tool. However, the number of all possible alignments between a sequence and a structure profile is exponential and it is thus more feasible to select the one with the maximum alignment score based on a given scoring scheme. From this perspective, sequence-structure alignment is an optimization problem and an algorithm is thus needed to find the alignment with the optimal alignment score.

RNA secondary structure may contain stacked base pairs: a group of stacked base pairs is also called a *stem*. In contrast, regions of unpaired residues are called *loops*. A structure is called *pseudoknot* if it contains stems that are structurally interweaving. For a structure that does not contain pseudoknots, Covariance Models (CMs) [21] have been developed to incorporate both sequence and structure information in a single statistical model. Based on the CYK algorithm, the optimal sequence-structure alignment can be performed in time $O(N^4)$, where N is the size of the structure profile. CMs have been used to search for pseudoknot free structures with high accuracy [34]. However, due to the large number of residues a genome generally contains, CM based sequence-structure alignment cannot be directly applied to search for structures that contain more than 300 residues.

Structure models [47, 14] have also been developed to model the crossing stems in pseudoknots. Based on these models, sequence-structure alignments can be performed with a dynamic programming algorithm similar to CYK algorithm. However, this algorithm needs $O(N^6)$ time and $O(N^4)$ space to align a sequence to a simple pseudoknot structure, where N is the size of the structure profile. Due to their high resource requirements, these models cannot be used to search a whole genome for pseudoknot structures.

In this dissertation, we develop a new structure model for all RNA secondary structures including pseudoknots. This structure model considers modeling the structure units in a secondary structure and the interactions among them with graph vertices and edges. A graph model thus can be constructed to model each given structure. Based on this new graph model, a new sequence-structure alignment algorithm is developed. Based on this sequence-structure alignment algorithm, genome annotation can be significantly sped up. In particular, our experiments on searching for RNA structures show that this new algorithm is in general at least 20 times faster than CM based searching while achieving the same accuracy. Moreover, the speed up is even more significant on structures that contain pseudoknots.

1.2 Protein Threading for Tertiary Structure Prediction

Protein tertiary structure prediction is a classic but very difficult problem in structural bioinformatics. Given the amino acids in a protein sequence, the goal of this problem is to determine its three dimensional structure. To date, two types of methods have been developed for this problem. They are knowledge based methods and *ab initio* ones. More specifically, knowledge based methods use protein sequences whose tertiary structures have been accurately determined and evaluate the similarity between a given new protein sequence and all sequences with known structure. The structure prediction can then be made based on this similarity. In contrast, *ab initio* methods consider the energy due to the interactions of amino acids in a protein sequence and are able to find stable three dimensional conformations by computing the global and local minima of this energy. So far, *ab initio* methods are computationally expensive and have yet been able to achieve prediction accuracy comparable with that of knowledge based methods due to the complexity of interactions among amino acids.

Protein threading is one of the most important knowledge based methods for predicting the tertiary structure of a protein sequence. In particular, threading methods group all protein sequences that fold into the same three dimensional structure into a *structure template*.

A structure template usually contains both the sequence and structure information obtained from its sequences. A database of all available structure templates can thus be constructed. A given protein sequence is then aligned with a computational tool to each structure template in the database and the prediction is based on the ranking of the alignment scores. For better accuracy, two-body interactions between amino acids whose spatial distance is within a certain cut-off distance must be modeled and considered while the alignment is performed.

However, the optimal sequence-structure alignment between a sequence and a structure template is an NP-hard problem [36] if the two-body interactions between amino acids are considered and gaps are allowed in the alignment. Two lines of effort have been made to develop methods to perform efficient sequence-structure alignment. On the one hand, heuristic methods have been developed to approximate the optimal sequence-structure alignment between a sequence and a structure template. However, the accuracy of these methods is not guaranteed.

On the other hand, based on the biochemical properties of protein sequences, methods that are practically efficient and can guarantee finding the optimal alignment have been developed. These methods include PROSPECT [64], which is the first program that can find the optimal alignment using a divide-and-conquer technique. RAPTOR [66] is another program that can perform optimal sequence-structure alignment. RAPTOR uses an integer linear programming approach and a branch-and-bound technique to reduce the size of the search space. Based on these optimal methods, optimal sequence-structure alignments can now be performed in a few minutes. However, tertiary structure prediction requires a given sequence to be aligned to all available structure templates in a database. Further improvement in the computational efficiency of alignment is thus needed to increase the overall efficiency of threading. Moreover, a faster algorithm for sequence-structure alignment enables the application of more sophisticated models for interactions among residues and thus can further improve the accuracy of prediction.

In this dissertation, we develop a new algorithm that can be used to perform sequence-structure alignment for protein threading. Similar to our assumption of RNA structures, we model the structure units and two-body interactions between amino acids in a structure template with graph vertices and edges. A structure template is thus modeled with a graph. The sequence-structure alignment is again reduced to a graph optimization problem. We show that, based on the biochemical property of a structure template, an algorithm that is practically efficient can be developed for this problem. We implemented this algorithm in a program called PROTTD (Protein Threading with Tree Decomposition) and compared its accuracy and efficiency with that of the PROSPECT. Our testing results show that this algorithm is significantly faster than PROSPECT while achieving the same or better alignment accuracy.

1.3 Sequence-Structure Alignment

The core part for both problems, the annotation of genomes for ncRNAs and protein threading, is the development of an algorithm that can perform sequence-structure alignment. We thus need to consider the general problem of sequence-structure alignment. In particular, we need to provide a formal definition for sequence-structure alignment. Based on this definition, we formulate the objective that needs to be optimized.

Definition 1.3.1 *A* structure template *T is a sequence of locations $\{L_1, L_2, \cdots, L_m\}$, and a relation R such that for $1 \leq i < j \leq m$, $R(L_i, L_j) = 1$ if L_i interacts with L_j and otherwise is 0, where m is the* size *of the template.*

Definition 1.3.2 *Suppose that Σ is an alphabet, given a sequence $S = \{s_1, s_2, \cdots, s_n\}$, where $s_i \in \Sigma$ for $i = 1, 2, \cdots, n$. An* alignment *$\mathcal{A}$ between S and a structure template $T = \{L_1, L_2, \cdots, L_m\, R\}$ is an order preserving placement of s_i's in locations in T.*

We note that some locations in the template may not be "occupied" by a residue. Such a location is considered to be aligned to a "gap" Δ.

Definition 1.3.3 *A given scoring scheme $\mathcal{S}$ consists of the following elements for all residues a, b and all locations L_i, L_j:*

1. *Singleton score $sing(a, L_i)$ which is fitness value associated with the placement of a at location L_i;*

2. *Pairwise score $pair(a, b, L_i, L_j)$ which is due to the interaction of a and b placed at locations L_i and L_j respectively;*

3. *A gap penalty function $P(\mathcal{A})$ which penalizes the residues and locations aligned to gaps in alignment $\mathcal{A}$.*

In general, we often use an *affine* gap penalty function to evaluate the gap penalty associated with a given alignment. In particular, two parameters g_o and g_e are introduced to model the penalty due to the opening and extension of a gap. Based on the affine gap penalty function, the penalty for a region that consists of d contiguous gaps can be computed by $g_o + g_e d$. The *alignment score* S_c under $\mathcal{S}$ for an alignment $\mathcal{A}$ is as follows.

$$S_c = \sum_{l(s_i)\neq\Delta} sing(s_i, l(s_i)) + \sum_{l(s_i)\neq\Delta, l(s_j)\neq\Delta} R(l(s_i), L(s_j))pair(s_i, s_j, l(s_i), L(s_j)) + P(\mathcal{A}) \quad (1.1)$$

where $l(s_i)$ is the location s_i is aligned to.

Based on Definition 1.3.3, an alignment score can be computed for each possible alignment between a sequence and a structure template. Our objective is to find the alignment with the maximum (or minimum) score over all possible alignments. We thus need to consider the following Sequence-Structure Alignment problem.

Problem 1.3.4 Sequence-Structure Alignment

Input: A sequence S, a structure template T and a scoring scheme $\mathcal{S}$.

Output: An alignment $\mathcal{A}$ between S and T such that its score S_c is the maximum (or minimum) over all possible alignments between S and T.

The optimal alignment score between a given sequence and a structure template is an important measure of the "fitness" of this sequence into the template. Without particular specification, we use alignment score to represent the optimal alignment score between a sequence and a structure template later in this dissertation. Unfortunately, the generic Sequence-Structure Alignment problem has been shown to be NP-hard in [36] by a reduction from the 3SAT problem.

As will be seen later in this dissertation, sequence-structure alignment is a problem of fundamental importance in structural bioinformatics. In particular, the computational difficulty associated with many problems in structural bioinformatics can be attributed to the difficulty of sequence-structure alignment or structure-structure alignment, which is one of the variants of the sequence-structure alignment. Most of the effort in this dissertation thus focuses on developing algorithms for this problem.

1.4 Dissertation Outline

This dissertation is organized as follows. In Chapter 2, a detailed survey of models and algorithms developed in previous work for searching for ncRNAs is presented. Similarly, we provide such a review for models and algorithms that have been developed for protein threading in Chapter 3. In Chapter 4, we describe in detail an important graph theoretical concept,*i.e.*, tree decomposition and describe some of its applications. The Development of our new algorithm for sequence-structure alignment is also based on this important concept. In Chapter 5 and 6 we supply a few technical details specific to each problem and the results of testing this algorithm on biological data. Chapter 7 concludes the dissertation and provides possible future work.

Chapter 2

Structure Models For Non-coding RNAs

2.1 Introduction

An RNA molecule contains a sequence of nucleotides. Nucleotides that are remote in the sequence can interact and form a *base pair*. Base pairs are often stacked and form a stem. In contrast, unpaired nucleotides in a contiguous region form a *loop*. An RNA secondary structure can be considered to be a combination of its stems and loops. Many secondary structures only contain nested and parallel stems. Some secondary structures may contain stems that structurally cross. These secondary structures are called *pseudoknots*.

To search a genome for a given secondary structure. We need to construct a structure profile that describes the secondary structure. Based on a structure profile, the genome can be scanned through and each sequence segment in the genome is aligned to the structure profile. Based on the alignment scores obtained on these sequence segments, we are able to report statistically significant ones as the possible locations for the given secondary structure.

The Stochastic Grammar Systems (SGS) have been used in computational linguistics to study the parsing of languages. Similarly, the base pairs can be considered to be two nucleotides that are derived simultaneously from a grammar system. Since two base pairs are either nested or parallel in a pseudoknot free structure, such a structure can also be considered to be a parsing tree of the sequence with respect to a given context free grammar. The productions in a Stochastic Context Free Grammar (SCFG) system are associated with probabilities. Based on a SCFG structure model, the optimal alignment between a given sequence and a structure is the parsing of the sequence that has the maximum probability.

This maximum probability can be computed with a dynamic programming similar to the CYK algorithm.

For a structure that contains pseudoknots, its crossing stems cannot be modeled with context free grammars. To deal with the context sensitiveness associated with pseudoknots, a few different grammar models have been developed to describe pseudoknots. For example, in [47], a new operator is introduced to the grammar system, pseudoknots can be derived from a grammar system with this new operator. In [14], a parallel stochastic grammar system (PCGS) with multiple grammar components is used to generate the crossing stems in a pseudoknot structure. A similar dynamic programming algorithm can be developed to perform the sequence-structure alignment between a sequence and a pseudoknot structure.

2.2 Structure Models for Pseudoknot Free Structures

2.2.1 Stochastic Context Free Grammars

Definition 2.2.1 *A grammar system $\mathcal{G}$ is a* stochastic context free grammar *if it is context free and each production is associated with a probability. In addition, probabilities of all productions with the same left hand side nonterminal sum up to* 1.0.

For a SCFG $\mathcal{G}$, we can obtain a SCFG $\mathcal{G}'$ that is equivalent to $\mathcal{G}$ and only contains the following types of productions.

1. $X \rightarrow a$;
2. $X \rightarrow Y|aY|Ya$;
3. $X \rightarrow aYb$;
4. $X \rightarrow YZ$.

where a, b are terminals and X, Y are nonterminals.

Based on the above definition, a stem formed between AGCAU and AUGCU can be derived from the following context free grammar. The derivation procedure can be clearly seen from Figure 2.1.

$$X_0 \rightarrow AX_1U \tag{2.1}$$

$$X_1 \rightarrow GX_2C \tag{2.2}$$

$$X_2 \rightarrow CX_3G \tag{2.3}$$

$$X_3 \rightarrow AX_4U \tag{2.4}$$

$$X_4 \rightarrow UX_5A \tag{2.5}$$

$$X_5 \rightarrow Y \tag{2.6}$$

These productions can be associated with probabilities and additional productions can also be included to generate other possible base pairs. For two nested stems, we can use the grammar sketched as following to generate both of them.

$$S \rightarrow X_0 \tag{2.7}$$

$$\cdots \tag{2.8}$$

$$X_m \rightarrow AY_0U \tag{2.9}$$

$$Y_0 \rightarrow AY_1 \tag{2.10}$$

$$\cdots \tag{2.11}$$

$$Y_l \rightarrow CZ_0G \tag{2.12}$$

$$\cdots \tag{2.13}$$

$$Z_n \rightarrow \cdots \tag{2.14}$$

where $S, X_0, \cdots, X_m, Y_0, \cdots, Y_l, Z_0 \cdots, Z_n$ are nonterminals and A, C, G, U are terminals. For two parallel stems, we only need to add a *bifurcation* production as follows.

$$X \rightarrow YZ \tag{2.15}$$

From Y and Z, the two parallel stems can be derived.

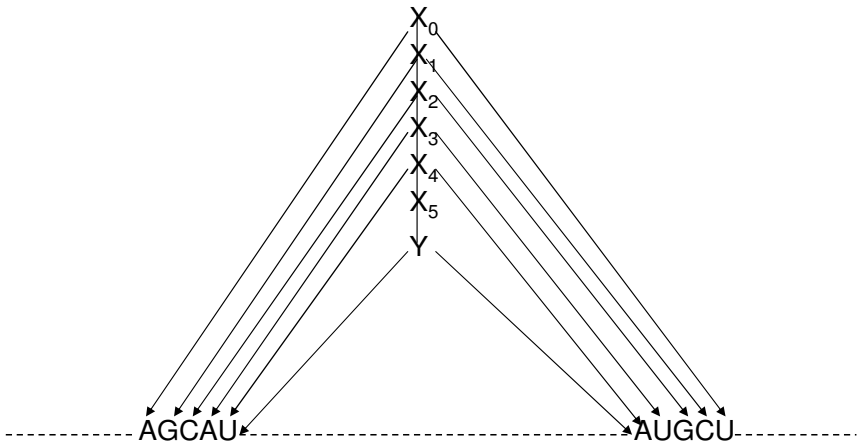

Figure 2.1: The derivation of a stem in an RNA secondary structure from a SCFG.

The optimal sequence-structure alignment between a sequence $s[1 \cdots, N]$ of N nucleotides and a structure of size M (M is often the number of nonterminals in the profile SCFG) can be performed with a CYK like dynamic programming algorithm. In particular, We construct a $M \times N \times N$ matrix $S[X_t, i, j]$ that stores the probability for deriving a subsequence $s[i \cdots, j]$ from nonterminal X_t ($1 \leq t \leq M$). The dynamic programming recursion relationship are as follows.

$$M[X_t, i, j] = \max \left\{ \begin{array}{l} \max_{i \leq k < j} \{M[X_l, i, k] \times M[X_m, k+1, j] \times P(X_t \rightarrow X_l X_m)\} \\ M[X_{r_1}, i+1, j] \times P(X_t \rightarrow s[i] X_{r_1}) \\ M[X_{r_2}, i, j-1] \times P(X_t \rightarrow X_{r_2} s[j]) \\ M[X_{r_3}, i+1, j-1] \times P(X_t \rightarrow s[i] X_{r_3} s[j]) \\ M[X_{r_4}, i, j] \times P(X_t \rightarrow X_{r_4}) \end{array} \right\}$$

Where function $P(.)$ denotes the probability associated with a given production in the SCFG. This relationship has taken into consideration five of the six possible types of productions where a given nonterminal X_t can be on the left hand side. The only remaining type $X_t \rightarrow a$

can be used to initialize the dynamic programming matrix.

$$M[X_t, i, i] = \max\{0.0, P(X_t \rightarrow s[i])\} \tag{2.16}$$

It is not difficult to see that the above dynamic programming needs $O(MN^3)$ time and $O(MN^2)$ space. In practice, in order to accurately evaluate the values of these probabilities, we often consider the logarithm of probabilities instead of the probabilities themselves. The multiplications in the above equation thus need to be replaced with additions.

2.2.2 Covariance Model

The Covariance Model (CM) is another statistical model developed by [22] to describe the formation of base pairs in the secondary structure of an RNA sequence. CM is an extension of the Hidden Markov Models (HMMs) [21] which have been extensively used to model the primary sequence information for a family of homologous sequences. A HMM consists of a *state set* which consists of a number of states where a given system can stay, a *symbol set* that specifies the "symbols" the system emit. a *transition matrix* that determines the probability p_{ij} for the system to transit from a given state i to state j. In addition, each given state in the state set is associated with a *set of emission probabilities* that specifies the probability for the system to emit each symbol in the symbol set when the system is in the given state.

HMMs have been extensively used in bioformatics to model the homologous sequence families [21]. In particular, for a given alignment of all sequences in the family, each column in the alignment can be associated with a state and each state has its set of emission probabilities that describe the probability for each nucleotide to appear in the given column. To model the insertions and deletions in an alignment, additional insertion and deletion states can be added to the corresponding HMM. Such a HMM is often called a *profile HMM* [21].

To align a sequence to a profile HMM, we can consider the nucleotides in the sequence to be symbols emitted by the HMM. There exist many state paths that can generate the given sequence of nucleotides. Each of such paths is associated with a probability value that can be

computed from the probabilistic parameters of the HMM. The optimal sequence-structure alignment problem is to identify the state path with the maximum probability.

The total number of state paths that can generate the same nucleotide sequence is exponential. However, the state path with the maximum probability can be computed with a dynamic programming algorithm. Such an algorithm is called Viterbi's algorithm, we can briefly describe this algorithm as follows. For convenience of notations, the sequence is represented with $S[1 \cdots, l]$ and $S[i]$ denotes the ith element in the sequence. We assume the set of states of the HMM is $S = \{s_1, s_2, \cdots, s_m\}$. The transition probability matrix for the HMM can be described with $\{p_{ij}\}$, where $1 \leq i \leq j \leq m$. We assume the set of symbols is Q and it contains $Q = \{q_1, q_2, \cdots, q_n\}$. The emission probability for symbol q_i $(1 \leq i \leq n)$ at state s_j $(1 \leq j \leq m)$ is e_{ij}. We use matrix $M[g, s_h]$ to store the maximum probability associated with the subsequence $S[1, \cdots, g]$, where the last symbol is emitted from state s_h. The recursion relation for the dynamic programming can be written as follows.

$$M[g, s_h] = \max_{1 \leq i \leq m} \{M[g-1, s_i] \times p_{ih}\} \times e_{S[g]h} \tag{2.17}$$

A profile HMM can only model the primary sequence content information of a family of homologous sequences. For RNA sequences, the secondary structure is also important since the biological function of an RNA sequence is often determined by its secondary structure. Similar to SCFGs, the CMs [22] are developed from the HMMs. In particular, a CM contains additional states that can emit two nucleotides that form a base pair together. Such states are called *pairing states*. In addition, *bifurcation states* are also added to generate parallel stems. It is not difficult to show that a CM is in fact equivalent to a stochastic regular grammar system. For example, in a CM, the system "transits" from a bifurcation state to two other states. Such a "transition" in fact plays the same role as a bifurcation production in a SCFG. Due to the equivalence between SCFGs and CMs, we do not distinguish them in later chapters of this dissertation.

2.3 Structure Models for Pseudoknot Structures

A pseudoknot structure contains crossing stems and it thus cannot be modeled with a context free grammar. In [47], a new operator is introduced to a grammar system and the crossing stems in a pseudoknot structure can be generated by applying this operator to the sequence that has been generated by the grammar system. In [14] an alternative stochastic grammar model is developed to generate the crossing stems through the communication of its context free grammar components. This grammar system is often called Stochastic Parallel Communicating Grammar Systems (SPCGSs). In this section, we focus on SPCGSs and show the way that they generate pseudoknot structures.

Definition 2.3.1 *A grammar system $\mathcal{G}$ is a* parallel communicating grammar system *if it satisfies the following.*

1. *The grammar contains m components $G_1, G_2, \cdots, G_m$, G_1 is the master component;*
2. *A nonterminal set $Q_s = \{Q_1, Q_2, \cdots, Q_m\}$ common to all grammar components, with the exception that Q_i is not in the nonterminal set of G_i ($1 \leq i \leq m$);*
3. *All components derive in parallel;*
4. *During the derivation if Q_i is in the derived result of grammar G_j ($i \neq j$), Q_i must be replaced by the sequence that have been obtained in grammar G_i and G_i needs to restart its derivation.*
5. *The output of the master component is the output of the grammar system.*

The Q_i's ($1 \leq m$) in a PCGS are also called the *querying symbols*. It has been shown in [14] that any pseudoknot structure can be derived from a PCGS grammar with a context free component and a few regular components. A stochastic PCGS can be obtained by associating the productions in each grammar component probabilities. From the result in [14], it is not difficult to know that a SPCGS can be constructed to model a family of sequences that fold into a common pseudoknot structure.

Definition 2.3.2 *For a SPCGS used to model a given pseudoknot structure, a nonterminal in the master component is a* P-structure *it derives an odd number of querying symbols and a* Pk-structure *if it derives even but non-zero number of querying symbols. A nonterminal is a* N-structure *if it derives a pseudoknot free substructure.*

Based on a SPCGS, a dynamic programming algorithm similar to the one for pseudoknot free structures can be developed to align a sequence to a pseudoknot structure. Table 2.1 provides a detailed description of the recursion relationships for the dynamic programming algorithm. Compared with the CYK algorithm, this algorithm needs two additional integer indices for P-structure nonterminals to store the stochastic scores associated with all possible locations of the subsequence obtained from other grammar components through communication. Therefore, the algorithm has space and time complexities of $O(N^4)$ and $O(N^6)$ respectively. Due to the requirement of computational resources, the algorithm cannot be used for efficient searching [14].

2.4 A Memory Efficient Algorithm

As shown in Table 2.1, in the original structural alignment algorithm, an exhaustive search is performed on all possible combinations of locations for query symbols to compute the optimal probabilities for Pk-structure nonterminals to derive a subsequence with pseudoknot structures. However, most of the combinations of locations do not really need to be considered since, biochemically, only very few of them can form stable stem structures. It is possible to identify the most likely locations of the regions without relying on the exhaustive search.

Since a pseudoknot structure is modeled with a bifurcation production that defines one of the crossing stems, the location of this stem can be determined with a local complementary alignment between the subsequences that result from the splitting point to separate the two P-structure nonterminals on the right side of the production. One of the subsequences must be reversed and complemented to convert the base pairs in a stem into matches in the alignment of two subsequences. The possible locations of the splitting point are exhaustively

NT	DP Recursions
K_i	$M[K_i, l, m] = \max\{M[P_s, l, k, p, q]M[P_t, k+1, m, u, v]P(K_i \rightarrow P_s P_t)\}$
P_i	$M[P_i, l, m, u, v] = \max\{M[N_j, l, k]M[P_s, k+1, m, u, v]P(P_i \rightarrow N_j P_s)\}$ $M[P_i, l, m, u, v] = \max\{M[P_s, l, k, u, v]M[N_j, k+1, m]P(P_i \rightarrow P_s N_j)\}$ $M[P_i, l, m, u, v] = M[Q, l, m, u, v]P(P_i \rightarrow Q)$
N_i	$M[N_i, l, m] = \max\{M[N_j, l, k]M[N_s, k+1, m]P(N_i \rightarrow N_j N_s)\}$ $M[N_i, l, m] = \max\{M[N_j, l+1, m-1]P(S[l] = a)P(S[m] = b)P(N_i \rightarrow aN_j b)\}$ $M[N_i, l, m] = \max\{M[N_j, l, m-1]P(S[m] = b)P(N_i \rightarrow N_j b)\}$ $M[N_i, l, m] = \max\{M[N_j, l+1, m]P(S[l] = a)P(N_i \rightarrow aN_j)\}$ $M[N_i, l, l] = \max\{P(S[l] = a)P(N_i \rightarrow a)\}$
Q	$M[Q, u, v, u, v] = 1.0$ $M[Q, l, m, u, v] = -\infty$ for all $l \neq u, m \neq v$

Table 2.1: The dynamic programming recursions for all types of nonterminals in the SPCGS model. For Pk-structure nonterminals, the maximum is taken over all integer k's that satisfy $l \leq k < m$ and all possible regions (p, q) and (u, v) obtained from the query symbol Q, where $l \geq p < q \leq k$ and $k+1 \leq u < v \leq m$ for a given k; for P-structure nonterminals, a four dimensional table must be maintained to store the intermediate probability values for all possible regions (u, v) obtained from the query symbol Q; the N-structure nonterminal follows the DP-recursions of the CYK algorithm; Q is the query symbol and we initialize $M[Q, u, v, u, v]$ to be 1.0 to start the dynamic programming. $M[X, i, j]$ is the maximum probability associated with nonterminal X to derive the subsequence from i to j; for P-structure nonterminal Y, $M[Y, i, j, u, v]$ is the probability for Y to derive the subsequence from i to j with the part from u to v obtained from the query symbol. S represents the sequence where we perform the alignment.

searched. However, for a given splitting point, the stem modeled with the query symbol is considered comprised of the regions with the maximum score of local complementary alignment, which can be determined with a variant of the Needleman and Wunsch's algorithm [42]. This approach hence only considers the stem location that is biochemically most likely and can effectively avoid the maintenance of the four dimensional table for P-structure nonterminals.

To further restrict the possible locations of the regions modeled with the query symbol, we introduce and define *offset pairs* for P-structure nonterminals, the set of offset pairs for a

Nonterminal	Productions	Recursions for Computing Offset Pairs
N_i	$N_i \rightarrow N_j N_s$	$len(N_i) = len(N_i) \cup \{k+h, k \in len(N_j), h \in len(N_s)\}$
	$N_i \rightarrow aN_jb$	$len(N_i) = len(N_i) \cup \{k+2, k \in len(N_j)\}$
	$N_i \rightarrow aN_j$	$len(N_i) = len(N_i) \cup \{k+1, k \in len(N_j)\}$
	$N_i \rightarrow N_jb$	$len(N_i) = len(N_i) \cup \{k+1, k \in len(N_j)\}$
	$N_i \rightarrow a$	$len(N_i) = 1$
P_i	$P_i \rightarrow N_j P_s$	$op(P_i) = op(P_i) \cup \{(k+l, r), k \in len(N_j), (l, r) \in op(P_s)\}$
	$P_i \rightarrow P_s N_j$	$op(P_i) = op(P_i) \cup \{(l+k, r), k \in len(N_j), (l, r) \in op(P_s)\}$
	$P_i \rightarrow Q$	$op(P_i) = op(P_i) \cup op(Q)$
Q	–	$op(Q) = \{(0, 0)\}$

Table 2.2: The recursion relationships used to compute the offset pairs of the P-structure nonterminals, the set of offset pairs for nonterminal P_i is denoted with $op(P_i)$. In order to compute $op(P_i)$, the set of lengths of all possible subsequences derivable from N-structure nonterminals must be determined, it is denoted with $len(N_i)$ for N-structure nonterminal N_i and can be computed recursively. The set of offset pairs for the query symbol Q is set to be $\{(0, 0)\}$.

P-structure nonterminal constitutes pairs of integers (l, r). An offset pair (l, r) indicates that left and right ends of the region derived from the query symbol must have distance no less than l and r from the left and right ends of the subsequence on which the local complementary alignment is performed. Every offset pair provides one possible position restriction for the region modeled with query symbol, and can be recursively determined from the productions in the master component of the SPCGS model. Table 2.2 shows the recursions on computing offset pairs for P-structure nonterminals.

The offset pairs are determined for all P-structure nonterminals to integrate the inherent conformational constraints to the memory efficient approach such that the optimal local complementary alignment is performed only on regions allowed by the model. Offset pairs are constraints imposed by the model and thus are independent of the length of the sequence.

Because the approach does not compute the probabilistic scores of P-structures for all possible locations of the stem modeled with the query symbol, it needs to compute the

NT	DP Recursions
P_i	$M[P_i, l, m] = \max\{M_{p,q}[N_j, l, k]M_{u,v}[P_s, k+1, m]P(P_i \rightarrow N_j P_s)\}$ $M_{u,v}[P_i, l, m] = \max\{M_{u,v}[P_s, l, k]M[N_j, k+1, m]P(P_i \rightarrow P_s N_j)\}$ $M_{u,v}[P_i, l, m] = M_{u,v}[Q, l, m]P(P_i \rightarrow Q)$
N_i	$M[N_i, l, m] = \max\{M[N_j, l, k]M[N_s, k+1, m]P(N_i \rightarrow N_j N_s)\}$ $M[N_i, l, m] = \max\{M[N_j, l, m]P(S[l] = a)P(S[m] = b)P(N_i \rightarrow aN_j b\}$ $M[N_i, l, m] = \max\{M[N_j, l, m-1]P(S[m] = b)P(N_i \rightarrow N_j b)\}$ $M[N_i, l, m] = \max\{M[N_j, l+1, m]P(S[l] = a)P(N_i \rightarrow aN_j)\}$ $M[N_i, l, l] = \max\{P(S[l] = a)P(N_i \rightarrow a)\}$
Q	$M_{u,v}[Q, u, v] = 1.0$ $M_{u,v}[Q, l, m] = -\infty$ for all $l \neq u, m \neq v$

Table 2.3: The recursions for computing the probabilistic score for P-structure nonterminals after the location of the stem has been determined for a given $l \leq k < m$ from the local complementary alignment as (p, q) and (u, v). The local alignment can guarantee that $l \leq p < q \leq k$ and $k+1 \leq u < v \leq m$. The dynamic programming matrices do not need to maintain the two additional integer indices to store the possible locations of the stem modeled with the query symbol because p, q, u and v are constants. However, they determine the initial condition to start the dynamic programming.

probability scores associated with the involved P-structures after the location of this stem has been determined with the local complementary alignment. This can be easily carried out with a dynamic programming approach by setting up the constraint that the query symbol Q must derive the regions. The recursions for performing this computation are shown in Table 2.3.

2.4.1 Techniques for Speeding up

Although the space complexity needed for structural alignment can be significantly reduced by this memory efficient alignment algorithm. The time complexity remains relatively high (of $O(N^6)$) and may become an serious problem when longer sequences need to be aligned. However, this algorithm can be significantly speeded up during the implementation by avoiding

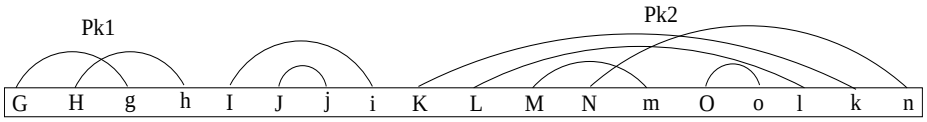

Figure 2.2: Diagram of the pairing regions of tmRNA pseudoknots 1 and 2 and the sequence between them. Upper case letters indicate base sequences that pair with the corresponding lower case letters. Not all structures are found in all sequences. This substructure of tmRNA, which contains 150 -250 nucleotides, is called Pk12.

recomputation. First, the aggregate time for local complementary alignments when the splitting point changes contiguously can be reduced, because the computation of the optimal local alignments for a given splitting point can be computed based on the results obtained for the preceding points. On the other hand, the local complementary alignment results may have some locality. In other words, the location determined for the stem modeled with the query symbol may remain unchanged for some given splitting points. This property of locality enables us to cache the probabilistic scores for P-structure nonterminals since it is very likely that the same local complementary alignment result would be obtained for the same splitting point in the future and the result can be directly

2.4.2 Experiments on the Memory Efficient Algorithm

In order to build the structure model, we need a set of sequences that have been aligned and annotated with respect to stem-loop and pseudoknot structures. The tmRNA database [29] provided such a dataset. We downloaded 85 aligned and structurally annotated sequences from the database. We constructed a phylogenetic tree of these sequences, then used this tree as the basis for dividing the dataset in two, so that the two halves each sampled the evolutionary diversity of the data. We used one half set as a training set to estimate the required probabilities for base pairs and for the grammar production rules. We used the

second half set as an evaluation set to compare the the results of our alignment program with the structural annotations provided by the database.

The tmRNA molecules have up to 4 pseudoknots in their structure. Figure 2.2 shows pseudoknot 1 (Pk1), pseudoknot 2 (Pk2) and the sequence region between them. Pk2 is complex in structure and may contain a stem loop that is present only in some training sequences. Pk1 and Pk2 are connected by a sequence region that contains two parallel sets of nested stems. In the dataset we used, Pk1 and Pk2 have average lengths of about 35 and 65 bases respectively. The two structures together with their separating regions are called Pk12 and contain about $150 - 250$ bases.

We assume that the region lengths of both stems and loops follow the geometric distribution and the probabilities associated with productions that extend the regions of loops and stems can be computed from their average lengths, frequencies of bases and base pairs for loops and stems respectively. We then align sequences in the testing set to the model and compare the conformations obtained from the optimal alignment results with their conformations denoted in the database.

Table 2.4 shows the results of simply counting the number of mismatches between the aligned conformation and the correct conformation as downloaded from the tmRDB database. On this basis, a comparison with the original structural alignment algorithm is made and shows that the memory efficient approach can be 80% as accurate as the original one. The mismatch scores for both approaches have a correlation of around 84%, indicating that difficulties arise on same sequences for both methods. The results demonstrate that the SPCGS model captures most of the conformational information in the sequences and the new approach can be as effective as the original one. An examination of the individual structure predictions shows that both approaches predict most structures similarly; however, on the individual sequence structure predictions that give both methods difficulty, the new approach tends to have many more mismatch errors.

Sequence	**Length(bs)**	**Methods**	**Mismatch Score**	
			Average	Standard Deviation
Pk1	$30-40$	Original	5.0	5.6
		Memory-Efficient	6.8	6.1
Pk2	$60-70$	Original	8.2	10.0
		Memory-Efficient	9.6	10.0
Pk12	$150-250$	Original	N/A	N/A
		Memory-Efficient	31	15

Table 2.4: The mismatch score is computed by comparing the aligned optimal conformations of sequences in testing data set with their accepted conformations as downloaded from the tmRDB. We use strings to represent conformations, and the mismatch score is essentially the edit distance between the string representatives of the two conformations. The Pk12 is not determined (n.d.) for the original alignment algorithm due to the unrealistic storage space requirement.

To evaluate the structural specificity of the SPCGS model, we compare the optimal alignment scores between sequences of a given structure and those of the same length and same base composition randomly reshuffled. The order of the nucleotides in each sequence from the testing data for Pk1 and Pk2 was randomized 50 times; the resulting sequences were then aligned to the SPCGS model with the memory efficient algorithm. This procedure provides the background distribution of optimal alignment scores for each sequence in the testing data set. The Z-score associated with each sequence can thus be computed from its background distribution. Figure 6.1 shows the Z-scores for each sequence in both testing data sets for Pk1 and Pk2 and, for comparison, their background Z-scores. It can be seen clearly from Figure 6.1 that most of the sequences are statistically significant with respect to the SPCGS model. The SPCGS model thus captures the structural signal specific to the pseudoknot structures. In particular, it achieves a good level of specificity for sequence-structure alignments.

Pseudoknot	Total	TP	Reported	SE(%)	SP(%)	Time (hr)	C+G(%)
Pk1	32	29	34	90.6	85.3	2.78	57.0
Pk2	28	25	31	89.3	80.6	69.52	57.0
Pk12	25	22	22	88.0	100.0	70.12	57.0
Pk1	32	31	35	96.9	88.6	2.86	67.0
Pk2	28	26	31	92.9	83.9	70.23	67.0
Pk12	24	22	22	91.7	100.0	71.45	67.0
Pk1	32	31	34	96.9	91.2	2.63	77.0
Pk2	28	26	29	92.9	89.7	71.19	77.0
Pk12	26	24	24	92.3	100.0	70.53	77.0
Pk1	32	31	32	96.9	96.9	2.56	87.0
Pk2	28	27	29	96.4	93.1	70.69	87.0
Pk12	25	24	24	96.0	100.0	70.42	87.0

Table 2.5: Results of experiments using the SPCGS model to find structural signals of pseudoknot structures on the random sequences with Pk1, Pk2 and Pk12 segments from tmRNA inserted respectively. "Total" is the total number of structures inserted; "Reported" is the total number of structures found; TP is the true positives, number of sequences correctly identified by the program with an error within ± 3 bases in both starting and ending locations; C+G is the sum of percentages of C and G in the background; SE and SP are sensitivity and specificity respectively. Experiments on Pk12 are carried out by searching for the locations of Pk1 and Pk2 respectively. A hit for Pk12 appears at locations where hits of Pk1 and Pk2 are found contiguous in location. This strategy identifies the locations of Pk12 with high values of specificity. The threshold is predetermined based on a Z-score of 2.0.

In addition to the structural alignment, the SPCGS model may provide a possible approach to searching biological genomes for the structural signals of noncoding RNAs including pseudoknot structures. In order to test its capability on searching, we inserted sequences from testing data set for Pk1, Pk2, and Pk12 into background sequences of 10^4 nucleotides; the background sequences are randomly generated with different base compositions. In particular, we use a window to scan through a genome sequence and use the memory efficient approach to align all sequence segments in the window to a SPCGS model that specifies the searched structural pattern. We select the maximum log-odds score of all sequence segments as the score for a given position in the genome sequence. It is then compared with a

threshold predetermined with a similar method as the one used in [34] (computed based on a Z−score of 2.0). A hit is reported if the score is greater than the threshold. Table 2.5 shows an evaluation of the results in terms of sensitivity and specificity. It can be seen from the table that, this alignment algorithm can achieve excellent accuracy (generally around $85\% - 95\%$ for both sensitivity and specificity) in recognizing the pseudoknot structures inserted into a random background. It is also clear that the performance of the approach does not vary significantly with the base composition of the background. Moreover, a slight improvement in both sensitivity and specificity is observed when the concentration of nucleotides C and G increases in the background.

We also used this alignment algorithm to search the genomes from a family of virus organisms for a domain that folds into a pseudoknot structure in the 3'UTR region and consists of five simple pseudoknots with each pseudoknot structures containing around 30 nucleotide bases [70]. We use Pk1-5 to represent them respectively. Due to the considerable amount of running time the program needs on long sequences, we trained the SPCGS model with the only 5 available sequences we have and we divided the pseudoknot structure into four pieces where each piece contains one or two simple pseudoknots; the genomes are then searched for each piece. We consider a real hit as comprised of hits that are from the results for different pieces and contiguous in locations on the genome. Table 2.6 shows the results of our experiments.

To summarize, this memory efficient alignment algorithm is able to recognize a complex multiple pseudoknot structure in viral genomes at essentially the correct location; however, some portions of these complex structures were not found correctly. It can be seen from Table 6.3 that the searching algorithm is able to recognize most of the structural signals of the pseudoknot structures in this particular domain. The Pk1 is not found on genomes of TMVC, TVV and RV at the corresponding locations where it is present in those of TMVF and TMV. Sequence alignments performed manually also demonstrate that the Pk1 is unlikely to appear in the part that contiguously precedes the Pk2 and Pk3. For the genomes of BVQ, CMV

Organism	SL(Pk1)	SL(Pk2-3)	SL(Pk4)	SL(Pk5)	RT (hr)	GL(bs)
TMV	6183 – 6237	6233 – 6290	6291 – 6356	6358 – 6395	6.53	6395
RL	6182 – 6237	6238 – 6289	6290 – 6357	6358 – 6395		
BVQ	5922 – 5978	5798 – 5857	Missing	5963 – 6003	6.12	6003
RL	N/A	N/A	N/A	N/A		
CMV	Missing	6262 – 6319	Missing	6387 – 6424	6.72	6424
RL	N/A	N/A	N/A	N/A		
OPV	6161 – 6215	6342 – 6401	Missing	6469 – 6506	6.81	6506
RL	N/A	N/A	N/A	N/A		
RV	5638 – 5692	6139 – 6198	6200 – 6263	6264 – 6300	6.48	6301
RL	N/A	6145 – 6197	6198 – 6263	6264 – 6301		
TMVC	Missing	6142 – 6201	6203 – 6266	6267 – 6303	6.48	6304
RL	N/A	6153 – 6205	6206 – 6271	6272 – 6304		
TMVF	6183 – 6237	6233 – 6290	6291 – 6357	6358 – 6395	6.37	6395
RL	6182 – 6237	6238 – 6289	6290 – 6357	6358 – 6395		
TVV	Missing	6150 – 6209	6211 – 6274	6275 – 6311	6.51	6311
RL	N/A	6156 – 6208	6209 – 6274	6275 – 6311		

Table 2.6: The searching results on the genomes from Tobacco Mosaic Virus (TMV) family, the 3'UTR pseudoknot structure is divided into four pieces with shorter lengths (less than 70 nucleotides each). For each genome, only one set of hits that are close or contiguous in locations is found. TMV is the Tobacco Mosaic Virus; BVQ is the Beet Virus Q; CMV is the Cucumber Mottle Virus; OPV is the Obuda Pepper Virus; RV is the Ribgrass Virus; TMVC and TMVF represent the TMV Crucifier and Fujian respectively. TVV is the Turnip Vein Virus; RL specifies the corresponding real location of each piece; we use N/A to mark the unavailable real location data; SL denotes the location found by the program; RT is the running time; GA and GL are the genome accession number and length in the number of base residues respectively.

and OPV, our results predict they should have a similar pseudoknot structure to TMV in their 3' UTR regions. However, in these genomes, the searching program fails to find the Pk4 in the region between Pk2-3 and Pk5, this may suggest that the structure on this region has been significantly changed by mutations or it is only a true negative of the searching program. In addition, we observed from the results that the Pk1 is not identified on locations that contiguously precedes Pk2 and Pk3 on the genomes of BVQ, CMV and OPV. However,

for two of them, the BVQ and OPV, the program finds a hit on locations close to Pk2 and Pk3. It is likely that the hits are false positives or the repeated structural patterns of Pk1 in locations nearby.

2.5 Summary of the Chapter

Due to the importance of the secondary structure of a noncoding RNA, a computational tool that can search genomes for noncoding RNA must include both the sequence and the structure information for the corresponding sequence family. SCFGs and CMs are two such models that have been developed to describe the nested and parallel stems in a secondary structure. Based on these models, the optimal sequence-structure alignment can be performed in time $O(MN^3)$, where M is the size of the structure profile and N is the length of the sequence.

Pseudoknot structures contain crossing stems and thus cannot be modeled with a context free grammar system due to its context sensitiveness. However, recent work on SPCGS has provided an elegant grammatical model for the description of pseudoknots. Based on this model, a dynamic programming algorithm can be used to optimally align a sequence of length N to a simple pseudoknot structure in time $O(N^6)$ and space $O(N^4)$.

To avoid the expensive space consumption needed to align a sequence to a pseudoknot structure, we have developed a memory efficient algorithm that can align an RNA sequence of length N to a SPCGS model in space $O(N^2)$. Based on the biochemical stability of the stem modeled with the query symbol in the SPCGS model, this algorithm determines the location of the crossing stem with a local complementary alignment algorithm and thus the exhaustive search used by the original alignment algorithm can be avoided. We have tested the sensitivity and specificity of this memory efficient algorithm on randomly generated sequences with inserted pseudoknot structures and a few biological genomes to annotate the structural signals of pseudoknotted noncoding RNAs. Our investigations have also provided a further understanding of both the SPCGS model and the pseudoknot structures and

demonstrated that the SPCGS model can serve as a good basis for profiling and searching for RNA pseudoknots.

Chapter 3

Protein Threading for Tertiary Structure Prediction

3.1 Introduction

The tertiary structure of a protein sequence is important for its biological functionalities. Laboratory techniques that can be used to determine protein tertiary structures include X-rays, NMR etc. The accuracy of these techniques is high. However, they are very expensive and time consuming and thus cannot keep up with the rapidly increased volumes of sequence data. Although computational methods are in general not as accurate as experimental techniques, they are significantly faster and promising to provide prediction results with high quality.

Currently, all available computational methods for tertiary structure prediction can be roughly classified into two categories: *ab initio* methods and knowledge based ones. In particular, *ab initio* methods consider the atomic interactions between the amino acids in a protein sequence and the corresponding energies. An algorithm that can minimize the total energy (or free energy) of the system is then used to predict the tertiary structure of a protein sequence. However, due to the complexity of atomic interactions, No efficient algorithms have yet been available to accurately find the three dimensional configuration with the minimum energy. *Ab initio* methods thus have not been extensively used for the prediction of protein tertiary structures.

Knowledge based approaches generally predict the tertiary structure of a protein sequence based on the available structure knowledge of other protein sequences. For example, It is possible to find homologs of a given sequence using BLAST search. The tertiary structure of the sequence can thus be predicted based on that of its homologs. However, this simple

method can only be used in cases where the homologs can be found with high confidence. To improve the prediction accuracy, it is often beneficial to include the sequence and structure information in a structure template for a family of sequences that share the same tertiary structure. A database of structure templates can thus be constructed based on all available three dimensional folds. To predict the tertiary structure of a protein sequence, we need to perform a sequence-structure alignment between the sequence and each structure template in the database. The prediction can be made based on the ranking of the alignment scores. Such methods are often called *protein threading*. It has been shown that including structure information in a structure template can significantly improve the prediction accuracy.

However, finding the optimal sequence-structure alignments is significantly more difficult than finding the optimal sequence-sequence alignments. It has been shown in [36] that the general problem of sequence-structure alignment is NP-hard if the structure template includes two body interactions and gaps are allowed in the alignment. A few heuristics have been developed to do sequence-structure alignment efficiently. However, these methods cannot guarantee the prediction accuracy. On the other hand, based on the biochemical properties of protein structures, a few optimal computational tools have been developed for the optimal sequence-structure alignment. For example, PROSPECT [64] employs a divide-and-conquer technique and can do optimal sequence-structure alignment in polynomial time for most of the structure templates. PROSPECT is the first computational tool that can perform efficient optimal sequence-structure alignments. RAPTOR [64] formulates the sequence-structure alignment as an integer linear programming problem and finds its optimal solution with branch-and-bound techniques. Currently, based on these computational tools, optimal sequence-structure alignment can be efficiently performed in a few minutes.

3.2 Energy Functions

The alignment between a sequence and a structure template is evaluated by its corresponding energy. For a given alignment (which may not be optimal), the energy of the alignment

consists of the contribution from a few different terms. Some of these terms are associated with the biochemical properties of protein sequences. In particular, the energy include the following terms.

$$E_{total} = E_{single} + E_{pair} + E_{mutate} + E_{gap} \tag{3.1}$$

where E_{single} is the singleton energy, which is the energy due to the placement of a given amino acids in a given local secondary structure environment. Such energy exists for each amino acids in the sequence and the total singleton energy is the sum of the singleton energy over all amino acids. We use $s[i]$ to denote the ith location in the sequence and $a[i]$ is the location that the ith amino acids is aligned to in the structure template. $ss[j]$ and $sol[j]$ represents the secondary structure and solvent accessibility of the jth location in the structure template. Based on these notations we can evaluate E_{single} based on the following equation.

$$E_{single} = \sum_{i=1}^{N} e_s(s[i], ss[a[i]], sol[a[i]]) \tag{3.2}$$

where $e_s(.)$ describes the singleton energy contributed from the alignment of a single amino acids. $e_s(s[i], ss[a[i]], sol[a[i]])$ is often evaluated with a statistical method based on available training sequences.

$$e_s(s[i], ss[a[i]], sol[a[i]]) = -k_B T \log\left(\frac{N_t N(s[i], ss[a[i]], sol[a[i]])}{N(s[i])N(ss[a[i]], sol[a[i]])}\right) \tag{3.3}$$

where N_t is the total number of amino acids in the training data, $N(s[i])$ is the number of amino acids $s[i]$; $N(ss[a[i]], sol[a[j]])$ is the number of amino acids that have secondary structure $ss[a[i]]$ and solvent accessibility $sol[a[i]]$; $N(s[i], ss[a[i]], sol[a[i]])$ is the number of amino acids $s[i]$ that have secondary structure $ss[a[i]]$ and solvent accessibility $sol[a[i]]$.

E_{pair} is the energy due to the two-body interactions of amino acids. In a structure template, only the two body interactions between amino acids that are within a cut-off distance is considered. In the structure templates used in PROSPECT, this cut-off distance is approximately 7.0Å. The number of two body interactions in a structure template can thus be significantly reduced. E_{pair} is the sum of the two energy of all pairs of interacting amino

acids.

$$E_{pair} = \sum_{i \neq j} R(a[i], a[j]) e_p(s[i], s[j]) \tag{3.4}$$

where $R(a[i], a[j]) = 1$ if locations $a[i]$ and $a[j]$ interact and $R(a[i], a[j]) = 0$ otherwise. $e_p(s[i], s[j])$ can be determined from the training sequences.

$$e_p(s[i], s[j]) = -k_B T \log \left(\frac{M_t M(s[i], s[j])}{M(s[i]) M(s[j])}\right) \tag{3.5}$$

where M_t is the total number of amino acids in the training protein sequences; $M(s[i])$ and $M(s[j])$ are the number of amino acids $s[i]$ and $s[j]$ respectively; $M(s[i], s[j])$ is the number of amino acids $s[i]$ and $s[j]$ that are within the cut-off distance. For other two energy terms, E_{mutate} evaluates the possibility for a given amino acids to be replaced with another one during evolution. E_{mutate} can be evaluated based on the available scoring matrices for protein sequence-sequence alignment. E_{gap} is used to evaluate the penalties associated with the insertions and deletions in an alignment. This energy function, however, is the one originally developed in PROSPECT I, an improved energy function will be provided in later chapters.

3.3 PROSPECT: A Divide-and-Conquer Optimal Algorithm

The divide-and-conquer technique used in computing the sequence-structure alignment can be sketched as follows. Since a structure template can be divided into structure templates with smaller size, the problem can also be divided into subproblems of smaller sizes. The algorithm thus solves these subproblems and combine the optimal solutions for these subproblems to get the optimal solution. This procedure can be recursively applied to the templates until the structure templates cannot be further divided.

To simplify the problem, we distinguish two types of structure units in a structure template. In general, *cores* represent α-helices, β-strands and a few other commonly seen structure units. We add an additional constraint such that locations in cores are not allowed to be aligned to gaps in a valid alignment. Based on this constraint, the divide-and-conquer

procedure is terminated when the sub-template contains only a single core and thus the alignment can be directly performed since no gaps are allowed in the alignment.

The only problem associated with this divide-and-conquer procedure is the two-body interactions between amino acids. In particular, dividing a structure template into substructure templates may "cut" the links that represent these two-body interactions. However, these links can be modeled with open links in the substructure templates. They have open ends in both of the substructure templates resulting from the division. For a subproblem with open links, the algorithm then needs to enumerate all possible amino acids for the open ends and find the optimal solution for each of such possibilities. the algorithm then combines the optimal solutions obtained on subproblems to get the optimal solution for a subproblem of larger size.

A careful implementation with the minimization of the maximum open link count can further improve the computational efficiency of this algorithm. It has been shown that, for most of the available structure templates, the number of open links is in generally small and this algorithm can thus find the optimal sequence-structure alignment for most of the available structure templates in polynomial time [64]. More specifically, the algorithm can optimally align a sequence to a structure alignment in time $O(Mn^{1.5C+1}+mn^{C+1})$ and space $O(Mn^{c+1})$, for a sequence with n amino acids and a structure template that contains m locations and M cores; C is a constant that is in general small (3 or 4 for around 78% of the available structure templates).

3.4 RAPTOR: An Integer Linear Programming Algorithm

The sequence-structure alignment problem can be formulated as an integer linear programming problem. Based on this property, RAPTOR develops a new algorithm that can do optimal sequence-structure alignment. In RAPTOR, an improved energy function has been implemented to evaluate alignments. In particular, an additional term E_{ss} is included in the total energy to take into account the matching of the predicted secondary structure of the

sequence and the secondary structure of the structure template. In addition, different energy terms are now assigned weights and the total energy is thus the weighted sum of all energy terms.

$$E = W_s E_{single} + W_p E_{pair} + W_{ss} E_{ss} + W_m E_{mutate} + W_g E_{gap} \tag{3.6}$$

Definition 3.4.1 ([65]) *A* contact map graph *is an undirected graph* $G = (V, E)$ *such that* $V = \{c_1, c_2, \cdots, c_M\}$*, where* c_i *represents the* i*th core along the structure backbone, and* $E = \{(c_i, c_j) |$ *there is an interaction between* c_i *and* c_j*, or* $|i - j| = 1\}$*.*

Based Definition 3.4.1, the alignment between a core c_i and a subsequence $[s_j, \cdots, s_j + l_i]$ in the sequence can be represented with (c_i, s_j), where s_j is the start of the subsequence. Such a pair is also called an *edge*, which is indeed an edge in a bipartite alignment graph. Based on this notation, we have the following results regarding alignments.

Definition 3.4.2 ([65]) *For any two different edges* $e_1 = (c_{i_1}, s_{j_1})$ *and* $e_2 = (c_{i_2}, s_{j_2})$*, if* $(loc_{i_1} - loc_{i_2})(s_{j_1} + loc_{i_2} - loc_{i_1} - s_{j_2}) \leq 0$*, then* e_1 *and* e_2 *are in* conflict*.*

Lemma 3.4.3 ([65]) *For any three different edges* $e_r = (c_{i_r}, s_{j_r})$*,* $r = 1, 2, 3$ *and* $loc_{i_1} < loc_{i_2} < loc_{i_3}$*, if* e_1 *conflicts with* e_2 *and* e_2 *conflicts with* e_3*, then* e_1 *conflicts with* e_3*.*

Lemma 3.4.4 ([65]) *For any three different edges* $e_r = (c_{i_r}, s_{j_r})$*,* $r = 1, 2, 3$ *and* $loc_{i_1} < loc_{i_2} < loc_{i_3}$*, if* e_1 *does not conflict* e_2 *and* e_2 *does not conflict* e_3*, then* e_1 *does not conflict* e_3*.*

Lemma 3.4.5 ([65]) *For any three different edges* $e_r = (c_{i_r}, s_{j_r})$*,* $r = 1, 2, 3$ *and* $loc_{i_1} < loc_{i_2} < loc_{i_3}$*, if* e_1 *conflicts with* e_3 *then* e_2 *conflicts with* e_3 *or* e_2 *conflicts with* e_1*.*

We use variables $x_{i,l}$ ($1 \leq i \leq M$ and $1 \leq l \leq n$) to represent the alignment relationship. In particular, $x_{i,l} = 1$ if (c_i, s_l) form an edge and otherwise $x_{i,l} = 0$. To model the two-body interactions in a structure template, we use variables $y_{(i_1,l_1),(i_2,l_2)}$ to represent the pairwise interaction between x_{i_1,l_1} and x_{i_2,l_2}, $y_{(i_1,l_1),(i_2,l_2)} = 1$ if $x_{i_1,l_1} = 1$, $x_{i_2,l_2} = 1$ and both edges

(c_{i_1}, s_{l_1}) (c_{i_2}, s_{l_2}) do not conflict. Using the same notation as that presented in [65]. We use $D[i]$ to represent the possible sequence positions that can be aligned to core c_i. And $R[i, j, l]$ denote all positions that can be aligned to c_j without conflicting with the edge (c_i, s_l). Based on these variables, the objective that needs to be minimized in the sequence-structure alignment problem is

$$\min W_m E_{mutate} + W_s E_{single} + W_p E_{pair} + W_g E_{gap} + W_{ss} E_{ss} \tag{3.7}$$

where

$$E_{mutate} = \sum_{i=1}^{M} \sum_{l \in D[i]} [x_{i,l} e_m(c_i, s_l)] \tag{3.8}$$

$$E_{single} = \sum_{i=1}^{M} \sum_{l \in D[i]} [x_{i,l} e_s(c_i, s_l)] \tag{3.9}$$

$$E_{ss} = \sum_{i=1}^{M} \sum_{l \in D[i]} [x_{i,l} e_{ss}(c_i, s_l)] \tag{3.10}$$

$$E_{pair} = \sum_{1 \le i < j \le M} \sum_{l \in D[i]} \sum_{k \in R[i,j,l]} [P(i,j) y_{(i,l),(j,k)} e_p(i, j, l, k)] \tag{3.11}$$

$$E_{gap} = \sum_{1 \le i}^{M} \sum_{l \in D[i]} \sum_{k \in R[i,i+1,l]} y_{(i,l),(i+1,k)} e_g(i, l, k) \tag{3.12}$$

where $P(i, j)$ is 1 if there are two body interactions between c_i and c_j and otherwise 0; $e_m(c_i, s_l)$ is the mutation energy computed from aligning c_i to s_l and $e_s(c_i, s_l)$ and $e_{ss}(c_i, s_l)$ are the corresponding singleton energy and energy of secondary structure matching; $e_p(i, j, l, k)$ is the sum of two body interactions due to the alignment of c_i to s_l and c_j to s_k; $e_g(i, l, k)$ is the gaps from the alignment between c_i and c_{i+1}.

The constraints for the integer linear programming problem are as follows.

$$\sum_{j \in D[i]} x_{i,j} = 1, i = 1, 2, \cdots, M \tag{3.13}$$

This set of constraints state that each core must be aligned to only one position in the sequence.

$$\sum_{l \ge l_0, l \in D[i]} x_{i,l} + \sum_{k \in D[i+1] - R[i,i+1,l_0]} x_{i+1,k} \le 1, l_0 \in D[i], i = 1, 2, \cdots, M-1; \tag{3.14}$$

This set of constraints state that the edges that determine the alignment of c_i and c_{i+1} do not conflict. According to Lemmas 3.4.3, 3.4.4, 3.4.5, this constraint can guarantee that no two edges for the cores in the alignment conflict.

$$\sum_{k \in R[i,j,l]} y_{(i,l),(j,k)} \leq x_{i,l}, \forall l \in D[i], i, j = 1, 2, \cdots, M; \tag{3.15}$$

$$\sum_{k \in R[j,i,k]} y_{(i,l),(j,k)} \leq x_{j,k}, \forall k \in D[j], i, j = 1, 2, \cdots, M; \tag{3.16}$$

This set of constraints state that for each pair of cores c_i and c_j, the two body interaction energy between them can be counted for at most once in the total energy of a given alignment.

$$\sum_{k \in R[i,j,l]} y_{(i,l),(j,k)} \geq x_{i,l} + \sum_{k \in R[i,j,l]} x_{j,k} - 1, l \in D[i], i, j = 1, 2, \cdots, M; \tag{3.17}$$

$$\sum_{k \in R[j,i,k]} y_{(i,l),(j,k)} \geq x_{j,k} + \sum_{l \in R[j,i,k]} x_{i,l} - 1, k \in D[j], i, j = 1, 2, \cdots, M; \tag{3.18}$$

This set of constraints state that for each pair of cores c_i and c_j, the two body interaction energy between them must be counted for at least once in the total energy of a given alignment. As the last points, we must have:

$$x_{i,j} \geq 0, j \in D[i], i = 1, 2, \cdots, M; \tag{3.19}$$

$$y_{(i,l),(j,k)} \geq 0, \forall l \in D[i], k \in D[j], i, j = 1, 2, \cdots, M. \tag{3.20}$$

A branch-and-bound algorithm can then be used to solve the integer programming problem formulated based on these constraints. More specifically, we can relax the above integer programming to a linear programming problem, which can be efficiently solved in polynomial time, if all the variables are assigned integer values in the optimal solution for the corresponding linear programming problem. If there exists one variable whose value is not an integer in the optimal solution, we then branch at this variable, i.e., consider the two possible integer values 0 and 1 that this variable can take and branches on the two resulting subproblems. This procedure can be applied recursively to find the optimal solution for this problem. Experiments on this algorithm has shown its superior performance in both accuracy and computational efficiency over most of other computational tools for protein tertiary

structure prediction. RAPTOR ranked the first among all non-meta servers in CAFASP3 (Third Critical Assessment of Fully Automated Structure Prediction). It also ranks the third among all non-meta servers in CASP6.

3.5 Summary of the Chapter

Protein tertiary structure prediction is a classic problem that is almost as old as bioinformatics itself. It has been under intensive study during the past two decades due to its difficulty and importance. So far, two types of methods have been developed to predict the tertiary structure of a protein sequence. *Ab initio* methods and knowledge based ones. *Ab initio* methods have yet to achieve satisfactory prediction results due to the complexity of atomic interactions among the amino acids in a protein sequence.

Protein threading is one of the most important knowledge based methods for protein tertiary structure prediction. The core part of a threading algorithm is an algorithm that can perform sequence-structure alignment between a protein sequence and a structure template. Due to the inherent complexity of protein structures, the generic problem of sequence-structure alignment is NP-hard [36]. However, using an appropriate cut-off distance (7Å), the number of two-body interactions in a structure template can be significantly reduced.

PROSPECT is the first algorithm that can do optimal sequence-structure alignment. PROSPECT uses a divide-and-conquer based algorithm that divides a structure template into substructure templates of smaller size, and then find the optimal solutions on these subproblems and combine them together. RAPTOR considers the problem from an alternative perspective and develops an integer programming formulation for the sequence-structure alignment problem. A new integer linear programming method is thus used in RAPTOR to find the optimal alignment.

Chapter 4

Tree Decomposition

4.1 Introduction

The idea of graph tree decomposition has profoundly affected research in both graph theory and algorithmic study. Tree decomposition was originally developed in a deep serial investigation of the graph minor theory [51]. Tree decomposition provides a new topological view of the graphs and moreover, important tools for the proof of the Wagner's conjecture, which states that an infinite series of graphs are well-quasi-ordered based on taking minors.

A *separator* of a graph is a subset of its vertices such that the removal of these vertices and edges incident on them disconnects the graph into at least two connected components. From perspectives of algorithm design, tree decomposition provide information on the graph separators and a divide-and-conquer based dynamic programming framework [4] can be developed to many graph optimization problems using these separators. For example, the Maximum Independent Set problem can be solved in time $O(2^t|V|)$ based on a tree decomposition with tree width t on a graph $G = (V, E)$. Similarly, an algorithm that needs time $O(2^{2t}|V|)$ can be developed for the Minimum Dominating Set problem.

In practice, many NP-hard graph optimization problems are generally formulated on graphs with bounded tree width. Algorithms based on this dynamic programming framework can possibly be used in practice to find the optimal solutions for these problems whose general instances are NP-hard.

4.2 Tree Decomposition

Definition 4.2.1 ([51, 52, 53, 54, 55, 56]) *Let $G = (V, E)$ be a graph, where V is the set of vertices in G, E denotes the set of edges in G. Pair (T, X) is a tree decomposition of graph G if it satisfies the following conditions:*

1. *$T = (I, F)$ defines a tree, the sets of vertices and edges in T are I and F respectively,*
2. *$X = \{X_i | i \in I, X_i \subseteq V\}$, and $\forall u \in V$, $\exists i \in I$ such that $u \in X_i$,*
3. *$\forall (u, v) \in E$, $\exists i \in I$ such that $u \in X_i$ and $v \in X_i$,*
4. *$\forall i, j, k \in I$, if k is on the path that connects i and j in tree T, then $X_i \cap X_j \subseteq X_k$.*

The tree width of the tree decomposition (T, X) is defined as $\max_{i \in I} |X_i| - 1$. The tree width of the graph G is the minimum tree width over all possible tree decompositions of G.

It is not difficult to see that a valid tree decomposition of a graph is in fact a partition of its vertices into different subsets. These subsets do not have to be disjoint and some vertices can thus be contained in more than one subsets. Such a subset is also called a *tree node*. Tree nodes are connected into a tree based on the graph topology. In particular, every vertex in the graph must be "covered" by at least one tree node and both vertices of each graph edge must be together contained in at least one tree node. Moreover, all tree nodes that contain a given graph vertex must form a connected subtree.

Tree decomposition provides an alternative view on graph topology. A tree node in a tree decomposition is often a separator of the graph. A divide-and-conquer based dynamic programming framework can thus be developed to find and combine partial optimal solutions on subproblems of smaller size. To combine partial optimal solutions, exhaustive search only needs to be performed on the vertices in the same tree node. This property leads to the development of efficient algorithms that can solve many NP-hard optimization problems on graphs with small treewidths.

4.3 A Framework for Dynamic Programming

4.3.1 Maximum Independent Set

An *independent set* in a graph is a subset of vertices such that any two vertices in the subset are not connected in the graph. The Maximum Independent Set problem is to find in a given graph the independent set of the maximum size. This problem has been known to be NP-hard [25]. However, it can be solved in linear time on graphs with bounded tree width.

Based on a tree decomposition T of tree width t for graph $G = (V, E)$. Without loss of generality, we assume that T is a binary rooted tree. We maintain a dynamic programming table in each tree node of T. A table stores all partial solutions obtained on the subgraph induced by the vertices contained in the subtree rooted at its tree node. For a tree node $X_k = \{v_1, v_2, \cdots, v_{t+1}\}$, The entries in a table contain all possible selections of vertices in the intersection of an independent set and the corresponding tree node. A table contains up to $t + 1$ columns where the ith column c_i specifies the decision bit for vertex v_i. More specifically, vertex v_i is included in the independent set if the decision bit of a given table entry is 1 at c_i and 0 otherwise. Two additional columns V and S are also included in a table. The V bit is used to indicate whether vertices selected in a given entry can form an independent set or not. The V bit of an entry is set to be 1 if the vertices selected can form an independent set and 0 otherwise. S stores the size of the maximum independent set based on the vertices selected in the entry.

A preprocessing procedure needs to be applied to the tree decomposition to mark in each tree node the vertices that no longer appear in its parents. The algorithm then follows a bottom-up procedure to fill the table for each tree node in the tree decomposition. In particular, for a leaf node of the tree, the algorithm enumerates all possible selections of vertices in the node and determine the values of V and S for each entry. To check the validity bit V for an entry, we can simply check whether two selected vertices are connected

or not. The S value for a valid entry can be computed by counting the number of selected vertices that are marked in the tree node.

For an internal tree node X_i, the algorithm needs to query the tables in its children X_j and X_k. In particular, for a given table entry e_m, the algorithm queries all valid entries in X_j that select the same vertices in $X_j \cap X_i$ as those in e_m and find the one with the maximum S value, we call this value S_j. Similarly, a value of S_k can also be obtained by querying the table of X_k. e_m is set to be invalid if no such entries exist in X_j or X_k, or two vertices selected in e_m are connected. Otherwise, we compute S by adding up the values of S_j, S_k and the number of selected vertices in e_m that are marked in X_i.

The algorithm then queries the table in the root node of the tree and finds the valid entry that has the maximum S value. Based on this entry, a bottom-up trace back procedure can be applied to find the vertices that are included in the maximum independent set. It is not difficult to show that this dynamic programming algorithm can correctly find the maximum independent set in a graph based on a tree decomposition. The number of entries a table need to maintain is up to $O(2^t)$. The computation time needed by this algorithm in a graph $G = (V, E)$ is thus bounded by $O(2^t|V|)$.

4.3.2 Minimum Dominating Set

The *dominating set* of a graph is a vertex subset such that every vertex not in this subset is connected to at least one vertex in the subset. The Minimum Dominating Set problem is to find in a graph the dominating set that is of the minimum size. This problem has also been shown to be NP-hard. Based on a tree decomposition of tree width t, the minimum dominating set of a graph $G = (V, E)$ can be found in time $O(2^{2t}|V|)$.

Similar to the dynamic programming algorithm developed for the Maximum Independent Set problem, we preprocess the tree decomposition and for each tree node mark the vertices that we maintain a dynamic programming table in each table. For a tree node $X_k = \{v_1, v_2, \cdots, v_{t+1}\}$. The table contains $2(t+1)$ columns including $s_1, s_2, \cdots, s_{t+1}$, which

are called *selection bits* and $d_1, d_2, \cdots, d_{t+1}$, which are *dominated bits*. In particular, s_i is set to be 1 if v_i is included in the dominating set and 0 otherwise. In contrast, d_i is set to be 1 if it has been dominated and 0 otherwise. An additional column V is set to be 1 if the vertices selected in a given entry can be extended to form a dominating set of the graph and 0 otherwise. For a given entry, column S stores the number of selected vertices that dominate all vertices contained in the subtree rooted at the tree node.

The algorithm then follows a similar bottom-up procedure to fill all the tables. For a leaf node, the algorithm enumerates all possible selections of its vertices. Based on a given selection, the dominated bits for each vertex can also be determined. V of a given entry is set to be 1 if all the vertices marked in the node has been dominated. S can be computed simply by counting the number of marked vertices that are selected.

For an internal node X_i with two children X_j and X_k. The algorithm also needs to query the tables in X_j and X_k to fill the table in X_i. More specifically, for an entry e_m in the table of X_i, we consider all valid entries in the table of X_j that are consistent with e_m. e_m is *consistent* with an entry e_n in the table of X_j if it satisfies the following.

1. e_m and e_n select the same vertices in $X_i \cap X_j$;

2. For any vertex $v \in X_i \cap X_j$, if its dominated bit is 1 in e_n, its dominated bit must also be 1 in e_m.

All valid entries consistent with e_m in the table of X_j can then be queried and the one with the minimum S value is considered, its S value is denoted with S_j. We can similarly get value S_k from the table in X_k. If no such entries can be found in either the table of X_j or X_k, or any marked vertex in X_i is not dominated, e_m is set to be invalid. For a valid entry e_m, the S value is equal to the sum of S_j, S_k and the number of selected vertices that are marked in X_i.

The size of the minimum dominating set can be obtained from the table in the root of the tree. A up-bottom trace back procedure can then be used to find the vertices that are

included in the minimum dominating set. The number of entries in a table can be up to $O(2^{2t})$ and the computation time of the algorithm is thus bounded by $O(2^{2t}|V|)$.

4.3.3 Maximum Cut

The Maximum Cut problem is another NP-hard problem that can be efficiently solved with a dynamic programming algorithm based on graph tree decomposition. In particular, given a graph $G = (V, E)$ and a partition P that partitions the vertices in V into two disjoint vertex sets V_1 and V_2, the *cut* of the partition is the number of edges in G such that one of the its vertex is from V_1 and the other is from V_2. The Maximum Cut problem is to find a partition P that achieves the maximum cut on a given graph G over all possible partitions.

A similar dynamic programming algorithm can be developed to solve the Maximum Cut problem based on a tree decomposition of the graph, the algorithm needs a preprocessing procedure that for each tree node, marks every graph edge that is contained in it but not contained in its parent. The algorithm also maintains a dynamic programming table in each tree node. Each entry table stores a possible partition of the vertices in the tree node. For tree node $X_k = \{v_1, v_2, \cdots, v_{t+1}\}$, the table contains $t + 1$ columns $c_1, c_2, \cdots, c_{t+1}$. Each column stores the decision bit on which partition the vertex v_i is assigned to. Without loss of generality, we assume that v_i is in V_1 if $c_i = 0$ and in V_2 otherwise. An additional column C is used to store the maximum cut that can be achieved by partitions that are consistent with the entry and on the subgraph induced on vertices contained in the subtree rooted at the tree node.

For a leaf node, the algorithm can enumerates all possible partitions of its vertices and fill its table with each of them as a table entry. To compute the C value for each entry, the algorithm simply counts the number of edges whose vertices are not in the same partition. For an internal tree node, the algorithm also needs to query the tables in the two children to determine the C value for each entry. For an entry e_m in the table of X_i, the algorithm queries the entries in the table of X_j that have the same partition on vertices in $X_i \cap X_j$

and finds the one with the maximum C value, we call it C_j. Similarly, we are able to obtain value C_k by querying the table in child X_k. Now the C value of entry e_m is simply the sum of C_j, C_k and the number of marked edges in X_i that are "cut" by e_m.

The value of the maximum cut can be obtained by querying the table in the root of the tree. A similar up-bottom trace back procedure can be used to find the partition with the maximum cut. A dynamic programming table in a tree node may contain up to $O(2^t)$ entries and the computation time needed by this algorithm is thus bounded by $O(2^t|V|)$.

4.4 The Subgraph Isomorphism Problem

Defintion 4.4.1 *Two graphs $G = (V, E)$ and $H = (M, N)$ are* isomorphic *if $|V| = |M|$ and there exists a bijective mapping f between V and M such that $(u, v) \in E$ if and only if $(f(u), f(v)) \in N$.*

We consider the following Subgraph Isomorphism problem.

Problem 4.4.2 Subgraph Isomorphism

Input: A graph $G = (V, E)$ and $H = (M, N)$;

Output: "yes" if H contains a subgraph that is isomorphic to G; otherwise output "no".

This problem can be easily shown to be NP-hard using a simple reduction from the Clique problem, which asks whether a given graph contains a clique of size k. However, it is possible to solve this problem in polynomial time when G and H are restricted to be certain types of graphs. For example, when G is of fixed size and H is planar, this problem can be solved in time $2^{O(|G|\log|G|)}|H|$, which is linear time since G is of fixed size. On the other hand, based on a tree decomposition of tree width t for H, this problem can be solved in time $O(|G|^t|H|)$, when G is of bounded degree. Recently it has been shown that this problem can be solved in polynomial time if G is of log-bounded fragmentation [30] and H is a graph with bounded tree width.

We show later that the sequence-structure alignment can be reduced to the Maximum Valued Subgraph Isomorphism problem and algorithms developed for this problem can thus be used to do sequence-structure alignment also. However, in problem instances reduced from the sequence-structure alignment problem. Graph G is generally of large size and the tree width of H is generally not small. All previously developed algorithms for subgraph isomorphism thus cannot be used to efficiently solve the sequence-structure alignment problem. New concepts and methods are thus needed to develop efficient algorithms for sequence-structure alignments through subgraph isomorphism.

4.4.1 Reduction to Subgraph Isomorphism

As we have defined in Chapter 1, an alignment is between a sequence and a structure profile (or template) is a placement of the sequence residues into the locations in the structure profile (or template). The optimal sequence-structure alignment is the alignment with the maximum (or minimum) alignment score. An alternative view of structure profile (or template) is to consider its structure units. For example, an RNA structure generally contains stems and loops and a protein structure template is comprised of cores and loops. Since loops generally consist of residue bases that do not interact with other parts of the structure. They are also called *trivial* structure units. An alignment can be considered to be the combination of the alignments on all structure units. The alignment score is also the sum of the alignment scores obtained on the sequence segments that are aligned to the structure units. Based on this view, we can consider solving the sequence-structure alignment problem at structure unit level instead of residue level.

A structure profile (or template) can be modeled with a graph. A nontrivial structure unit in a profile can be represented with a graph vertex. To model the interactions between structure units, the vertices of two interacting structure units can be connected with a non-directed edge. In addition, vertices for structure units neighboring along the sequence backbone are connected with a directed edge. A directed edge also represents the trivial

structure units between two nontrivial ones. The resulting graph is called the *structure graph* for the structure profile.

The sequence that is to be aligned to the structure profile can be preprocessed and the possible sequence segments that can be aligned to each structure unit can be determined from the alignment scores. These sequence segments are *candidates* for the given structure unit. Similarly, each candidate can be represented with a graph vertex and vertices for candidates of interacting structure units can be connected to form a *sequence graph*. We use G to denote a structure graph and H for a sequence graph.

It is clear that an alignment between a sequence and a structure corresponds to a subgraph contained in the sequence graph H and isomorphic to the structure graph G. Each candidate can be valued based on the alignment score between its sequence segment and the profile of the structure unit. In addition, each edge in H can also be associated with a value based on the alignment score between the corresponding sequence segment and a loop structure. The optimal structure-sequence alignment thus corresponds to a maximum (or minimum) valued subgraph in the sequence graph that is isomorphic to the structure graph.

4.4.2 Subgraph Isomorphism with Bounded Map Width

In the Subgraph Isomorphism problem, we are given a host graph H and a guest graph G and the goal is to decide whether there exists a subgraph in H that is isomorphic to G. In practice, it is often the case that H contains a large number of such subgraphs. The one that is of interest is the one that maximizes (or minimizes) certain objective function. A variant of the Subgraph Isomorphism problem can be formulated as follows.

Problem 4.4.3 Maximum Valued Subgraph Isomorphism

Input: A guest graph $G = (V, E)$, a host graph $H = (M, N)$, each vertex $v \in M$ is associated with a value $V(v)$ and each edge $e \in N$ is associated with a value $V(e)$.

Output: A subgraph H' in H such that H' is isomorphic to G and the sum of the values of vertices in H' reaches its maximum value.

The sequence-structure alignment problem can be reduced to the Maximum Valued Subgraph Isomorphism problem. Due to the biochemical property of biological sequences and structures, the number of candidates for a structure unit is generally a small number. In addition, the tree width of the structure graph G is often a small number. From these restrictions, a new variant of the Subgraph Isomorphism problem can be developed.

Definition 4.4.4 *Given two graphs $G = (V, E)$ and $H = (M, N)$ and a mapping f from V to 2^M, the* map width *of the mapping f is* $\max_{v \in V} |f(v)|$.

We consider the Maximum Valued Subgraph Isomorphism problem, where graph G is of bounded tree width and the map width for the isomorphic mapping is also bounded. This problem has been shown to be W[1]-hard [15], which suggests that it is unlikely to develop a uniform polynomial time algorithm for this problem when both the tree width of the structure graph and the map width of the isomorphic mapping are fixed parameters.

For an instance of the sequence-structure alignment problem, the vertices in the structure graph obtained from the structure profile is ordered along the backbone of the structure. In addition, the sequence graph obtained from the is partial ordered and does not contain directed cycle. These additional properties are important, since an efficient algorithm can then be developed to find in the sequence graph the maximum valued subgraph isomorphic to the structure graph, based on a tree decomposition of the structure graph.

4.4.3 A Dynamic Programming Algorithm

We assume that the map width of the isomorphic mapping between the structure graph and the sequence graph is bounded by k. We maintain a dynamic programming table in each tree node of the tree decomposition. Each entry in the table for a tree node is a combination of the candidates for vertices in the tree node. The table for a tree node $X_k = \{v_1, v_2, \cdots, v_{t+1}\}$ contains $t + 1$ columns $c_1, c_2, \cdots, c_{t+1}$, where c_i specifies the candidate for vertex v_i in each table entry. Two additional columns V and S are used to indicate the validity and the maximum (or minimum) value of the isomorphic mappings that conform to a given entry.

The algorithm applies a preprocessing procedure to process the tree decomposition, for each tree node, it marks the structure units (including loops) that are not covered in its parent. The algorithm then follows a bottom-up procedure to fill all the dynamic programming tables. For a leaf node, the algorithm enumerates all possible combinations of the candidates for its vertices and check whether the constraint of being isomorphic is satisfied or not. The validity bit V is set to be 1 if it is the case. The value S for a valid entry can be computed by adding up the values of all the marked structure units.

For an internal node X_i with two children X_j and X_k. To determine the validity bit and the S value for a given entry e_m in X_i, we enumerate in the table of X_j all valid entries that assign the same candidates to vertices in $X_i \cap X_j$ as those in e_m. The entry with the maximum (or minimum) score can be determined and we denote this value to be S_j. A value of S_k can be similarly obtained from the table in X_k. The validity bit of e_m is 0 if the candidates selected in e_m do not conform to the constraint of being isomorphic or no valid entries in X_j or X_k can be queried. The S value for e_m can be computed by adding S_j, S_k and the values for structure units marked in X_i due to the candidates in e_m.

The maximum (or minimum) value of the subgraph can be obtained by finding in the table of the tree root the entry with the maximum (or minimum) value. Based on this entry, a up-bottom trace back procedure can be applied to find the subgraph with the maximum (or minimum) value. A table may contain up to $O(k^t)$ entries and this dynamic programming algorithm thus needs $O(k^t|V|)$ time to do the sequence-structure alignment. It is not difficult to verify the correctness of this algorithm, based on the properties that vertices in a structure graph are ordered and those in a sequence graph are partially ordered.

4.5 Summary of the Chapter

Tree decomposition is an important concept in the fields of both graph theory and algorithm design. Tree decomposition provides an important means for both graph theory and

algorithm study. Based on tree decomposition, a separator based framework for dynamic programming algorithm has been developed, this framework can be used to efficiently solve many NP-hard optimization problems when the given graph is of small tree width. This chapter provides a few examples on the application of this framework, including the Maximum Independent Set problem, the Minimum Dominating Set problem and the Maximum Cut problem.

The sequence-structure alignment problem can be reduced to the Maximum Valued Subgraph Isomorphism problem. Given a restricted instance of this problem, where the tree width of the guest graph and the map width of the isomorphic mapping are both fixed, this problem cannot be solved with a uniform polynomial time algorithm. However, since vertices in a structure graph are ordered along the structure backbone and those in a sequence graph are partially ordered, a dynamic programming algorithm can be developed to solve this problem in time $O(k^t|V|)$. On the other hand, the parameterized complexity for the general Maximum Valued Subgraph Isomorphism problem, where a partial order may not exist for the vertices in the host graph, remains open.

Chapter 5

Annotating Non-coding RNAs with Tree Decomposition

5.1 Introduction

As we have seen in Chapter 2, non-coding RNAs (ncRNAs) are biologically important and play fundamental roles in a variety of biological processes such as gene regulation, chromosome replication, and RNA modification [24, 44, 69]. Recently, with the large amount of available sequence data, homologous searching based on computational methods has become one of the important approaches to the identification of new ncRNAs [38, 48, 49]. The core part of such a search program is an algorithm that aligns a target sequence to an RNA profile. To optimally identify the structure of remote homologs, the profile needs to include conserved conformations caused by long distance nucleotide base pairs (stems) as well as sequence conservation.

Most existing RNA search programs [38, 34, 13, 37] are based on the Covariance model (CM) developed by Eddy and Durbin [22] which enables the profiling of base pairs as well as single nucleotides. While a CM can achieve high accuracy on searching for pseudoknot-free structures, it cannot profile the crossing stems of a pseudoknot. In general, CM based search is computationally inefficient on structures with more than 300 nucleotides. For instance, the commonly used CYK structure-sequence alignment algorithm requires a computation time $O(N^4)$ for a profiled pseudoknot-free RNA containing N nucleotides [21]. To reduce the computation time needed for searching on long genomes or large sequence databases, a preprocessing step can be used to filter out portions of a genome which are unlikely to contain the desired pattern [6, 38, 63]. The filtration based methods can significantly reduce

the search time but the amount of speedup may not be guaranteed. These techniques have yet to be applied to searches for structures containing pseudoknots.

On the other hand, searching for pseudoknots may be significantly speeded up with heuristic approaches. For example, ERPIN, a search tool developed by Gautheret and Lambert [26], disassembles the secondary structure of an RNA family into separate stem loops. It scans the genome to search for possible hit locations for each stem loop structure and reports a hit when a combination of hit locations for different stem loops can conform with the overall structure. ERPIN does not allow gaps in the alignment and can therefore miss important remote homologs. Another approach, first proposed by Brown and Wilson [13] and further developed by us [37], models pseudoknots with the intersection of several SCFG or CM components. The optimal alignment score of a sequence is computed by combining the scores obtained from aligning the sequence to all components separately. This approach has the same drawback in computation time as CM based methods, therefore is not suitable for moderately large RNA structures.

In this chapter, we describe in detail a novel RNA structure (including pseudoknot) profiling model. We use a structure graph to model a secondary structure and a sequence graph to represent a sequence. The sequence-structure alignment problem is then reduced to the maximum valued subgraph isomorphism problem. We then apply the algorithm we have developed in Chapter 4 to solve this problem. Based on this algorithm, an optimal alignment can be found in time $O(k^t N^2)$ for a given integer parameter k. The value of k can be effectively determined by a statistical cut off and is also small in nature. Compared with the dynamic programming algorithm used in CM based search, our new algorithm is significantly faster.

We conducted experiments on several ncRNA families to test the accuracy and efficiency of the searching algorithm. Our experiments showed that, using a significantly reduced amount of computation time, the searching algorithm based on this new model can achieve the same accuracy as the CM based searching does. Specifically, on average, the algorithm

is about 24 and 50 times faster than CM based methods on searching for pseudoknot free sequences that contain around 90 and 150 nucleotides respectively. Our experiments also demonstrated an even more significant advantage of the algorithm over the CM based searching in computation time when the profiled RNAs contain pseudoknots. As a test of the model on real genomes, we used the algorithm to search for the tmRNA gene in two bacterial genomes, and the telomerase RNA gene on two yeast genomes. Both the tmRNA and the telomerase genes were very accurately detected on both genomes in days, a task that would have needed months of computation time if a CM based searching model has been used.

5.2 Methods and Models

We view the consensus secondary structure of an RNA family as a topological relation among basic structural units, each of which is a stem or a loop. Our new structure model consists of two components: a *conformational graph* that represents the relationship among all basic structural units, and a set of simple CMs and profile HMMs, each modelling a stem or a loop.

In the conformational graph H, each vertex defines either of the base pairing regions of some stem. The graph is a mixed graph containing both directed and undirected edges. Each undirected edge connects two base pairing regions that form a stem. Two base regions are connected with a directed edge (from 5' to 3') if they are the two ends of a loop. Technically, we add two additional vertices s (called *source*) and t (called *sink*) to the graph. Figure 6.1(a) and (b) show the consensus structure of an RNA family and the corresponding conformational graph. A consensus structure is usually obtained from a multiple structural alignment of a family of RNAs whose structure information is known. Therefore, in addition to the conformational graph, statistical models such as CMs and profile HMMs can be constructed for all stems and loops involved in the structure.

In this framework, a *target sequence* is a segment in a (possibly long) genome sequence. We use the profile of each stem to scan the target sequence to identify all pairs of regions in the target sequence that have statistically significant scores of (structural) alignment with the stem profile. These pairs of regions are called *images* of the stem. We define *parameter* k to be the maximum number of images of a stem over all stems in the structure. k has two interesting properties. First, k is a function of a statistical cut-off value. For example, for any stem, the number of images scored above a certain Z-score threshold is inversely proportional to the threshold value. Second, the value k is generally small in nature, especially when a more effective statistical cut-off is applied.

Given the set of images of all profiled stems in the structure, an *image graph* can be constructed. Similar to the construction of a conformational graph, each vertex defines one of the two base pairing regions of some stem and each undirected edge connects two base pairing regions that form a stem, but now a directed edge connects every two base regions (5' to 3') so long as they do not overlap. Based on the construction, each vertex u in the conformational graph H can only be mapped to a specific set of k vertices in the image graph G, each of which is called an *image of the vertex* u. Figure 6.1(c) and (d) illustrate the mapping from stems to their images and the corresponding image graph constructed.

The optimal structure-sequence alignment between an RNA structure profile and a target sequence is equivalent to the following generalized subgraph isomorphism problem: given a conformational graph H and an image graph G, find an one-to-one mapping f from vertices in H to their images in a subgraph S of G such that

1. (u, v) is an edge in H if and only if $(f(u), f(v))$ is an edge in S,

2. for any set of vertices in G representing overlapping regions on the target sequence, at most one of them can be selected to the subgraph S, and

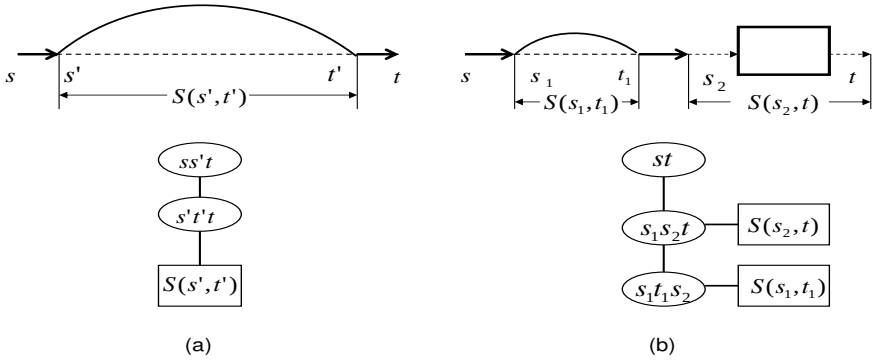

Figure 5.1: (a) The tree decomposition for secondary structure that contains an outer stem formed by s' and t'. (b) The tree decomposition for secondary structure that contains a leading structure unit with an outer stem (s_1, t_1).

3. the total score achieves the maximum when calculated from the score sum of the simultaneous alignment of all regions selected by the mapping to the stems and loops in the profile.

The defined problem is an optimization problem, different from the classical subgraph isomorphism decision problem. Section 3 gives an optimal algorithm for this optimization problem.

As we have seen in Chapter 2, searching a genome for a desired structure is accomplished by scanning through it with a window of a length determined by the size of the profiled structure. For a target sequence within the window frame, the optimal structure-sequence alignment is performed. Locations of the window with statistically significant scores are considered hits.

5.3 Tree Decomposition of Conformational Graphs

Although finding the optimal tree width and tree decomposition for a general graph is NP-hard [7], the conformational graph of a pseudoknot-free structure is simply an outer-planar

graph which has tree width 2 [7]. We describe briefly in the following a linear time recursive algorithm to find an optimal tree decomposition for such conformational graphs.

A pseudoknot-free structure can only be either of the following two cases: a single outermost stem "containing" all other stems within, and parallel stems. The tree decomposition algorithm just needs to deal with these two situations recursively.

(a) If the graph has a single outmost stem (s', t'), the algorithm generates two connected tree nodes $\{s, s', t\}$ and $\{s', t', t\}$. Then it recursively produces a subtree (decomposition) for the part in between s' and t', and connects the root of the subtree to node $\{s', t', t\}$, as shown in Figure 5.1(a). It returns the node $\{s, s't\}$ as the root of the tree decomposition.

(b) If the graph consists of parallel stems, as shown in Figure 5.1(b), the algorithm generates a tree node $\{s_1, t_1, s_2\}$ for the first stem (s_1, t_1) and connect it to another tree node $\{s_1, s_2, t\}$. Recursively, it produces a subtree for the part in between s_1 and t_1 and connects the root of the subtree to node $\{s_1, t_1, s_2\}$. Similarly, it creates a subtree for the part in between s_2 and t and connects the root of the subtree to node $\{s_1, s_2, t\}$. It further connects the $\{s_1, s_2, t\}$ to the third node $\{s, t\}$ which is returned as the root of the tree decomposition.

Now we consider *pseudoknot structures*, which are secondary structures with as least two stems that structurally cross. Tree decomposition for the conformational graph of a pseudoknot structure can be obtained by extending a tree decomposition for the conformational graph of a pseudoknot-free structure, since a pseudoknot structure can be viewed as the combination of a maximal pseudoknot-free structure with some *crossing* stems. Adding a crossing stem (Figure 5.2(a)) edge (u, v) to the conformational graph of the pseudoknot-free structure, called the *primary conformational graph*, may only increase the tree width by 1. This can be achieved by first including the vertices u and v into the conformational graph and finding a tree decomposition using the algorithm specified earlier in this section (Figure 5.2(b)). The tree decomposition is then extended by including v in every tree node on the path of the tree from the node containing u to the node containing v, thus accommodating

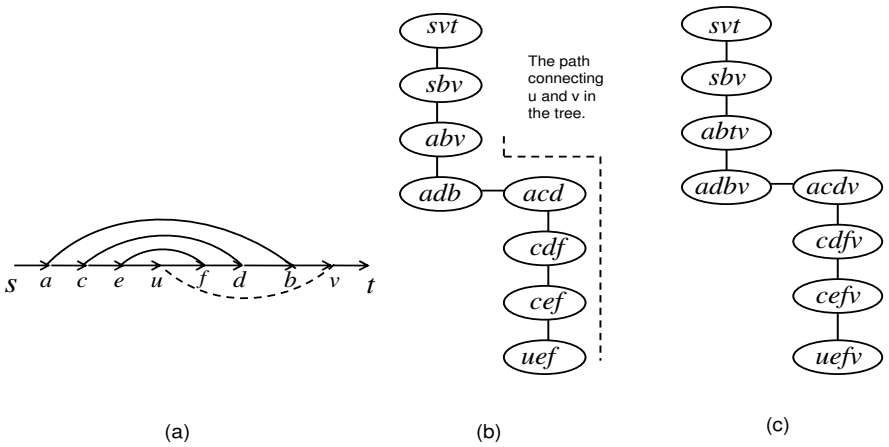

Figure 5.2: (a) The conformational graph of a pseudoknot structure. The dashed edge represents the crossing stem. (b) A tree decomposition of the graph without considering stem (u, v). (c) A tree decomposition of the graph by adding v to tree nodes on the path connecting u and v to contain the additional edge (u, v) in one tree node $(uefv)$.

the edge (u, v). (Figure 5.2(c)). Theoretically, if there are c crossing stems in a pseudoknot structure, the tree width of the corresponding conformational graph has tree width at most $2 + c$. Real RNA pseudoknots have a much smaller tree width. For example, Figure 5.3 shows the structure of tmRNA that contains 4 pseudoknots. It is a pseudoknot-free structure combined with crossing stems $(H, h), (N, n)$, (S, s), $(W, w), (X, x)$, (Z, z), and $(\#, 3)$. The tree width of the corresponding conformational graph is at most 4 since only crossing stems (W, w) and (X, x) are not independent of each other, increasing the tree width by 2.

The above technique can be improved by considering adding one "stack" of nested crossing stems, instead of one crossing stem, to the primary conformational graph at a time. Theoretically, we can prove that the tree width can only increase at most by 4 for the addition of a "stack" of crossing stems independent of the number of stems in this "stack". A *minimal* set of crossing stems consists of stems whose removal turns the structure into a pseudoknot free one, but the structure resulting from the removal of any of its proper subset contains at

least one pseudoknot. We assume that a pseudoknot structure contains d parallel structure units and each structure unit contains l_i minimal crossing stems. Moreover, the minimal crossing stems can be divided into c_i different groups of nested stems. It is not difficult to see that the tree width of such a structure graph is bounded by $2 + \max_{1 \leq i \leq d}\{\min\{l_i, 4c_i\}\}$.

5.4 Algorithms

We apply the algorithm developed in Chapter 4 to compute the maximum values subgraph in the sequence graph that is isomorphic to the structure graph. The computation time needed by the dynamic programming process is $O(k^t n)$ over a tree decomposition of tree width t and tree size n without including the time for alignment. Let N be the size of the overall structure profile containing m stems of lengths $s_1, \ldots, s_m$ and r loops of lengths $l_1, \ldots, l_r$. Then $N = \sum_{i=1}^{m} 2s_i + \sum_{j=1}^{r} l_j$ and $n = O(m)$. Since for each stem and loop profile, its optimal alignment to a counterpart in the target sequence takes a quadratic time, the total time for the optimal alignment algorithm is $O(k^t N^2)$.

5.5 Tests and Evaluation Results

We performed experiments to test the accuracy and efficiency of the algorithm and compared the performance of the algorithm with that of the CM-based searching. The training data was obtained from Rfam database [32], for each family, we choose up to 60 sequences with their pair-wise identities lower than 80% from the structural alignment of seed sequences. In practice, to obtain a reasonably small value for the parameter k, the upper bound on the number of images that a stem can map to, we constrain the images of a stem within certain region, called the *constrained image region* of the stem, in the target sequence. For this, we assume that, for homologous sequences, the distances from the pairing region of a given stem to the 3' end follow a Gaussian distribution. We compute the mean and standard deviation of distances from its two pairing regions to the 3' end of the sequence respectively, evaluated over all training sequences. For training data representing distant homologs of an

RNA family, we can effectively divide data into groups so that a different but related profile can be built for each group and used for search. This ensures a small value for the parameter k in the models.

As a first profiling and searching experiment, we inserted several RNA sequences from the same family into a random background generated with the same base composition as the sequences in the family. We then used both our algorithm and a CM-based searching algorithm we previously developed [37] to search for the inserted sequences. We compared the sensitivity and specificity of both searching algorithms on several different RNA families. To test the performance of the algorithm on real genomes, we used the algorithm to search for non-coding RNA genes in real biological genomes.

5.5.1 Searching for Pseudoknot Free Sequences

We used both the tree-decomposition based and the CM based algorithm to search for about 30 pseudoknot free RNA structures inserted in a random background of 10^5 nucleotides generated with the same base composition. We determined the statistical distribution for the alignment scores with a random sequence of 3000 nucleotides, which is generated with the same base composition as that of the sequence to be searched, with a method similar to that used by RSEARCH [34]. An alignment score with a Z-score greater than 5.0 is reported as a hit in both searching programs. In our experiments, for each stem, the algorithm selects k images with the maximum alignment scores within the constrained image region of the stem. In order to evaluate the impact of the parameter k on the accuracy of the algorithm, we carried out the same searching experiments for each given k.

Table 5.1 shows that, on tested RNA families, the tree decomposition based algorithm achieves the same searching accuracy as that of the CM based algorithm when the parameter k is equal to or larger than 6. From Table 5.2, compared to the CM based searching, the tree decomposition based algorithm requires a significantly reduced amount of computation time

RNA	LE	CM based		Tree decomposition based							
				$k = 5$		$k = 6$		$k = 7$		$k = 8$	
		SE	SP	SE	SP	SE	SP	SE	SP	SE	SP
ECRE	61	80.65	100	74.19	100	80.65	100	80.65	100	80.65	100
EOR	73	100	100	100	100	100	100	100	100	100	100
Let_7	84	100	100	95.8	100	95.8	100	100	100	100	100
Lin_4	72	100	100	100	88.9	100	94.11	100	94.11	100	94.11
Purine	103	93.10	100	93.10	96.43	93.10	96.43	93.10	96.43	93.10	96.43
SECIS	68	100	97.30	100	97.30	100	97.30	100	97.30	100	97.30
S_box	112	100	100	100	92.86	100	92.86	100	96.30	100	96.30
TtRNA	86	100	96.67	100	96.67	100	96.67	100	96.67	100	96.67

Table 5.1: A comparison of the searching accuracy of the tree decomposition based and CM based algorithms in terms of sensitivity and specificity. LE is the average length of sequences in the family, SE and SP are sensitivity and specificity in percentage respectively. ECRE, EOR, TtRNA represent Entero_CRE, Entero_OriR and Tymo-tRNA-like.

when the parameter k is 6. On most of the tested families, the tree decomposition based searching is more than 20 times faster than the CM based searching.

5.5.2 Searching for Sequences with Pseudoknots

We also performed searching experiments on several RNA families that contain pseudoknot structures. For each family, we inserted about 30 structures that contain pseudoknot structures into a background randomly generated with the same base composition as that of the inserted sequences. The training data was also obtained from the Rfam database [32] where we selected up to 40 sequences with pair wise identity lower than 80% from the seed alignment for each family. We used both the tree decomposition based algorithm and the CM based algorithm to identify the inserted sequences. For both algorithms, the threshold of alignment scores for reporting a hit is determined by a Z-score value 5.0.

Tables 5.3 and 5.4 show the comparisons of both searching accuracy and computation time for both algorithms. It is evident that, on families with pseudoknots, the tree decomposition

RNA	CM based	Tree decomposition based							
		$k = 5$		$k = 6$		$k = 7$		$k = 8$	
	RT	RT	SU	RT	SU	RT	SU	RT	SU
ECRE	57.96	2.60	22.2×	2.85	20.3×	3.21	18.1×	3.38	17.2×
EOR	103.08	4.77	21.6×	4.91	21.0×	5.26	19.6×	5.42	19.0×
Let_7	157.11	13.94	11.3×	14.97	10.5×	16.38	9.6×	16.92	9.3×
Lin_4	132.51	2.45	54.1×	3.22	41.2×	4.25	31.2×	5.10	26.0×
Purine	179.29	6.61	27.1×	7.09	25.3×	8.49	21.1×	9.61	18.7×
SECIS	185.21	8.48	21.8×	9.14	20.3×	10.23	18.1×	10.89	17.0×
S_box	756.27	26.10	29.0×	29.76	25.4×	34.76	21.8×	41.01	18.4×
TtRNA	185.05	4.34	42.6×	5.01	37.0×	6.10	30.3×	7.07	26.2×

Table 5.2: The computation time for both searching algorithms on all pseudoknot free RNA families. RT is the computation time in minutes, SU is the amount of speed up compared to the CM based searching algorithm.

based algorithm achieves the same accuracy as that of the CM based algorithm when the parameter k reaches a value of 7. In particular, the computation time needed by the algorithm is about 66 and 38 times less than that of the CM based algorithm on Alpha_RBS and Tombus_3_IV, the two families that contain more than 100 nucleotides. This demonstrates the promising advantage of the tree decomposition based algorithm over the CM based searching method in computation time when the structural pattern for which one is searching contains more than 100 nucleotides.

5.5.3 Search on Biological Genomes

To test the performance of the algorithm on real genomes, we used the algorithm to search biological genomes for structure patterns that contain pseudoknots. For example, the secondary structure formed by nucleotides in the 3' untranslated region in the genomes of the corona virus family contains a pseudoknot structure. This pseudoknot was recently shown to play important roles in the replication of the viruses in the family [28]. We selected four

RNA	LE	CM based		Tree decomposition based							
				$k = 5$		$k = 6$		$k = 7$		$k = 8$	
		SE	SP	SE	SP	SE	SP	SE	SP	SE	SP
Alpha_RBS	110	100	96.00	91.67	88.00	95.80	92.00	100	96.00	100	96.00
AFSE	55	100	100	92.86	100	96.43	100	100	100	100	100
HDVR	95	100	100	100	97.37	100	97.37	100	97.37	100	97.37
IFN_gamma	170	100	100	100	100	100	100	100	100	100	100
Tombus_3_IV	95	100	100	92.31	100	100	100	100	100	100	100
corona_pk3	65	100	94.80	100	97.37	100	97.37	100	97.37	100	97.37

Table 5.3: The searching accuracy for both tree decomposition based and CM based algorithms on RNA sequences containing pseudoknots. AFSE and HDVR represent Antizyme_FSE and HDV_ribozyme.

genomes from the corona virus family and used the algorithm to search for this pseudoknot. For bacteria, the tmRNA is essential for the trans-translation process and is responsible for adding a new C-terminal peptide tag to the incomplete protein product of a broken mRNA [41]. The secondary structure of tmRNA contains four pseudoknots and Figure 5.3 provides a sketch of the stems that constitute the secondary structure of a tmRNA. The tree decomposition based algorithm was also used to search for tmRNA genes on the genomes of two bacteria organisms, *Haemophilus influenzae* and *Neisseria meningitidis*. Both of the genomes contain more than 10^6 nucleotides. Among the bacteria containing tmRNAs, these two are relatively distant from each other evolutionarily. To test the accuracy and efficiency of the algorithm on genomes with a significantly larger size, we used the algorithm to search for the telomerase RNA gene in the genomes of two yeast organisms, *Saccharomyces cerevisiae* and *Saccharomyces bayanus*, both of which contain more than 10^7 nucleotides. Telomerase RNA is responsible for the addition of some specific simple sequences onto the chromosome ends [19].

RNA	CM based	Tree decomposition based							
		$k = 5$		$k = 6$		$k = 7$		$k = 8$	
	RT	RT	SU	RT	SU	RT	SU	RT	SU
Alpha_RBS	27.85	0.24	116.0×	0.31	90.1×	0.42	66.3×	0.55	50.6×
AFSE	0.94	0.10	9.4×	0.13	7.2×	0.18	5.2×	0.23	4.1×
HDVR	6.54	0.22	29.7×	0.34	19.2×	0.52	12.6×	0.79	8.3×
IFN_gamma	31.24	0.47	66.5×	0.72	43.4×	1.07	29.2×	1.52	20.6×
Tombus_3_IV	15.45	0.17	90.9×	0.27	57.2×	0.40	38.6×	0.57	27.1×
corona_pk3	2.89	0.12	24.1×	0.15	19.3×	0.20	14.5×	0.26	11.1×

Table 5.4: The computation time for both searching algorithms on all RNA families that contain pseudoknots. The amount of RT is in hours.

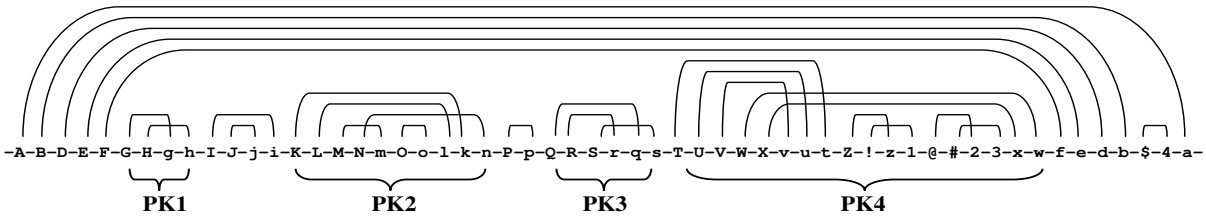

Figure 5.3: Diagram of stems in the secondary structure of a tmRNA. Upper case letters indicate base regions that pair with the corresponding lower case letters. The four pseudoknots constitute the central part of the tmRNA gene and are labeled as Pk1, Pk2, Pk3, Pk4 respectively.

	ncRNA	Tree decomposition based			CM based		
		Left offset	Right offset	Time	Left offset	Right offset	Time
BCV	3'PK	0	0	0.053	0	0	1.24
MHV	3'PK	0	0	0.053	0	0	1.27
PDV	3'PK	0	0	0.048	0	0	1.17
HCV	3'PK	0	0	0.047	0	0	1.12
HI	tmRNA	−1	−1	44.0	0	0	1700
NM	tmRNA	0	0	52.9	0	0	2044
SC	TLRNA	−3	−1	492.3	–	–	–
SB	TLRNA	−3	2	550.2	–	–	–

Table 5.5: A comparison of the accuracy and efficiency for both algorithms on searching biological genomes. OR is the name of the organism; GL is the length of the genome in multiples of 10^5 nucleotides. BCV is Bovine corona virus; MHV is Murine hepatitus virus; PDV is Porcine diarrhea virus; HCV is Human corona virus; HI and NM represent *Haemophilus influenzae* and *Neisseria meningitidis* respectively. SC and SB represent *Saccharomyces cerevisiae* and *Saccharomyces bayanus* respectively. RT is the single CPU time needed to identify the ncRNA in hours. For tmRNA and telomerase RNA searches, RT is estimated from the time needed by a parallel search with 16 processors.

The parameter k used in the tree decomposition based algorithm for searching all genomes is 7. Table 5.5 provides the real locations of the searched patterns, the locations annotated by the tree decompostion based and CM based algorithms respectively. The table clearly shows that, compared with CM based searching, the tree decomposition based model and searching algorithm are able to achieve the same accuracy with a significantly reduced amount of computation time. Both our new program and the CM base program have 100% sensitivity and specificity for searches in genomes. Searching a genome of moderate size for a structural pattern as complex as tmRNA gene only needs days of computation time, instead of months.

5.6 Summary of the Chapter

In this chapter, we describe in detail a novel graph theoretical model for profiling RNA structures including pseudoknots. This approach profiles the fundamental structural units that form the secondary structure of an RNA family separately, and the structural relations among the structural units are described with a conformational graph. Based on this generic framework, an image graph can be constructed by determining the possible locations of each stem on a target sequence. The target sequence can be efficiently aligned to the profiling model by computing the maximum valued subgraph isomorphic to the conformational graph in the image graph. Our experiments demonstrated that this approach is able to achieve the same searching accuracy as CM based methods while requiring only a small fraction of the computation time needed by them. Based on this profiling model and the optimal alignment algorithm, we are able to accurately determine the locations of ncRNAs with complex structural patterns in genomes of a moderate size in days.

The time complexity of the alignment is $O(k^t N^2)$, for an RNA family that contains N nucleotides and has a conformational graph with tree width t. Parameter k is an upper bound of the number of images of each stem on a target sequence and can be effectively determined with a statistical cut-off value based on the constrained mapped regions of a stem. Our experiments also showed that, on most tested RNA families, a value of 7 is sufficient for achieving the same accuracy as that of the CM based searching methods.

Chapter 6

Efficient Threading with Tree Decomposition

6.1 Introduction

As we have seen in Chapter 3, predicting the native-like structural fold of a protein from its primary sequence is an important but difficult problem in computational biology. Threading based methods [32, 58, 64, 66] have become one of the most important approach to the prediction of protein tertiary structures. Based on the observation that most of protein sequences fold into a limited number of structure folds, a threading method predicts the tertiary structure of a sequence by evaluating the compatibility between a sequence and each available structure fold with both sequence and structural information [32]. In general, the core computational part of a threading based method is the sequence-structure alignment, where a protein sequence is aligned to the available structure templates in a database and structure prediction is made based on the statistical alignment scores. Developing accurate and efficient algorithms for sequence-structure alignments is thus one of the primary goals for improving the performance of tertiary structure prediction.

A few different threading approaches have been developed to incorporate structural information into structure templates or alignment algorithms. For example, the program GenTHREADER [32] considers the two-body interactions after obtaining a sequence-structure alignment and uses a neural network based method to evaluate the confidential measure for each predicted tertiary structure. Compared with other approaches, GenTHREADER achieved an improved average prediction accuracy around 77%. FUGUE [58] uses the environment-specific amino acid substitution tables and structure-dependent gap penalties for sequence-structure alignment. FUGUE significantly improved the performance

of fold recognition at both superfamily and fold levels. However, later work [64, 35, 66, 65] has shown that, considering the two-body interactions between amino acids while performing the sequence-structure alignment can improve prediction accuracy significantly.

Protein threading problem is NP-hard [36] when two-body interactions are considered and gaps are allowed in a sequence-structure alignment. To reduce the computational cost, heuristics have been incorporated into the alignment process without guarantee of the optimality [27, 32]. On the other hand, threading algorithms based on optimal sequence-structure alignment have also been developed. For example, PROSPECT [64] can find a global optimal alignment with a divide-and-conquer method. Its improved version, PROSPECT II, outperforms some popular fold recognition approaches at all structural similarity levels including family, superfamily and fold levels. RAPTOR [66, 65] formulates the two-body interactions contained in a structure template with a set of linear constraints and uses the linear integer programming approach to minimize the energy function, where computational cost is reduced with a branch-and-bound technique. The performance of RAPTOR on fold recognition improves that of others at both superfamily and fold levels. However, tertiary structure prediction generally requires aligning a sequence to all structure templates in a large database and the overall computational cost for both approaches thus remains high. In addition, an efficient sequence-structure alignment algorithm would allow the implementation of more sophisticated modeling of amino acid interactions with refined and more accurate energy functions.

In this chapter, we introduce a new algorithm for efficient sequence-structure alignments. We model a structure template with a conformational graph and preprocess a sequence to construct an image graph. The optimal alignment between a sequence and a structure template corresponds to finding the minimum valued subgraph isomorphism between the conformational graph and the image graph. Based on graph tree decomposition, alignment between a sequence of M amino acids and a structure template of size N can be performed in time $O(MN + k^t n)$, where n is the number of cores in the structure template, t is the tree

width of the conformational graph. Parameter k, usually small, represents an upper bound for the possible number of locations in the sequence that can be aligned to a given core. In particular, our experiments show that among 3890 protein tertiary structure templates compiled using PISCES [62], only 0.8% of them have treewidth $t > 10$ and 92% have $t < 6$, when using 7.5 Å C_β-C_β distance cut-off for defining two-body interactions. The naturally small tree width t and small parameter k allows the alignment algorithm to run efficiently.

The technique of tree decomposition has been used in protein side-chain packing when back bone is known [67]. A graph theoretical algorithm proposed in [68] formulates the protein threading problem as a graph labeling problem, which can be solved with graph tree decomposition. But the algorithm was not implemented and tested for protein threading. Our work formulates the alignment into subgraph isomorphism. The introduction of parameter k makes it possible to achieve efficiency in threading.

In order to test and evaluate the efficiency and accuracy of the algorithm, we implemented the algorithm in program PROTTD and compared its performance with that of PROSPECT II [35] and RAPTOR. We used protein pairs in the Dali data set [31] to test the alignment accuracy of PROTTD. The alignment accuracy was evaluated by comparing the obtained alignments with those provided by FAST [71]. Our experiments showed that, on average, PROTTD is about 50 times faster than PROSPECT II to obtain better or same alignment accuracy. In addition, we tested the algorithm on the lindahl data set [58] for fold recognition and compared the accuracy with that of RAPTOR at all similarity levels. Our testing results showed that PROTTD achieved significantly improved fold recognition accuracy on superfamily and fold levels.

6.2 Threading Models

6.2.1 Energy Function for Protein Threading

An alignment between a sequence and a structure template is an order-preserved placement of sequence amino acids into the residue locations in the structure template. In general,

alignments are scored with a given energy function and the goal of protein threading is to find the alignment with the minimum score. We used an improved energy function compared with the one used in PROSPECT II [35], which is the weighted sum of contributions from five different energy terms including E_m, E_s, E_p, E_g and E_{ss}. They represent mutation energy, singleton energy, pairwise energies, gap penalty and an energy term that arises from the secondary structure matching respectively. The overall alignment score E_t can be computed as follows.

$$E_t = W_m E_m + W_s E_s + W_p E_p + W_g E_g + W_{ss} E_{ss} \tag{6.1}$$

where W_m, W_s, W_p, W_g and W_{ss} are relative weights for the corresponding energy terms.

The mutation energy E_m is computed with the position-specific score matrices (PSSM) [1] of the structure template and can be formulated as follows.

$$E_m = \sum_{i=1}^{N} \sum_{j=1}^{20} s_{ij} \log \frac{t_{kj}}{t_j} \tag{6.2}$$

where s_{ij} is the frequency of amino acid j in the ith location of the sequence. t_{kj} is the probability for amino acid j in the kth location of the structure template, t_j is the probability that it would be found by chance. N is the size of the sequence. We assume the ith amino acid is aligned to the kth location of the structure template.

The singleton energy term evaluates the preference of the placement of an amino acid in a residue location with certain secondary structure and solvation accessibility. Secondary structure for a given location can be helix, sheet or loop while solvation solvation accessibility can belong to one of the three categories: buried, exposed and intermediate based on the comparison of its solvation accession area with two cut-off thresholds [64]. We thus can compute E_s by

$$E_s = -\sum_{i=1}^{N} \sum_{j=1}^{20} p_{kj} \log \frac{p(j, ss, sa)}{p(j)p(ss, sa)} \tag{6.3}$$

where $p(j, ss, sa)$ is the probability for amino acid j to reside in a residue location with secondary structure ss and solvation accessibility sa; $p(j)$ is its overall probability; and $p(ss, aa)$ is the probability for a residue location with secondary structure ss and solvation

accessibility sa. We assume the ith amino acid is aligned to the kth location in the structure template.

Two residue locations in the structure template are considered to interact if their spatial distance is less than a cut-off value. We compute the overall contribution from two-body interactions by

$$E_p = -\sum_{d(k,l)<d_c} \sum_{s=1}^{20} \sum_{t=1}^{20} f_{is} f_{jt} \log \frac{p(s,t)}{p(s)p(t)} \tag{6.4}$$

where $p(s,t)$ is the probability that amino acids s and t are within the cut-off distance, $p(s)$ and $p(t)$ are overall probabilities for s and t; k and l are the locations in the structure template where residue locations i and j are aligned to; $d(k,l)$ is the spatial distance between the residue locations k and l in the template. We assume d_c is the cut-off distance.

Penalty for gaps in an alignment is evaluated using the affine gap penalty function. Specifically, two parameters, penalties for opening a gap and extending a gap are α and β respectively. The overall gap energies E_g can be thus computed by summing the gap penalties over all groups of contiguous gaps.

$$E_g = \sum_s (\alpha + \beta|s|) \tag{6.5}$$

where s represents a group of contiguous gaps, $|s|$ is its length.

For secondary structure matching energy, we assume residue location i in the sequence is aligned to location k with secondary structure ss_k in the structure template. The overall secondary structure matching energy is thus computed with

$$E_{ss} = \sum_{i=1}^{N} (\gamma - \tau \times p_{ss}(i, ss_k)) \tag{6.6}$$

where γ, τ are constants and $p_{ss}(i, ss_k)$ is the predicted probability for residue location i to be of secondary structure type ss_k. In this paper, we used optimized relative weights for energy terms in our experiments.

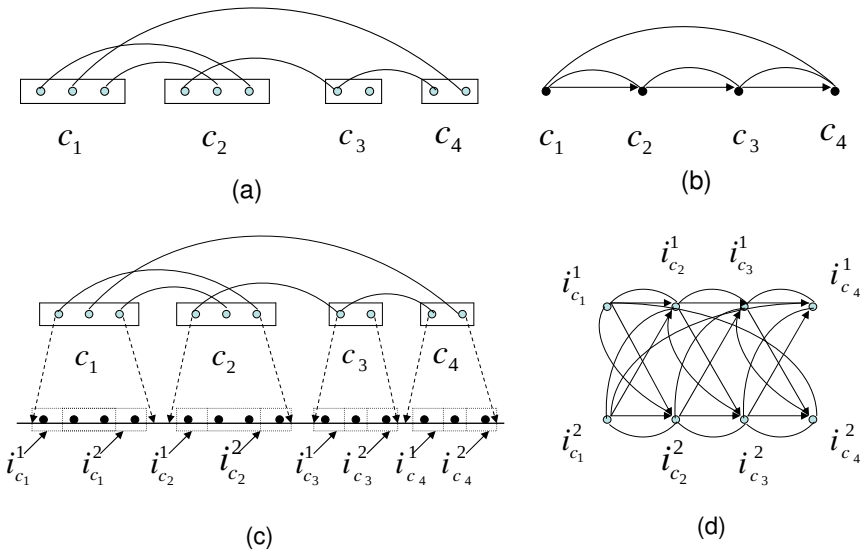

Figure 6.1: (a) A protein structure template that contains both cores and loops. (b) The corresponding conformational graph. (c) The mapped region and images for each core in the structure template (bottom). The dashed lines specify the possible mappings between cores and their images. Core c_i has two possible images ${i^1}_{c_i}$ and ${i^2}_{c_i}$. (d) The image graph formed by the images of cores in the template.

6.2.2 Problem Description

Structural units in a structure template include cores and loops. A *core* contains a row of residue locations where the tertiary structure is highly conserved during the evolution. In contrast, a *loop* consists of the residue locations in between two consecutive cores and its tertiary structure can be highly variable during the evolution. To reduce the computational difficulty for the sequence-structure alignment, gaps are not allowed to appear in core regions. The optimal alignment between a sequence and a structure template is an alignment that minimizes the overall alignment energy E_t under the constraint that cores in the structure template must be aligned with no insertions or deletions.

In this threading algorithm, we define the possible residue locations that can be aligned to a given core to be its *images*. Since residue locations aligned to a core contain no insertions or deletions, for a given sequence, they must be contained in a region limited by the lengths of other cores in the structure template. Such a region exists for each core in the template and is defined as its *mapped region*. To determine the images for a given core, we use its profile specified in the structure template to scan its mapped region and select the residue locations with the k lowest alignment scores, where k is a small parameter that can be determined with a statistical cut-off.

Based on the structure units contained in a structure template, we model the two-body interactions among them with a *conformational graph*. In particular, we use vertices to represent cores in the structure template and cores next to each other in the backbone are connected with directed edges from left to right. In contrast, undirected edges connect two vertices if there exists a two-body interaction with its interacting amino acids contained in the two corresponding cores. Figure 6.1(a)(b) shows a structure template and its corresponding conformational graph.

For a given sequence that needs to be aligned, the images of each core can be efficiently determined in linear time. Using vertices to represent images, an *image graph* can be constructed using an approach similar to the one we have used to construct a conformational graph. Specifically, two images i_u and i_v are connected with an undirected edge if their corresponding cores u and v are connected with an undirected edge in the conformational graph; i_u is connected to i_v with a directed edge if i_u is to the left of i_v and there exists a directed edge from u to v in the conformational graph. Figure 6.1(c)(d) shows the images determined on a sequence for each core in a structure template and the image graph constructed from them. In addition, we assign values to vertices and edges in an image graph, the value associated with a vertex is its alignment score on the corresponding core profile; a directed edge represents a loop and its value is the score of aligning the sequence part between its two ends to the corresponding loop profile in the structure template; the value of an undirected

edge is computed by summing up the energies of all two-body interactions with two ends from the two cores respectively.

An alignment thus corresponds to an embedding of the conformational graph into the image graph. The alignment score is the sum of the values of vertices and edges selected in the image graph to embed the conformational graph. Using the notion of conformational graph and image graph, the problem of optimally aligning a sequence to a structure profile can be formulated as a minimum valued subgraph isomorphism problem. Specifically, a *constraint mapping* f from the conformational graph $G = (V, E)$ and the image graph $H = (I, U)$ satisfies the following conditions.

1. $\forall u \in V$, $f(u) \in I$,
2. There exists integer $k > 0$ such that $\forall u \in V$, $\exists M(u) \subseteq I$, $f(u) \in M(u)$, $|M(u)| \leq k$ and,
3. (u, v) is an edge in G if and only if $(f(u), f(v))$ is an edge in H.

where k is the maximum number of images for all vertices in G. It is the *map width* of the mapping. The objective is to determine the constraint mapping f_m that minimizes the following function:

$$W(f_m) = \sum_{u \in V} w_v(f_m(u)) + \sum_{(u,v) \in E} w_e(f_m(u), f_m(v)) \tag{6.7}$$

where $w_v(x)$ is the value associated with a vertex $x \in I$, $w_e(x, y)$ is the value associated with the edge $(x, y) \in U$.

6.3 Algorithms

We apply the same algorithm we have developed for the Minimum Valued Subgraph Isomorphism problem in Chapter 4. The computation time of the algorithm is thus $O(k^t n)$ where n is the number of vertices in the graph. However, in the preprocessing stage, the algorithm needs to compute the mutation energy that arises from placing residue location

i in the sequence to location j in the structure template for all possible combinations of i and j. The overall time complexity for the algorithm is thus $O(MN + k^t n)$, where M and N are the sizes of the sequence and the structure template respectively. It is clear from the time complexity that the tree width of the available tree decomposition can affect the computation time of the algorithm significantly. Although computing the tree decomposition with the minimum tree width for a given graph is an NP-hard problem [3]. For many graphs, a few heuristics or approximate algorithms [2, 7, 8, 9, 10, 11, 12] can be used to efficiently obtain tree decompositions with small tree width.

6.4 Experiments and Results

We have implemented the tree decomposition based threading algorithm in program PROTTD. PROTTD used a separator based heuristic developed by Bodlaender and Koster [11] to obtain a tree decomposition for each structure template. We tested PROTTD on the Dali data set [31] to evaluate its alignment accuracy and computational efficiency. We used PROTTD to perform sequence-structure alignment for each pair of the 940 protein pairs. We measured the computation time needed for aligning each pair and evaluated the accuracy of structural alignments by comparing the alignments obtained by PROTTD with the structural alignments provided by FAST [71]. In addition, we compared the alignment accuracy and computational efficiency with those of PROSPECT II, which can optimally align a protein sequence to a structure template.

6.4.1 Alignment Accuracy and Computation Time

We constructed the conformational graph for each of the 3890 available structure templates compiled using PISCES [62] and used PROTTD to obtain a tree decomposition for it. We observed that, for around 92% of the available structure templates, the tree widths of their conformational graphs are bounded by 6. This fact suggests that the tree decomposition

TW	PT	NC$(0, 10)$	NC$(10, 20)$	NC$(20, 30)$	NC$(30, 40)$	NC$(40, 50)$	NC$(50, -)$
1	12.04	230	0	0	0	0	0
2	17.85	448	18	1	1	0	0
3	21.86	523	151	15	4	1	0
4	15.61	323	438	81	6	2	0
5	10.78	72	381	132	20	2	0
6	5.30	9	236	128	37	6	3
≥ 7	7.99	1	125	234	105	35	13

Table 6.1: The statistics on tree widths of the conformational graphs for all available 3890 structure templates compiled in PISCES [62]. Column TW specifies value of the tree width; PT is the percentage of structure templates with the specified tree width; NC(a, b) is the number of structure templates that have the specified tree width and their number of cores is greater than a and less than or equal to b.

based alignment can achieve a high computational efficiency. Table 6.1 provides the statistics on the tree widths of all available 3890 structure templates obtained with PROTTD.

To evaluate the alignment accuracy of PROTTD and its computational efficiency, we varied the number of constrained images that a core can be aligned to and performed sequence-structure alignment for protein pairs in the Dali test set. The alignment accuracy can be obtained by identifying the percentage of perfectly aligned residues and those aligned with a shift of less than four amino acids. We computed the alignment accuracy for both PROTTD and PROSPECT II on a given protein pair by comparing the alignments against its FAST structural alignment. We observed that, for most of the sequence pairs in the test set, PROTTD can yield structural alignments as accurate as those provided by PROSPECT II when the number of constrained images is around 7. Table 6.2 shows a comparison of its alignment accuracy with PROSPECT II on several selected protein pairs in the Dali test set. Table 6.3 shows the computation time of PROSPECT II and the amounts of relative speed up gained with PROTTD on these pairs. It can be clearly seen from both tables that, on average, PROTTD can achieve better or the same alignment accuracy as

PR	TP	CN	TW	PROTTD				PROSPECT II
				$k = 3$	$k = 5$	$k = 7$	$k = 9$	
1hyha	1ldm	21	5	82.45	90.57	90.57	90.57	85.33
1gca	2dri	21	6	88.16	93.52	93.52	93.52	90.73
1gtma	1leha	19	5	80.71	83.69	83.69	83.69	83.69
1cdka	1csn	13	4	79.52	81.63	82.54	82.54	80.83
1cb2a	1tml	16	5	70.63	74.34	75.41	75.41	73.55
1prn	2por	17	3	68.02	71.10	71.10	71.10	71.10
2omf	2por	17	3	59.38	63.48	64.21	64.21	61.76
1lcl	1slta	10	4	60.41	63.63	65.02	65.02	62.59
1bfg	1hce	12	5	47.83	52.06	53.84	53.84	51.79
1mfa	3cd4	11	3	55.32	60.54	60.54	60.54	60.54

Table 6.2: The alignment accuracy obtained by PROTTD on a few sequence-structure alignments in the Dali test set using different constrained image numbers k. PR is protein sequence in the sequence-structure alignment; TP is that of the structure template; CN is the total number of cores in the structure template and TW is the tree width of its conformational graph; alignment accuracy is in percentage.

that of PROSPECT II when its parameter k is greater than 5. It only uses a small fraction of the computation time needed by PROSPECT II to obtain sequence-structure alignments with better or same accuracy.

6.5 Summary of the Chapter

In this chapter, we introduce an efficient parameterized graph algorithm for protein threading. Based on this algorithm, we are able to align a sequence of length M to a structure template with n cores and of size N in time $O(MN + k^t n)$, where t is the tree width of the conformational graph and k is the upper bound of constrained images for cores. The comparison of its performance with that of PROSPECT II and RAPTOR showed that the approach is promising for solving problems of higher computational complexity. For example, recent work [67] has shown that tree decomposition based approach can provide

PR	TP	CN	TW	PROTTD(s)				PROSPECT II(s)	SP
				$k=3$	$k=5$	$k=7$	$k=9$		
1hyha	1ldm	21	5	2.72	4.09	12.23	23.96	256.04	62.60×
1gca	2dri	21	6	2.72	5.11	17.47	26.96	376.32	73.64×
1gtma	1leha	19	5	4.91	5.93	16.12	34.86	319.91	53.95×
1cdka	1csn	13	4	3.17	4.20	6.38	18.63	150.40	35.81×
1cb2a	1tml	16	5	3.68	7.19	15.74	33.03	220.29	30.64×
1prn	2por	17	3	2.80	3.49	4.71	7.24	180.49	51.71×
2omf	2por	17	3	3.33	4.70	5.51	8.05	190.46	40.52×
1lcl	1slta	10	4	0.65	0.93	1.01	2.21	75.51	81.19×
1bfg	1hce	12	5	0.51	1.78	5.69	12.34	98.52	55.34×
1mfa	3cd4	11	3	1.65	2.43	4.13	9.59	150.27	61.84×

Table 6.3: The computation time in seconds needed by PROTTD on the same protein pairs in the Dali test set using different constrained image numbers k. SP is the relative amount of speed up while PROTTD is used to achieve an alignment accuracy close to that of PROSPECT II.

optimal solutions for the protein side-chain packing problem efficiently. In [68], the tree decomposition based approach was shown to be more efficient than the linear programming approach on the side-chain packing problem. We thus expect that the two subproblems, protein threading and side-chain packing can be formulated in a single optimization problem. In addition, based on graph tree decomposition, an efficient algorithm can possibly be developed for this problem and we expect that the accuracy for tertiary structure prediction can be further improved.

Chapter 7

Conclusions and Future Work

This dissertation studies the sequence-structure alignment problem in structural bioinformatics from an alternative yet original perspective. In particular, we view a structure profile (or template) as the combination of structure units, which may include stems and loops for an RNA structure and cores and loops for a protein structure template. The score of an alignment between a sequence and a structure is the sum of the alignment scores obtained on sequence segments that are aligned to the structure units in the template.

The sequence-structure alignment problem can then be reduced to a graph optimization problem. In particular, each nontrivial structure unit in a structure profile (or template) can be represented with a graph vertex and two vertices that represent interacting nontrivial structure units are connected with non-directed edges. In addition, the backbone of the structure can be represented by a set of directed edges that connect the vertices of two structure units neighoring along the sequence backbone. A structure profile (or template) can then be described by the resulting structure graph.

A sequence that is to be aligned to the structure profile can be preprocessed by the structure profile for each structure unit and the possible sequence segments that can be aligned to each structure unit are selected based on the alignment scores. Each of these candidates can be represented with a graph vertex and a sequence graph can be similarly constructed to model the sequence. The vertices and edges in the sequence can be associated with values computed from the alignment scores obtained on the candidates.

The sequence-structure alignment problem is then reduced to finding in the sequence graph a maximum valued (or minimum valued) subgraph that is isomorphic to the structure

graph. Based on the biochemical property of structure units, the number of candidates for a given structure unit is often bounded by a small integer. Such an integer is defined to be the map width of the isomorphic mapping. Moreover, most of the structure graphs that we have investigated for modeling noncoding RNA secondary structure and protein threading have bounded tree width. Based on the constraints imposed by the map width of the isomorphic mapping and the tree width of the structure graph, a variant of the Maximum Valued Subgraph Isomorphism problem was developed in Section 4.4. However, this problem remains computationally intractable in general.

Vertices in a structure graph are ordered based on the structure backbone, and those in a sequence graph are partially ordered due to the presence of overlapping candidates. Based on these two additional properties, the Maximum Subgraph Isomorphism problem can be efficiently solved based on a tree decomposition of the structure graph. We have implemented this algorithm and tested its performance on searching genomes for non-coding RNAs and protein threading for tertiary structure prediction. Our testing results show that computational tools based on this algorithm are significantly faster than previous methods while achieving the same or better accuracy.

In the following sections, we discuss several problems that appear ripe for future research. These problems are related to what we have done on the sequence-structure problem. We believe that methods similar to those described above can possibly be used to solve these problems.

7.1 Alternative Structure Models for Noncoding RNAs

In Chapter 4, we developed a graph model to describe the secondary structure of an RNA family and reduce the sequence-structure alignment to a maximum valued subgraph isomorphism problem. In practice, an RNA family often consists of several subfamilies. The structure of a subfamily is slightly different from that of others. It is possible that some sequences contain structures with structure units from two or more subfamilies. For each

subfamily, a different structure model is thus needed to describe the corresponding structure unit. To improve the sensitivity of the search, we can further consider the probability for the structure model from each of the subfamily. In addition, deletions and insertions of substructures can also happen during the process of evolution. This information also needs to be incorporated into the structure model to improve its flexibility.

We thus need to slightly change the strategy we have used to select the candidates for each structure unit. In particular, we consider using all possible structure models for a given stem to scan the sequence and candidates are selected for each of the structure models. The rest of the algorithm can remain unchanged and a sequence-structure alignment with the optimal score can be determined. The sensitivity of this new method can possibly be improved since this alternative model based search can identify sequences that contain structure units from different subfamilies. These sequences are highly likely to be the intermediate conformations for the RNA molecule during the process of evolution.

The deletion of a stem could be modeled by introducing a "void" candidate to its candidate list. Several different loops can be combined into a single loop due to the selection of such a candidate. However, we may still consider these loops separately to simplify the implementation. The insertion of stems is more difficult to accommodate and need further investigation in the future.

We have implemented part of the strategies described above in our searching program and our testing results on searching have shown that alternative model based search can significantly improve the sensitivity of searching. Our future work may focus on developing new structure models that can smoothly deal with the insertion and deletion of substructures and we expect that these new structure models can be used to search for remote homologs.

7.2 Protein Threading with Variable Cut-off Distances

An interesting but difficult problem in protein threading is to consider the change of the computational complexity of performing sequence-structure alignment when the cut-off dis-

tance for two-body interactions varies from zero to infinity. It is not difficult to see that sequence-structure alignment can be solved in polynomial time when the cut-off distance is 0 and it is an NP-hard problem when the cut-off distance is infinity. An important problem associated with this is where and how the tractability of the problem changes.

To be more specific, we provide a geometric picture on the problem. We can model a residue with a sphere in three dimensional space. The diameter of the sphere is the cut-off distance. We use a graph vertex to represent a sphere and two vertices are connected if the corresponding spheres intersect. The resulting graph is also a structure graph. Based on our work on sequence-structure alignment, the tree width of this structure graph can be expected to be important for developing efficient algorithms for sequence-structure alignment.

Along the lines of the complexity results for graph optimization problems on planar graphs, we offer the following conjecture.

Conjecture 7.2.1 *If the radius of the residue is R_r, there exists a critical $d_c = f(R_r)$ such that when the cut-off distance d satisfies $d > d_c$, the sequence-structure alignment problem is NP-hard. In addition, for each fixed d that is finite, there exists a subexponential algorithm for computing the optimal sequence-structure alignment.*

A reduction from an NP-hard problem or a counter example might be needed to prove or disprove this conjecture. The most interesting problem is the determination of the function f. We believe this function is closely related to the geometry of three dimensional space.

7.3 Summary of the Chapter

This chapter summarizes the previous chapters of this dissertation. Sequence-structure alignment is a classic problem in structure bioinformatics. The structure of a biological sequence can be complex and the general sequence-structure alignment problem is thus computationally intractable. This dissertation provides an original perspective on this problem and shows that it can be reduced to the Maximum Valued Subgraph Isomorphism problem.

This problem can be efficiently solved when the guest graph and host graph satisfy certain constraints.

This chapter also describes several problems that are related to the sequence-structure alignment problem and seem to deserve further investigation. These problems include searching noncoding RNAs with alternative models and protein threading with variable cut-off distances. These problems are either of practical or theoretical value and finding answers to them would be an interesting extension of the work presented in this dissertation.

Bibliography

[1] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs", *Nucleic Acids Research*, 25(17): 3389-3402, 1997.

[2] E. Amir, "Efficient approximation for triangulation of minimum treewidth", *Proceedings of 17th Conference on Uncertainty in Artificial Intelligence*, 7-15, 2001.

[3] S. Arnborg, D. G. Corneil, and A. Proskurowski, "Complexity of finding embeddings in a k-tree.", *SIAM Journal on Algebraic and Discrete Methods*, 8: 277-284, 1987.

[4] S. Arnborg and A. Proskurowski, "Linear time algorithms for NP-hard problems restricted to partial k-trees.", *Discrete Applied Mathematics*, 23: 11-24, 1989.

[5] E. Bachoore and A. Proskurowski, "Characterization and recognition of partial 3-trees", *SIAM Journal on Algebraic and Discrete Methods*, 7: 305-314, 1986.

[6] V. Bafna and S. Zhang, "FastR: Fast database search tool for non-coding RNA.", *Proceedings of the 3rd IEEE Computational Systems Bioinformatics Conference*, 52-61, 2004.

[7] H. L. Bodlaender, "Classes of graphs with bounded tree-width.", *Tech. Rep. RUU-CS-86-22*, Dept. of Computer Science, Utrecht University, the Netherlands, 1986.

[8] H. L. Bodlaender, "Better algorithms for the pathwidth and treewidth of graphs.", *Proceedings of the 18th international Colloquium on Automata, Languages and Programming*, Springer Verlag, Lecture Notes in Computer Science, 510: 544-555, 1991.

[9] H. L. Bodlaender, "Better algorithms for the pathwidth and treewidth of graphs.", *Proceedings of the 18th international Colloquium on Automata, Languages and Programming*, Springer Verlag, Lecture Notes in Computer Science, 510: 544-555, 1991.

[10] H. L. Bodlaender, "A linear time algorithm for finding tree-decompositions of small treewidth", *SIAM Journal on Computing*, 25: 1305-1317, 1996.

[11] H. L. Bodlaender and A. M. C. A. Koster, "Safe separators for treewidth", *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments*, 70-94, 2004.

[12] H. L. Bodlaender, A. M. C. A. Koster, and T. Wolle, "Contraction and Treewidth Lower Bounds", *Proceedings of the 12th Annual European Symoposium on Algorithms*, 628-639, 2004.

[13] M. Brown and C. Wilson, "RNA Pseudoknot Modeling Using Intersections of Stochastic Context Free Grammars with Applications to Database Search.", *Pacific Symposium on Biocomputing*, 109-125, 1995.

[14] L. Cai, R. Malmberg, and Y. Wu, "Stochastic Modeling of Pseudoknot Structures: A Grammatical Approach.", *Bioinformatics*, 19, $i66 - i73$, 2003.

[15] L. Cai, Y. Song, X. Huang, C. Liu, and J. Zhao, "A Case Study of Parameterized Reduction to MSO logic", manuscript.

[16] J. Chen, I. A. Kanj, and W. Jia, "Vertex Cover: Further Observations and Further Improvements", *Journal of Algorithms* 41(2): 280-301, 2001.

[17] F. Clautiaux, J. Carlier, A. Moukrim, and S. Négre, "New lower and upper bounds for graph treewidth", *Proceedings of International Workshop on Experimental and Effiicent Algorithms, Lecture Notes in Computer Sciences*, 2647: 70-80, 2003.

[18] B. Courcelle, "The Monadic Second Order Theory of Graphs I: Recognisable Sets of Finite Graphs", *Information and Computation*, 85(1):12-75, 1990.

[19] A. T. Dandjinou, N. Lévesque, S. Larose, J. Lucier, S. A. Elela, and R. J. Wellinger, "A Phylogenetically Based Secondary Structure for the Yeast Telomerase RNA.", *Current Biology*, 14: 1148-1158, 2004.

[20] R. G. Downey and M. R. Fellows, "Parameterized Complexity", Monographs in Computer Science, *Springer*, 1997.

[21] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*, Cambridge Univ. Press, 1998.

[22] S. Eddy and R. Durbin, "RNA sequence analysis using covariance models.", *Nucleic Acids Research*, 22: 2079-2088, 1994.

[23] D. Eppstein, "Subgraph Isomorphism in Planar Graphs and Related Problems.", *Journal of Graph Algorithms and Applications*, 3.3: 1-27, 1999.

[24] D. N. Frank and N. R. Pace, "Ribonuclease P: unity and diversity in a tRNA processing ribozyme.", *Annu Rev Biochem.*, 67: 153-180, 1998.

[25] M. R. Gary and D. S. Johnson, "Computers and Intractability", *Freeman*, 1979.

[26] D. Gautheret and A. Lambert, "Direct RNA motif definition and identification from multiple sequence alignments using secondary structure profiles.", *Journal of Molecular Biology*, 313: 1003-1011, 2001.

[27] A. Godzik, A. Kolinski, and J. Skolnick, "Topology fingerprint approach to the inverse folding problem", *Journal of Molecular Biology*, 227: 227-238, 1992.

[28] S. J. Geobel, B. Hsue, T. F. Dombrowski, and P. S. Masters, "Characterization of the RNA components of a Putative Molecular Switch in the 3' Untranslated Region of the Murine Coronavirus Genome.", *Journal of Virology*, 78: 669-682, 2004.

[29] B. Knudsen, J. Wower, C. Zwieb, J. Gorodkin, "tmRDB (tmRNA database).", *Nucleic Acids Research*, 29(1): 171-172, 2001.

[30] M. Hajiaghayi and N. Nishimura, "Subgraph isomorphism, log-bounded fragmentation and graphs of (locally) bounded treewidth", *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science*, 305-318, 2002.

[31] L. Holm and C. Sander, "Decision support system for evolutionary classification of protein structures", Proceedings of International Conference on Intelligent Systems for Molecular Biology, 5: 140-146, 1997.

[32] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S. R. Eddy, "Rfam: an RNA family database.", *Nucleic Acids Research*, 31: 439-441, 2003.

[33] D. T. Jones, "Protein secondary structure prediction based on position-specific scoring matrices", *Journal of Molecular Biology*, 292(2): 195-202, 1999.

[34] R. J. Klein and S. R. Eddy, "RSEARCH: Finding Homologs of Single Structured RNA Sequences.", *BMC Bioinformatics*, 4:44, 2003.

[35] D. Kim, D. Xu, J. Guo, K. Ellrott, and Y. Xu, "Prospect II: protein structure prediction program for genome-scale applications", *Protein Engineering*, 16(9): 641-650, 2003.

[36] R. H. Lathrop, "The protein threading problem with sequence amino acid interaction preferences is NP-complete", *Protein Engineering*, 7(9): 1059-1068.

[37] C. Liu, Y. Song, R. Malmberg, and L. Cai, "Profiling and Searching for RNA Pseudoknot Structures in Genomes.", Proceedings of 2005 International Workshop in Bioinformatics Research and Applications, to appear.

[38] T. M. Lowe and S. R. Eddy, "tRNAscan-SE: A Program for Improved Detection of Transfer RNA genes in Genomic Sequence.", *Nucleic Acids Research*, 25: 955-964, 1997.

[39] B. Lucena, "A new lower bound for tree-width using maximal cardinality search.", *SIAM Journal on Discrete Mathematics*, 16: 345-353, 2003.

[40] J. Matousek and R. Thomas, "On the complexity of finding iso- and other morphisms for partial k-trees.", *Discrete Mathematics*, 108: 343-364, 1992.

[41] N. Nameki, B. Felden, J. F. Atkins, R. F. Gesteland, H. Himeno, and A. Muto, "Functional and structural analysis of a pseudoknot upstream of the tag-encoded sequence in E. coli tmRNA.", *Journal of Molecular Biology*, 286(3): 733-744, 1999.

[42] S. B. Needleman, and C. D. Wunsch, " A general method applicable to the search for similarities in the amino acid sequence of two proteins.", *Journal of Molecular Biology*, 48: 443-453, 1970.

[43] G. L. Nemhauser and L. E. Trotter, "Vertex packing: structural properties and algorithms", *Mathematical Programming*, 8: 232-248, 1975.

[44] V. T. Nguyen, T. Kiss, A. A. Michels, and O. Bensaude, "7SK small nuclear RNA binds to and inhibits the activity of CDK9/cyclin T complexes.", *Nature* 414: 322-325, 2001.

[45] K. G. Olesen and A. L. Madsen, "Maximal prime subgraph decomposition of Bayesian networks", *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 32: 21-31, 2002.

[46] L. Perković and B. Reed, "An improved algorithm for finding tree decompositions of small tree width", *Proceedings of the 25th International Workshop on Graph Theoretic Concepts in Computer Science, Lecture Notes in Computer Science*, 1665: 148-154, 1999.

[47] E. Rivas and S. Eddy, "The language of RNA: a formal grammar that includes pseudoknots.", *Bioinformatics*, 16: 334-340, 2000.

[48] E. Rivas and S. R. Eddy, "Noncoding RNA gene detection using comparative sequence analysis.", *BMC Bioinformatics*, 2:8, 2001.

[49] E. Rivas, R. J. Klein, T. A. Jones, and S. R. Eddy, "Computational identification of noncoding RNAs in E. coli by comparative genomics.", *Current Biology*, 11: 1369-1373, 2001.

[50] E. Rivas and S. R. Eddy, "A dynamic programming algorithm for RNA structure prediction including pseudoknots.", *Journal of Molecular Biology*, 285: 2053-2068, 1999.

[51] N. Robertson and P. D. Seymour, "Graph Minors II. Algorithmic aspects of tree-width.", *Journal of Algorithms*, 7: 309-322, 1986.

[52] N. Robertson and P. D. Seymour, "Graph Minors III. Planar tree width.", *Journal of Combinatorial Theory Series B*, 36: 49-64, 1984.

[53] N. Robertson and P. D. Seymour, "Graph Minors IV. Tree width and well-quasi-ordering.", *Journal of Combinatorial Theory Series B*, 48: 227-254, 1990.

[54] N. Robertson and P. D. Seymour, "Graph Minors V. Excluding a planar graph.", *Journal of Combinatorial Theory Series B*, 41: 92-114, 1986.

[55] N. Robertson and P. D. Seymour, "Graph Minors X. Obstructions to tree decomposition.", *Journal of Combinatorial Theory Series B*, 52: 153-190, 1991.

[56] N. Robertson and P. D. Seymour, "Graph Minors XIII. The disjoint paths problem.", *Journal of Combinatorial Theory Series B*, 63: 65-110, 1995.

[57] P. D. Seymour and R. Thomas, "Call routing and the rat catcher", *Combinatorica* 14: 127-241, 1994.

[58] J. Shi, T. L. Blundell, and K. Mizuguchi, "FUGUE: sequence-structure homology recognition using environment-specific substitution tables and structure- dependent gap penalties", *Journal of Molecular Biology*, 310(1): 243-257, 2001.

[59] Y. Song, C. Liu, X. Huang, R. Malmberg, Y. Xu, and L. Cai, "Efficient parameterized algorithms for biopolymer structure-sequence alignment", *Proceedings of the Fifth*

Workshop on Algorithms in Bioinformatics, Lecture Notes in Bioinformatics 3692, 376-388, 2005.

[60] Y. Song, C. Liu, R. L. Malmberg, F. Pan, and L. Cai, "Tree decomposition based fast search of RNA structures including pseudoknots in genomes", *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference*, 223-224, 2005.

[61] Y. Uemura, A. Hasegawa, Y. Kobayashi, and T. Yokomori, "Tree adjoining grammars for RNA structure prediction.", *Theoretical Computer Science*, 210: 277-303, 1999.

[62] G. Wang and R. L. Dunbrack, Jr. "PISCES: a protein sequence culling server", *Bioinformatics*, 16: 257-268, 2000.

[63] Z. Weinberg and W. L. Ruzzo, "Faster genome annotation of non-coding RNA families without loss of accuracy.", *Proceedings of the Eighth Annual International Conference on Computational Molecular Biology*, 243-251, 2004.

[64] Y. Xu, D. Xu and E. C. Uberbacher, "An Efficient Computational Method for Globally Optimal Threading", *Journal of Computational Biology*, 5: 597-614, 1998.

[65] J. Xu, M. Li, D. Kim, and Y. Xu, "RAPTOR: Optimal Protein Threading by Linear Programming", *Journal of Bioinformatics and Computational Biology*, 1(1): 95-117, 2003.

[66] J. Xu, M. Li, D. Kim, and Y. Xu, "Protein threading by linear programming", *Biocomputing: Proceedings of the 2003 Pacific Symposium*, 264-275, 2003.

[67] J. Xu, "Rapid Problem Side-Chain Packing via Tree Decomposition", Proceedings of the Ninth Annual International Conference on Research in Computational Molecular Biology, 423-429, 2005.

[68] J. Xu, F. Jiao, and B. Berger, "A Tree-Decomposition Approach to Protein Structure Prediction", *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference*, 247-256, 2005.

[69] Z. Yang, Q. Zhu, K. Luo, and Q. Zhou, "The 7SK small nuclear RNA inhibits the Cdk9/cyclin T1 kinase to control transcription.", *Nature* 414: 317-322, 2001.

[70] V. V. Zeenko, L. A. Ryabova, A. S. Spirin, H. M. Rothnie, D. Hess, K. S. Browning, and T. Hohn, "Eukaryotic Elongation Factor 1A Interacts with the Upstream Pseudoknot Domain in the 3' Untranslated Region of Tobacco Mosaic Virus RNA.", *Journal of Virology*, 76(11): 5678-5691, 2002.

[71] J. Zhu and Z. Weng, "FAST: A Novel Protein Structure Alignment Algorithm", *Proteins: Structure, Function and Bioinformatics*, 58(3): 618-627, 2005.