Maximum Spanning k-Trees: Models and Applications

by

Liang Ding

(Under the Direction of Liming Cai)

Abstract

The spanning k-tree for bounded k is a notion extended from the spanning tree of a graph, which serves many applications ranging from network reliability to machine learning. k-trees are intimately related to graphs of bounded tree width; problems involving k-trees are potentially have efficient solutions. However, on given general graphs, the problem to produce a maximum spanning k-tree (MSkT) is NP-hard, even for $k = 2$, and remains intractable for many well-known restricted families of graphs.

This thesis investigates effective models derived from MSkT, where efficient algorithms are designed to compute optimal and near optimal solutions with different objective functions defined on the models. The development of the models is motivated by the increasing real-world interests that seek answers about complex relations from given input data. This research is of particular interest to coping with the computational intractability that routinely arises from problems in many emerging applications, such as bioinformatics, machine learning, big data analytics, and social networks. The success of the models has been based on non-conventional, non-trivial graph metrics that can well characterize many application problems and can lead to efficient graph optimization algorithms.

Index words: Spanning k-tree, Backbone k-tree, RNA 3D structure, Machine learning, Bayesian network learning, Gene expression analysis, Computational biology, Cancer bioinformatics

Maximum Spanning k-Trees: Models and Applications

by

Liang Ding

B.S., Zhengzhou University, China, 2007

M.S., University of Texas - Pan American, 2011

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

2016

©2016

Liang Ding

All Rights Reserved

Maximum Spanning k-Trees: Models and Applications

by

Liang Ding

Approved:

Major Professor:	Liming Cai
Committee:	E. Rod Canfield Russel L. Malmberg Ying Xu

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
August 2016

Acknowledgments

First of all, I would like to thank my major advisor, Liming Cai, for his guidance throughout my Ph.D.'s program. I very much appreciate his theoretical and bioinformatics insights. I also very grateful of our frequent meetings along with numerous email communications.

I would like to thank my advisory committee members for their time and helpful advices to clarify my research ideas and refine my thesis. In particular, I am thankful to Russel L. Malmberg for his thorough revisions of the paper drafts and helpful bioinformatics insights for the development of my research. I also greatly appreciate Ying Xu and the members in the Computational Systems Biology Laboratory for sharing the data and their valuable opinions on the cancer bioinformatics research.

I have also had many helpful discussions with other faculties in the Department of Computer Science and the graduate students in the RNA Informatics Lab. I am specially thankful to Robert W. Robinson and Guojun Li for their guidance in the development of k-tree models. And thanks to Sal LaMarca, Mohammad Mohebbi, Abdul Samad, Xingran Xue, among others.

Table of Contents

List of Figures

Page

List of Tables

Page

Chapter 1

Introduction

Treewidth is an important metric on graphs to measure the degree to which a graph is tree-like. Stemming from the work in graph minors [Robertson and Seymour, 1986], treewidth and its associated technical notion of tree decomposition have provided useful approaches to cope with the computational intractability for a large number of graph-theoretic problems [Arnborg, 1985; Arnborg and Lagergren, 1991; Bern et al., 1987; Bodlaender, 1988; Courcelle, 1990; Matousek and Thomas, 1992; Eppstein, 1999; Demaine and Hajiaghayi, 2008]. Typically, on input graphs of treewidth bounded by constant integer $k \geq 1$ (along with a tree decomposition), many NP-hard problems can be solved in polynomial $O(n^{O(k)})$ or even linear or quadratic time. However, rather than taking a graph of bounded treewidth as input, graph problems formulated for more sophisticated applications are often required to produce such a graph as output. One such problem is to take as input an unrestricted graph and produce a spanning subgraph, called a *spanning k-tree*, as output.

A generalization of the tree concept [Beineke and Pippert, 1971; Rose, 1974], the notion of k-tree is intimately related to graphs of bounded treewidth. In fact, a graph has treewidth at most k if and only if it is a subgraph of a k-tree [Bodlaender, 1993]. The *maximum spanning k-tree* (MSkT) problem seeks to produce a spanning k-tree that maximizes the total weight of edges contained in the desired k-tree [Bern, 1987]. The problem MSkT models a wide range of applications, including network deployment [Bern, 1987], communication reliability [Cai and Maffray, 1993; Cai, 1995; Liao and Zhang, 2009], and statistical network learning [Karger and Srebro, 2001; Srebro, 2001; Bradley and Guestrin, 2010]. However, MSkT is NP-hard even for $k = 2$ [Bern, 1987; Cai and Maffray, 1993] and remains inherently hard for

many constrained families of graphs [Cai and Maffray, 1993; Liao and Zhang, 2009] including, for example, the families of split graphs, graphs with maximum degree at most $3k + 2$, and planar graphs (for $k = 2$).

Unlike the problems investigated in the past more than three decades, where computation was often based on given graphs of small tree width, newer, more challenging research problems often seek to discover such graphs from large scale yet often unstructured data. However, due to the immense space of possible graphs, discovering an optimal or near optimal graphs that can characterize seamlessly the input data is formidable even for tree width ≤ 2. Therefore, beside the tree width, it is necessary to find constraints that can be imposed on the graphs, where the constraints are from intrinsic properties with respect to a specific problem. This thesis summarizes the studies of discovering tree width bounded graphs, specifically, spanning k-tree graphs, with various constraints on the graphs. Some of the constraints, such as single backbone constraint and multiple backbones constraint are motivated by the characteristics of the problems in computational biology and machine learning. While other constraints, e.g., spanning tree constraint, are obtained from analyzing the existing real data.

This thesis begins by introducing the concept of k-tree on the family of *backbone graphs* [Ding et al., 2016c], which contain a labeled Hamiltonian path that the produced maximum spanning k-tree is required to include. Spanning k-trees on backbone graphs can ideally model sophisticated yet tameable structures imposed on linear lists. We then examine the special properties of spanning k-tree on the backbone graphs. Efficient algorithms and their optimality are discussed in the following sections.

We also relate the graph theory of backbone k-trees to linguistic grammars [Ding et al., 2014a]. Chapter 3 introduces a class of novel stochastic grammars, called *stochastic k-tree grammar* (SkTG), for the analysis of context-sensitive languages. With the new grammar rules, co-occurrences of distant terminals are characterized and recursively organized into k-tree graphs. It is shown that probabilistic analysis of k-trees over strings are computable

in polynomial time $n^{O(k)}$.

Chapter 4 introduces a method, based on backbone k-tree, to tackle a key step in the RNA 3D structure prediction problem: the prediction of the nucleotide interactions that constitute the desired 3D structure [Ding et al., 2015]. The backbone k-tree is adopted to tightly constrain the nucleotide interaction relationships considered for RNA 3D structures. It is shown that the new model makes it possible to efficiently compute the optimal set of nucleotide interactions (including the non-canonical interactions in all recently revealed families) from the query sequence along with known or predicted canonical basepairs. The preliminary results indicate that in most cases the new method can predict with a high accuracy the nucleotide interactions that constitute the 3D structure of the query sequence.

In the chapter that follows, we proceed to present our works for atomic-grain 3D structure prediction from single RNA sequence [Ding et al., 2016b]. Through modeling the RNA 3D structure as a collection of interconnected geometric motifs formed by selected quadruplets (4-cliques) of nucleotides. A two-step process is used in this work. Both canonical and non-canonical interactions between nucleotides are first predicted using the method illustrated in the earlier chapter and organized into a non-geometric 3-tree graph. Then a 3D model is computed by optimally assigning to 4-cliques in the predicted graph with geometric motifs identified from a knowledge base. Because both steps utilize exact graph-theoretic algorithms, the new method proves quite effective. The preliminary results show the decent performance of the method for 3D structure prediction on RNA sequences including those of length beyond 100 nucleotides.

The last chapter of the thesis describes our preliminary work of using k-tree model to learn Markov or Bayesian networks. We intend to address the under performance issue in Markov/Bayesian network learning due to its computational intractability with the notion of k-tree and its potentially efficient algorithms. The aims of this project also include the investigation of the tractability for a class of problems associated with k-tree optimization and the translation of learned networks of correlations into causal relationships. This thesis

discusses the preliminary ideas of linking the k-tree model with network learning and gene network learning for causality inference in cancer bioinformatics.

Chapter 2

Maximum Spanning k-Trees on Backbone Graphs

In this chapter, we investigate efficient algorithms for MSkT on the family of graphs that always contain a labeled Hamiltonian path (called *backbone graphs*). We tailor the problem MSkT for such graphs to require the output spanning k-tree produced by MSkT algorithms to contain the Hamiltonian path.

Spanning k-trees on backbone graphs can ideally model sophisticated yet tameable structures imposed on linear lists, such as neurocognitive linguistic sentences and bio-molecular sequences, both of great interest in biomedical research. In particular, this current work is initially motivated by an algorithmic need from research in bio-molecule 3D structure prediction, one of the most challenging areas in bioinformatics where algorithm efficiency is still a bottleneck toward deliverable solutions [Zhang, 2008; Roy and Zhang, 2012; Abual-Rub and Abdullah, 2008; Istrail and Lam, 2009; Leontis and Westhof, 2012b; Rangwala and Karypis, 2010]. Nevertheless, the authors, it has recently been observed [Xu, 2005; Xu and Berger, 2005; Song et al., 2005; Xu et al., 2007; Huang et al., 2008b; Shareghi et al., 2012] that known protein and RNA 3D structures in PDB [Berman et al., 2000a] overwhelmingly have small treewidth in their interaction topology graphs. Thus MSkT on backbone graphs turns out to be an appropriate model for building 3D molecular structures. In particular, given a molecule sequence containing n residues, we can construct a (complete) graph of n vertices forming a Hamiltonian path and edges for potential interactions among non-consecutive residues. A yielded maximum spanning k-tree from such a complete graph consists of the backbone and most plausible residue-residue interactions subject to the k-tree topology, which can be used as a basis for a 3D construction (see Figure 2.1). Therefore, efficient algorithms for MSkT

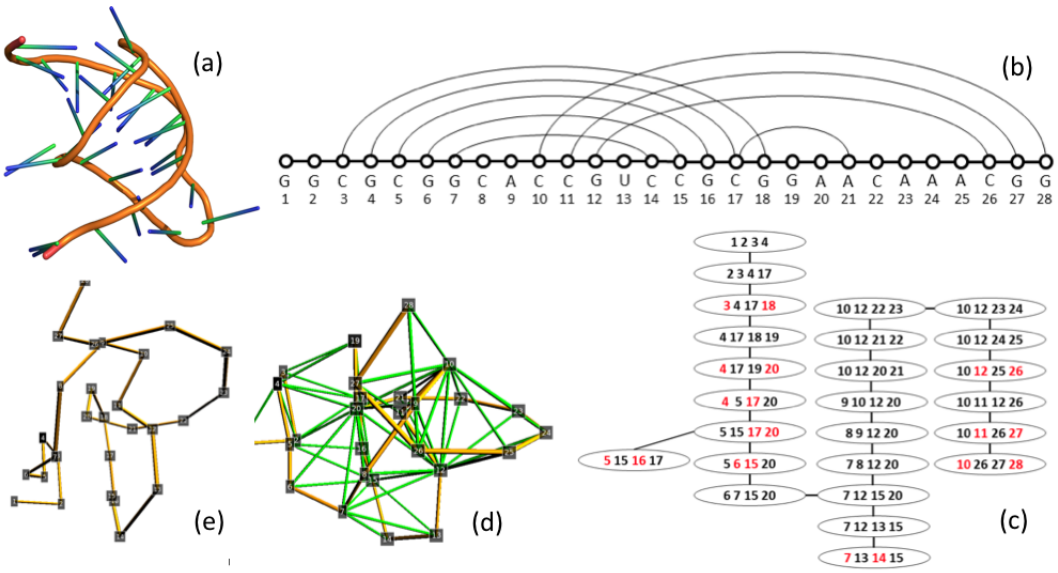


Figure 2.1: An illustration for the role of the (backbone) maximum spanning k-tree problem in RNA tertiary structure prediction. (a) The tertiary structure of an RNA molecule example (PDB ID: 1L2X) with 28 nucleotides; (b) a backbone graph for the RNA sequence, with edges corresponding to residue-residue interactions in the known tertiary structure (all other possible edges are not shown); (c) a maximum spanning partial k-tree, $k = 3$, found for the backbone graph, represented in a tree topology of $(k+1)$-cliques (see section 3); (d) 3D geometric model of the maximum spanning k-tree with one tetrahedron for every 4-clique; and (e) with non-backbone geometric lines removed, the folding of backbone is the predicted preliminary 3D structure (to be refined).

on backbone graphs become significant to feasible 3D structure prediction from molecular sequences.

In this chapter, we reveal several important properties possessed by spanning trees on backbone graphs, which have allowed us to develop an efficient dynamic programming algorithm for MSkT on backbone graphs. The time and space complexities of the algorithm are $O(k(k+1)^{k+2}n^{k+1})$ and $O(2^{k+1}n^k)$, respectively. We further show evidence that the n^{k+1} factor in the time complexity is unlikely to be improved. First, we prove that, when the objective function is sum of weights on $(k+1)$-cliques in the k-tree, the problem remains NP-hard and W[1]-hard, therefore excluding the parameterized tractability for MSkT on backbone graphs. Second, reduction in the exponent $k+1$ of n^{k+1} would meet with the

barrier to improve the efficiency of context-free language parsing. On the other hand, we point out that both the time and space complexities of our algorithm are suitable for its implementation in many situations of the biological real world, where $n \leq 250$ and $k \leq 4$.

2.1 Preliminaries

Definition 2.1.1. [Beineke and Pippert, 1971] Let $k \geq 1$ be an integer. A *k-tree* is a graph that can be generated via the following recursive steps:

1. The clique K_{k+1} is a k-tree (of $k+1$ vertices);
2. Let G be a k-tree (of n vertices). Adding a new vertex to G forms a new K_{k+1} (with any k-clique K_k already existing in G) results in a k-tree (of $n+1$ vertices).

We note that the special case of k-tree when $k = 1$ is simply the tree under the usual sense. In addition, according to [Bodlaender, 1993], a graph has tree width $\leq k$ if and only if it is a subgraph of a k-tree.

Definition 2.1.2. Let $G = (V, E)$ be a graph and $k \geq$ be an integer. A *spanning k-tree* of G is a k-tree that is a spanning subgraph of G.

For $k = 1$, a spanning k-tree is a spanning tree under the usual sense. Like in the spanning tree case, there may be many spanning k-trees for a given graph. Optimization problems can be defined to find a spanning k-tree achieving the optimal value of some objective function. For spanning k-trees, various objective functions may be formulated. The traditional maximum spanning tree MST problem has its objective function as $obj(T) = \sum_{(u,v)\in T} \omega(u, v)$, the sum of edge weights $\omega(u, v)$ on the spanning tree T. We may consider the sum to be defined based on how the tree can be formed:

$$obj(T) = obj(T') + \omega(x, y)$$

where tree T is formed from tree T' and new vertex y that forms an edge with some vertex x already in T'. Because spanning tree is the special case of spanning k-tree when $k = 1$, we generalize the idea of objective function for spanning k tree as follows.

Let k-tree G be formed from k-tree G' and new vertex x_{k+1} that forms a $(k+1)$-clique with some k-clique $\{x_1, x_2, \ldots, x_k\}$ already in G'. Then

$$obj(G) = obj(G') + \lambda(\, \mathcal{E}(x_1, x_2, \ldots, x_k, x_{k+1})\,) \tag{2.1}$$

where $\mathcal{E}(x_1, x_2, \ldots, x_k, x_{k+1}) = \langle \omega(x_1, x_2), \omega(x_1, x_3), \ldots, \omega(x_k, x_{k+1}) \rangle$, i.e., the m-tuple of weights of all edges in the $(k+1)$-clique $\{x_1, x_2, \ldots, x_k, x_{k+1}\}$, for $m = \frac{1}{2}k(k+1)$, and λ is a function: $\bigcup_{m=1}^{\infty} \mathbb{R}_{\geq 0}^{m} \to \mathbb{R}_{\geq 0}$.

We do not restrict the class which the function λ in formula (2.1) belongs to, so it can cover a wide ranges of problems involving spanning k-trees. For example, when $k = 1$ and λ is set the identity function, formula (2.1) defines the objective function for the maximum spanning tree problem.

In particular, the maximum spanning k-tree problem previously investigated in [Bern, 1987; Cai and Maffray, 1993; Srebro, 2001; Liao and Zhang, 2009] has the objective function formulated by (1) with

$$\lambda(\langle \omega(x_1, x_2), \omega(x_1, x_3), \ldots, \omega(x_k, x_{k+1}) \rangle) = \sum_{i=1}^{k} \omega(x_i, x_{k+1})$$

i.e., the total weight sum of all edges in the spanning k-tree.

In many applications however, the sum of "weights" of the $(k+1)$-cliques contained in a spanning k-tree may be more interesting. Such objective functions can be appropriate for applications modeled by the spanning k-tree, because a real-world weight function on a $(k+1)$-clique may not simply be replaced with the sum of edge weights in the clique. This is particularly true in the bio-molecule structure modeling for which interaction energy functions tend to be multi-bodies instead of binary.

Definition 2.1.3. Assume function $\lambda : \bigcup_{m=1}^{\infty} \mathbb{R}_{\geq 0}^{m} \to \mathbb{R}_{\geq 0}$. Then *problem* MS$kT^{\lambda}$ problem is, given any integer $k \geq 1$ and graph $G = (V, E)$ with edge weight function $\omega : E \to \mathbb{R}_{\geq 0}$, to find a spanning k-tree $H = (V, F)$ and $F \subseteq E$, which maximizes objective function $\sum_{\kappa \in H} \lambda(\mathcal{E}(\kappa))$ (where the notation $\kappa \in H$ indicates that the $(k+1)$-clique κ belongs to the spanning k-tree H).

The earlier mentioned the maximum spanning k-tree problem MSkT investigated in [Bern, 1987; Cai and Maffray, 1993; Srebro, 2001; Liao and Zhang, 2009] is MSkT$^{\lambda}$ problem for a simple case of λ. It was proved NP-hard for every fixed $k \geq 2$ [Bern, 1987], giving strong evidences that the problems MSkT$^{\lambda}$, for non-trivial functions λ, are not tractable for any constant bound $k \geq 2$ on general graphs. In this chapter, we investigate efficient algorithms for MSkT$^{\lambda}$ on the family of graphs with a labelled Hamiltonian path, which are of interest to biomedical informatics research.

Definition 2.1.4. A graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$, is a *backbone graph* if $E = D \cup A$, i.e., the edges are the disjoint union of two edge sets D and A, such that $D = \{(i, i+1) : i = 1, 2, \cdots, n-1\}$. And edge set D is called the *backbone* of G. Accordingly, the edges in D and the edges in A are called backbone and non-backbone edges of G respectively.

We will also use the following terminologies in this chapter. A *backbone k-tree* is a backbone graph that is a k-tree. A *spanning k-tree on a backbone graph* is a spanning k-tree of the graph which contains the backbone. We tailor Definition 2.1.3 for backbone graphs to require that the produced spanning k-trees to contain the backbone (i.e., the Hamiltonian path).

Definition 2.1.5. Assume function $\lambda : \bigcup_{m=1}^{\infty} \mathbb{R}_{\geq 0}^{m} \to \mathbb{R}_{\geq 0}$. *Problem* MS$kT^{\lambda}$ *on backbone graphs* (denoted as MSkT$^{\lambda}$-B) is, given any integer $k \geq 1$ and backbone graph $G = (V, E)$ with backbone $D \subseteq E$ and edge weight function $\omega : E \to \mathbb{R}_{\geq 0}$, to find a spanning k-tree $H = (V, F)$ and $D \subseteq F \subseteq E$, which maximizes objective function $\sum_{\kappa \in H} \lambda(\mathcal{E}(\kappa))$.

2.2 Properties of Spanning k-Trees on Backbone Graphs

In this section, we reveal important properties of spanning k-trees, especially on backbone graphs, which will facilitate our algorithm design for MSkT$^{\lambda}$-B, i.e, problem MSkT$^{\lambda}$ on backbone graphs.

We first introduce a representation for k-trees. Any k-tree generated by the recursive rules given in section 2.1 can be represented as an ordered sequence of $(k+1)$-cliques $\kappa_0, \kappa_1, ..., \kappa_m$, for $m = n-k-1$, where κ_0 is called the *base* $(k+1)$-*clique* of the k-tree. For $j = 1, 2, ..., m$, clique κ_j is defined as

$$\kappa_j = \kappa_i \setminus \{x\} \cup \{y\} \text{ for some } i < j,\ x \in \kappa_i \text{ and for all } l < j, \text{ where } y \notin \kappa_l \tag{2.2}$$

In other words, clique κ_j is obtained from clique κ_i by replacing vertex x in clique κ_i constructed earlier with a new vertex y. This construction will be denoted by $\kappa_j = \kappa_i|_y^x$. We call κ_j a *child* of κ_i and κ_i the *parent* of κ_j. Similarly, we can also define an *ancestor* and an *descendant* of a $(k+1)$-clique in the usual sense. It is not difficult to see the construction of κ_j for $j = 1, 2, \ldots, m$ given in (2.2) ensures that κ_j has exactly one parent. Furthermore, by (2.2), no vertex can be used to create two different cliques; two different cliques containing the same vertex must share a nearest common ancestor that also contains the vertex. Thus we have

Proposition 2.2.1. Let $\kappa_0, \kappa_1, ..., \kappa_m$ be a $(k+1)$-clique sequence for any k-tree. Then

1. The parent-child relationships over the clique sequence defines a tree topology rooted at κ_0 with the other cliques being tree nodes;

2. For any pair of $i \neq j$, $i, j = 0, 1, \ldots, m$, $\kappa_i \cap \kappa_j \subseteq \kappa$ for every clique κ on the path between κ_i and κ_j on the tree topology.

We elaborate additional technical properties for $(k+1)$-clique sequences. We define

$$U_{\kappa_i} = \{v \in \kappa : \kappa \text{ is a descendant of } \kappa_i\} \cup \kappa_i,$$

i.e., the set of all vertices contained in descendants of κ_i. For the root clique κ_0, $U\kappa_0 = \{1, 2, \cdots, n\}$ obviously. Based on (2.2) and Proposition 2.2.1, we provide the following additional technical observations.

Proposition 2.2.2. For any $(k+1)$-clique κ_i in a $(k+1)$-clique sequence of a k-tree , $\kappa_j = \kappa_i|_y^x$, where $i < j$, the following properties hold:

1. $U_{\kappa_j} \subsetneq U_{\kappa_i}$;
2. $y \in U_{\kappa_i}$and $x \notin U_{\kappa_j}$;
3. For every child clique κ_l of κ_i, if $l \neq j$, then $y \notin U_{\kappa_l}$.

Now we derive concepts and properties of k-trees on backbone graphs. We still use the notion of $(k+1)$-clique sequences to discuss corresponding k-trees.

Definition 2.2.1. Let κ be a $(k+1)$-clique in a $(k+1)$-clique sequence for a backbone k-tree and $v \notin \kappa$ be a vertex in the k-tree. The *stretch* of v in κ is the maximal set of consecutive vertices including v but none of the vertices in κ. We denote this set by stretch(κ, v).

In particular, if $\kappa = \{x_1, x_2, ..., x_{k+1}\}$ with $x_1 < x_2 < ... < x_{k+1}$ and $v \notin \kappa$, but $x_i < v < x_{i+1}$ for some i, $0 \leq i \leq k+1$[1]. Then stretch$(\kappa, v) = \{x_i+1, \ldots, v-1, v, v+1, \ldots, x_{i+1}-1\}$. We observe the following properties for stretches.

Proposition 2.2.3. Let $\kappa = \{x_1, x_2, ..., x_{k+1}\}$ with $x_1 < x_2 < ... < x_{k+1}$ and u, $v \notin \kappa$ be two different vertices with $x_i < v < x_{i+1}$ and $x_i < u < x_{i+1}$ for some i, $0 \leq i \leq k+1$. Then

1. stretch(κ, v) = stretch(κ, u);

[1]Technically, here we assume $x_0 = 0$ and $x_{k+2} = n+1$.

2. κ contains at most $(k+2)$ non-empty stretches.

Theorem 2.2.4. Let κ be any $(k+1)$-clique in a $(k+1)$-clique sequence for a k-tree . Then $\forall v \notin \kappa$, stretch$(\kappa, v) \subsetneq U_\kappa$ or stretch$(\kappa, v) \cap U_\kappa = \emptyset$.

Proof. We will prove the theorem by considering two cases: (1) $v \in U_\kappa$, and (2) $v \notin U_\kappa$ separately.

(1) Assume $v \in U_\kappa$. We prove by induction on integer $\delta \geq 0$ that for any $w = v - \delta$, if $w \in$ stretch(κ, v), then $w \in U_\kappa$.

- **Basis:** $\delta = 0$, i.e., $w = v$; by assumption $v \in U_\kappa$.
- **Assumption:** Assume that if $w = v - \delta \in$ stretch(κ, v), then $w \in U_\kappa$.
- **Induction:** Consider $w' = v - \delta - 1 \in$ stretch(κ, v). We have $w = w' + 1 = v - \delta \in U_\kappa$ by the assumption. Since (w, w') is a backbone edge in the k-tree, there must be a clique κ' such that $(w, w') \in \kappa'$. If κ' is the descendant of κ, then $w' \in U_\kappa$. Otherwise since $w \in U_\kappa$ by the assumption, w is in some clique κ'' which is not a descendant of κ. But w must appear in every clique on the path from κ'' to κ' by Proposition 2.2.1, contradicting with the fact that it is not in κ.

Likewise, we can prove by an induction on integer $\delta \geq 0$ that for any $w = v + \delta$, if $w \in$ stretch(κ, v), then $w \in U_\kappa$.

(2) Assume $v \notin U_\kappa$. We prove by induction on integer $\delta \geq 0$ that for any $w = v - \delta$, if $w \in$ stretch(κ, v), then $w \notin U_\kappa$.

- **Basis:** $\delta = 0$, i.e., $w = v$; by assumption $v \notin U_\kappa$.
- **Assumption:** Assume that if $w = v - \delta \in$ stretch(κ, v), then $w \notin U_\kappa$.
- **Induction:** Consider $w' = v - \delta - 1 \in$ stretch(κ, v). Let $w = w' + 1 = v - \delta$; thus $w \in$ stretch(κ, v), $w \notin U_\kappa$ by the assumption. Since (w, w') is a backbone edge in the k-tree, there must be a clique κ' such that $(w, w') \in \kappa'$. κ' cannot be

a descendant of κ because of the assumption $w \notin U_\kappa$. If $w' \in U_\kappa$, we may assume it is contained in clique κ'', a descendant of κ. By Proposition 2.2.1, w' must be on every clique on the path from κ' to κ'', contradicting the fact that it is not in κ.

Likewise, we can prove by induction on integer $\delta \geq 0$ that for any $w = v + \delta$, if $w \in \text{stretch}(\kappa, v)$, then $w \notin U_\kappa$.

□

Corollary 2.2.4.1. Let $\kappa = \{x_1, x_1, ..., x_{k+1}\}$ be any $(k+1)$-clique in a $(k+1)$-clique sequence of a k-tree with $x_1 < x_2 < ... < x_{k+1}$. If $\kappa' = \kappa|_y^{x_i}$, for some i, $1 \leq i \leq k+1$, then

1. $stretch(\kappa, y) \subseteq U_{\kappa'}$; and
2. $stretch(\kappa', x_i) \cap U_{\kappa'} = \emptyset$.

Proof. (1) It follows the theorem 2.2.4 on account of $y \in U'_\kappa$ by Proposition 2.2.2. (2) Because $x_i \notin U_{\kappa'}$, it follows the theorem 2.2.4. □

Definition 2.2.2. Let κ be a $(k+1)$-clique in some $(k+1)$-clique sequence. Then the *Importable Set* of κ is the set of vertices contained in the descendent cliques of κ, excluding those vertices already in κ. We denote the importable set of κ by I_κ, i.e., $I_\kappa = U_\kappa \setminus \kappa$.

Proposition 2.2.5. Let κ be a $(k+1)$-clique in some $(k+1)$-clique sequence. Then

1. If $\kappa' = \kappa|_y^x$ then $I_{\kappa'} = I_\kappa \setminus \text{stretch}(\kappa', x) \setminus \{y\}$;
2. If $\kappa' = \kappa|_{y_1}^{x_1}$ and $\kappa'' = \kappa|_{y_2}^{x_2}$, for $y_1 \neq y_2$, then $I_{\kappa'} \subseteq I_\kappa \setminus \text{stretch}(\kappa', y_2)$;
3. If $\kappa' = \kappa|_{y_1}^{x_1}$ and $\kappa'' = \kappa|_{y_2}^{x_2}$, for $y_1 \neq y_2$, then $I_{\kappa'} \cap I_{\kappa''} = \phi$.

Lemma 2.2.6. Let $\kappa_0, \kappa_1, ..., \kappa_m$, where $m = n - k - 1$, be a $(k+1)$-clique sequence of a k-tree. Then I_{κ_0} consists of at most $k+2$ disjoint non-empty sets of consecutive vertices delimited by the $k+1$ vertices in κ_0. And I_{κ_i}, for every $i = 1, 2, \ldots, m$, consists of at most $k+1$ disjoint non-empty sets of consecutive vertices.

Proof. We assume $\kappa_0 = \{x_1, x_2, \cdots, x_{k+1}\}$ with $x_1 < x_2 < \cdots < x_{k+1}$. Since $I_{\kappa_0} = U_{\kappa_0} \setminus \kappa_0$, it is clear that the $k+1$ vertices in κ_0 separate I_{κ_0} into at most $k+2$ disjoint non-empty sets of consecutive vertices.

To prove I_{κ_i}, for every $i = 1, 2, \ldots, m$, consists of at most $k+1$ disjoint non-empty sets of consecutive vertices, we use induction on the number of of steps S taken to produce κ_i from the base clique κ_0.

- **Basis:** $S = 1$. $\kappa_i = \kappa_0|_{y_i}^{x_i}$. By Proposition 2.2.5, $I_{\kappa_i} = I_{\kappa_0} \setminus \text{stretch}(\kappa_0, x_i) \setminus \{y_i\}$. If $stretch(\kappa_0, x_i)$ is empty, the original number of disjoint sets of consecutive vertices in I_{κ_0} is at most k; if $stretch(\kappa_0, x_i)$ is not empty, its removal reduces the number to at most k. Moreover, new vertex y_i must fall in between x_{j-1} and x_j for some $j \neq i$, splitting stretch(κ_0, y_i) in I_{κ_0} into two, yielding at most $k+1$ disjoints set in I_{κ_i}.

- **Assumption:** Assume the claim holds for I_{κ_i} for clique κ_i produced at step S or less.

- **Induction:** Let κ_j be any child of κ_i created at step $S+1$, with $\kappa_j = \kappa_i|_{y_j}^{x_j}$. By assumption, I_{κ_i} consists of at most $k+1$ disjoint sets of consecutive vertices. By Proposition 2.2.5, $I_{\kappa_j} \subseteq I_{\kappa_i} \setminus \text{stretch}(\kappa_i, x_j) \setminus \{y_j\}$. Similar to the proof of the basis, removing x_j causes the number of disjoint sets of consecutive vertices in I_{κ_i} is at most k. And introducing y_j may separate a stretch into two and resulting in at most $(k+1)$ disjoint non-empty sets of consecutive vertices.

□

Theorem 2.2.7. Let κ be a $(k+1)$-clique in a $(k+1)$-clique sequence of a k-tree and $v \notin \kappa$ be a vertex in the k-tree such that stretch(κ, v) $\neq \emptyset$ and stretch(κ, v) $\subsetneq U_\kappa$. If $\kappa' = \kappa|_{y_i}^{x_i}$ and $\kappa'' = \kappa|_{y_j}^{x_j}$, where y_i, $y_j \in$ stretch(κ, v), then $y_i = y_j$ and $x_i = x_j$.

Proof. We assume $\kappa = \{x_1, x_2, \cdots, x_{k+1}\}$ with $x_1 < x_2 < \cdots < x_{k+1}$. Without loss of generality, we assume $x_t < y_i \leq y_j < x_{t+1}$ for some $0 \leq t \leq k+1$. First, $y_i + 1 \neq y_j$. Otherwise, (y_i, y_j) is a backbone edge. By Proposition 2.2.2, y_i is only in $U_{\kappa'}$ and y_j is

only in $U_{\kappa''}$ which causes loss of the backbone edge (y_i, y_j). Also note that $i, j \notin \{t, t+1\}$. This is because assume $i = t$ (the proofs for other cases are similar), by Proposition 2.2.5, $I_{\kappa'} = I_\kappa \setminus \text{stretch}(\kappa', x_i) \setminus \{y_i\}$ and $I_{\kappa''} = I_\kappa \setminus \text{stretch}(\kappa'', x_j) \setminus \{y_j\}$. Thus removing x_i from κ results in removing all the vertices before y_i, which causes loss of the backbone edge $(y_i - 1, y_i)$ in the k-tree. On the other hand, if $y_i + 1 < y_j$, there exists a vertex y_m between y_i and y_j. Since $i, j \notin \{t, t+1\}$, $I_{\kappa'}$ and $I_{\kappa''}$ both contain y_m, violating the condition $I_{\kappa'} \cap I_{\kappa''} = \emptyset$ in Proposition 2.2.5. Thus we has proved $y_i = y_j$.

Now we have to have $x_1 = x_2$, otherwise y_i would belong to two children κ' and κ'', but not to κ, violating the second property in Proposition 2.2.1. □

Theorem 2.2.8. Let $\kappa_0, \kappa_1, ..., \kappa_m$, where $m = n - k - 1$, be a $(k+1)$-clique sequence of a k-tree. Then root clique κ_0 has at most $(k+2)$ children; and κ_i, for every $i = 1, 2, \ldots, m$, has at most $(k+1)$ children.

Proof. By Lemma 2.2.6, there are at most $(k+2)$ disjoint non-empty sets of consecutive vertices in I_{κ_0} and at most $(k+1)$ disjoint non-empty sets of consecutive vertices in I_{κ_i} $(1 \leq i \leq m)$. By Theorem 2.2.7, for each such set, at most one child clique of κ_i $(0 \leq i \leq m)$ can be created through $\kappa_i|_y^x$ by selecting some vertex $x \in \kappa_i$ and some y from the importable set. □

2.3 Dynamic Programming Algorithm

In this section, we introduce a dynamic programming algorithm for the MSkT$^\lambda$-B problem on backbone graph, which runs in polynomial time $O(n^{k+1})$ for every fixed k. We point out that this is not to enumerate all $(k+1)$-cliques to examine their parent-child relationships since such method would incur another time factor of n to yield $O(n^{k+2})$-time complexity. Our less straightforward algorithm comes from the following observation: every $(k+1)$-clique in any $(k+1)$-sequence of a k-tree can be obtained by adding a new vertex to an existing k-clique. By the properties in Section 2.2, the resulted $(k+1)$-clique has at most $k+2$

children. At the same time, the $(k+1)$-clique consists of $k+1$ number of k-cliques, from which the children, each being a $(k+1)$-clique, can be formed independently and recursively. We present some details in the following.

In the case of no confusion, $\kappa = \{x_1, x_2, \cdots, x_k\}$ is used to represent a k-clique throughout this section. By Proposition 2.2.3, κ has at most $k+1$ stretches (of contiguous vertices) delimited by the k vertices in κ. We use A_κ to denote the set of all non-empty stretches for κ. For any subset $S \subseteq A_\kappa$, we denote $\mathcal{U}(S) = \bigcup_{s \in S} s$, i.e., the union of all stretches s in S. For any vertex $p \in \mathcal{U}(S)$, we use $\kappa \otimes \{p\}$ to denote the $(k+1)$-clique formed by κ together with p, which also results in at most $k+2$ stretches by splitting one of the stretches in S into two with vertex p. We denote this set of stretches with $[S, p]$. At the same time, we define $\kappa_i = \kappa|_p^{x_i}$ for $i = 1, 2, \ldots, k$ and $\kappa_{k+1} = \kappa$ and let $[\kappa, p]$ denote the set $\{\kappa_r \,|\, 1 \leq r \leq k+1\}$. We then define $\Phi(\kappa, S, p) = \{\phi \,|\, \phi : [S, p] \to [\kappa, p]\}$.

Note that for all functions ϕ in Φ are actually constrained. That is, every stretch in $[S, p]$ can only be mapped to one of $k-1$ k-cliques in $[\kappa, p]$, except the first and the last stretches which can be mapped to one of k k-cliques. To be specific, let $\kappa \otimes \{p\} = \{x_1, x_2, \cdots, x_{k+1}\}$, with $x_1 < x_2 < \cdots < x_{k+1}$. Then any stretch with vertices in between x_i and x_{i+1} can only be mapped to some k-clique containing vertices x_i and x_j and the number of such k-cliques is $k-1$.

We formulate in the following the objective function with recurrences for the MSkT$^\lambda$-B problem, where function λ satisfying statement (1) in section 2. Let $G = (V, E)$ be an input backbone graph with $V = \{1, 2, \ldots, n\}$. Then for any k-clique $\kappa \subseteq V$, subset of stretches $S \subseteq A_\kappa$, and vertex $p \in \mathcal{U}(S)$, we define $M(\kappa, S)$ to be the maximum objective function

value of a k-tree rooted at $(k+1)$-clique $\kappa \otimes \{p\}$, for some $p \in \mathcal{U}(S)$. Then we have

$$M(\kappa, S) = 0 \quad \text{if } S = \emptyset \tag{2.3}$$

$$M(\kappa, S) = \max_{p \in \mathcal{U}(\mathcal{S})} \max_{\phi \in \Phi(\kappa, S, p)} P(\kappa, S, p, \phi) + \lambda(\mathcal{E}(\kappa \otimes \{p\})) \tag{2.4}$$

$$P(\kappa, S, p, \phi) = \sum_{\substack{1 \le r \le k+1 \\ \kappa_r \in [\kappa, p],\, \phi^{-1}(\kappa_r) \neq \emptyset}} M(\kappa_r, \phi^{-1}(\kappa_r)), \tag{2.5}$$

The goal of the algorithm is to seek to maximize the following function:

$$\max_{\kappa \in [n]^k} M(\kappa, A_\kappa) \tag{2.6}$$

where $[n]^k$ is the set of all k-cliques of vertices drawn from $V = \{1, 2, \ldots, n\}$.

We now explain the algorithm with the recurrences (2.3)-(2.6). Our algorithm computes the maximum weight of a spanning k-tree rooted at a $(k+1)$-clique formed by some k-clique. Recurrence (2.6) gives in the initial step by examining all k-cliques $\kappa \in [n]^k$ and all the possible stretches that κ has. Formula (2.3) gives the empty stretch set ($S = \emptyset$) as the terminating condition for the algorithm. Recurrences (2.4) and (2.5) provide details as follows for how to compute $M(\kappa, S)$, where $S \subseteq A_\kappa$.

For the considered k-clique κ with the allowed set S of stretches, every vertex p from the allowed stretches in S is examined. Adding p to κ to form $\kappa \otimes \{p\}$ as a newly created $(k+1)$-clique for the sought spanning k-tree. The new $(k+1)$-clique $\kappa \otimes \{p\}$ has the weight contribution defined as $\lambda(\mathcal{E}(\kappa \otimes \{p\}))$. The new set of stretches $[S, p]$ because of the addition of p is the same as S, except the stretch in S, from which p was chosen, is split into two by p. These new set of stretches are mapped by function ϕ to the newly created $k+1$ number of k-cliques κ_r, with each such clique using the stretches $\phi^{-1}(\kappa_r)$ assigned by ϕ to recursively and independently children $(k+1)$-cliques for the sought spanning k-tree. Recurrence (2.4) maximizes the weight of the spanning k-tree by considering all possible vertices p and functions ϕ. Recurrence (2.5) branches the computation by independently

computing based on every one of the $k+1$ k-clique κ_r and sum the total weights in all branches.

In the following paragraphs, we formally prove that the recurrences (2.3)-(2.6) are correct for problem MSkT$^{\lambda}$-B (i.e., the problem of finding the maximum spanning k-tree on backbone graphs).

Definition 2.3.1. Let $(k+1)$-cliques κ_i, κ_j be two children of some $(k+1)$-clique in the tree topology of a spanning k-tree. κ_i and κ_j are *identical siblings* if $|\kappa_i \cap \kappa_j| = k$.

Lemma 2.3.1. Any spanning k-tree G can be transformed into a k-tree G' without identical siblings, where G' has the same weights as G.

Proof. The lemma holds naturally when G doesn't have identical siblings. Assume G has identical siblings. Let $\kappa_i = \{x_1, \cdots, x_p, y_1, x_{p+1}, \cdots, x_k\}$, $\kappa_j = \{x_1, \cdots, x_q, y_2, x_{q+1}, \cdots, x_k\}$ be two $(k+1)$-cliques that are identical siblings in G, where $\kappa_i \cap \kappa_j = \{x_1, \cdots, x_k\}$. We can transform the identical siblings relationship into parent child relationship by pruning the k-subtree rooted at κ_j and connecting it to κ_i. To be specific, we have two steps:

1. Let $\kappa_j = \kappa_i|_{y_2}^{y_1}$, i.e., prune the k-subtree rooted at κ_j and make κ_i the parent of the κ_j;

2. Update the importable set of κ_i to $I_{\kappa_i} \cup I_{\kappa_j}$.

It is clear that above transformation preserves the weights. To achieve an k-tree G' without identical siblings, we apply above transformation on G level by level from root to leaves. Note that at a certain level, the transformation can be applied multiple times until there is no identical siblings. □

Figure 2.2 illustrates (a) identical siblings in a $k$tree, and (b) transformation of the k-tree to one without identical sibling. We note that constructing k-trees without identical siblings allows a reduction in the search space. Our algorithm actually produces a maximum spanning k-tree without identical siblings.

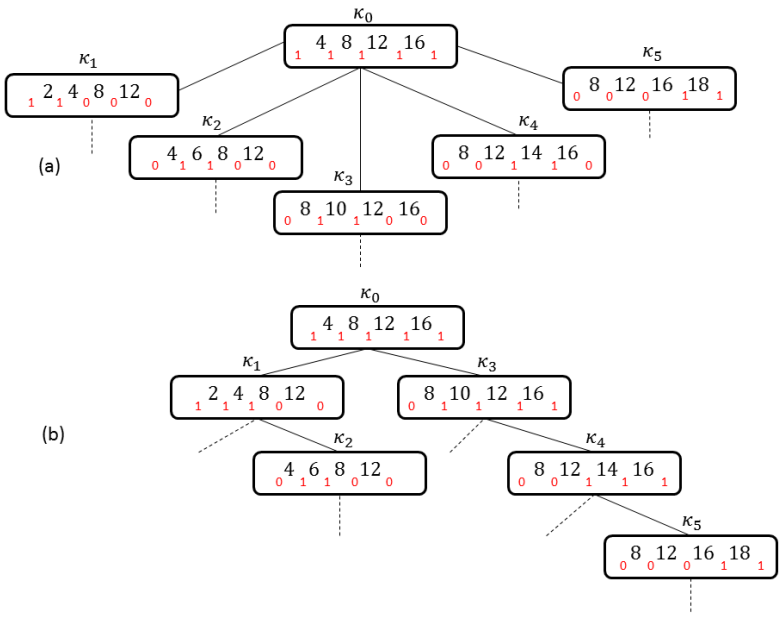


Figure 2.2: An illustration on the transformation from a k-tree with identical siblings to a k-tree without identical siblings: (a) a k-tree rooted at κ_0, where κ_1 and κ_2 are identical siblings, κ_3, κ_4 and κ_5 are identical siblings. (b) a k-tree without identical siblings which has the same weight as (a). The binary number vector for each $(k+1)$-clique is the bitmap for corresponding importable set.

Lemma 2.3.2. The algorithm for MSkT$^{\lambda}$-B examines all spanning k-trees without identical siblings.

Proof. Let G be any spanning k-tree without identical siblings with base $(k+1)$-clique $\kappa_0 = \{x_1, x_2, \cdots, x_{k+1}\}$. We will prove inductively that any $(k+1)$-clique in G is produced by the algorithm along the same tree topology of G. In particular, assume that $(k+1)$-clique κ_i is produced by the algorithm, we will prove that every one of its children $(k+1)$-cliques (if they exist) are produced by the algorithm as well.

First, κ_0 is produced as the root $(k+1)$-clique by our algorithm. This is because recurrence (2.6) traverses all possible k-cliques $\kappa \in [n]^k$. There must be one $\kappa = \kappa_0 \setminus \{x_i\}$, for some $1 \leq i \leq k+1$, that is considered. The subsequent step with (2.4) can introduce x_i as vertex p to form the desired $(k+1)$-clique $\kappa \otimes \{x_i\} = \kappa_0$ which is then produced by the algorithm.

Now we assume $(k+1)$-clique κ_i is produced by the algorithm, with importable set I_{κ_i}. It

has to go through recurrence (2.4) such that $\kappa_i = \kappa \otimes \{p\}$ for some k-clique κ and some vertex $p \in \mathcal{U}(S)$, where S is the set of stretches associated with κ. It is clear that $I_{\kappa_i} = \mathcal{U}([S, p])$. Let the children of κ_i be $\kappa_{i_1}, \kappa_{i_2}, \cdots, \kappa_{i_j}$ $(1 \leq j \leq k+1)$. Since G doesn't have identical siblings, each k-clique from $\{\kappa_i \cap \kappa_{i_1}, \kappa_i \cap \kappa_{i_2}, \cdots, \kappa_i \cap \kappa_{i_j}\}$ must be unique. By recurrence (2.4) and (2.5), the algorithm maximizes over all possible ways to map the stretches of κ_i to the above k-cliques. In particular, for $l = 1, 2, \ldots, j$, if $(k+1)$-clique κ_{i_l} has importable set I_{i_l}, through recurrence (2.5) the algorithm can assign the set of stretches S_{i_l} to k-clique $\kappa_i \cap \kappa_{i_l}$ such that $I_{i_l} = \mathcal{U}([S_{i_l}, p_{i_l}])$ where $p_{i_l} \in \kappa_{i_l} \setminus \kappa_i$ that can be picked by recurrence (2.4). And the algorithm produces $(k+1)$-clique $(\kappa_i \cap \kappa_{i_l}) \otimes p_{i_l}$, i.e., κ_{i_l}, as a child of κ_i.

Again since G doesn't have identical siblings, the k-clique from $\{\kappa_i \cap \kappa_{i_1}, \kappa_i \cap \kappa_{i_2}, \cdots, \kappa_i \cap \kappa_{i_j}\}$ are different. It is not difficult to see that the mapping functions in Φ guarantee all three necessary conditions for importable sets stated in Proposition 2.2.5.

□

By Lemmas 2.3.1 and 2.3.2, we conclude

Theorem 2.3.3. The algorithm for problem MSkT$^{\lambda}$-B computes answers correctly.

Finally, we analyze the running time of the algorithm. First, traversing all the k-cliques in $[n]^k$ in recurrence (2.6) needs $O(n^k)$ time . And the first maximization over p in recurrence (2.4) takes $O(n)$ time. Thus the polynomial running time on n is $O(n^{k+1})$. For the exponential factor on k, we count $|\Phi|$ for a fixed $(k+1)$-clique $\kappa \otimes \{p\}$ on all subset $S \subseteq A_k$. When $|S| = k+2$, $|\Phi| = k^2(k-1)^k$; when $|S| = k+1$, $|\Phi| \leq \binom{k+2}{k+1} k^2(k-1)^{k-1}$; simiarly, we can count $|\Phi|$ for other cases. So for a fixed $(k+1)$-clique, we have

$$k^2(k-1)^k + \binom{k+2}{k+1} k^2(k-1)^{k-1} + \cdots + \binom{k+2}{1} k + 1 < (k+1)^{(k+2)}$$

The summation in recurrence (2.5) takes $O(k)$ time. Therefore, the running time of the algorithm is $O(k(k+1)^{k+2} n^{k+1})$. Also derived from the above analysis is the space complexity $O(2^{k+1} n^k)$.

2.4 Optimality of Algorithm's Time Complexity

We are interested in knowing if our developed algorithm for MSkT$^{\lambda}$ is optimal in the computational complexity. In particular, we would like to know if there are polynomial time algorithms for the problem, with a polynomial degree independent of k or a polynomial degree d for some $d < k + 1$ (d may be a function of k). In this section, we show strong evidence that these scenarios are unlikely.

We first prove that, problem MSkT$^{\lambda}$-B remains NP-hard for a simple function λ. In particular, we construct a reduction from the k-Clique problem to MSkT$^{\lambda}$-B that essentially preserves the parameter k. Since the former is W[1]-hard, our proof also implies fixed-parameter intractability for the latter.

We define the following function $\lambda : \bigcup_{m=1}^{\infty} \mathbb{R}_{\geq 0}^{m} \to \mathbb{R}_{\geq 0}$ such that for any $m \geq 1$ and $\langle e_1, e_2, \cdots, e_m \rangle \in \mathbb{R}_{\geq 0}^{m}$,

$$\lambda(\langle e_1, e_2, \ldots, e_m \rangle) = 1 \text{ if and only if } e_i > 0, \forall i \, 1 \leq i \leq m$$

By definition 2.1.5, we obtain problem MSkT$^{\lambda}$-B with the above defined function λ. We formulate in the following the related canonical decision problem, where notation $\kappa \in H$ is to represent the statement that $(k+1)$-clique κ belongs to k-tree H.

Problem Decision MSkT$^{\lambda}$-B:

Input: integer $k \geq 1$, graph $G = (V, E)$ with backbone $D \subseteq E$, weight $\omega : E \to \mathbb{R}_{\geq 0}$, and $W > 0$;

Question: Does G have a spanning k-tree $H = (V, F)$ such that $\sum_{\kappa \in H} \lambda^*(\mathcal{E}(\kappa)) \geq W$?

Theorem 2.4.1. The problem Decision MSkT$^{\lambda}$-B is NP-complete.

Proof. It is not difficult to see that in a polynomial time a certificate H encoding a spanning k-tree can be checked to determine if the defined objective function value is at least W. Therefore, the problem is in NP.

The hardness is proved with a reduction from k-CLIQUE. Given a graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ as an instance of k-CLIQUE, we construct an instance of complete backbone graph $G' = (V', E')$ as follows in polynomial time:

1. $V' = \{1, 2, \ldots, n\}$;
2. $E' = \{(i, j) : i < j \text{ and } i, j = 1, 2, \ldots, n\}$;
3. backbone $D = \{(i, i+1) : i = 1, 2, \ldots, n-1\}$;
4. weight function $\omega(i, j) = 1$ if $(x_i, x_j) \in E$; $\omega(i, j) = 0$ otherwise;
5. $k' = k - 1$; $W = 1$.

For given $k \geq 3$, if the constructed G' has a spanning k'-tree H with objective function value at least 1, where $k' = k - 1$, the contribution of the value must have come from at least one $(k+1)$-clique in the spanning tree. Let this clique be $\{i_1, i_2, \ldots, i_k\}$ in H. Since the function λ on the clique has value 1 if and only if every one of the arguments of function λ, i.e., weight $\omega(i_h, i_l)$, for every $h, l = 1, 2, \ldots, k$, is non-zero, Based on the construction of G', these arguments are indicators for edges among the vertex set $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ in the original graph G. Thus G has a clique of size k.

On the other hand, if G has a clique of size k, then G' has a spanning k'-tree H of objective function value $\geq W = 1$. To see this, let the clique of size k in G be $\{v_{i_1}, v_{i_2}, \cdots, v_{i_k}\}$. Let k-clique $\kappa_0 = \{i_1, i_2, \ldots, i_k\}$. We show in the following that there is a spanning k-tree rooted at κ_0 for G'.

Without loss of generality, we assume $i_1 < i_2 < \cdots < i_k$ in κ_0. By section 3, κ_0 can have s non-empty stretches, for $s \leq k + 1$. Then on backbone graph G' a $(k-1)$-tree H rooted at κ_0 with at most s branches can be constructed with each branch covering a stretch. In particular, for every $j = 0, 1, \ldots, k$, the jth stretch is $\{i_j + 1, i_j + 2, \ldots, i_{j+1} - 1\}$. Let $m_j = i_{j+1} - i_j - 1$, the length of the jth stretch. If $m_j \neq 0$, the jth stretch yields a branch of k-cliques represented by the following k-clique sequence: $\kappa_0, \kappa_{j_1}, \kappa_{j_2}, \ldots, \kappa_{j_{m_j}}$,

where $\kappa_{j_1} = \kappa_0|_{i_j+1}^{i_j}$, $\kappa_{j_2} = \kappa_{j_1}|_{i_j+2}^{i_j+1}$, $\ldots$, $\kappa_{j_{m_j}} = \kappa_{j_{m_j-1}}|_{i_j+m_j}^{i_j+m_j-1}$ (note that $i_j + m_j = i_{j+1} - 1$). Details of boundary cases $j = 0$ and $j = k$ are only slightly different from the general case and thus are omitted. H covers all backbone edges of G' and is a spanning k-tree for G'. In addition, since $\lambda(\kappa_0) = 1$, the weight of this $(k-1)$-tree is $\geq \lambda(\kappa_0) \geq W$.

□

The proof of the above theorem constructs a reduction that is actually also a parameter-preserving polynomial time reduction between the two parameterized problems. Because k-clique is W[1]-complete [Downey and Fellows, 1995] and it cannot be solved in time $f(k)n^{o(k)}$ for any function f unless W[1]=FPT [Chen et al., 2006], we conclude

Corollary 2.4.1.1. DECISION MSkT$^\lambda$-B is $W[1]$-hard.

Theorem 2.4.2. Unless the W-hierarchy collapses, the MSkT$^\lambda$-B problem cannot be solved in time $f(k)n^{o(k)}$ for any function f.

Efficient algorithms for exact computation of backbone MSkT are always desirable but there may be some limitations beyond which what we can do. The MSkT problem is the core parsing task for stochastic k-tree grammar (see Section 3) that include stochastic context-free grammar (SCFG) as a special case (for $k = 2$). Optimal parsing for SCFG can be done in $O(n^3)$ (e.g., via the CYK algorithm) and there seems to be just a little room for improving the complexity. In particular, there are algorithms for parsing CFG in times $O(n^3/\log^2 n)$ and $n^{3-\epsilon}$ for some $\epsilon > 0$. The techniques of these two kinds of improved algorithms rely heavily on the Four Russians algorithm [Aho et al., 1974] and a more advanced idea [Lee, 2002] for matrix multiplication, respectively. The matrix multiplication problem can be done in time $O(m^{2.376})$. We believe these ideas can be adopted to improve algorithms for backbone maximum spanning k-tree problem to $O(n^{k+1-\delta})$ for some small $\delta > 0$. We note that we may not expect δ to be substantially large for the general case of k, since it would imply much faster algorithms for matrix multiplication, which seems very unlikely [Lee, 2002]. Nevertheless, even if $\delta = 0.5$, an algorithm for backbone maximum spanning k-tree

of running time $O(n^{k+1-\delta}) = O(n^{k+1}/\sqrt{n})$ is worth pursuing. In addition to its theoretical merit, it will have $\sqrt{n}$ times of speed up and memory reduction.

Chapter 3

Stochastic k-Tree Grammar and Its Application in Biomolecular Structure Modeling

Stochastic formal language systems, typically the stochastic context-free grammar (SCFG), have been significantly valuable to various applications. Such a system essentially consists of a finite set of rules that syntactically dictate generation of strings for a desired language. Any generation process of a language string is a series of Chomsky rewriting rule applications and thus yields a syntactic structure associated with (the terminal occurrences in) the string. Because syntactic rules often are nondeterministic, there may be more than one syntactic process to generate the same string [Salomaa, 1981; Hopcroft et al., 2007]. Stochastic versions of such formal systems may be established by associating a probability distribution with the rules. Compounding the probabilities of rules used in a generation process of a string gives rise to the probability for the corresponding syntactic structure admitted by the string [Searls, 1993; Durbin et al., 1998]. Therefore, a stochastic language system defines a probability space for all the syntactic structures admitted by the string. At the same time, it also defines a probability space for all the strings in the language.

In addition to the apparent wide application in natural language processing [Lari and Young, 1990; Jurafsky et al., 1995; Waters and MacDonald, 1997; Klein and Manning, 2003; Sánchez et al., 2005; Antoine Rozenknop, 2006], SCFG has also been extensively adopted for statistical analysis of biomolecular structures [Sakakibara et al., 1994; Durbin et al., 1998; Chiang et al., 2006; Dill et al., 2007; Searls, 2010]. A biomolecule consists of a string of linearly arranged residues that can spatially interact to fold the string into a 3D structure

of biological significance. Interactions between residues are interpreted as co-occurrences of lexical objects in each parsing of the string. SCFG can conveniently model nested and parallel relationships of the interacting residues on a biomolecule. Figure 1 shows an RNA molecule with parallel and nested canonical base parings (in gray, lighter lines) between nucleotides, which is context-free. Indeed, SCFG has enabled the development of a number of effective computer programs for the prediction of RNA secondary structure [Nawrocki et al., 2009; Z. and L., 2004; Knudsen and Hein, 2003; Achawanantakun et al., 2010; Rivas et al., 2012]. Such programs are also computationally efficient by taking the advantage of dynamic programming algorithms permitted by context-free rules.

Nevertheless, SCFG cannot account for crossing interactions of a context-sensitive nature, e.g., the interactions in Figure 1 denoted by both gray (lighter) and pink (darker) lines. Since crossing, distant interactions are the signature of a biomolecule forming a tertiary (3D) structure, adequate modeling of such interactions with a stochastic grammar would have the potential for effective analysis and even prediction of biomolecular tertiary structures. Modeling context-sensitive languages with Chomsky context-sensitive grammars can be inconvenient and may incur computational intractability [Martin et al., 1994; Hopcroft et al., 2007]. Previous work in more constrained languages has studied mildly context-sensitive grammars, typically the Tree-Adjoining Grammar [Joshi, 1985] and its equivalent variants [Joshi and Vijay-Shanker, 1991; Vijay-Shanker and Weir, 1994], to model limited cross-serial dependencies arising in natural language processing. There has been limited success in the applications of such grammars in biomolecular structure modeling [Uemura et al., 1999; Searls, 2010; Chiang et al., 2006]; they were mostly used for the characterization of local, secondary structures. The global structure of a biomolecule involving cross relationships between arbitrarily distant residues may be beyond limited cross-serial dependencies.

In this chapter, we introduce a novel stochastic grammar called *stochastic k-tree grammar* (SkTG), for the analysis of context-sensitive languages. With succinct grammar rules, co-occurrences of distant terminals are recursively characterized as *k-trees*. A k-tree is a chordal

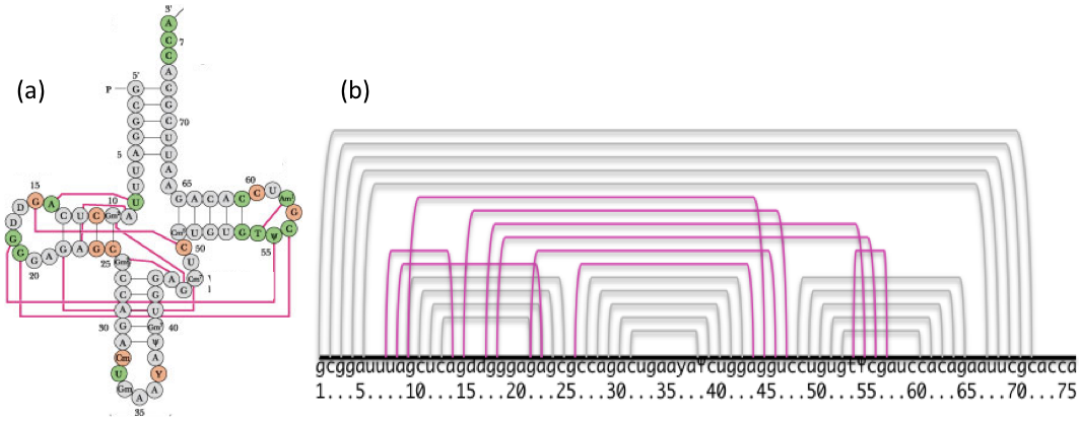


Figure 3.1: A single RNA molecule can fold back on itself to form secondary and tertiary structures through bio-residue interactions. (a) The secondary structure of tRNA (Phe of yeast, PDB id: 1EHZ)) consists of parallel and nested canonical base parings (gray, lighter connections) between nucleotides, which is context-free. The tertiary structure formed with additional non-canonical tertiary interactions (pink, darker connections) between nucleotides is context-sensitive. (b) Illustration of the bio-residues interactions of the tRNA molecule in terms of co-occurrences of terminals on a language string.

graph that does not contain cliques of size more than $k+1$ as a graph minor [Patil, 1986; Arnborg and Proskurowski, 1989]. For small values of k, k-trees are tree-like graphs; they are adopted in this work to constrain crossing relationships of terminal occurrences on language strings. Such constrained context-sensitivity has been discovered in biomolecular structures; recent studies have revealed that graphs describing bio-residue interactions found in resolved biomolecular 3D structures are actually (subgraphs of) k-trees, typically for $k \leq 4$ [Xu and Berger, 2006; Song et al., 2006; Xu et al., 2007; Huang et al., 2008a, 2010]. Therefore, the new grammar SkTG offers a viable approach to statistical modeling, analysis, and prediction of biomolecular tertiary structures.

Previous studies showed that statistical analysis problems over general k-trees are extremely difficult, in particular, NP-hard even for $k = 2$, excluding the possibility to feasibly implement such a framework [Srebro, 2001; Zimand, 2004; Sergio Caracciolo et al., 2008]. However, with the linear chain of vertices constrained on k-trees, we are able to show, for

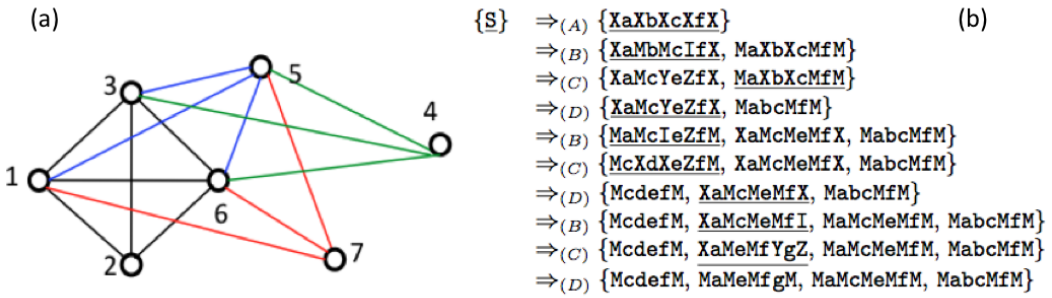


Figure 3.2: (a) A generation of a 3-tree of 7 vertices by Definition 4.1.1. (b) A derivation of string `abcdefg` with 3-tree grammar rules introduced in Definition 3.1.3, with the types of applied grammar rules shown and the LHS of every applied rule underscored. The derivation also results in an induced 3-tree, the same graph shown in (a).

the first time, that the k-tree parsing problem is solvable in polynomial-time for every fixed value of k. In particular, we will show that SkTG makes it possible to define a probability space for all k-treestructures admitted by any given language string. We will demonstrate efficient dynamic programming algorithms for computing the most probable k-tree structure for any given string. In this chapter, we will also discuss the application in the prediction of biomolecular tertiary structures that has motivated this work.

3.1 k-trees and the k-tree Grammar

Definition 3.1.1. [Patil, 1986] Let integer $k \geq 1$. The class of k-trees are defined with the following inductive steps:

1. A k-tree of $k+1$ vertices is a clique of $k+1$ vertices;

2. A k-tree of n vertices, for $n > k+1$, is a graph consisting of a k-tree G of $n-1$ vertices and a vertex v, which does not occur in G, such that v forms a new $(k+1)$-clique with some size-k clique already in G.

Figure 3.2 (a) shows of a 3-tree with seven vertices. By Definition 4.1.1, the order in

which 4-cliques formed is: initially $\{1, 2, 3, 6\}$ (black edges), vertex 5 and blue edges added, then vertex 7 and red edges added, and finally vertex 4 and green edges added.

3.1.1 The k-tree Grammar

Chomsky grammars derive a language sentence by series of rewritings on a single symbolic string. Instead, our new grammar derives a language sentence by rewritings on *multiple* symbolic strings, thus resulting in multiple symbolic strings. The language sentence generated in such a derivation consists of the terminal symbols that occur in the resulting multiple symbolic strings; the positional ordering of the derived terminals is completely determined by the derivation.

Let Σ be an alphabet, $\mathcal{N}$ be the set of non-terminals, and ϵ be the empty string. We call a symbolic string an *m-alternating string*, if it has the format $X_0a_1X_1 \cdots a_mX_m$ for some $m \geq 0$, such that $X_i \in \mathcal{N} \cup \{\epsilon\}$ for all $0 \leq i \leq m$ and $a_i \in \Sigma$ for all $1 \leq i \leq m$.

Definition 3.1.2. Let $\alpha = X_0a_1X_1 \cdots a_mX_m$ be an m-alternating string for some $m \geq 0$. Let ω be the substring $X_ia_{i+1} \cdots X_j$ in α, for some $0 \leq i \leq j \leq m$, and $\sigma \in (\mathcal{N} \cup \Sigma)^*$ be a string. Then $\alpha|_{\sigma}^{\omega}$ is the string obtained from α with the substring ω being substituted by σ.

For two non-overlapping substrings ω_1 and ω_2 in α, we use $\alpha|_{\sigma_1,\sigma_2}^{\omega_1,\omega_2}$ to denote the string obtained from α with ω_1 being replaced by σ_1 and ω_2 being replaced by σ_2 at the same time. We also allow aggregation $\forall i$ to denote multiple simultaneous substitutions involving all applicable indexes i. In particular, $\alpha|_{Y_i}^{\forall iX_i}$ is the string obtained from α by replacing X_i with Y_i for every i

Definition 3.1.3. Let $k \geq 2$ be a fixed integer. A *k-tree grammar* is a 6-tuple $\Gamma = (\Sigma, \mathcal{N}, \mathcal{R}, M, I, S)$, where Σ is a finite alphabet, $\mathcal{N}$ is a set of nonterminals, S, I, and M are the *starting, importing* and *masking* nonterminals in $\mathcal{N}$, respectively, and $\mathcal{R}$ is a set of grammar rules. Each rule has the format of $\alpha \to \mathcal{A}$, where α is either S or a $(k+1)$-alternating string and $\mathcal{A}$ is a subset of $(k+1)$-alternating strings, and has one of the following four types.

(In the following we assume $\alpha = X_0 a_1 X_1 \cdots a_{k+1} X_{k+1}$, where $\forall i = 1, \cdots, k+1$, $a_i \in \Sigma$, and $\forall j = 0, \cdots, k+1$, $X_j \in \mathcal{N}$.)

1. $S \to \{\beta\}$, for $\beta = Y_0 b_1 Y_1 \cdots b_{k+1} Y_{k+1}$, where $\forall i = 1, \cdots, k+1, \quad b_i \in \Sigma$, and $\forall j = 0, 1, \cdots, k+1, \quad Y_j \in \mathcal{N} - \{M, I\}$.

2. $\alpha \to \{\beta, \gamma\}$, where $\exists s$, $0 \leq s \leq k+1$, $X_s \neq M$, such that

   (1) $\beta = \alpha|_{Y_i}^{\forall i X_i}$, $\gamma = \alpha|_{Z_i}^{\forall i X_i}$, $Y_s = I$, and $Z_s = M$.

   (2) $\forall i = 0, 1, \cdots, k+1$, if $X_i = M$ then $Y_i = Z_i = M$; else either $Y_i = X_i$ and $Z_i = M$, or $Y_i = M$ and $Z_i = X_i$.

3. $\alpha \to \{\beta\}$, where $\exists s$, $0 \leq s \leq k+1$, $X_s = I$, and $\exists t$, $0 \leq t \leq k$, $t - s \geq 1$ or $s - t > 1$, $X_t = X_{t+1} = M$, such that $\beta = \alpha|_{YaZ}^{X_s}|_{M}^{X_t a_{t+1} X_{t+1}}$, for some $Y, Z \in \mathcal{N} - \{M, I\}$ and some $a \in \Sigma$.

4. $\alpha \to \{\beta\}$, such that $\beta|_{Y_i}^{\forall i X_i}$ and $\forall i = 0, 1, \cdots, k+1$, if $X_i = M$ then $Y_i = M$; else $Y_i = \epsilon$.

We note that rules of types (B) and (C) are tightly related by the importing nonterminal I. In particular, a rule of type (C) can be used if and only if a related rule of type (B) has been used.

Definition 3.1.4. Let $\Gamma = (\Sigma, \mathcal{N}, \mathcal{R}, M, I, S)$ be a k-tree grammar. Let set $\mathcal{T} \subseteq (\Sigma \cup \mathcal{N})^+$. Let $\alpha \in \mathcal{T}$, $\alpha \to \mathcal{A} \in \mathcal{R}$, and define $\mathcal{T}' = \mathcal{T} - \{\alpha\} \cup \mathcal{A}$. We say that $\mathcal{T}$ *derives* $\mathcal{T}'$ with rule $\alpha \to \mathcal{A}$ and denote it by $\mathcal{T} \Rightarrow_{\alpha \to \mathcal{A}} \mathcal{T}'$ (or simply $\mathcal{T} \Rightarrow \mathcal{T}'$ when the used rule is clear in the context).

We call $\mathcal{T} \Rightarrow^* \mathcal{T}'$ a *derivation* if and only if either $\mathcal{T} = \mathcal{T}'$ or there are $\mathcal{T}''$ and $\alpha \to \mathcal{A}$ such that $\mathcal{T} \Rightarrow_{\alpha \to \mathcal{A}} \mathcal{T}''$ and $\mathcal{T}'' \Rightarrow^* \mathcal{T}'$ is a derivation.

Let $\mathcal{T} \subseteq (\Sigma \cup \mathcal{N})^+$ be a subset. A terminal *occurs* in $\mathcal{T}$ if it occurs in some string contained in $\mathcal{T}$. Binary relation $\preceq$ on the set of all terminal occurrences in $\mathcal{T}$ is such that,

for any two terminal occurrences a_i and a_j in $\mathcal{T}$, $a_i \preceq a_j$ if and only if (a) $a_i = a_j$, or (b) a_i occurs to the left of a_j in the same string, or (c) there is a terminal occurrence a_h such that a_i occurs to the left of a_h in the same string and $a_h \preceq a_j$.

Theorem 3.1.1. Let $\mathcal{T} \subseteq (\Sigma \cup \mathcal{N})^+$ be a subset and $\{S\} \Rightarrow^* \mathcal{T}$ be a derivation. Then the binary relation $\preceq$ on the set of all terminal occurrences in $\mathcal{T}$ is a total order.

Proof. (Sketch) By induction on m, the number of terminal occurrences in $\mathcal{T}$, where $\{S\} \Rightarrow \mathcal{T}$.

Basis: $m = k + 1$. $\mathcal{T}$ can contain only one string and the last rule used must be of type (D). Therefore, all the terminal occurrences are next to each other on the only string in $\mathcal{T}$, thus forming the total order.

Assumption: for m terminal occurrences in $\mathcal{T}$, the claim is true.

Induction: we assume that there are $m+1$ terminal occurrences in $\mathcal{T}$. Let $\mathcal{T}_1$ be such that $\{S\} \Rightarrow^* \mathcal{T}_1$ and $\mathcal{T}_1 \Rightarrow^* \mathcal{T}$ for which rule $\alpha \rightarrow \{\beta, \gamma\}$ of type (B) and $\beta \rightarrow \theta$ of type (C) are used to introduce a new terminal occurrence a. Let L be the set of m terminal occurrences in $\mathcal{T}_1$. By the assumption, the binary relation $\preceq$ on L is a total order. Note that terminal a co-occurs with other k terminals in the same string θ. Without loss of generality, we assume a occurs to the right of terminal occurrence b and to the left of terminal occurrence c. Then $b \preceq a$ and $a \preceq c$, and for any other terminal occurrence $d \in L$, either $d \preceq b$ or $c \preceq d$, thus either $d \preceq a$ or $a \preceq d$ by the definition of $\preceq$. So the binary relationship $\preceq$ on set $L \cup \{a\}$ is also a total order. □

Definition 3.1.5. Let $\Gamma = (\Sigma, \mathcal{N}, \mathcal{R}, M, S)$ be a k-tree grammar and $\mathcal{T} \subseteq (\Sigma \cup \{M\})^+$ such that $\{S\} \Rightarrow^* \mathcal{T}$. A string $a_1 a_2 \cdots a_n \in \Sigma^+$, $n \geq 3$, is *the underlying string* of $\mathcal{T}$, if for every $1 \leq i < n$, substring $a_i a_{i+1}$ occurs in some string in $\mathcal{T}$. In addition, the language defined by the grammar Γ is

$$L(\Gamma) = \{s \in \Sigma^+ : \mathcal{T} \subseteq (\Sigma \cup \{M\})^+, \{S\} \Rightarrow^* \mathcal{T}, \text{ and } uls(\mathcal{T}, s)\}$$

where predicate $uls(\mathcal{T}, s)$ asserts that s is the underlying string of $\mathcal{T}$.

For example, Figure 2(b) shows a derivation of $\mathcal{T}$ that contains four symbolic strings, for which the string `abcdefg` of 7 terminals is the underlying string.

3.1.2 Structure Space for Individual Strings

The introduced k-tree grammars are context-sensitive that can define crossing relationships among terminals. Let subset $\mathcal{T} \subseteq (\Sigma \cup \mathcal{N})^+$. We call two terminal occurrences *syntactically related* if they appear in the same RHS of some rule used in some derivation $\{S\} \Rightarrow^* \mathcal{T}$. We characterize such relationships of terminal occurrences in $\mathcal{T}$ with notions of graphs.

Definition 3.1.6. Let Γ be a k-tree grammar. Let $\mathcal{T} \subseteq (\Sigma \cup \mathcal{N})^+$ be such that $\{S\} \Rightarrow^* \mathcal{T}$. The *induced graph of* $\mathcal{T}$ is a labeled graph $G_{\mathcal{T}} = (V, E)$, in which vertices have one-to-one correspondence (i.e., labeled) with the terminal occurrences in $\mathcal{T}$ and edges connect vertices corresponding to syntactically related terminal occurrences. The *structure space* $\mathcal{E}(s)$ of any given string $s \in L(\Gamma)$ is defined as

$$\mathcal{E}(s) = \{G_{\mathcal{T}} : \mathcal{T} \subseteq (\Sigma \cup \{M\})^+ \text{ and } uls(\mathcal{T}, s)\}$$

For example, Figure 2(a) is the induced graph for the final set of four symbolic strings in the derivation shown in Figure 2(b), for which `abcdefg` is the underlying string.

Definition 3.1.7. Let $s = s_1 \cdots s_n \in \Sigma^+$ be a string of length n. A (labeled) graph $G = (V, E)$, where $V \subseteq \{1, 2, \cdots, n\}$, is *faithful* to s if

1. $\forall\, i \in V$, vertex i is labeled with s_i; and
2. $\forall\, i, j \in V$, if $i < j$ and $\neg\exists\, l \in V\ i < l < j$, then $(i, j) \in E$.

Lemma 3.1.2. Let $\{S\} \Rightarrow^* \mathcal{T}'$ with the underlying string $s = s_1 s_2 \cdots s_n \in \Sigma^+$. Then for any $\mathcal{T}$ such that $\{S\} \Rightarrow^+ \mathcal{T} \Rightarrow^* \mathcal{T}'$, the induced graph of $\mathcal{T}$ is a faithful k-tree to string s.

Proof. (Sketch) We prove by induction on l, the number of grammar rule applications in the derivation $\{S\} \Rightarrow^+ \mathcal{T}$ to show the induced graph $G_{\mathcal{T}}$ of $\mathcal{T}$ is both a k-tree and faithful to s.

$l = 1$. This is the case that rule $\{S\} \rightarrow \{X_0 a_1 X_1 \cdots a_{k+1} X_{k+1}\}$ is first used. Thus $G_{\mathcal{T}}$, where $\mathcal{T} = \{X_0 a_1 X_1 \cdots a_{k+1} X_{k+1}\}$, consists of $k+1$ vertices $\{i_1, i_2, \cdots, i_{k+1}\}$ labeled with terminal co-occurrences $\{a_1, a_2, \cdots, a_{k+1}\}$. $G_{\mathcal{T}}$ is a $(k+1)$-clique, thus a k-tree. It also is faithful to s since it satisfies condition (b) as no vertices other than $\{i_1, i_2, \cdots, i_{k+1}\}$ are present.

We assume the lemma to be true for the case that fewer than l rules are applied. We now prove it is also true for the case that l rules applied, $l \geq 2$. Let $\mathcal{T}_1$ be such that $\{S\} \Rightarrow^* \mathcal{T}_1 \Rightarrow^* \mathcal{T}$ and $\mathcal{T}_1 \Rightarrow^* \mathcal{T}$ be realized by either a rule of type (D) or a rule of type (B) and then a rule of type (C).

In the case of a rule of type (D) used to realize $\mathcal{T}_1 \Rightarrow^* \mathcal{T}$, no new terminal occurrences are introduced to $\mathcal{T}$. Thus $G_{\mathcal{T}_1} = G_{\mathcal{T}}$, proving the lemma by the assumption.

In the case of a combination of rules of types (B) and (C), one new vertex h, labeled with the new terminal occurrence b in the RHS of the rule of type C, is introduced to $G_{\mathcal{T}}$. New vertex h, along with the vertices labeled with $a_1, \cdots, a_t, a_{t+2}, \cdots, a_{k+1}$, forms a $(k+1)$-clique, thus $G_{\mathcal{T}}$ is a k-tree. In addition, let i and j be two vertices in $G_{\mathcal{T}}$ such that $i < j$ and there is no vertex between them. If neither is labeled with the terminal occurrence b, they should belong to $G_{\mathcal{T}_1}$ as well. By the assumption they satisfy condition (b) of Definition 3.1.7. If i (resp. j) is labeled with b, the rule of type (C) ensures that (h, i) (resp. (h, j)) is included in the new $(k+1)$-clique, thus in $G_{\mathcal{T}}$. Therefore, $G_{\mathcal{T}}$ is a faithful k-tree to s. □

Let $\{S\} \Rightarrow^* \mathcal{T}$ for which s, $|s| = n$, is the underlying string. Then by Lemma 3.1.2, $G_{\mathcal{T}}$ is a k-tree of n vertices faithful to s. According to Definition 3.1.6, edge $(i, i+1)$ is in $G_{\mathcal{T}}$, for all $1 \leq i \leq n-1$. Hence, $G_{\mathcal{T}}$ contains the annotated Hamiltonian path $\{(i, i+1) : 1 \leq i \leq n-1\}$. We thus have the following.

Theorem 3.1.3. Let Γ be a k-tree grammar and string $s \in L(\Gamma)$. The structure space $\mathcal{E}(s)$ is a set of k-trees, each containing the Hamiltonian path $\{(i, i+1) : 1 \leq i \leq n-1\}$, where

$n = |s|$.

On the other hand, we are interested in such k-tree grammars that for every string s in the defined language, the structure space $\mathcal{E}(s)$ contains all possible k-trees (of size $n = |s|$) constrained by the annotated Hamiltonian path. In the following, we show that such k-grammars do exist.

Recall Definition 4.1.1 for creating all possible k-trees. Let $\kappa = \{i_1, i_2, \cdots, i_{k+1}\}$ be an existing $(k+1)$-clique, with $i_1 < i_2 \cdots < i_{k+1}$. We call any new $(k+1)$-clique a *child of* κ if it is formed by a newly introduced vertex along with exactly k vertices already in κ.

Lemma 3.1.4. Let $\kappa = \{i_1, i_2, \cdots, i_{k+1}\}$ be an existing $(k+1)$-clique. Then with the Hamiltonian path constraint, κ can have at most $k+2$ children.

Proof. (Sketch) A new $(k+1)$-clique can be created by introducing a new vertex in one of the $k+2$ intervals $(1, i_1), (i_1, i_2), \cdots, (i_{k+1}, n)$ to connect to exactly k vertices in the clique κ. Therefore, it suffices to show that, for each of the $(k+2)$ intervals, at most one new $(k+1)$-clique can be created.

Without loss of generality, assume two different new $(k+1)$-cliques κ_1 and κ_2 are created with two new vertices h and l drawn from the same interval (i_j, i_{j+1}), respectively, where $i_j < h < l < i_{j+1}$. Apparently (h, l) is not an edge. Nor can there be a path $\{(h, h+1), (h+1, h+2), \cdots, (h+m, l)\}$, where $h + m = l - 1$, for any $m \geq 1$. This is because a new vertex between h and l will only be introduced as a part of descendant of either κ_1 or κ_2 but not both. Therefore, there must be r, $0 \leq r \leq m$, such that edge $(h + r, h + r + 1)$ is not accounted for as a part of the Hamiltonian path. □

Theorem 3.1.5. Let $k \geq 2$ be a fixed integer. There exists a k-tree grammar Γ such that $L(\Gamma) = \Sigma^*$ and, for any given string $s \in L(\Gamma)$ of length n, the structure space $\mathcal{E}(s)$ contains all k-trees constrained by the Hamiltonian path $\{(i, i+1) : 1 \leq i < n\}$.

Proof. (Sketch) It suffices to show that such a desired k-tree grammar has a finite number of rules.

Recall the four types of grammar rules given in Definition 3.1.3. Each rule $\{S\} \to \{\beta\}$ of type (A) induces a $(k+1)$-clique corresponding to the co-occurrence of $k+1$ terminals in β. Such rules can be at most $O(|\Sigma|^{k+1}|\mathcal{N}|^{k+2})$ in number. Each rule $\alpha \to \{\beta, \gamma\}$ of type (B) and each rule $\beta \to \rho$ of type (C) work together to induce an additional $(k+1)$-clique from the $(k+1)$-clique whose vertices are labeled with the $k+1$ terminals that co-occurr in α. As a result of the rule applications two symbolic strings are derived. One symbolic string contains k existing terminals selected from those in α to co-occur with a new terminal occurrence b, while the other symbolic string retains the co-occurrences of $k+1$ terminals in α but "masks off" the segment that introduces b. The latter symbolic string allows rules of types (B) and (C) to be repeatedly applied to induce more $(k+1)$-cliques from the same terminal occurrences in α. By Lemma 3.1.4, rules of types (B) and (C) are bounded by $O(|\Sigma|^{k+2}|\mathcal{N}|^{k+4}k^2)$ in number. Finally, type (D) rules are used to terminate recursion without deriving new terminal occurrence. They are bounded by $O(|\Sigma|^{k+1}|\mathcal{N}|^{k+2})$ in number as well. □

3.2 Probability Computation with k-tree Grammars

3.2.1 Stochastic k-tree Grammars

Definition 3.2.1. A *stochastic k-tree grammar* (SkTG) is a pair (Γ, θ), where $\Gamma = (\Sigma, \mathcal{N}, \mathcal{R}, M, I, S)$ is a k-tree grammar and θ is a function: $\mathcal{R} \to [0, 1]$ such that for every $\alpha \in (\Sigma \cup \mathcal{N})^+$,

$$\sum_{\alpha \to \mathcal{A} \in \mathcal{R}} \theta(\alpha \to \mathcal{A}) = 1$$

We interpret the probability model θ associated with grammar rules as follows. $\theta(S \to \{\beta\})$ is the probability for co-occurrence of the $k+1$ terminals in β. θ associated with all such type (A) rules gives a probability distribution over all co-occurrences of $k+1$ terminals. In addition, θ distributes probabilities between rules of type (B) and of type (D) to account

for the expected number of co-occurrences of $k+1$ terminals that share the same set of at least $k-1$ terminal occurrences. $\theta(\alpha \to \beta)$ of a type (C) rule is probability for co-occurrence of the $k+1$ terminals in β conditional on co-occurrence of the $k+1$ terminals in α.

Definition 3.2.2. Let $\mathcal{T} \subseteq (\Sigma \cup \mathcal{N})^+$ be such that $\{S\} \Rightarrow^* \mathcal{T}$. Then the *probability of derivation* $\{S\} \Rightarrow^* \mathcal{T}$ with (Γ, θ) is defined recursively as

$$Prob(\mathcal{T}|\Gamma, \theta) = \sum_{r \in \mathcal{R},\, \mathcal{T}' \Rightarrow_r \mathcal{T}} Prob(\mathcal{T}'|\Gamma, \theta) \times \theta(r)$$

with the base case $Prob(\{S\}|\Gamma, \theta) = 1$.

Definition 3.2.3. Let (Γ, θ) be a SkTG. Then for any given string $s \in L(\Gamma)$, its probability with (Γ, θ) is defined as

$$Prob(s|\Gamma, \theta) = \sum_{\{S\} \Rightarrow^* \mathcal{T},\, uls(\mathcal{T}, s)} Prob(\mathcal{T}|\Gamma, \theta)$$

Therefore, the probability of s under the model (Γ, θ) is computed as the sum of probabilities of all derivations of s by the grammar. In other word, $Prob(s|\Gamma, \theta)$ is the likelihood for the string s to possess at least one k-tree structure. We observe that

Proposition 3.2.1. Let (Γ, θ) be a SkTG, the strings in the language $L(\Gamma)$ form a probabilistic space, i.e.,

$$\sum_{s \in L(\Gamma)} Prob(s|\Gamma, \theta) = 1$$

Alternatively, it is of interest to know the most likely structure possessed by a given string s. This then is to compute the maximum probability of a derivation $\{S\} \Rightarrow^* \mathcal{T}$ for which s is the underlying string. Similar to the total probability computation, we can define maximum probability recursively,

Let $\mathcal{T} \subseteq (\Sigma \cup \mathcal{N})^+$ be such that $\{S\} \Rightarrow^* \mathcal{T}$. Then the *maximum probability of derivation*

$\{S\} \Rightarrow^* \mathcal{T}$ is defined recursively as

$$Maxp(\mathcal{T}|\Gamma,\theta) = \max_{r\in\mathcal{R},\, \mathcal{T}'\Rightarrow_r\mathcal{T}} Maxp(\mathcal{T}'|\Gamma,\theta)\times\theta(r)$$

with the base case $Maxp(\{S\}|\Gamma,\theta) = 1$.

Definition 3.2.4. Let (Γ,θ) be a stochastic k-tree grammar. Then for every given string $s \in L(\Gamma)$, the maximum probability of a derivation for s is defined as

$$Maxp(s|\Gamma,\theta) = \max_{\{S\}\Rightarrow^*\mathcal{T},\, uls(\mathcal{T},s)} Maxp(\mathcal{T}|\Gamma,\theta)$$

And *the most likely structure* for s with (Γ,θ) is the induced graph $G_{\mathcal{T}^\diamond}$ of the subset $\mathcal{T}^\diamond \subseteq (\Sigma\cup\{M\})^+$ decoded from $Maxp(s|\Gamma,\theta)$, where

$$\mathcal{T}^\diamond = arg\max_{\{S\}\Rightarrow^*\mathcal{T},\, uls(\mathcal{T},s)} Maxp(\mathcal{T}|\Gamma,\theta)$$

3.2.2 Dynamic Programming Algorithms

We now show probability computations with SkTG can be done efficiently. We outline a dynamic programming strategy for computing the maximum probability function $Maxp$. The computation for the total probability function is similar. Let $s = s_1\cdots s_n$, where $s_i \in \Sigma$, for $1 \le i \le n$, be a given terminal string.

Definition 3.2.5. Let $\alpha = X_0a_1X_1\cdots a_{k+1}X_{k+1} \in (\Sigma\cup\mathcal{N})^+$ be a symbolic string and $\kappa = (l_1, l_2, \cdots, l_{k+1})$ be $k+1$ ordered integers where $1 \le l_1 < l_2, \cdots, l_{k+1} \le n$. (α,κ) is *a consistent pair* if

(1) $a_i = s_{l_i}$, $1 \le i \le k+1$, and

(2) For $i = 0, 1, \cdots, k+1$, $X_i = \epsilon$ iff $l_i = l_{i+1} - 1$ ($l_0 =_{df} 1$ and $l_{k+2} =_{df} n$).

Now given a pair (α,κ), we define function $f(\alpha,\kappa)$ to be the maximum probability for a derivation $\{\alpha\} \Rightarrow^* \mathcal{T}$, where $\mathcal{T}\subseteq (\Sigma\cup\{M\})^+$ for which s is the underlying string. Then

function f can be recursively defined according to types of α and the types of rules α is involved with in $\mathcal{R}$.

1. $\alpha \in (\Sigma \cup \{M\})^+$:

$$f(\alpha, \kappa) = \begin{cases} 1 & (\alpha, \kappa) \text{ is a consistent pair} \\ 0 & \text{otherwise} \end{cases}$$

2. $\alpha \in (\Sigma \cup \mathcal{N})^+$ but $\alpha \neq S$:

$$f(\alpha, \kappa) = \max_{r \in \mathcal{R}} \begin{cases} f(\beta, \kappa) f(\gamma, \kappa)\theta(r) & r = \alpha \to \{\beta, \gamma\}, \text{ type (B)} \\ \max\limits_{l_s < h < l_{s+1}, \kappa' = \kappa|_h^{l_{t+1}}} f(\beta, \kappa')\theta(r) & r = \alpha \to \{\beta\}, \text{ type (C)} \\ f(\beta, \kappa)\theta(r) & r = \alpha \to \{\beta\}, \text{ type (D)} \end{cases}$$

   where for the case of r being a type (C) rule, s and t are known values given in $\beta = \alpha|_{YbZ}^{X_s}|_M^{X_t a_{t+1} X_{t+1}}$, satisfying $(s - t) > 1$ or $(t - s) \geq 1$, and $\kappa' = \kappa|_h^{l_{t+1}}$ represents the ordered set modified from κ by replacing l_{t+1} with h.

3. $\alpha = S$:

$$f(S, \kappa) = \max_{S \to \{\beta\} \in \mathcal{R}} f(\beta, \kappa)\theta(S \to \{\beta\})$$

Theorem 3.2.2. $Maxp(s|\Gamma, \theta) = \max\limits_{\kappa \in [n]^{k+1}} f(S, \kappa)$, where $[n]^{k+1}$ is the set of all combinations of $k + 1$ integers in $[n] = \{1, 2, \cdots, n\}$.

Proof. (Sketch) We prove by induction on the number m of rule applications in a process to generate the string s with the maximum probability, where $m \geq 2$. The base case $m = 2$ is obvious. The proof of inductive step examines all possible types of rules used in the last step. □

A dynamic programming algorithm can be implemented to compute function $f(\alpha, \kappa)$.

This is to establish a table to store computed values of function f through the use of the formulae provided above (the cases 1 through 3). The table has $k+2$ dimensions, one for all α's in the grammar and the other $k+1$ are for all κ's, resulting in the $O(n^{k+2}|\Gamma|)$-time and $O(n^{k+1}|\Gamma|)$-space complexities, respectively, for every fixed k.

3.3 Applications and Discussions

We have introduced the stochastic k-tree grammar (SkTG) for the purpose of modeling context-sensitive yet tamable crossing co-occurrences of terminals. The recursive rules of the new grammar permit association of probability distributions in a natural way. The resulting dynamic programming algorithms for probability computation with SkTG are efficient enough, with potential for statistical analysis of real-world structures. This work is in progress in both application and further theoretical investigation.

3.3.1 Application in Biomolecular Structure Prediction

This work was initially motivated by the need in the analysis of biomolecules for tertiary structure prediction. A biomolecular sequence, e.g., ribonucleic acid (RNA) or protein, is a linear chain of residues interacting spatially to form a 3D structure functionally important [Noller, 1984; Murzin et al., 1995]. One of the most desirable computational biology tasks is to predict the tertiary structure from the sequence information only [J. et al., 2009; Y, 2008]. The newly introduced SkTG offers a viable approach to this task. We briefly outline the application as follows.

Biomolecular sequences are natural strings definable over some finite alphabet Σ (e.g., $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$ for nucleic acids). A class of biomolecular sequences can be defined as a language with a SkTG in which grammar rules model statistically not only the sequential composition but also structural composition of the sequences. The task of designing SkTGs, much like that for SCFGs, is non-trivial and may often be based on experience. Equipping a

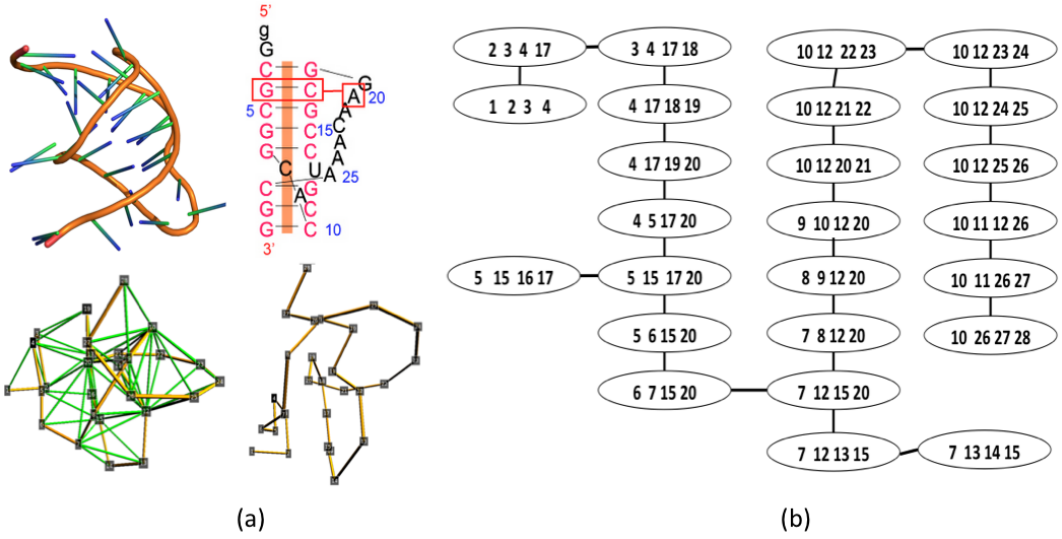


Figure 3.3: Illustration of a tertiary structure prediction from BWYV (beet western yellows virus) RNA molecule sequence (PDB ID: 1L2X) that contains 28 nucleotides, coaxial helices connected with two loops, and an A-minor motif. Top of (a): Tertiary structure (drawn via `pymol`) and details of nucleotide interactions (`http://www.biomath.nyu.edu/motifs/`); (b) The induced 3-tree (containing desired interactions) corresponding to a derivation of the sequence with the maximum probability. The 3-tree is presented in terms of the tree topology connecting the created 4-cliques in the 3-tree. Bottom-left of (a): 3D representation of the 3-tree with one tetrahedron for every 4-clique; Bottom-right of (a): only backbone edges are kept from the tetrahedron representation, serving as a preliminary structure prediction from the sequence. We note that more accurate structural motif modeling of individual 4-cliques would allow more accurate prediction of the overall tertiary structure.

designed SkTG Γ with probability parameters θ may be done through learning from known biomolecules (with or without known structures) (see next subsection for a briefly discussion).

With an SkTG (Γ, θ), using the dynamic programming algorithm (developed in section 3) we can compute the maximum probability of an induced k-tree, e.g., $k = 3$, from a given query sequence. In such an application, every $(k+1)$-clique κ in the desired k-tree may potentially admit one of many possible configurations (i.e., all possible interaction topologies along with permissible geometry shapes) for the $k+1$ residues in κ. Therefore, the dynamic programming algorithm can be tailored to include the third argument C_κ in the probability function f defined in section 4, where C_κ is the set of all possible configurations incurred by $(k+1)$-clique κ. The information about C_κ can often be obtained from known tertiary structures of biomolecules as well. Figure 3.3 illustrates this approach used in a tertiary structure prediction for a small RNA molecule.

3.3.2 Further Theoretical Issues

SkTG is a natural extension from SCFG; in particular, k-tree grammars, for $k = 2$, can define all context-free languages. In addition, the outlined dynamic programming algorithm (in section 3.2) to compute the maximum probability can be improved. In fact, in a related work [Ding et al., 2013], the authors have developed an algorithm of time $O(n^{k+1})$, for every fixed value of k, for computing the maximum spanning k-tree that includes a designated Hamiltonian path. On the other hand, due to the long standing barrier of $O(n^3)$ for parsing context-free languages, this also suggests the time complexity upper bound $O(n^{k+1})$ has optimal order of growth in n for each $k \geq 2$.

We further note that the above efficiency issue is closely related with the parameterized complexity [Downey and Fellows, 1999] of the following problem: computing the maximum probability of an input sentence to be produced by an input SkTG, for which k is considered a variable parameter. By the above observation, such a problem is likely parameterized intractable. Nevertheless, the interesting question remains whether an additional small pa-

rameter (e.g., significant in applications) may be associated with such problems for further improvement of computational efficiency.

Estimation of probability parameters θ for given k-tree grammars deserves more thorough investigation and it is not within the scope of this chapter. However, we point out that it is highly possible to develop efficient parameter estimation algorithms for SkTG. This is because $O(n^{k+1})$-time algorithms may exist for computing the maximum and total probabilities of given language strings. Much like the analogous algorithms for SCFG, these algorithms can be used to re-estimate probability parameters θ given an initial parameter θ_0, through an EM algorithm.

Finally, we feel that future work is also needed to investigate the relationship between the k-tree grammar and other grammars that already exist (e.g., the Tree-Adjoining Grammar and its generalized versions [Joshi and Vijay-Shanker, 1991]) for constrained context-sensitive languages.

Chapter 4

RNA Nucleotide Interaction Prediction with Backbone k-Tree Model

In the past decade, there have been many revelations of the importance of non-coding RNAs to cellular regulatory functions and thus a growing interest in the computational prediction of RNA 3D structure [Laing and Schlick, 2010; Leontis and Westhof, 2012a]. Nevertheless, RNA 3D structure prediction from a single RNA sequence is a significant challenge. One major unresolved issue is the immense space of tertiary conformations even for a short RNA sequence. Existing methods usually employ random sampling algorithms for computation feasibility, which assemble sampled tertiary motifs into native-like structures [Das and Baker, 2007; Ding et al., 2008; Jonikas et al., 2009; Parisien and Major, 2002; Popenda et al., 2012; Sharma et al., 2008]. To reduce the chance to miss native structures, the assembly algorithms have mostly been guided with constraining structural models. For example, MC-Fold/MC-Sym [Parisien and Major, 2002] assumes the 3D structure consists of 4-nt cyclic tertiary motifs constructible from the predicted secondary structure. Rosetta [Das and Baker, 2007; Das et al., 2010] *de novo* assembles 3D structure from a database of 3-nt tertiary fragments. Other methods follow samplings that preserve the secondary structure [Bida and Maher, 2012; Popenda et al., 2012; Reinharz et al., 2013]. However, these constraining models do not necessarily ensure that native conformations are examined. The state-of-the-art methods have yet to deliver the desired prediction accuracy for RNA sequences of lengths beyond 50 nucleotides [Laing and Schlick, 2010].

In this chapter, we introduce a novel method to predict nucleotide interactions from

known or predicted canonical basepairs as a key step toward accurate prediction of 3D structure. Accurate knowledge of the nucleotide interactions is crucial to predicting the 3D structure of an RNA and subsequently predicting its functional roles. To predict nucleotide interactions, our method is guided by a novel graph model called a *backbone k-tree*, for small integer k, to globally constrain the nucleotide interaction relationships (NIRs) that constitute the 3D structure. In such a k-tree graph, nucleotides are organized into groups of size $k+1$, such that NIRs are permitted only for nucleotides belonging to the same group and groups are connected to each other with a tree topology (see section 2). This model was inspired by our recent discovery of the small treewidth of the NIR graphs for more than 3,500 RNA chains extracted from 1,984 RNAs whose structure has been resolved (Figure 4.1). Treewidth is a graph metric, which indicates how much a graph is tree-like [Bodlaender and Koster, 2010; Van Leeuwen, 1990]. We have been able to develop dynamic programming (DP) algorithms with $O(n^{k+1})$ time and space complexities, efficient for small k, to compute the optimal backbone k-tree spanning over the nucleotides on the query sequence, given a scoring function [Ding et al., 2014a, 2016a].

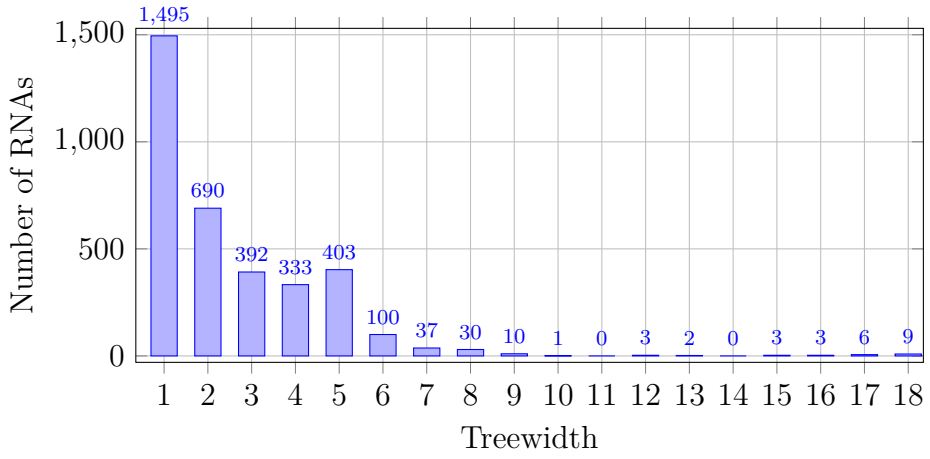


Figure 4.1: Treewidth distribution of NIR graphs of more than 3,500 RNA chains from the RNA Structure Atlas [Sarver et al., 2008]. The RNAs with treewidth larger than 18 are omitted due to their very small number. These treewidths are actually upper bounds computed by an approximation algorithm [Bodlaender and Koster, 2010]; it is likely that the exact treewidths of the NIR graphs may be smaller.

To ensure that the computed optimal k-tree can actually yield the set of nucleotide in-

teractions that constitutes the native 3D structure, our method proposes to identify detailed patterns of nucleotide interactions for every group of $k+1$ nucleotides found in known RNA 3D structures and to score every such pattern. We consider nucleotide interactions from the established geometric nomenclatures and families [Leontis and Westhof, 2001; Leontis et al., 2002; Stombaugh et al., 2009], including base-base, base-phosphate, base-ribose [Zirbel et al., 2009; Zirbel, 2011], base-stacking interactions as well as the phosphodiester bonds between two neighbouring nucleotides on the backbone. To test our method, we adopted an improved 3-tree model, and pre-computed candidates of interaction patterns for every group of 4 given nucleotides (see Figure A.1 in Appendix A). These annotated atom-level interaction patterns have been extracted from resolved 3D structures of RNAs in the RNA Structure Atlas [Sarver et al., 2008]. To avoid overfitting, only nucleotide interactions from RNAs of length ≤ 100 nucleotides were selected. To score such patterns, we trained artificial neural networks (ANNs) to compute the confidence of every admissible nucleotide interaction pattern for every group of 4 given nucleotides. We filtered out unlikely interaction patterns and kept only those with high confidences. With this 3-tree model, our algorithm efficiently predicts an optimal set of nucleotide interactions from the query sequence (along with canonical base pairs) within computational time $O(n^3)$. We have implemented the algorithm into a program called BkTree as a part of a 3D structure prediction framework (Figure A.2 in Appendix A).

We evaluated our methods through testing BkTree on two sets of data. First, the performance of nucleotide interaction prediction was measured on a set of 43 RNAs of lengths ranging from 26 to 128 nucleotides, a benchmark used by the survey of state-of-the-art 3D structure prediction methods [Laing and Schlick, 2010]. The resolved, atom-level interactions of these high resolution RNAs were extracted with FR3D [Sarver et al., 2008]. Sensitivities (STY), positive predictive values (PPV) and Matthews correlation coefficients (MCC) [Laing and Schlick, 2010] of the nucleotide interactions predicted by BkTree were calculated for these 43 RNAs. The overall performance was also compared with previous programs MC

[Parisien and Major, 2002], Rosetta [Das and Baker, 2007], and NAST [Jonikas et al., 2009]. Second, performance of nucleotide interaction prediction was also evaluated by testing BkTree on a set of 13 single RNA chains from PDB [Berman et al., 2000b] of lengths between 100 and 200 nucleotides, whose nucleotide interaction patterns have not been extracted for the construction of ANNs. These evaluations show that BkTree predicted nucleotide interactions with high accuracies across the tested RNAs, including RNAs whose interaction patterns were not used for training. Our method also impressively outperformed the other methods on the overwhelming majority of the tested RNAs and showed a great potential for handling RNAs beyond short lengths.

4.1 Model and Methods

In this work, we consider RNA nucleotide interactions of atomic-resolution of all known types, Table A.1 in Appendix A summarizes these nucleotide interactions by their geometric families.

We use notation $\langle X, Y, t\rangle$ for a type t interaction between nucleotides X and nucleotide Y (from 5' to 3'), where $X, Y \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$. Let $I_{XY} = \{\langle X, Y, t\rangle : t \text{ is an interaction type}\}$ and $I = \bigcup_{X,Y\in\{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{U}\}} I_{XY}$.

4.1.1 Backbone k-Tree Model

Let $S = S_1S_2...S_n$ be an RNA sequence, in which $S_i \in \{A, C, G, U\}$, for $1 \leq i \leq n$. The *nucleotide interaction relation* (NIR) *model* for the native structure of S is a pair $\langle G; A\rangle$, where $G = (V, E)$ is called the *NIR graph* of S with vertex set $V = \{1, 2, \ldots, n\}$, and A is an *association* such that for every pair $i < j$, $A(i, j) \subseteq I_{S_iS_j}$ is the set of interactions between nucleotides S_i and S_j in the native structure and $A(i, j) \neq \emptyset$ implies $(i, j) \in E$. Note that because of phosphodiester bonds between neighboring nucleotides, the NIR graph G always contains the Hamiltonian path $(i, i + 1)$, $i = 1, 2, \ldots, n - 1$; these edges are named *backbone*

edges.

In our recent investigation [Ding et al., 2014b,a], we constructed NIR graphs for all RNAs whose 3D structures were known from RNA Structure Atlas [Reinharz et al., 2013]. We discovered that an overwhelming majority of these RNAs are of small treewidths (Figure 4.1). Theoretically, if a graph has treewidth bounded by k, any clique obtained by deleting vertices and edges and contracting edges of the graph can contain at most $k+1$ vertices [Arnborg et al., 1990]. Thus the distribution of treewidths suggests that NIRs in the RNA 3D structures are in general not arbitrarily complex.

The concept of treewidth is closely related to, and may be better explained with the notion of *k-tree*, which is central to this work.

Definition 4.1.1. [Patil, 1986] Let integer $k \geq 1$. The class of *k-trees* are graphs defined by the following inductive steps:

1. A k-tree of $k+1$ vertices is a clique of $k+1$ vertices;

2. A k-tree of n vertices, for $n > k+1$, is a graph consisting of a k-tree G of $n-1$ vertices and a vertex v, which does not occur in G, such that v forms a $(k+1)$-clique with some k-clique already in G.

Figure A.3 (a) and (b) in Appendix A show a 3-tree of 7 vertices. It is well known that for any $k \geq 1$, a graph is of treewidth $\leq k$ if and only if it is a subgraph of some k-tree [Arnborg and Proskurowski, 1989]. This also suggests that every graph of treewidth $\leq k$ can be augmented with additional edges into a k-tree. Thus, given a NIR model $\langle G; A\rangle$, where NIR graph G is of a treewidth $\leq k$, one can augment G to a k-tree with additional edges, and for each newly added edge (i, j), let $A(i, j) = \emptyset$. Since such k-trees contain all backbone edges, they are called *backbone k-trees*.

Definition 4.1.2. A *backbone k-tree model* is a NIR model $\langle G; A\rangle$ in which NIR graph G is a backbone k-tree.

This allows us to conclude that for small values of k, backbone k-tree models exist for an overwhelmingly majority of RNAs whose native structures are known. Figure A.3 (c) in Appendix A shows a backbone 3-tree as the NIR graph for a short sequence consisting of 7 nucleotides.

To predict the set of nucleotide interactions from a query sequence $S = S_1S_2, \ldots S_n$, we propose to identify a backbone k-tree model $\langle G; A\rangle$, where $G = (V, E)$ and $A(i, j) \subseteq I_{S_iS_j}$ such that $A(i, j) \neq \emptyset \Longrightarrow (i, j) \in E$. To ensure the identified model actually corresponds to the set of nucleotide interactions that constitute the native structure of the query sequence, we will quantify nucleotide interactions for the optimization computation of such a backbone k-tree model.

4.1.2 Quantification of Nucleotide Interactions

Definition 4.1.3. Let q be a $(k + 1)$-clique in a backbone k-tree of query sequence S. An *interaction pattern* (ip) for clique q is a set $A(q)$ of nucleotide interactions, for some association A, such that $A(q) = \cup_{i,j\in q}A(i, j)$.

Given an ip $A(q)$ for clique q, the *subgraph of q induced by $A(q)$*, denoted with $H_{q,A(q)} = (q, E_{q,A(q)})$, is such that $(i, j) \in E_{q,A(q)}$ if and only if $A(i, j) \neq \emptyset$. Figure A.1 in Appendix A illustrates the examples of a $(k + 1)$-clique q, two ips of q and their induced subgraphs.

Definition 4.1.4. Let q be a $(k + 1)$-clique in a backbone k-tree of query sequence S. The *confidence* of a given ip $A(q)$ for clique q is defined as

$$f(q, A(q), S) = \sum_{(i,j)\in E_{q,A(q)},\, \langle S_i,S_j,t\rangle\in A(i,j)} c^{(i,j),t}_{q,H_{q,A(q)}} \tag{4.1}$$

where $c^{(i,j),t}_{q,H_{q,A(q)}}$ is the *confidence* of interaction $\langle S_i, S_j, t\rangle$ given q and the subgraph $H_{q,A(q)}$ induced by ip $A(q)$.

In Section 4.2, we will introduce artificial neural networks (ANNs) that compute confi-

dence $c^{(i,j),t}_{q,H_{q,A(q)}}$.

For every clique q, with $\mathcal{P}(q)$, we denote the finite set of all ips for q. In the practical application, we may only include those ips in $\mathcal{P}(q)$ which have "high" confidences (e.g., above certain threshold).

Definition 4.1.5. Let k be any fixed integer ≥ 2. The *nucleotide interaction prediction* problem NIP(k) is, given an input query sequence S, to identify a backbone k-tree model $\langle G^*; A^* \rangle$, such that

$$(G^*; A^*) = \arg \max_{\langle G;A \rangle} \{ \sum_{q \text{ in } G,\, A(q) \in \mathcal{P}(q)} f(q, A(q), S) \} \tag{4.2}$$

4.1.3 Overview of the Method

To solve the NIP(k) problem, our method consists of three major components.

1. Data repositories include NIPDB and NIPCTable. NIPDB is a database of all possible interaction patterns. To build the database, we first extracted a set $\mathcal{P}(q)$ of nucleotide interaction patterns for every $(k+1)$-clique q, which were found in the known 3D structures of RNAs with length ≤ 100 nucleotides. Then an unique identifier was assigned to each such clique by taking into account both the nucletides and their backbone distances. See Figure A.1 in Appendix A for examples.

   NIPCTable is a matrix for compatibility between every pair of ips for two cliques that share all but one vertex (nucleotide). To compute for the optimization problem formulated with (2), for every two $(k+1)$-cliques q_1 and q_2 that are adjacent in the k-tree G, $A(q_1)$ and $A(q_2)$ are required to be compatible in the sense that the two interaction sets among the k common nucleotides of q_1 and q_2 are identical. The compatibility of all pairs of ips in NIPDB forms a binary matrix. For the efficiency, the compatibility can be precomputed before the prediction program is executed. The nucleotides in an ip are ordered from $5'$ to $3'$. Given two ips I_1 and I_2 each with $k+1$ nucleotides, we

enumerate all $(k+1)^2$ ways of mapping k nucleotides of I_1 to k nucleotides of I_2. For each of the mappings, if the selected two sets of k nucleotides are not identical, two ips are not compatible; otherwise, we further verify the interactions among the two sets of k nucleotides and the compatibility holds when they are identical.

2. A set of ANNs; each computes the confidence for a specific interaction between two given nucleotides on the query sequence.

3. A dynamic programming algorithm that computes the solution to equation (4.2).

Given the query sequence S and the known or predicted canonical basepairs on S, our method first employs ANNs to compute $c^{(i,j),t}_{q,H_{q,A(q)}}$, the confidence of the interaction $\langle S_i, S_j, t\rangle$ in the interaction pattern $A(q)$ for $(k+1)$-clique q that involves vertices i and j , where $H_{q,A(q)}$ is the subgraph induced by ip $A(q)$. This is done for every pair of $i < j$, every interaction type t, every $(k+1)$-clique q and every ip $A(q) \in \mathcal{P}(q)$ for q. Then it computes the confidence score $f(q, A(q), S)$ for every ip $A(q) \in \mathcal{P}(q)$ of every $(k+1)$-clique q, using formula (4.1). Afterward, it runs the dynamic programming algorithm to solve equation (4.2).

4.2 Algorithms

4.2.1 ANNs for Computing Interaction Confidence

We constructed ANNs that compute confidences of nucleotide interactions, one ANN for every specific nucleotide interaction $\langle S_i, S_j, t\rangle$ contained in a given specific interaction pattern $A(q)$ of a given $(k+1)$-clique q. We use $\mathcal{N}^{(i,j),t}_{q,H_{q,A(q)}}$ to denote such an ANN and $c^{(i,j),t}_{q,H_{q,A(q)}}$ for the confidence score that the ANN computes. Each ANN $\mathcal{N}^{(i,j),t}_{q,H_{q,A(q)}}$ consists of an input layer, two hidden layers (with 8 and 16 nodes, respectively), and an output layer (Figure A.4 in Appendix A). The output layer is a single unit producing a confidence value for interaction $\langle S_i, S_j, t\rangle$. The input layer consists of input units representing the selected global and local features shown in Table A.2 in Appendix A. The features included the sequence length and

the distance between the involved nucleotides as well as neighboring nucleotide types. In addition, we included the information of *assumed* canonical base pairs within the query sequence.

We adopted conventional methods to construct and train each ANN [Mitchell, 1997], typically the technique of back-propagation with gradient descent, using a fixed-size network. The learning rate 0.03 were the values that yielded the best results for some representative ANNs. The training data for the ANNs were from RNA Structure Atlas. We removed all RNAs of lengths larger than 100 nucleotides and RNAs with missing nucleotides. This resulted in a subset of 895 RNAs of single chains. Then all the $(k+1)$-cliques q along with their features of every RNA in the subset were enumerated to form a whole set T. Due to different number of features, the number of $(k+1)$-cliques associated with different q, interaction pattern $A(q)$, and subgraph $H_{q,A(q)}$ in T vary considerably. As a result, for most of the ANNs, only a small portion of 895 RNAs were used for training and testing. A 10-fold cross validation was used to avoid over-fitting.

4.2.2 Algorithm for NIP(k) problem

Our algorithm solves the NIP(k) problem by producing a pair $\langle G^*; A^* \rangle$ satisfying equation (4.2) for the query sequence. In particular, the backbone k-tree G^* constrains the nucleotide interaction relationship topology, together with the association A^* of nucleotide patterns with all $(k+1)$-cliques in G^*, to achieve the maximum confidence score. The algorithm maximizes the confidence score of a backbone k-tree spanning over the query sequence nucleotides by a dynamic programming process. To derive recurrences for the dynamic programming, we followed the basic process of creating k-trees given in Definition 1. The inclusion of backbone edges in the k-trees disallows introducing edges in arbitrary order and thus makes the search space much smaller. We briefly explain this algorithm in the following paragraphs.

By *interval* $[i..j]$, for $i \leq j$, we mean the set of consecutive integers between i and j,

inclusive. Two intervals $[i..j]$ and $[h..l]$ are *non-overlapping* if either $j \leq h$ or $l \leq i$. Let the query sequence be S of length n and q be a $(k+1)$-clique formed by $k+1$ vertices drawn from $\{1, 2, \ldots, n\}$. Let C be a set of non-overlapping intervals and $A(q) \in \mathcal{P}(q)$ be an ip for clique q. We define function $M(q, C, A(q), S)$ to be the maximum confidence of a k-tree constructed beginning from clique q, which includes backbone edge $(i, i+1)$ for every pair of integers i and $i+1$ both contained in some interval in C. Then we obtain the following recurrence:

$$
\begin{aligned}
& M(q, C, A(q), S) \\
& \quad = \max_{x \in q,\, y \notin q,\, y \in [i..j] \in C,\, p = q|_y^x} \\
& \qquad \{ \max_{A(p) \in \mathcal{P}(p),\, \mathcal{R}(C_1, C_2),\, \mathcal{Q}(A(p), A(q))} \{ M(p, C_1, A(p), S) \\
& \qquad\quad + M(q, C_2, A(q), S) + f(q, A(q), S) \} \}
\end{aligned} \tag{4.3}
$$

where abbreviations $q|_y^x = q \cup \{y\} \setminus \{x\}$, $\mathcal{Q}(A(p), A(q))$ asserts that the chosen ip $A(p)$ be compatible with the ip $A(q)$, and $\mathcal{R}(C_1, C_2)$ represents the choices of two sets of intervals, C_1 and C_2, which satisfy the following constraints

1. $\{[i..y], [y..j]\} \subseteq C_1$, $\{[w..x], [x..z]\} \subseteq C_2$, for applicable w and z; and
2. $C_1 \cup C_2 = C \cup \{[i..y], [y..j]\} \setminus \{[i..j]\}$, and $C_1 \cap C_2 = \emptyset$.

Recurrence (5.4) offers a bottom-up process to compute $M(q, C, A(q), S)$. Intuitively, the idea is to create a new clique p from q by introducing a new nucleotide vertex y. There may be one or more sub-k-trees, some stemming from p while others from q (but not including vertex y). Since these sub-k-trees will never join together again, interval sets are used to ensure backbone edges will be properly created. In particular, the set of backbone edges in the k-tree corresponding to the value of function $M(q, C, A(q), S)$ contains only those edges between consecutive indexes specified in the intervals in C. Initially, C may include intervals allowing all backbone edges.

The confidence score of the produced k-tree is computed as the sum of confidence scores of ips chosen for all involved $(k+1)$-cliques. The chosen ips need to be compatible across the cliques

when they share nucleotide interactions or even just nucleotides. This is ensured by the assertion $\mathcal{Q}(A(q), A(p))$ by looking up table NIPCTable. In addition, any pattern of interactions between a single nucleotide and multiple others has to exist in the structure database.

To complete the recurrence (5.4), we need the following base case:

$$M(q, C, A(q), S) = 0 \quad \text{if } C = \emptyset$$

which will be first computed in a bottom-up dynamic programming strategy.

4.2.3 Improved Algorithms

Implementation of the above outlined dynamic programming algorithm would require $O(n^{k+1})$ memory space and $O(n^{k+2})$ computation time for every fixed value of k. Following the same idea but creating $(k+1)$-cliques from k-cliques instead has lead to an improved algorithm, with a few more sophisticated steps to navigate through k-cliques. The improved algorithm uses $O(n^k)$ memory and $O(n^{k+1})$ time for every fixed value of k [Ding et al., 2016a].

For $k = 3$, the time efficiency can be further improved to $O(n^3)$ with a constrained backbone k-tree model that requires every (k + 1)-clique to contain at least one backbone edge $(i, i+1)$ for some i. Testing has shown that the constrained backbone k-tree model did not weaken the capability to account for sophisticated nucleotide interactions as the "standard" backbone k-tree model. The constrained model may reduce biologically unfavorable interaction patterns, e.g., those not involving locally related nucleotides.

4.2.4 Implementation

We have implemented the new method into a prototype system. The NIPDB database construction was coded in Python, where the Prody package [Bakan et al., 2011] was adopted to search the RNA Structure Atlas. NIPCTable, the matrix for ip consistence was developed using Python. Building and training of all ANNs were realized with WEKA package [Hall et al., 2009] in nearly a month. Programs were coded in Java to compute confidences of ips admissible for every $(k+1)$-clique in the query sequence.

We implemented in C++ the dynamic programming algorithm into a program called BkTree based on the constrained backbone 3-tree model. We ran the evaluation tests on a Red Hat 4.8.2-7 server with 4 Intel Quad core X5550 Xeon Processors, 2.66GHz 8M Cache and 70GB Memory. The server runs nearly an hour for predicting a sequence of 100 nucletodies.

4.3 Performance Evaluation

4.3.1 Test Data

We implemented our method in the program BkTree. We evaluated our method through testing BkTree on two sets of RNAs of high resolution structures. One was a list of 43 RNAs that had been used as a benchmark set in the survey of state-of-the-art 3D structure prediction methods [Laing and Schlick, 2010]. Eighteen of the RNA sequences are of length ≥ 50 nucleotides. In developing the ANNs for computing interaction confidences, 7 of these RNAs were not included in set T. The second set was 13 high resolution (3.5Å or better) single chain RNAs of lengths from 101 to 174 nucleotides, none of which was included in T.

Given the recent progress made in RNA secondary structure prediction [Laing and Schlick, 2010; Reinharz et al., 2013], we believe that canonical base pairs may be routinely predicted with a fair accuracy. Therefore, we have allowed the program BkTree to accept known or predicted canonical base pairs along with the query sequence as input. Note that the knowledge of canonical base pairs does not necessarily imply the whole secondary structure, which is often a part of input to most of the existing RNA 3D prediction methods. In our test, we extracted canonical base pairs of a RNA from FR3D analyzed interactions [Sarver et al., 2008].

4.3.2 Overall Performance

We evaluated the quality of the predicted nucleotide interactions by the sensitivity (STY) and positive predictive value (PPV) against the FR3D-analyzed interactions [Sarver et al., 2008]. In order to take into account the effects of both true positive and false positive rates in one measure, the *Matthews correlation coefficient* (MCC), defined in [Laing and Schlick, 2010] as MCC

$:= \sqrt{\text{PPV} \times \text{STY}}$, was also calculated.

Figure A.5 in Appendix A summarizes the overall performance of BkTree on the benchmark set. On a large majority of RNAs, the sensitivity is decently high. Note that the STY and PPV calculations excluded the canonical base pairs. The sensitivity result indicates that our method has a high accuracy in identifying non-canonical interactions that may be crucial to 3D structures. This is true even for those longer RNAs. We further note that for the 7 RNAs that are not in T, BkTree also performed very well.

We point out that in Table A.5 in Appendix A almost all of the relatively low MCC values (below 0.8) were caused by relatively low sensitivity (STY) values. These low sensitivity values were due to that the backbone 3-tree is too weak to model RNAs of structures more complex than helices and junctions, such as pseudoknots. This is evident by the column EdgeDiff, which is the ratio of total number of edges in the NIR graph of the RNA to the number of edges that the constrained 3-tree model is able to include. k-tree models, with higher k values, can include all edges of the NIR graph and thus is expected improve the performance of prediction (see Section 5 for more discussions on how such k-tree models can be efficiently implemented).

4.3.3 Performance Comparison with Other Methods

We compared our program BkTree with the programs MC, Rosetta, and NAST on the capability to predict nucleotide interactions. These other state-of-the-art methods had been surveyed and evaluated in [Laing and Schlick, 2010] based on their ability to identify both base pairing and base stacking interactions only. We removed base-phosphate and base-ribose interactions from our prediction results. We incorporated the canonical base pairs into our results because these other methods include all interactions from the input secondary structure.

Figure 4.2 shows the MCC plots for MC, Rosetta, NAST, and BkTree on the benchmark set of RNAs. Data of RNAs failed by a program were not included in the calculation. We note that for every RNA, these other programs produced more than one conformation so the results were averaged for these comparisons. The figure demonstrates that BkTree overall outperformed the other three programs in predicting non-canonical base pairing and base stacking interactions.

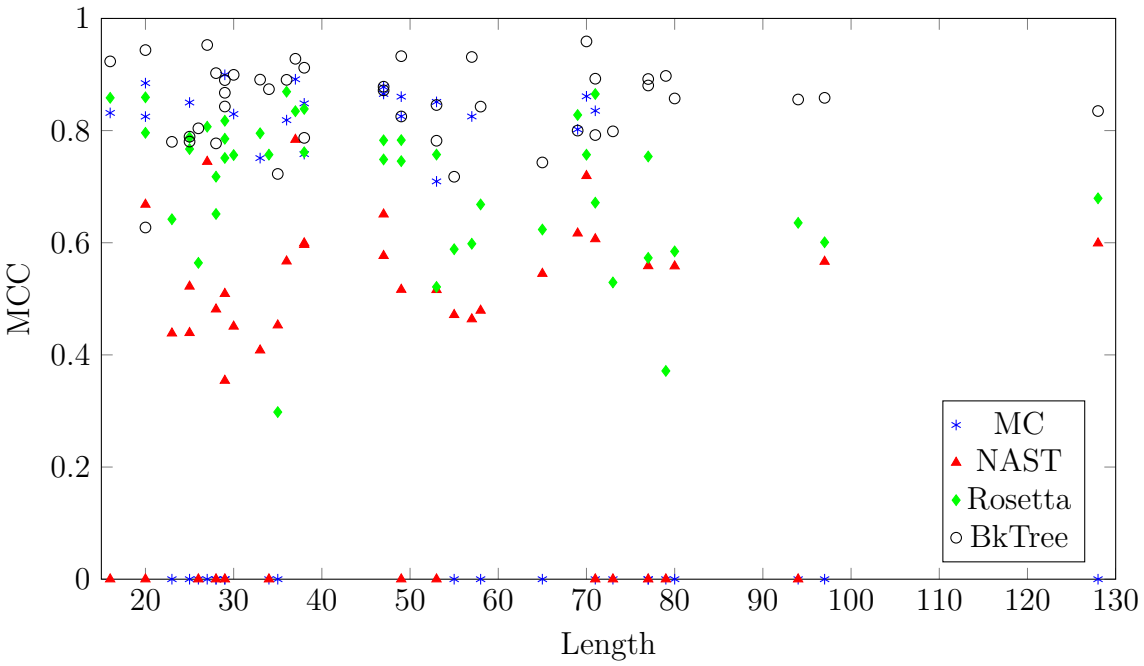


Figure 4.2: Comparison of the MCC generated by MC, NAST, Rosetta and BkTree. The MCC of the 43 RNAs are calculated by including canonical base pairs in the results and sorted by their lengths. The plot was derived by merging the results obtained by BkTree and the data computed in the survey (Laing, 2014; Laing and Schlick, 2010). In that survey, the 3D structure predictions with the other 3 methods were based on resolved secondary structures and the secondary structures were included in the calculations. Therefore, the canonical base pairs were also been added to the prediction results by BkTree.

In addition, Table A.3 in Appendix A gives comparisons on average performance across the 43 RNAs between the four methods. In general, Bktree produced much better average results than Rosetta and NAST, and comparable average results with MC, for which BkTree shows better average STY value than MC, whereas MC gives better average PPV. On MCC values, BkTree had an average over MC. On RNAs of length $\geq$ 50 nucleotides, BkTree maintained almost the same average MCC as it did on the whole set.

4.3.4 Performance on long RNAs

We also evaluated performance of BkTree on 13 longer RNAs with diverse structures, including riboswitches, pseudoknots, synthetic RNAs and RNAs containing multi-way junctions. These RNAs

have lengths from 101 to 174 nucleotides and thus they were not included in the training data for ANNs. Table A.4 in Appendix A shows that the performance on these long RNAs is comparable to that of the 43 benchmark RNAs. In particular, the average MCC of the predictions is 0.787, suggesting the capability of our method to predict nucleotide interactions for RNAs of lengths beyond 100 nucleotides.

4.4 Discussions

Our method is a non-conventional framework to predict RNA nucleotide interactions (of all known types) without simultaneous prediction of 3D structure. The underlying backbone k-tree model drastically reduces the space of plausible nucleotide interaction relations, permitting not only efficient but also effective prediction of nucleotide interactions. The evaluation test results have highlighted the potential of our method as a viable step toward accurate 3D structure prediction of RNA sequences beyond short lengths.

Our work has taken advantage of the recent growth of knowledge in the rich, high-resolution nucleotide interaction data. In particular, our method predicts the most plausible set of interactions based on confidence scores of individual interactions computed with artificial neural networks (ANNs). The neural networks were trained and tested with a small subset of single chain RNAs (all of lengths ≤ 100 nucleotides) extracted from the established database RNA Structure Atlas. For each ANN that computes an individual interaction, a 10-folds cross validation was used to avoid overfitting. Indeed, test results on RNAs not in the training data, especially those of lengths > 100, have apparently justified the rationale of the proposed confidence scores.

The evaluation tests have revealed that our method is robust in the sense that only choices of k for the k-tree model may affect the prediction results. A careful look at Figure A.5 in Appendix A shows that these RNAs are of more complex structures and their nucleotide interaction relationships are actually beyond the capability of the backbone 3-tree model built in the program BkTree. In particular, column EdgeDiff of Table 4 shows that all these seven RNAs have more than a few edges in their NIR graphs which cannot be included by even the best backbone 3-tree model. For example, the 3-tree model can miss 6 and 9 edges in the NIR graphs of the long RNAs 1LNG and

1MFQ, respectively. These edges correspond to some important nucleotide interactions including those between the hairpins of two helices in these two signal recognition particle RNAs. Failure to predict these crossing interactions may result in considerably under performance in their 3D models.

Therefore, backbone k-tree models, for $k > 3$ are expected to improve performance for RNAs of complex structures. However, one major concern with such a model is the possibly impractical complexity $O(n^{k+1})$, for $k \geq 4$, of implementation with the developed dynamic programming algorithm. Our more recent study reveals that such obstacle can be surmounted by taking advantage of some inherent properties of backbone k-trees constructed from known RNA structures. For example, our survey on more than 600 RNAs with known 3D structures (data not shown) suggests that in backbone 4-tree models of these RNAs 5-cliques are basically of two types. One type of clique is that the 5 nucleotides can be partitioned into at most 3 groups, each containing vertices close to each other on the backbone. The other type of clique models $4 \sim 5$ sporadic nucleotides as a part of a tertiary motif connecting 3 or 4 small regions of the backbone. Both type of 5-cliques are thus of number $O(n^3)$ in total, potentially leading to an $O(n^3)$-time (and space) implementation of the dynamic programming algorithm with the backbone 4-tree model for nucleotide interaction prediction.

The implemented neural networks for interaction pattern scoring were designed and trained exclusively for the constraint backbone 3-tree model. Thus to implement the backbone k-tree model with a higher k, an updated knowledge-based scoring system is needed. Using the same idea of the neural networks, due to the increased number of features, a feature selection scheme will need to be developed. Then the neural networks have to be retrained.

Chapter 5

RNA 3D Structure Prediction with Backbone k-Tree Model

Given the significant functional roles played by non-coding RNAs (ncRNAs), it is highly desirable to develop software tools that can accurately predict RNA 3D structure [Laing and Schlick, 2010; Leontis and Westhof, 2012a], especially from a single RNA sequence. There have been a number of tools developed for this purpose [Das and Baker, 2007; Sharma et al., 2008; Ding et al., 2008; Jonikas et al., 2009; Parisien and Major, 2002; Bida and Maher, 2012; Popenda et al., 2012; Reinharz et al., 2013]. While the methods vary, most of the front-runners are based on *de novo* assembly of 3D fragments into a full structure. Due to the immense space of 3D conformations, fragment assembly based methods are often guided by a predicted (or known) secondary structure and/or enhanced with sampling techniques.

In particular, Rosetta [Das and Baker, 2007; Das et al., 2010] *de novo* assembles 3D structure from a database of local 3-nt tertiary fragments. MC-Fold/MC-Sym [Parisien and Major, 2002] assumes the 3D structure consists of 4-nt cyclic tertiary motifs. Such enclosed loops are often parts of helices and junctions in the predicted secondary structure and may not include other global, more distant interactions that are critical to the overall conformation. The recent survey [Laing and Schlick, 2010] shows that such a method failed to yield meaningful 3D models for more than half of the RNAs in a benchmark of 43 RNAs, most of which contain 80 or fewer nucleotides. The underperformance highlights the limit of the existing methods and the inherent difficulty in computing native-like conformations from the immense 3D conformation space, even with the help of sampling techniques or secondary structure constraints.

We have developed a non-conventional structure prediction method which proceeds in two stages. First we predict the nucleotide interactions, then second, from these, we predict the full 3D structure. The nucleotide interactions are predicted using the backbone k-tree method [Ding

et al., 2015] described in Chapter 4. In the backbone k-tree graph model, nucleotides are organized into groups of size $k+1$ (e.g., quadruplets for $k=3$), such that interactions are permitted only for nucleotides belonging to the same group and groups are connected to each other with a tree topology. This approach was motivated by our recent discovery [Ding et al., 2014a] that the nucleotide interaction relationship graphs for RNA chains whose structure is known are overwhelmingly of small tree width. The model considers all nucleotide interactions from the established geometric nomenclatures and families [Leontis and Westhof, 2001; Leontis et al., 2002; Stombaugh et al., 2009], including base-base, base-phosphate, base-ribose [Zirbel et al., 2009; Zirbel, 2011], base-stacking interactions as well as the phosphodiester bonds between two neighbouring nucleotides on the backbone.

Then upon the predicted backbone k-tree model, an optimal 3D model is computed by global substantiation of all the groups $k+1$ nucleotides with corresponding 3D motifs. These annotated atom-grain motifs for interaction patterns were extracted from resolved 3D structures of RNAs in the RNA Structure Atlas [Sarver et al., 2008]. In this chapter, we will present in detail the method and algorithm for 3D modeling with the backbone k-tree model, once the nucleotide interactions have been successfully predicted.

5.1 Model and Methods

Our method to predict an RNA 3D structure consists of two major steps (see Figure A.2 in Appendix A). First, it predicts an optimal backbone k-tree model $\langle G^*; A^* \rangle$ such that

$$\langle G^*; A^* \rangle = \arg\max_{\langle G;A \rangle} \{ \sum_{q \in G} f(q, A(q), S) \} \tag{5.1}$$

in which A^* assigns a set of consistent ips, one for each clique in G^*. Section **??** will summarize how to define scoring function $f(q, A(q), S)$ and how to optimize the sum in equation (5.1) through dynamic programming. The detailed description of the method for nucleotide interaction prediction has appeared in Chapter 4 and our publication [Ding et al., 2015].

Second, based on the predicted backbone k-tree model $\langle G^*; A^* \rangle$, the method computes an

optimal 3D modeling

$$g^* = \arg\min_{g} \sum_{p,q \in G^*, \mathcal{N}(p,q)} \gamma(g(A^*(p)), g(A^*(q))) \tag{5.2}$$

where g assigns a geometric motif $g(A^*(q))$, called an *ip-motif*, to the interaction pattern $A^*(q)$ for clique q and predicate $\mathcal{N}(p,q)$ asserts that p and q are two cliques neighboring in graph G^* (i.e., their corresponding nodes are neighboring in the tree topology).

Function γ quantifies the how well geometric motifs for interaction patterns of neighboring cliques fit to each other. In Section 3 we will present the details for function δ and how to calculate it. We will also show an algorithm to optimize the computation in equation (5.2) in linear time.

Our 3D modeling method is based on a predicted backbone 3-tree model (where $k = 3$ was chosen in our implementation). Our method produces a most plausible set of geometric motifs for the predicted interaction patterns and assemble the motifs into an optimal 3D model. In this section, we describe the details of the 3D modeling method.

5.1.1 Geometric Motif Alignment

Given an interaction pattern (ip) for a quadruplet of nucleotides, a large number of ip-motifs can be associated with the ip. In our development, we extracted ip-motifs from the known RNA 3D structures through the use of software FR3D [Sarver et al., 2008]. Considering the number of ip-motifs for every ip is usually large and the geometrical redundancy between these candidates, we used an additional cluster step to group the geometry ip-motifs. In particular, we applied an all-to-all geometry alignment to all candidates for each ip. The 0.5Å RMSD (root-mean-square-deviation) cutoff value was adopted in the alignment, which means the RMSD values is not larger than 0.5Å for every two geometry candidates in each group after clustering. Then we selected one ip-motif candidate to represent each group. All ip-motif representatives were built into NIPGDB, namely the nucleotide interaction pattern geometry database. Figure 5.2 shows two representative motifs for the two ips in Figure 5.1, respectively.

The function $\gamma(g(A^*(p)), g(A^*(q)))$ in equation (5.2) measures the geometric consistency between the two ip-motifs $g(A^*(p))$ and $g(A^*(q))$. The measurement was also realized by comput-

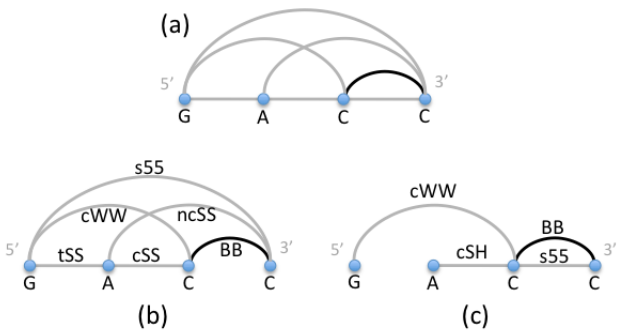


Figure 5.1: (a) A 4-clique formed by quadruplet of nucleotides {G, A, C, C}, with gray edges to indicate that interactions between every pair of nucleotides are possible, where the dark edge indicates the known phosphodiester bond between the two cytosines neighboring on the backbone; (b) and (c) two different interaction patterns (ips, found from the known 3D structures) that can be assigned to the clique in (a), where BB represents the phosphodiester bond, s55 and s35 are 5'-5' and 3'-5'stacking, respectively, the other interactions are all base-base, defined by the edges of the triangle base model [Leontis et al., 2002]) participating in the interactions.

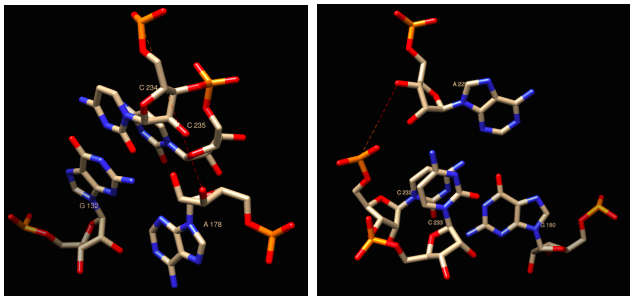


Figure 5.2: Two geometric motifs (ip-motis) (left and right) for the two interaction patterns (ips) shown in Figure 5.1 (b) and (c), respectively. These two ip-motif formed by quadruplet $\{\mathtt{G}_{132}, \mathtt{A}_{178}, \mathtt{C}_{234}, \mathtt{C}_{235}\}$ and quadruplet $\{\mathtt{G}_{180}, \mathtt{A}_{229}, \mathtt{C}_{232}, \mathtt{C}_{233}\}$, respectively. Both are found in the known structure of RNA 1NBS with 155 nts (the indexes starts from 86) from PDB; images were rendered with UCSF Chimera.

ing the minimum RMSD value through a rigid superimposition on atoms common to both ip-motifs. In particular, let $x \subseteq p \cap q$ be the 3 common nucleotides shared by cliques p and q. Let $g(A^*(p))|_x \subseteq g(A^*(p))$ and $g(A^*(q))|_x \subseteq g(A^*(p))$ be the two respective subsets of 3D coordinates of atoms that belong to the 3 nucleotides in x. The superimposition between these two subsets is done by a transformation $\mathbf{T}$ that transforms $g(A^*(p))|_x$ to $g(A^*(p))|_x^{\mathbf{T}}$ in the coordinate system of

$g(A^*(q))|_x$. Therefore, we defined

$$\gamma(g(A^*(p)), g(A^*(q))) = rmsd(g(A^*(p))|_x^{\mathbf{T}}, g(A^*(q))|_x)$$

where function *rmsd* calculates the minimum value of RMSD for the superimposition. The alignment function *rmsd* used in this work is imported from Bio.PDB package (`biopython.org/DIST/docs/api/Bio.PDB-module.html`), i.e., the *Superimposer* object, which was developed based on the method of singular value decomposition (SVD).

In equation (5.2), the optimal 3D modeling is to minimize overall RMSD by identifying mutually consistent ip-motifs, one for every ip in the predicted backbone 3-tree model. Because there may be more than one ip-motif candidate for every ip, a naive examination of all ip-motif combinations for all the ips would result in infeasible algorithms. Our work took advantage of the backbone 3-tree topology and yielded a non-trivial linear time algorithm for optimal 3D modeling.

5.2 Optimization Algorithm

We assume that the tree topology is a rooted tree for the predicted backbone 3-tree graph G^*. Each node in the tree corresponds uniquely to a 4-clique in the graph (see Figure **??**); we will use words "tree node" and "clique" interchangeably in this chapter. Let q be a 4-clique in G^*. We define $\mathcal{C}(q)$ be the set of 4-cliques that are children of q in the tree topology. Then equation (5.2) is equivalent to

$$g^* = \arg\min_g \sum_{q \in G^*} \sum_{p \in \mathcal{C}(q)} \gamma(g(A^*(p)), g(A^*(q))) \tag{5.3}$$

To describe a dynamic programming algorithm for the optimization computation desired by equation (5.2), we need to introduce additional notions. We assume $\langle G^*; A^* \rangle$ to be the predicted backbone 3-tree model and g to be a choice of ip-motifs for all interaction patterns in the graph model.

Definition 5.2.1. Let q be a 4-clique in G^*. The *aggregate motif* for ip $A^*(q)$ based on g is defined

recursively as

$$\mathcal{G}(A^*(q)) = g(A^*(q)) \cup \bigcup_{p \in \mathcal{C}(q),\, x \subseteq q \cap p} \mathcal{G}(A^*(p))|_{\overline{x}}^{\mathbf{T}_p}$$

where $\mathbf{T}_p$ represents the transformation determined by ip motifs $g(A^*(p))$ and $g(A^*(q))$ and $\mathcal{G}(A^*(p))|_{\overline{x}}^{\mathbf{T}_p}$ is the transformed aggregate motif for ip $A^*(p)$ which does not contain the atoms of nucleotides in set $x = p \cap q$.

Note that Definition 5.2.1 includes the base case that the aggregate motif for $A^*(q)$ is simply $g(A^*(q))$ when q is a leaf (i.e., $\mathcal{C}(q) = \emptyset$) in the tree topology. Not only does the definition introduce an explicit way to assemble a 3D model based on the choice g of ip-motifs, but also it suggests a dynamic programming approach to identify a choice g^* that minimizes the objective function.

Let $M(q, A^*)$ to be minimum overall RMSD computed over all the ips in the subtree topology rooted at node (clique) q. Then we obtained the following recurrence:

$$M(q, A^*) = \min_g \sum_{p \in \mathcal{C}(q)} \gamma(g(A^*(p)), g(A^*(q))) + M(p, A^*) \tag{5.4}$$

with base case: $M(q, A^*) = 0$ for leaf q.

Recurrence (4) with the base case has allowed us to develop a dynamic programing algorithm to compute function $M(q, A^*)$. For every clique q, the computation of $M(q, A^*)$ yields an aggregate motif as defined by Definition 5.2.1. The algorithm computes $M(q, A^*)$ in a bottom-up fashion, from leaf cliques to the root. The ultimate value of the optimization function is $M(r, A^*)$, where r is the root of the tree topology, with an identified set g^* of ip-motifs for the ips in the given (predicted) backbone 3-tree model $\langle G^*; A^* \rangle$. The optimal 3D model is $\mathcal{G}(A(r))$ obtained based on g^*.

As the number of cliques in the backbone 3-tree is proportional to the number n of nucleotides, the algorithm runs in time $O(ntm)$ where t is the time needed to transform a set of $O(n)$ atom coordinates, and m is the maximum number of ip-motifs for every ip.

5.3 Performance Evaluation

5.3.1 Implementation

The 3D modeling stage of the pipeline was implemented in python into a program called BkTree3D. We ran the evaluation tests on a Red Hat 4.8.2-7 server with 4 Intel Quad core X5550 Xeon Processors, 2.66GHz 8M Cache and 70GB Memory.

5.3.2 Test Data

Our 3D structure prediction pipeline was evaluated with the same benchmark of 43 high-resolution (3.4Å or higher) RNAs used in the survey [Laing and Schlick, 2010]. These RNAs, of which 18 are with length $\geq$ 50 nucleotides, contain diverse structure complexities including hairpins, internal loops, pseudoknots and multi-way junctions. The prediction results from our recent works [Ding et al., 2015], including both predicted interactions tested with the same benchmark of RNAs and the output 3-trees, were used directly as inputs to our 3D modeling programs. Note that for the consistency purpose, we leave out an RNA (PDB ID: 2F8K) due to its recent updates in the PDB database.

5.3.3 Overall Performance

We evaluated the quality of the predicted 3D structures with the RMSD values calculated against their native structures. Since the RMSD does not account for local deviations, base-pairing and base-stacking accuracies, we also calculated the Deformation Index, a measure that accounts for both RMSD and Matthews correlation coefficient (MCC $:= \sqrt{\text{PPV} \times \text{STY}}$), the quotient between them, where STY and PPV stand for sensitivity and positive predictive value respectively.

We conducted 5 types of independent tests on the benchmark set to show the effectiveness of our 3D modeling method. In the first test, the predicted interactions and backbone 3-trees from our recent results [Ding et al., 2015] were used as inputs to BkTree3D. Figure A.9 in Appendix A summarizes the overall performance. For small sequences ($\leq$ 50 nucleotides) with simpler struc-

tures (hairpins and internal loops), BkTree3D can predict accurately the 3D structures with the average RMSD 1.94 Å. Noticeably, BkTree3D also produced decent results for the 3 sequences with pseudoknots. In terms of long sequences ($>$ 50 nucleotides), our method shows great potentials to predict majority of the structures. In particular, 11 out of 18 sequences achieved RMSD values below 10 Å.

By replacing the inputs from the predicted interactions with the resolved interactions in RNA 3D Atlas [Sarver et al., 2008] but using the same predicted backbone 3-trees, we performed the second test for the purpose of evaluating our 3D modeling algorithm. The results, summarized in Figure A.10 in Appendix A, show that the majority of the predicted structures have lower RMSD values. In particular, 35 out of 42 RNAs achieved RMSD values below 5 Å. We can see from Figure 5.4 the decreasing of the average RMSD value as compared to the first test, indicating that accurate predictions of interactions are important to the 3D modeling stage.

In the third test, we applied a pre-processing step described in Section A.1 in Appendix A on the predicted interactions before executing BkTree3D. Figure 5.4 shows that the preprocessing steps permitted us to fix some of the predicted interactions and hence improved the results, where the average RMSD values over all RNAs in the benchmark set dropped from 5.09 Å to 4.85 Å.

In the fourth test, we removed from the NIPGDB all ip-motifs obtained from the RNA that is being predicted. The results are listed in Figure A.10 in Appendix A. As compared with the results of the third test in Figure 5.4, only minor increasing of the RMSD values can be noticed.

BkTree3D is capable of generating multiple suboptimal solutions. In the last test, 5 structures were generated for each of the RNA in the benchmark set. The results are listed in Figure A.11 in Appendix A. Compared with the results with other 4 tests in Figure 5.4, the fifth test produces a slightly higher average RMSD value.

5.3.4 Performance Comparison with Other Methods

We compare our 3D modeling program with the state-of-the-art structure prediction methods including MC, Rosetta and a recently developed program called RNA-MoIP [Reinharz et al., 2013] that preformed well on the long sequences with more than 50 nucleotides. We provide a compari-

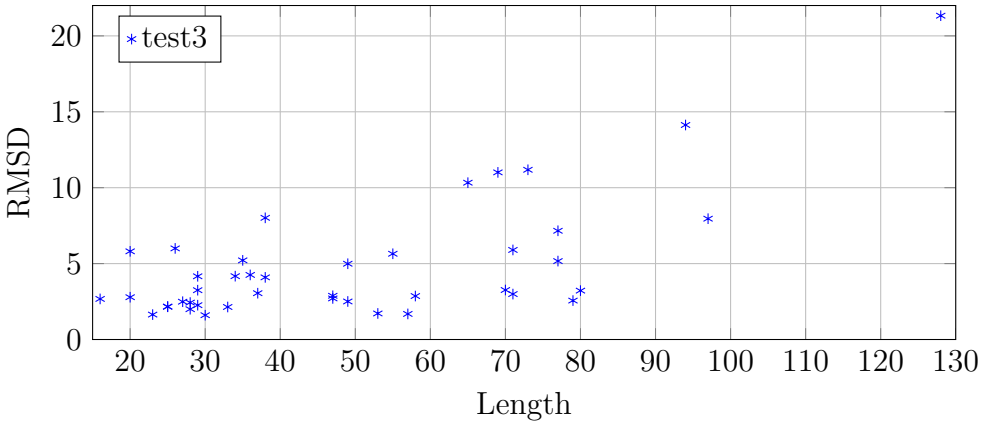


Figure 5.3: RMSD values generated by BkTree3D in the third test for the benchmark set of 42 RNAs sorted by their lengths.

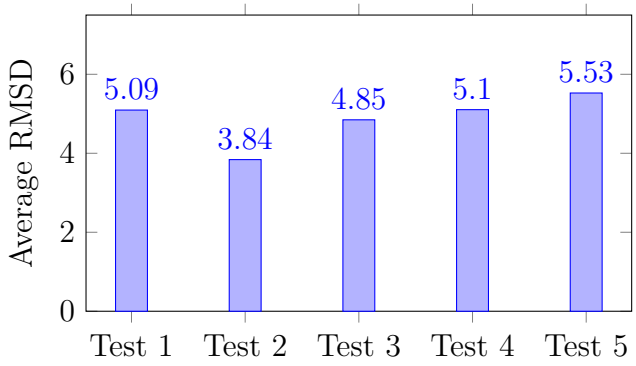


Figure 5.4: Comparison of average RMSD values of 5 tests on the benchmark set of 42 RNAs. In test5, the average RMSD of the 5 structures produced by BkTree3D is calculated first for every RNA. Then the overall average is calculated using the averages of 42 RNAs.

son of the alignment between the predicted and native structures using the RMSD and DI metrics. The comparison with MC and Rosetta is based on 4 representative structures chosen in [Laing and Schlick, 2010] which typically contain two hairpins and two junctions. Since both MC and Rosetta allow the prediction of multiple structures, we chose to use the best and the average values of their solutions. Figure A.12 in Appendix A summarizes the comparisons with MC and Rosetta. For every one of the 4 RNAs, the RMSD achieved by BkTree3D is significantly smaller than the best values achieved by MC and Rosetta.

In Table 5.1, BkTree3D is compared with RNA-MoIP on 9 large RNAs that were used as the benchmarks in evaluating RNA-MoIP, where their results were reported in [Reinharz et al., 2013].

As RNA-MoIP has not been designed to predict pseudoknot, this set of RNAs is pseudoknot-free and 8 of them have a 3-way junction with the other a 4-way junction. This is not an ideal comparison as we have input to BkTree3D program the known canonical Watson-Crick base pairs instead of the predicted secondary structure. The overall quality of the predicted structures is comparable to the structures by RNA-MoIP. In particular, for 5 out of the 9 RNAs, our method achieved an RMSD value much smaller than the minimum RMSD achieved by RNA-MoIP. On the tRNA 2DU3, our RMSD is 2.993Å, very close to the average RMSD 2.91Å achieved by RNA-MoIP. For a slightly longer RNA sequence 1LNG, our RMSD is slightly larger than the average RMSD yielded by RNA-MoIP. Only on the longest sequence 1MFQ, our method performed worse than RNA-MoIP.

Table 5.1: RMSD comparison between BkTree3D and RNA-MoIP. Boldface indicates the smaller RMSD value of each RNA. "-" indicates the RNA that RNA-MoIP failed on.

	Length	RNA-MoIP			BkTree3D
		Min	Avg	SD	
3E5C	53	-	-	-	**1.716**
1DK1	57	2.95	4.76	0.99	**1.686**
1MMS	58	5.66	7.65	0.86	**2.864**
2DU3	71	2.23	**2.91**	0.44	2.993
3D2G	77	5.34	7.35	1.34	**7.162**
2HOJ	79	3.19	7.19	2.31	**2.564**
2GDI	80	-	-	-	**3.22**
1LNG	97	2.73	**6.30**	1.91	7.959
1MFQ	128	9.07	**14.34**	5.01	21.33

5.4 Discussion and Conclusion

We have presented a non-conventional framework to predict RNA all-atom 3D structures by a two-step process: nucleotide interaction prediction and 3D modeling. Both steps were built upon the backbone k-tree model, which makes it possible to drastically reduce the space of 3D conformations and permit efficient exact algorithms to calculate the optimal 3D structures. The evaluating results have shown that utilizing the accurately predicted interactions from our recent results and coupling a simple score scheme can yield predictions that outperform the state-of-the-art methods.

The framework of two-step process offers operability and flexibility; in particular, the predicted interaction interactions from the first step provide a scaffold to the 3D structure to be modeled, enabling diagnostics and treatment before the second step for 3D modeling. Therefore, the interaction prediction can be critical to the accuracy of the 3D modeling. In particular, a single missing long-distance interaction may cause a major conformation change in the predicted structure. For example, as shown in Figure 5.5 a where the predicted 3D structure (blue) of a pseudoknot RNA is superimposed to its native structure (green), the predicted 3D structure has two widely opened arms due to a missing kiss-hairpin interaction. The gap can be narrowed down when the long-distance interaction was added back to the predicted interactions (Figure 5.5b). Alternatively, a stacking (s55) interaction inserted to both sides of the helix forming the hairpin loop forced both sides of the helix twist closer to each other, leading to a smaller gap as well (Figure 5.5). Based on this observation, we have introduced a pre-processing step that establishes a serial of knowledge-based rules to modify the predicted interactions before the 3D modeling step. The pre-processing step includes removing or replacing certain stacking interactions to achieve a more native-like backbone spin. The proposed rules help to improve the result of 3D prediction in average (see Test 3 in Figure 5.4). The details of the rules are discussed in the Appendix A. However, a potentially more accurate method for nucleotide interaction prediction is through the use of backbone k-tree models, for larger k (e.g., $k = 4$). The graph model of larger tree width can be robust enough to account for all interactions in sophisticated structures such as kissing hairpins and other types of pseudoknots.

We would also like to note that the performance of the 3D modeling method for the longer sequences (over 100 nucleotides) is relatively lower than that for shorter sequences. Besides the choice of a larger parameter k, another future improvement of the prediction quality is to enrich and refine the geometric motif database NIPGDB. In particular, the resolved RNAs that have been selected for building the geometry motif database are currently limited to the length ≤ 100. More resolved RNAs can be used to improve the effectiveness of NIPGDB. In addition, the database works well in providing motifs candidates to "dense" interaction patterns where every nucleotide has at least one interaction with at least one other nucleotide in the corresponding quadruplet. However, our test results show that the prediction performance may drop when many quadruplets are instantiated with less "dense" interaction patterns. Our most recent study reveals that such

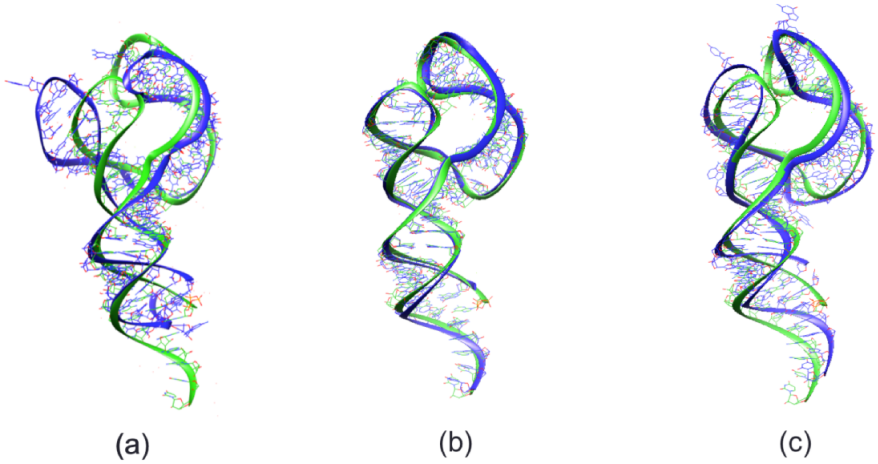


Figure 5.5: Comparison of predicted structures (blue) for 2QUS (69 nucleotides, pseudoknot) superimposed to its native structure (green) using three different sets of interactions: (a) predicted interactions from the first step of our pipeline (11.01Å); (b) resolved interactions from RNA 3D Atlas (3.001Å); (c) predicted interactions with additional stacking (s55) inserted to both sides of the helix forming the hairpin loop (5.58Å).

an issue can be addressed with smaller motifs extracted from the known structures with more non-trivial calculations in the dynamic programming algorithm for 3D modeling. =

Chapter 6

Bayesian Network Learning with k-tree Model

Bayesian network is a probabilistic model that represents random variables and their dependencies with a directed acyclic graph. The most challenging task for Bayesian network learning from data for complex domains is to optimize joint probability distribution functions for the random variables. Because such the optimization is computational intractable [Chickering et al., 1994] and the complexity remains even the network has bounded treewidth [Dagum and Luby, 1993]. It is well known that the complexity of exact inference in a Bayesian network is associated with the treewidth of the network [J. H. P. Kwisthout and van der Gaag, 2010]. Therefore, learning Bayesian network with bounded treewidth has gained increasing attention recently due to its existence of efficient inference. However, algorithms for Bayesian learning have resort to heuristics and approximation methods [Daly and Aitken, 2011; Suzuki, 1999; Yuan and Malone, 2013]. Such algorithms cannot guarantee the learning accuracy even with assumptions of tree-like topologies [Dasgupta, 1999]. As a result, their performances vary significantly for different applications.

Our aim is to address the under performance issue in Bayesian network learning with the notion of k-tree and its potentially efficient algorithms with meaningful constraints on the k-tree. Our preliminary investigations (see Figure 6.1) shows that k-tree graph can ideally model the topology of real-world complex systems. With the k-tree modeling, the Bayesian network learning can be formulated as an optimization problem with objective function derived from the data. We develop efficient algorithms by discovering and utilizing meaningful constraints from the data. Actually, effective algorithms have been developed by forcing the Bayesian network to satisfy certain conditions. For example, a simple yet effective algorithm is designed [Teyssier and Koller, 2012] based on the observation that, given an ordering on the variables in the network, finding the optimal Bayesian network is not NP-hard. In fact, this ordering constraint is equivalent to the backbone

condition that has been adopted in developing the backbone k-tree model. We give the ordering of the variables a simple name as backbone. Because the model is initially developed with the motivations from the bio-molecular structure prediction problems, where a sequence of the bio-molecules is known. However, for most of the real world application, determining an appropriate ordering is itself a difficult problem. In this chapter, we explore some more general and practical constraints that can be learning efficiently from the data. With the constraints, we are able to develop efficient dynamic programming algorithms that compute the optimal solution based on a score function.

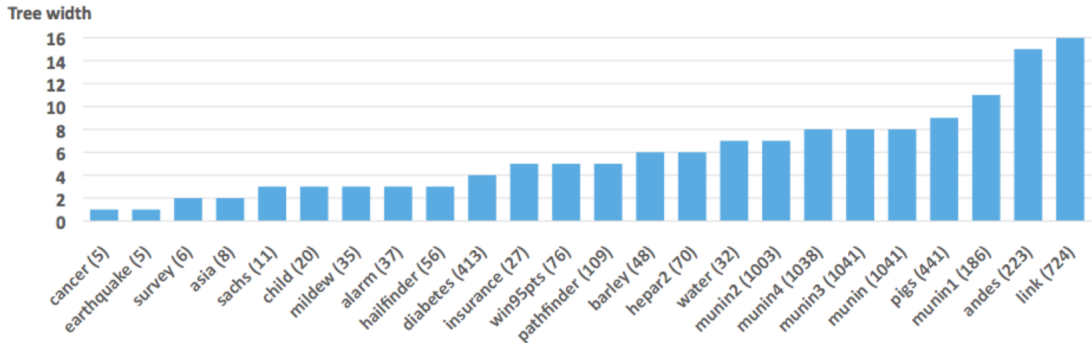


Figure 6.1: Tree width upper bounds of 24 Bayesian networks of a wide variety of types, obtained from Bayesian data repository www.bnlearn.com [Nagarajan et al., 2013]. The number beside the name of each network is the total number of nodes in the network. The tree width upper bounds were calculated with an approximation algorithm software tool [Bodlaender and Koster, 2010]; the real tree widths are likely to be much smaller.

We plan to apply the effective Bayesian network learning methods, specifically k-tree model with spanning tree constraint, to analyze the relations of Fenton reaction and cancer initialization and growth. The analysis will be based on high throughput transcriptomics data collected from real cancer tissue samples. Given a dataset, the objective is find Bayesian networks that include the spanning tree to optimize a score function. As initial steps, we construct a Chow-Liu tree [Chow and Liu, 1968] for each of the data set. By comparing the correlations that are included in the Chow-Liu tree with the entire correlations, we find that some important correlations are not included in the Chow-Liu tree due to the limitations of spanning tree, which indicates the necessity of using an enhanced model, e.g., k-tree model to learn the networks. Our final goal is to build a detailed causal network of genes related to cancer initiation and growth, which clearly indicates the causal genes of the cancer-related activities.

6.1 The Bayesian k-tree Model

For a set of random variables $X = X_1, X_2, \cdots, X_n$ of a complex system, out task is to identify an "optimal" directed acyclic graph (DAG) G, whose nodes represent the random variables and edges characterize the dependencies of the variables. Each edge of the DAG encode a parent-child relationship $X_i|X_{pa_i}$, where X_{pa_i} is the parent set of variable X_i. The dependencies of the random variables must satisfy Markov property, i.e., each X_i is independent of its non-descendants given its parents X_{pa_i}. A Bayesian network defines a unique joint probability distribution over X as

$$P(X_1, \cdots, X_n) = \prod_{i=1}^{n} P(X_i|X_{pa_i}), \tag{6.1}$$

where $P(X_i|X_{pa_i})$ is a conditional probability distribution of X_i given its parents in G.

A k-tree can be represented with the vertices in the order in which they are created. A *creation order* of a k-tree G of n vertices is defined as

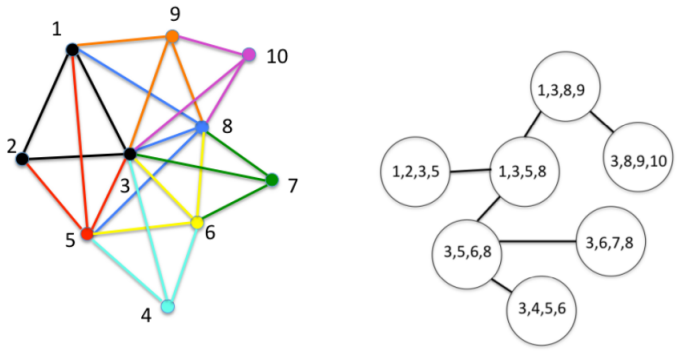
$$O_G = C_{k+1}x_{k+1}C_{k+2}x_{k+2}\cdots C_n x_n, \tag{6.2}$$

where C_i is k-clique in G, and $C_i \subseteq \{x_{k+1}, \cdots, x_{i-1}\} \cup_{j<i} C_j$, for all $i = k+1, \cdots, n$, such that C_{k+1} is created in step 1 of Definition 2.1.1 and x_i is created by connecting to C_i in step 2 of Definition 2.1.1. Figure 6.2 shows an example of 3-tree containing 10 vertices, its tree decomposition representation, and a creation order of the 3-tree example.

Definition 6.1.1. Let $k \geq 1$ be an integer. A Bayesian k-tree is a Bayesian network over n random variables $X_1, \cdots, X_n$ that has a k-tree of n vertices to represent the dependencies of the random variables.

Proposition 6.1.1. Let $X = x_1, \cdots, x_n$ be random variables of a Bayesian k-tree G, with the creation order $O_G = C_{k+1}x_{k+1}C_{k+2}x_{k+2}\cdots C_n x_n$. Then the joint probability distribution function of the corresponding Baysian network can be computed as

$$Q(X) = Q(x_1, \cdots, x_n) = P(x_1, \cdots, x_k) \prod_{i=k+1}^{n} P(x_i|C_i) \tag{6.3}$$



{1,2,3} 5 {1,3,5} 8 {1,3,8} 9 {3,8,9} 10 {3,5,8} 6 {3,5,6} 4 {3,6,8} 7

Figure 6.2: Left: A 3-tree of 10 vertices; Right: a tree decomposition representation of the 3-tree; Bottom: a creation order for the 3-tree example.

The proposition holds because $Q(x_1, \cdots, x_n) = P(x_1, \cdots, x_n|O_G)$ and $P(x_1, \cdots, x_n|O_G) = P(x_n|x_1, \cdots, x_{n-1}, O_G)P(x_1, \cdots, x_{n-1}|O_G)$. The last equality holds for the reason that x_n shares edges only with vertices in C_n. Solving the recurrence gives equation.

6.1.1 Optimal Learning of Bayesian k-tree

Let Q be the Bayesian k-tree-constrained model. Then it can be considered an approximation of the unconstrained Bayesian model P. The approximation can be measured using the Kullback-Leibler divergence between two models [Kullback and Leibler, 1951; Lewis, 1959]

$$I(P, Q) = \sum_x P(X) \log \frac{P(X)}{Q(X)} \geq 0 \tag{6.4}$$

where the last equality holds if and only if $P(x) = Q(x)$. The problem learning the optimal Markov k-tree is the task to find model Q that minimizes the divergence.

Definition 6.1.2. The mutual information $I(x_i, x_j)$ between two variables x_i and x_j is given by

$$I(x_i, x_j) = \sum_{x_i, x_j} P(x_i, x_j) \log(\frac{P(x_i, x_j)}{P(x_i)P(x_j)}). \tag{6.5}$$

It is well-known that $I(x_i, x_j)$ is non-negative. And from the above definition, it is easy to

derive $I(x_i, C_i) = \sum_{x_i, C_i} P(x_i, C_i) \log(\frac{P(x_i, C_i)}{P(x_i)P(C_i)})$

Proposition 6.1.2. The maximization of $\sum_{i=k+1}^{n} I(x_i, C_i)$ yields an optimal Bayesian k-tree.

The proof the above is very similar to the proof of tree dependence is an optimum approximation in [Chow and Liu, 1968].

6.1.2 Tree k-tree Model

In our previous investigations of using backbone k-tree model to predict RNA 3D structures, with the natural order of the nucleotides of a RNA sequence, we can assume the vertices of the input graph G are labeled with integers $\{1, 2, \cdots, n\}$. The backbone k-tree model requires that all the backbone edges $\{(i, i+1) : 1 \leq i < n\}$ are contained in both G and the objective spanning k-tree, which makes it possible to compute the constrained spanning k-tree efficiently.

Definition 6.1.3. A *condition* for problem MSkT is a designated subgraph that are contained by both the input graph and the output spanning k-tree.

Proposition 6.1.3. Under the backbone condition, the problem MSkT can be solved in time $O(n^{k+1})$, for every fixed $k \geq 1$.

The above proposition has been justified in Chapter 2. Although the backbone condition makes the developing of efficient algorithms possible, for most of the real world problems, determining an appropriate ordering is itself a difficult problem. A natural extension of the backbone is a condition with a constant number of backbones. Actually, it is not hard to show that if a constant number of backbones are forced to be in both the input graph G and the output spanning k-tree, the problem MSkT can be solved in time $O(n^{k+1})$, for every fixed $k \geq 1$. We are more interested in the condition of a spanning tree, i.e., a spanning tree T is required to be included in both the input graph and the spanning k-tree. An important advantage of the spanning tree condition is that a meaningful spanning tree can be achieved efficiently using the famous greedy algorithms [Chow and Liu, 1968]. Moreover, the spanning tree condition can be easily extended to spanning forest condition. Generally,

Definition 6.1.4. Let G be an arbitrary k-tree. A condition satisfied by G is called a *genuine condition* if every $(k+1)$-clique in G separates the condition (a subgraph) into more than one connected nonempty component.

Proposition 6.1.4. Under a genuine condition, the problem MSkT can be solved in time $O(n^{k+1})$ for every fixed $k \geq 1$.

For a spanning tree to be a genuine condition, the spanning tree must be split into more than one connected component by an arbitrary k-clique. Therefore, the spanning tree cannot have degree more than k. It is relatively simple to achieve a spanning tree from the application data. But computing a spanning tree with bounded degree is NP-hard (can be reduced to Hamiltonian Path problem easily). Therefore, heuristic algorithms may be needed to construct a spanning tree with bounded degree. In the following section, we have applied the algorithm in [Chow and Liu, 1968] to build a spanning tree from analysis of gene expression data for cancer research. As we found that, the spanning tree for this real world application has small degree. And the degree can be further reduced with the prior knowledge of the gene correlations.

6.2 Analysis of Gene Networks for Cancer Research

In the section, we apply the proposed methods on Bayesian k-tree learning for the studying of cancer development. We work with researchers in the Computational Systems Biology Laboratory (CSBL) at the University of Georgia on this project. Fenton reaction has long been believed associated to the initiation and progression of multiple cancer types due to the production of highly oncogenic metabolite named hydroxyl radical [Torti and Torti, 2013; Toyokuni, 2009]. However, there is lack a detailed mechanism that how the Fenton reaction causes cancer. The preliminary studies by CSBL raise a hypothesis that the oxidative stress may serve as a direct pressure that drives the cell proliferation and cause the formation of cancer [Xu et al., 2014]. To the best of our knowledge, there is no effective biological marker can be used to measure Fenton reaction in cancer patients, and the highly micro-environment-dependent Fenton reaction cannot be reflected by cell or animal

based models, which cause the difficulty to study this problem through experiment based methods.

$$Fe^{2+} + H_2O_2 \rightarrow Fe^{3+} + HO\cdot + OH^- \tag{6.6}$$

$$Fe^{3+} + H_2O_2 \rightarrow Fe^{2+} + HOO\cdot + H^+ \tag{6.7}$$

In this study, we aim to test the hypothesis by building a causal model between the Fenton reaction related genes and cell proliferation and cancer associated genes by using high throughput transcriptomics data collected from real cancer tissue samples. Our planned investigations will be conducted on 15-20 types of sporadic cancers that have RNA-Seq data collected from more than 100 samples and available in TCGA database [Tomczak et al., 2015], including bladder urothelial carcinoma, breast invasive carcinoma, cervical squamous cell carcinoma, endo-cervical adenocarcinoma, colon adenocarcinoma, esophageal carcinoma, glioblastoma multiforme, head and neck squamous cell carcinoma, kidney renal clear cell carcinoma, kidney renal papillary cell carcinoma, liver hepatocellular carcinoma, lung adenocarcinoma, lung squamous cell carcinoma, ovarian serous cystadenocarcinoma, pancreatic adenocarcinoma, prostate adenocarcinoma, rectum adenocarcinoma, skin cutaneous melanoma, stomach adenocarcinoma, and thyroid carcinoma. The RNA-Seq data measures the gene expression level of around 20,000 genes encoded in human genome in each collected sample that reflect the biological activities associated to the function of each gene.

The preliminary studies by CSBL also suggest that the Fenton reaction can be quantified by using the gene expression level of 12 iron-sulfur clustering metabolism, 19 oxidative stress and 65 protein damage response genes, by which they have proved the Fenton reaction generally exists in all of the selected cancer types [Xu et al., 2014]. We intend to construct causal models by linking the Fenton reaction related genes against 20 cell proliferation marker genes and about 500 cancer associated genes in each cancer type. Out final goal is to build a detailed causal model that explain which cell proliferation or cancer associated genes are directly influenced by Fenton reaction in each selected cancer type. The result cannot only increase our basic understanding of the roles of Fenton reaction in cancer formation and progression but also can be applied in identification of biomarkers or drug targets for anti-oxidant (hydroxyl radical) therapies in cancer treatment [Fuchs-Tarlovsky, 2013].

6.2.1 Data Set

The data set, provided by the Computational Systems Biology Laboratory at the University of Georgia, contains gene expression level of 12 iron-sulfur clustering metabolism, 19 oxidative stress, 65 protein damage response genes, 20 cell proliferation marker genes and about 400 cancer associated genes. The data set is filtered, normalized and discretized before being used to produce the results in SectionpreResultsCancer,

6.2.2 Preliminary Results

We construct a spanning tree (see Figure 6.3) using kruskal's algorithm with edge weight defined as mutual information of two random variables [Chow and Liu, 1968]. Each node of the spanning tree represents a gene that is colored by the pathway it belongs to. There are 6 different pathways: Iron related genes, Oxidative stress, Nucleotide synthesis, Glycosaminoglycan, Lipid peroxidation, Collagen. Though the learned tree structure is simple, it shows some important correlations that are consistent to the our observations and hypothesis. For example, oxidative stress responsive gene is a reactant of Fenton reaction. So they have strong correlations that can be recognized in the spanning tree (green nodes and cyan nodes). Iron related genes (Iron containing proteins and synthesas) is also a key reactant of Fenton reaction. Glycosaminoglycan, one kind of extracellular component, is highly likely to be involved in Fenton reaction related biological process. Nucleotide synthesis, one out come of Fenton reaction, is hoped to be driven by Fenton reaction. Lipid peroxidation is unknown if is driven by Fenton reaction. Collagen, one kind of extracellular component, may not be involved in Fenton reaction related biological process very much. It can be observed in the spanning tree that the collagen genes are relatively independent with other genes.

However, by comparing the heat maps of all correlations and the correlations that are captured in the learned spanning tree (see Figure 6.4 and Figure 6.5). We identify some meaningful correlation regions that are included in the learned spanning tree. This may indicate a requirement of a more powerful model, such as k-tree model, to capture all important collations.

We also notice that the largest degree of the learned spanning tree is 16. Though there are only a very small number of nodes that have degrees close to 16, it may still be too large to use

tree k-tree model. We hope that we could analyze the learning spanning tree and remove some redundant or unimportant correlations using the prior knowledge or the hypothesis to reduce the degrees.

6.3 Causality Inference

A Bayesian network is model of dependencies of the random variables. However, the biologists are more interested in understanding the flow of causality in the biological systems. For example, an expression of a gene is an immediate cause of the expression of another gene. At first glance, we would think there is no direct connection between correlation and causality. However, it has been studied [Spirtes et al., 1989; Pearl and Verma, 1995; N et al., 2000] for years that when we can learn a causal network. It is generally believed that it is very difficult to obtain a complete causality network from data, but learning partial causality inference from data is possible. In the future, we hope to use the learned multiple constrained Bayesian k-trees to derive partial causalities. An inherent advantage of using Bayesian k-trees to study the causality network is that the dependencies in Bayesian k-trees are depicted between a random variable x_i and a clique C_i of random variables. And the creating order of a Bayesian k-tree provides an effective way to model the dependencies. We believe this will have a significant impact on predicted casual relationships from real data.

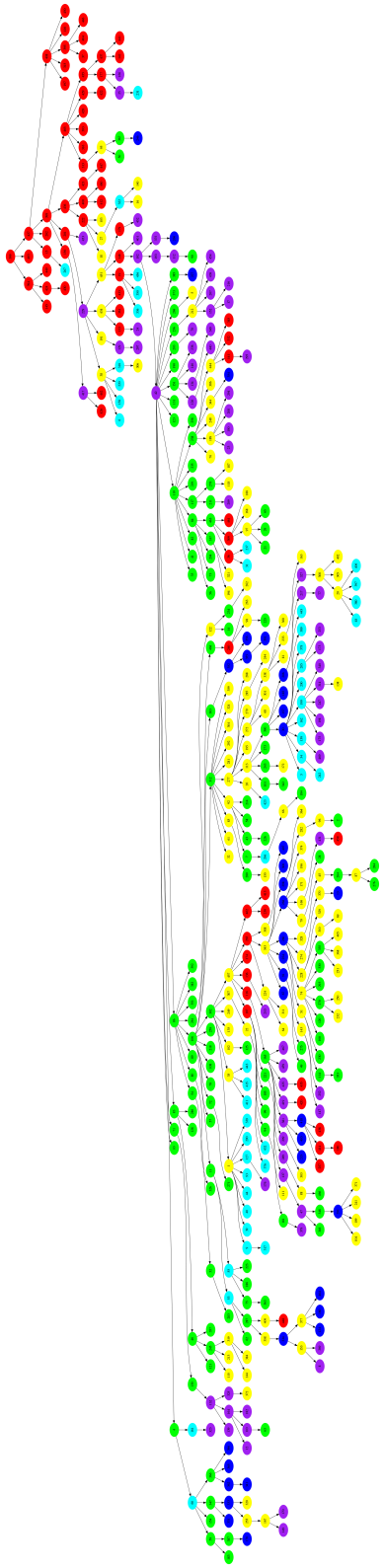


Figure 6.3: A spanning tree learning from the data set. Tree nodes are colored by its pathways: Iron related genes (green), Oxidative stress (cyan), Nucleotide synthesis (yellow), Glycosaminoglycan (purple), Lipid peroxidation (blue), Collagen (red)

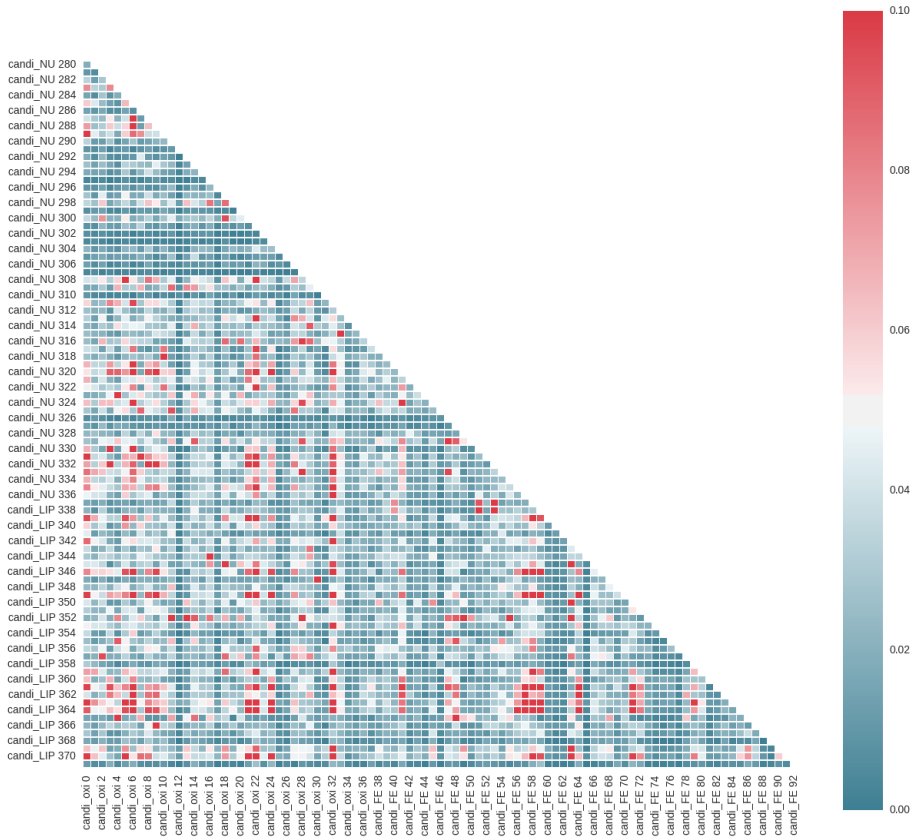


Figure 6.4: Heat map for all correlations for the data set.

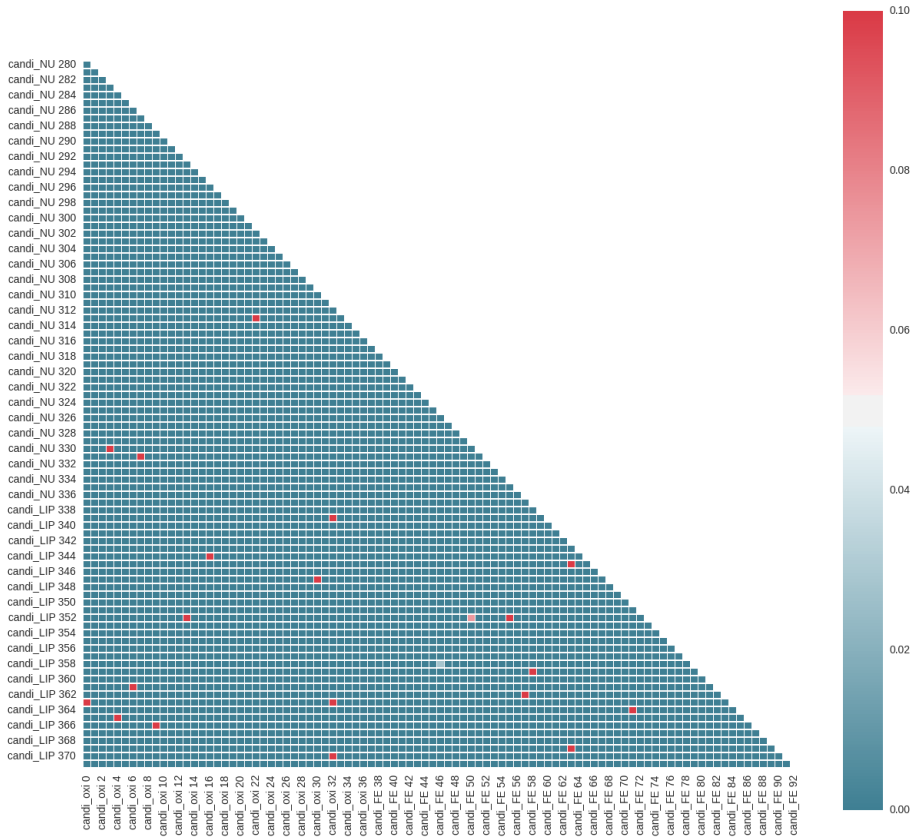


Figure 6.5: Heat map for the correlations captured by the learned spanning tree.

Appendix A

Supplementary Materials

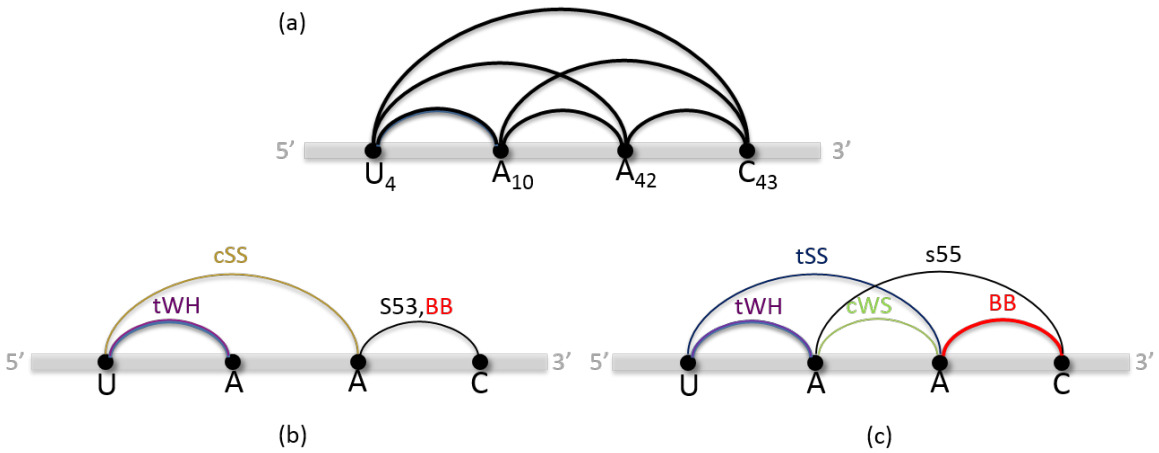


Figure A.1: (a) An example of 4-clique q with nucleotides $U_4, A_{10}, A_{42}, C_{43}$, where $4, 10, 42, 43$ are the indices of the nucleotides in the sequence. The identifier of q is UAAC001, where UAAC are the ordered nucleotides, 0 (resp. 1) encodes the number of nucleotides between two nucleotides along the backbone is larger than (resp. equal to) 0. Two possible interaction patterns (ips) for the clique of identifier UAAC001 in (a), with interactions labeled between the nucleotides (see Table 1), can be found by searching the identifier in the interaction patterns database NIPDB (see Section 2.3 in the main text). NIPDB has been established by extracting ips from the known RNAs of length < 100 nucleotides, e.g., one of the sources of the ip in (b) is chain D of the tRNA (PDB ID: 2DU3) with PDB numbers $\{908, 914, 946, 947\}$ for the 4 nucleotides. All extracted ips are grouped according to their identifiers in NIPDB. (b) and (c) also show the subgraphs induced by two ips, respectively, where each interaction induces one edge of the subgraph and if multiple interactions induce the same edge, only one edge is shown.

A.1 Pre-processing Step

The predicted interaction can be critical to the accuracy of the 3D modeling. Based on Table A.10, the predicted 3D structures using the resolved interactions are generally more accurately than those using the predicted interactions. One of the major reasons is that the stacking interactions with

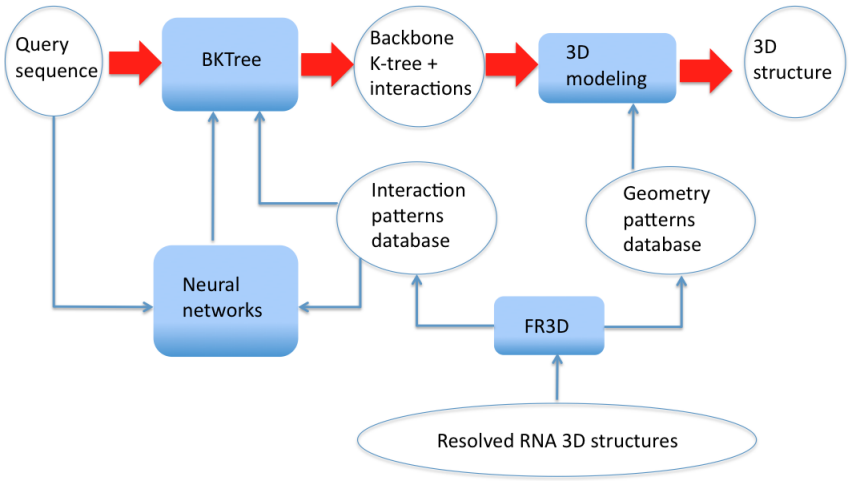


Figure A.2: New framework for the RNA 3D structure prediction.

Table A.1: Categories, types and families of RNA nucleotide interactions (from 5' to 3'), mostly summarized from works (Leontis and Westhof, 2001; Leontis *et al.*, 2002; Zirbel *et al.*, 2009; Zirbel, 2011). It also includes the phosphodiester interaction between two neighboring nucleotides.

Categories	Types (Interaction Families)	Num.
Base pairs	cWW, tWW, cWH, tWH, cHW, tHW, cWS, tWS, cSW, tSW, cHH, tHH, cHS, tHS, cSH, tSH, cSS, tSS	18
Base phosphates	0BPh, 1BPh, 2BPh, 3BPh, 4BPh, 5BPh, 6BPh, 7BPh, 8BPh, 9BPh	10
Base riboses	0BR, 1BR, 2BR, 3BR, 4BR, 5BR, 6BR, 7BR, 9BR	9
Base stackings	s35, s53, s33, s55	4
Backbone-backbone	phosphodiester(BB)	1

different facing conformations were treated as one type of stacking when we trained the neural networks. As a result, the provided stacking facing conformation that might not always lead to the correct prediction. Therefore, we have introduced a pre-processing step that establishes a serial of knowledge-based rules to modify the predicted interactions before using them in the 3D modeling step.

Figure A.8 show examples of four uncommon base stackings that cause uncommon shapes in the modeled 3D structures. The pre-processing step solves the problem by removing or replacing certain

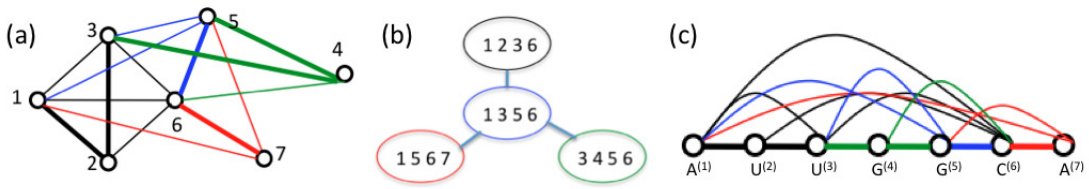


Figure A.3: (a) 3-tree of 7 vertices by Definition 1, with the order of forming four 4-cliques: from clique $\{1, 2, 3, 6\}$ (black edges), vertex 5 and blue edges added, then vertex 7 and red edges added, and finally vertex 4 and green edges added. (b) Illustration of the graph of (a) with a tree-topology connecting the four 4-cliques. (c) A backbone 3-tree for sequence `AUUGGCA`, of the same topology as shown in (a); backbone edges are in bold.

interactions and results in a more common backbone spin. More specifically, we introduced rules including removing contiguous s53; removing s55 or s33 within 3 intervals; removing contiguous interactions in cis or trans conformation and replacing contiguous s33 with s35. The proposed rules help to improve the result of 3D prediction generally (see Fig. 6 in the main text).

Table A.2: Features selected from a given $(k+1)$-clique q and the subgraph $H_{q,A(q)}$ for training ANN $\mathcal{N}^{(i,j),t}_{q,H_{q,A(q)}}$. CBP is an abbreviation for canonical base pair. A *Component* (Cp) is defined as the maximal subsequence consisting of two or more nucleotides each involved in a CBP.

Feature	Value	Comments
Seq. length	An integer	Length of a training sequence containing q.
Distances	k integers	Distances between every two nucleotides in the sequential order in q.
Number of Cps	k integers	Number (one of $\{0, 1, 2, 3, -1\}$) of Cps on the subsequence between every two nucleotides in the sequential order. 3 means there are at least 3 Cps; -1 means the two nucleotides are neighboring nucleotides on the sequence.
Neighbor nts.	$k+1$ 4-mers	One 4-mer (of letter `A, C, G, U`) for every nucleotide in q, where the first two letters and the last two letter of the 4-mer indicate the two nts to the left and to the right of the nucleotide, respectively, and letter `N` is used when there is no neighbor.
Neighbor CBPs	$k+1$ 4-mers	One 4-mer (of binary bits) for every nucleotide in q, where the first two bits and the last two bits of the 4-mer indicate the two nts to the left and to the right of the nucleotide are involved in CBPs, respectively, and letter `N` is used when there is no neighbor.
Edge properties	up to $\frac{k(k+1)}{2}$ integers	For every edge in the subgraph $H_{q,A(q)}$ of q, value 0 indicates both nts are involved in a CBP; -1 (resp. +1) indicates exclusively left (resp. right) nt is involved in a CBP; 2 indicates either is near a CBP; and -2 indicates both are far away (distant beyond 3 nts) from a CBP.

Table A.3: Average performances of MC, Rosetta, NAST and BkTree, with results in two categories: average over all successfully resolved RNAs and average over all successfully resolved RNAs of length > 50. The best performance data are displayed in bold.

	All RNAs				RNAs of length >50			
	Success/Total	STY	PPV	MCC	Success/Total	STY	PPV	MCC
MC	21/43	80.7	**86.2**	0.8344	6/18	77.1	**86.0**	0.8145
Rosetta	43/43	62.8	80.3	0.7101	18/18	53.4	78.5	0.6474
NAST	30/43	44.5	68.2	0.5508	12/18	44.0	71.4	0.5604
BkTree	43/43	**88.6**	81.3	**0.8482**	18/18	**86.0**	82.7	**0.8433**

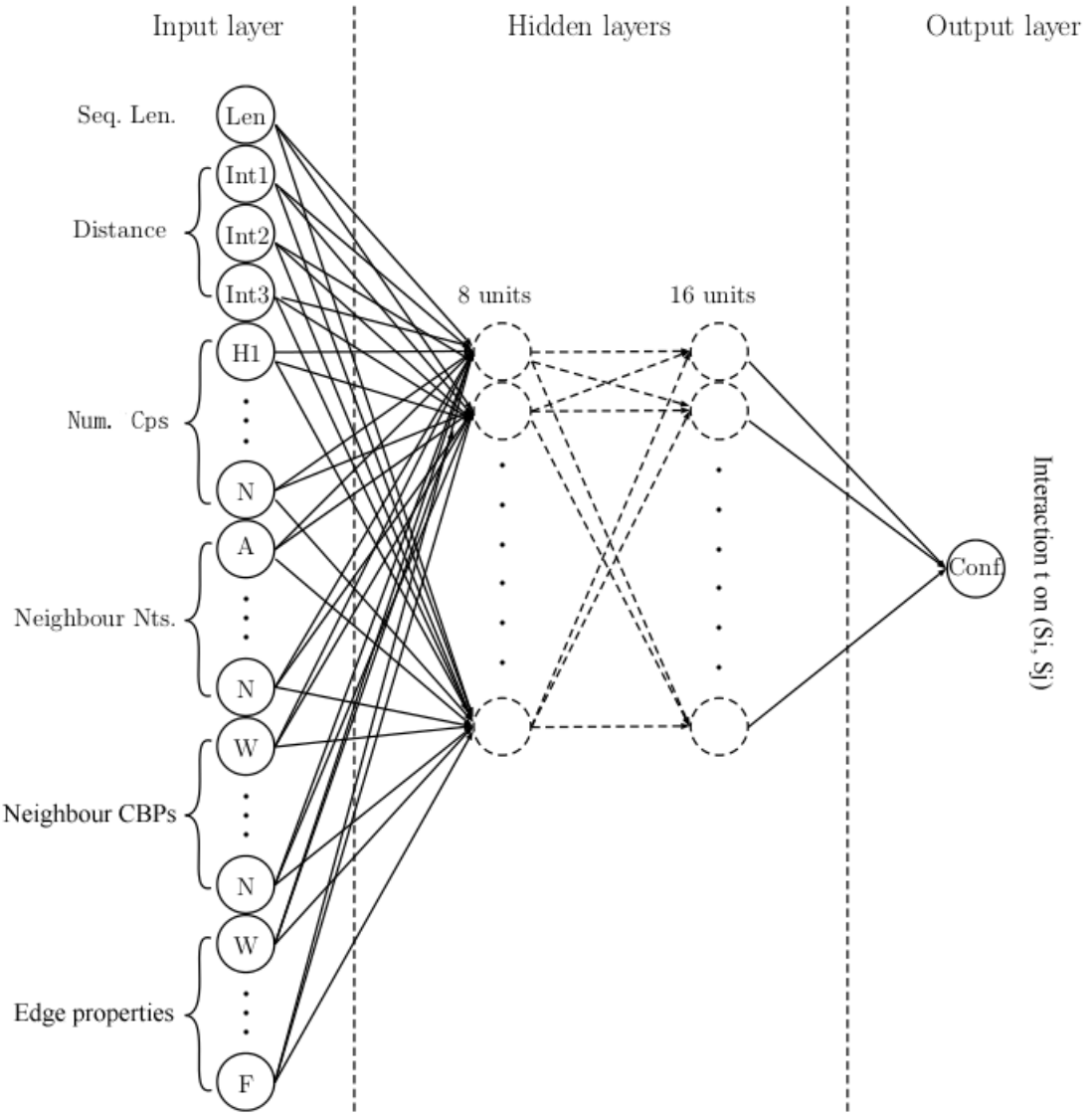


Figure A.4: Structure of ANN $\mathcal{N}^{(i,j),t}_{q,H_{q,A(q)}}$. For every feature listed in Table 2, there is a node in the input layer for each value of the feature. The hidden layer is comprise of two layers with 8 nodes and 16 nodes respectively. The output layer is a single node representing the confidence of interaction t on (S_i, S_j).

Figure A.5: Nucleotide interaction prediction results by BkTree on the benchmark set used in the survey (Laing and Schlick, 2010). The number of canonical base pairs (CPBs) and number of non-canonical interactions (NCIs) are listed. The STY, PPV and MCC were calculated, excluding the canonical bases pairs used as a part of the input. Column 8 shows the edge difference (see Section 2) for each of the RNAs. The data of the 7 RNAs displayed with the bold font are not in set T.

PDB ID	Length	# CBPs	# NCIs	STY	PPV	MCC	EdgeDiff	Structure complexity
2F8K	**16**	**6**	**14**	**85**	**85**	**0.8571**	**12/12**	**Hairpin**
2AB4	**20**	**6**	**20**	**100**	**90**	**0.9534**	**12/12**	**Hairpin**
361D	**20**	**5**	**17**	**70**	**57**	**0.6351**	**10/10**	**Hairpin**
2ANN	23	3	24	75	66	0.7071	12/12	Hairpin
1RLG	25	5	22	95	63	0.7793	15/15	Hairpin, internal loop
2QUX	25	9	22	90	71	0.8058	15/15	Hairpin
387D	26	4	23	86	68	0.7744	10/10	Hairpin
1MSY	27	6	39	97	92	0.9502	23/23	Hairpin
1L2X	28	8	34	88	88	0.8823	23/22	Pseudoknot
2AP5	28	8	29	82	66	0.7427	23/21	Pseudoknot
1JID	29	8	31	93	72	0.8235	20/20	Hairpin, internal loop
1OOA	29	8	29	93	72	0.8242	15/15	Hairpin, internal loop
430D	29	6	37	94	77	0.8577	21/21	Hairpin, internal loop
3SNP	30	12	31	93	85	0.8932	22/22	Hairpin, internal loop
2OZB	33	10	33	93	79	0.8641	26/26	Hairpin, internal loop
1MJI	34	10	44	84	84	0.8409	29/28	Hairpin, internal loop
1ET4	35	8	40	67	84	0.7546	28/25	Pseudoknot
2HW8	36	12	44	93	80	0.8655	30/30	Hairpin, internal loop
1I6U	37	15	47	91	89	0.9053	28/28	Hairpin, internal loop
1F1T	38	10	38	81	63	0.7184	29/29	Hairpin, internal loop
1ZHO	**38**	**13**	**46**	**95**	**83**	**0.8911**	**31/31**	**Hairpin, internal loop**
1S03	47	18	53	88	79	0.8404	37/37	Hairpin, internal loop
1XJR	47	15	55	83	80	0.8215	35/35	Hairpin, internal loop
1U63	49	17	50	94	65	0.7833	36/36	Hairpin, internal loop
2PXB	49	16	66	98	94	0.9632	38/38	Hairpin, internal loop
2FK6	53	20	58	77	70	0.7385	43/33	Pseudoknot, 3-way junction
3E5C	53	21	65	84	73	0.7877	40/39	3-way junction (riboswitch)
1MZP	55	17	73	64	73	0.6876	44/38	Hairpin internal
1DK1	57	24	65	100	89	0.9436	46/45	3-way junction
1MMS	58	20	86	74	82	0.7814	57/45	3-way junction
3EGZ	65	23	72	70	66	0.6849	39/35	3-way junction (riboswitch)
2QUS	69	26	81	75	76	0.7577	48/42	Pseudoknot, 3-way junction
1KXK	70	28	87	96	92	0.9440	54/54	Hairpin, internal loop
2DU3	71	27	75	78	70	0.7433	50/43	4-way junction (tRNA)
2OIU	**71**	**29**	**84**	**90**	**83**	**0.8692**	**51/51**	**3-way junction (riboswitch)**
1SJ4	73	19	83	78	81	0.7976	53/38	Pseudoknot, 4-way junction
1P5O	77	29	86	97	77	0.8716	51/51	Hairpin, internal loop
3D2G	77	28	103	80	88	0.8435	70/60	3-way junction (riboswitch)
2HOJ	79	27	100	87	84	0.8572	64/58	3-way junction (riboswitch)
2GDI	**80**	**32**	**100**	**84**	**80**	**0.8197**	**63/56**	**3-way junction (riboswitch)**
2GIS	94	36	125	87	82	0.8485	79/67	Pseudoknot, 4-way junction (riboswitch)
1LNG	97	38	124	85	79	0.8254	76/70	3-way junction (SRP)
1MFQ	**128**	**49**	**164**	**81**	**76**	**0.7895**	**101/92**	**3-way junction (SRP)**

Table A.4: List of MCC values predicted using BkTree, excluding the canonical bases pairs used as a part of the input, for 13 long RNAs.

PDB ID	Len.	# CBPs	# NCIs	MCC	Structure complexity
1Z43	101	30	128	0.8549	3-way junction (SRP)
3SUX	101	30	121	0.7295	Pseudoknot, 3-way junction (Riboswitch)
3F2Y	109	31	135	0.7400	Pseudoknot, 4-way junction (Riboswitch)
1NBS	120	31	147	0.8117	4-way junction (Ribonuclease P RNA)
4KQY	120	39	138	0.7671	Pseudoknot (Riboswitch)
1L9A	126	37	146	0.8497	3-way junction (Synthetic RNA)
3NDB	136	43	174	0.8431	3-way junction (SRP)
2QBZ	153	47	204	0.7982	3-way junction (Synthetic RNA)
1U9S	155	50	205	0.7643	4-way junction (Synthetic RNA)
2R8S	159	45	204	0.7483	3-way junction
4ERL	161	55	219	0.8292	Pseudoknot, 5-way junction (Riboswitch)
4GXY	162	46	190	0.6666	4-way junction (Riboswitch)
3DIL	174	62	234	0.8337	Pseudoknot, 5-way junction (Riboswitch)

Figure A.6: List of performance values predicted using MC, Rosetta and BkTree on 4 representative RNAs chosen by (Laing and Schlick, 2010). The results generated by MC and Rosetta are obtained from the survey paper (Laing, 2014; Laing and Schlick, 2010). We display the results of each RNA in two categories, where the average and the best performances of up to five folds are shown in the upper and lower category respectively. The highest values among the results of three methods are displayed in bold.

		MC				Rosetta				BkTree			
	PDB	1KXK	1XJR	2OIU	2QUS	1KXK	1XJR	2OIU	2QUS	1KXK	1XJR	2OIU	2QUS
	NT	70	47	71	69	70	47	71	69	70	47	71	69
Avg	STY	81	76	76	78	74	71	63	58	97	91	92	80
	PPV	89	87	92	86	85	83	87	86	94	84	86	80
	MCC	0.849	0.813	0.836	**0.819**	0.793	0.767	0.740	0.706	**0.958**	**0.878**	**0.892**	0.8
	RMSD	9.49	8.74	16.85	18.41	17.23	11.63	18.10	15.73	**5.46**	**2.71**	**5.02**	**9.07**
	DI	11.16	10.74	20.14	22.44	21.69	15.21	24.72	22.80	**5.69**	**3.08**	**5.62**	**11.33**
Best	STY	91	90	93	87	88	87	91	90	97	91	92	80
	PPV	84	82	80	79	76	78	75	73	94	84	86	80
	MCC	0.874	0.859	0.862	**0.829**	0.817	0.794	0.826	0.810	**0.958**	**0.878**	**0.892**	0.8
	RMSD	8.71	5.72	14.02	15.92	12.65	8.88	15.98	12.07	**5.46**	**2.71**	**5.02**	**9.07**
	DI	10.03	7.08	16.78	19.19	15.53	10.97	21.58	15.03	**5.69**	**3.08**	**5.62**	**11.33**

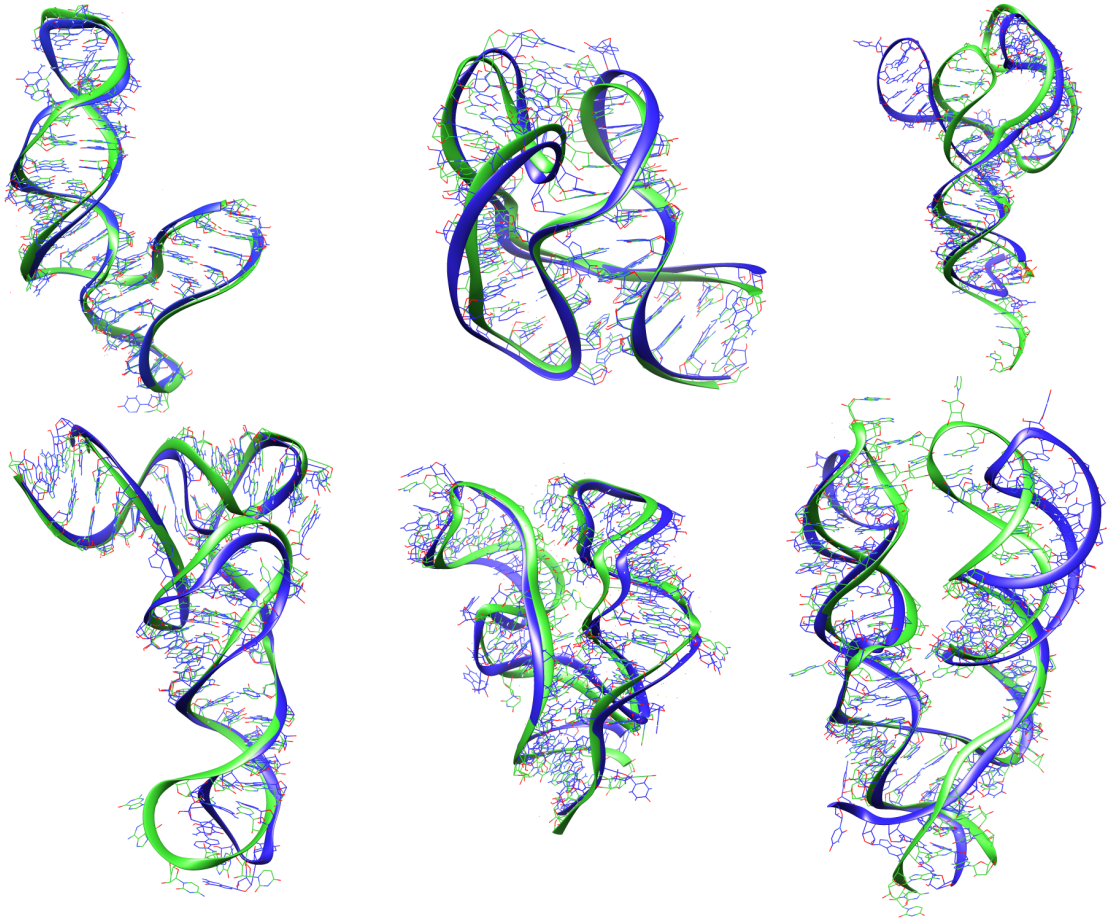


Figure A.7: Examples of predicted 3D models (blue) superimposed with respective native structures (green). The superimpositions in the frist row from left to right: 1DK1 (57 nucleotides, 3-way junction, 1.029Å); 1MMS (58 nucleotides, 3-way junction, 2.032Å); 2QUS (69 nucleotides, pseudoknot, 9.864Å). The superimpositions in the second row from left to right: 2DU3 (71 nucleotides, 4-way junction, tRNA, 3.751Å); 3D2G (77 nucleotides, 3-way junction, riboswitch, 4.552Å), 1LNG (97 nucleotides, 3-way junction, SRP, 10.862Å).

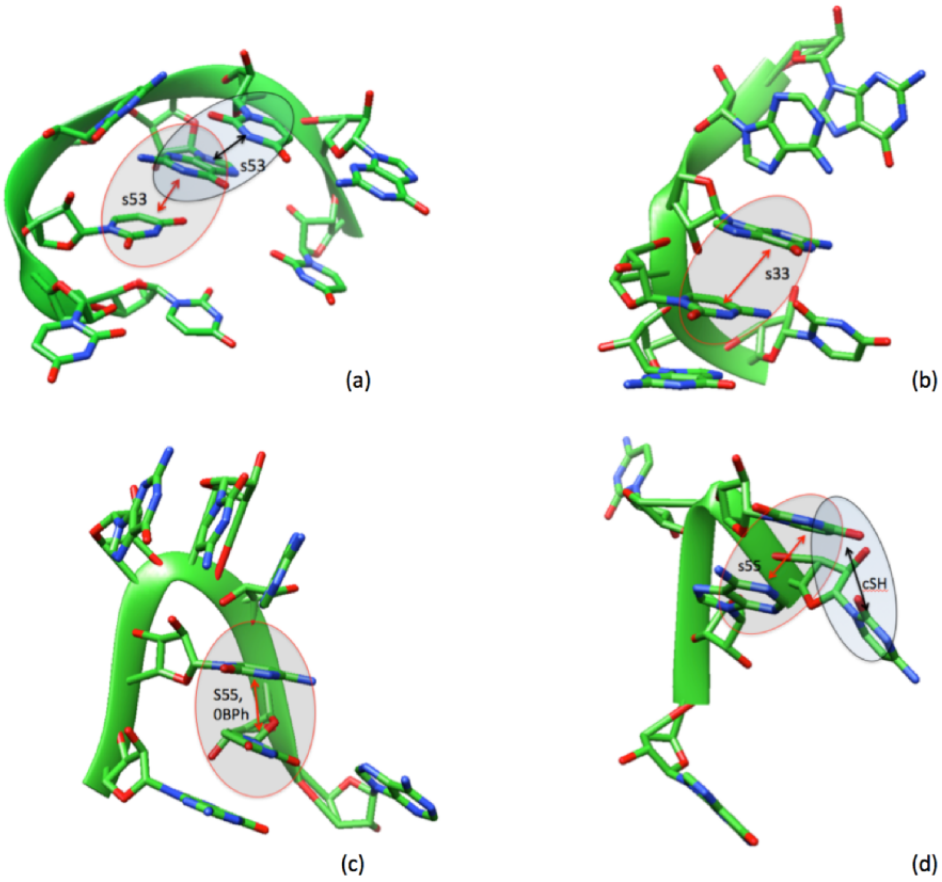


Figure A.8: Examples of uncommon shapes of RNA backbone spins. Stacking interactions other than s35 between neighboring nucleotides will force the backbone twist in a way that its torsion angles are far off their average values.

Figure A.9: RMSD and DI values of the prediction results by BkTree3D on the benchmark set used in the survey (Laing and Schlick, 2010). The number of canonical base pairs (CPBs) and number of non-canonical interactions (NCIs) are listed. The columns labeled STY, PPV, MCC and EdgeDiff were from our recent results (Ding *et al.*, 2015), where the predicted interactions were used as inputs to BkTree3D. The data of the 7 RNAs displayed with the bold font are not used in the training of the neural networks in the interaction prediction stage.

PDB ID	Length	# CBPs	# NCIs	STY	PPV	MCC	RMSD	DI	EdgeDiff	Structure complexity
2F8K	**16**	**6**	**14**	**85**	**80**	**0.828**	**2.911**	**3.515**	**12/12**	**Hairpin**
2AB4	**20**	**6**	**20**	**100**	**90**	**0.953**	**2.753**	**2.887**	**12/12**	**Hairpin**
361D	**20**	**5**	**17**	**100**	**73**	**0.859**	**2.525**	**0.936**	**10/10**	**Hairpin**
2ANN	23	3	24	100	96	0.979	4.351	4.440	12/12	Hairpin
1RLG	25	5	22	95	60	0.756	2.630	3.475	15/15	Hairpin, internal loop
2QUX	25	9	22	90	86	0.889	3.162	3.556	15/15	Hairpin
387D	26	4	23	95	78	0.866	6.069	7.000	10/10	Hairpin
1MSY	27	6	39	97	97	0.974	1.304	1.338	23/23	Hairpin
1L2X	28	8	34	97	94	0.956	3.534	3.694	23/22	Pseudoknot
2AP5	28	8	29	79	65	0.721	6.231	8.631	23/21	Pseudoknot
1JID	29	8	31	87	69	0.776	2.346	3.021	20/20	Hairpin, internal loop
1OOA	29	8	29	68	60	0.646	4.165	6.442	15/15	Hairpin, internal loop
430D	29	6	37	89	80	0.847	4.335	5.116	21/21	Hairpin, internal loop
3SNP	30	12	31	93	82	0.880	1.198	1.360	22/22	Hairpin, internal loop
2OZB	33	10	33	93	83	0.887	3.497	3.941	26/26	Hairpin, internal loop
1MJI	34	10	44	75	86	0.807	1.758	2.178	29/28	Hairpin, internal loop
1ET4	35	8	40	77	86	0.816	4.565	5.588	28/25	Pseudoknot
2HW8	36	12	44	90	85	0.879	6.673	7.586	30/30	Hairpin, internal loop
1I6U	37	15	47	91	89	0.905	3.095	3.418	28/28	Hairpin, internal loop
1F1T	38	10	38	78	66	0.725	4.258	5.869	29/29	Hairpin, internal loop
1ZHO	**38**	**13**	**46**	**91**	**85**	**0.884**	**3.264**	**3.689**	**31/31**	**Hairpin, internal loop**
1S03	47	18	53	88	82	0.855	3.485	4.075	37/37	Hairpin, internal loop
1XJR	47	15	55	89	94	0.916	2.494	2.721	35/35	Hairpin, internal loop
1U63	49	17	50	90	70	0.795	2.087	2.623	36/36	Hairpin, internal loop
2PXB	49	16	66	100	98	0.992	1.140	1.148	38/38	Hairpin, internal loop
3E5C	53	21	65	90	80	0.856	2.042	2.384	40/39	3-way junction (r: riboswitch)
1MZP	55	17	73	78	86	0.821	11.23	13.675	44/38	Hairpin internal
1DK1	57	24	65	95	88	0.919	1.029	1.119	46/45	3-way junction
1MMS	58	20	86	77	89	0.834	2.032	2.435	57/45	3-way junction
3EGZ	65	23	72	77	74	0.762	6.433	8.441	39/35	3-way junction (r)
2QUS	69	26	81	70	76	0.731	9.864	13.488	48/42	Pseudoknot, 3-way junction
1KXK	70	28	87	90	90	0.908	3.230	3.555	54/54	Hairpin, internal loop
2DU3	71	27	75	82	77	0.800	3.751	4.686	50/43	4-way junction (tRNA)
2OIU	**71**	**29**	**84**	**91**	**89**	**0.905**	**5.629**	**6.213**	**51/51**	**3-way junction (r)**
1SJ4	73	19	83	73	82	0.778	13.99	17.973	53/38	Pseudoknot, 4-way junction
1P5O	77	29	86	95	79	0.871	6.826	7.834	51/51	Hairpin, internal loop
3D2G	77	28	103	79	87	0.833	4.552	5.462	70/60	3-way junction (r)
2HOJ	79	27	100	85	87	0.863	4.024	4.662	64/58	3-way junction (r)
2GDI	**80**	**32**	**100**	**82**	**81**	**0.815**	**5.176**	**6.343**	**63/56**	**3-way junction (r)**
2GIS	94	36	125	84	80	0.828	14.87	17.951	79/67	Pseudoknot, 4-way junction (r)
1LNG	97	38	124	86	83	0.849	10.86	12.786	76/70	3-way junction (SRP)
1MFQ	**128**	**49**	**164**	**79**	**80**	**0.800**	**24.58**	**30.723**	**101/92**	**3-way junction (SRP)**

Figure A.10: RMSD values from the results of three independent tests using BkTree3D on the benchmark set used in the survey [Laing and Schlick, 2010]. The resolved interactions from RNA 3D Atlas were used as inputs to BkTree3D to produce the results in RMSD_r column. For RMSD_p column, the predicted interactions were preprocessed before inputting to BkTree3D. In addition to the preprocessing of the predicted interactions, the geometry candidates of a testing RNA were removed from GDB before GDB was used by BkTree3D for the results in RMSD_g column.

PDB ID	Length	RMSD_r	RMSD_p	RMSD_g	Structure complexity
2F8K	16	2.02	2.674	2.694	Hairpin
2AB4	20	3.01	2.787	2.571	Hairpin
361D	20	3.99	5.809	5.825	Hairpin
2ANN	23	3.04	1.642	2.649	Hairpin
1RLG	25	4.61	2.169	4.021	Hairpin, internal loop
2QUX	25	1.05	2.156	2.156	Hairpin
387D	26	0.06	6.002	6.002	Hairpin
1MSY	27	1.00	2.49	2.337	Hairpin
1L2X	28	1.45	2.433	3.750	Pseudoknot
2AP5	28	2.85	1.997	2.711	Pseudoknot
1JID	29	1.77	3.249	4.180	Hairpin, internal loop
1OOA	29	0.159	2.262	2.262	Hairpin, internal loop
430D	29	2.26	4.165	4.282	Hairpin, internal loop
3SNP	30	2.132	1.602	2.263	Hairpin, internal loop
2OZB	33	1.365	2.143	2.347	Hairpin, internal loop
1MJI	34	1.105	4.173	4.173	Hairpin, internal loop
1ET4	35	3.816	5.214	5.214	Pseudoknot
2HW8	36	1.574	4.255	4.542	Hairpin, internal loop
1I6U	37	1.569	3.051	3.051	Hairpin, internal loop
1F1T	38	2.501	8.015	8.174	Hairpin, internal loop
1ZHO	38	2.07	4.094	4.328	Hairpin, internal loop
1S03	47	1.512	2.707	2.907	Hairpin, internal loop
1XJR	47	1.355	2.88	2.88	Hairpin, internal loop
1U63	49	1.421	4.991	5.293	Hairpin, internal loop
2PXB	49	0.884	2.506	2.506	Hairpin, internal loop
3E5C	53	7.598	1.716	1.716	3-way junction (r: riboswitch)
1MZP	55	4.862	5.648	5.833	Hairpin internal
1DK1	57	4.944	1.686	1.686	3-way junction
1MMS	58	4.768	2.864	2.973	3-way junction
3EGZ	65	4.893	10.33	10.33	3-way junction (r)
2QUS	69	3.001	11.01	11.13	Pseudoknot, 3-way junction
1KXK	70	2.666	3.261	3.261	Hairpin, internal loop
2DU3	71	0.558	2.993	2.993	4-way junction (tRNA)
2OIU	71	1.21	5.9	5.9	3-way junction (r)
1SJ4	73	7.333	11.18	11.18	Pseudoknot, 4-way junction
1P5O	77	11.47	5.166	5.166	Hairpin, internal loop
3D2G	77	2.461	7.162	7.162	3-way junction (r)
2HOJ	79	8.283	2.564	3.583	3-way junction (r)
2GDI	80	2.227	3.22	3.22	3-way junction (r)
2GIS	94	5.713	14.13	14.34	Pseudoknot, 4-way junction (r)
1LNG	97	19.91	7.959	10	3-way junction (SRP)
1MFQ	128	20.79	21.33	20.79	3-way junction (SRP)

Figure A.11: RMSD values of 5 top structures predicted by BkTree3D on the benchmark set used in the survey (Laing and Schlick, 2010). The predicted interactions were preprocessed (see Section X) before inputting to BkTree3D. The best and average results were also listed for each RNA.

PDB ID	Length	RMSD1	RMSD2	RMSD3	RMSD4	RMSD5	Best	Average	Structure complexity
2F8K	16	2.674	2.644	2.644	3.544	3.544	2.644	3.004	Hairpin
2AB4	20	5.809	4.950	4.975	4.939	5.005	4.939	5.139	Hairpin
361D	20	2.787	2.472	2.466	2.543	2.431	2.431	2.497	Hairpin
2ANN	23	1.642	2.474	2.526	2.367	2.224	2.224	2.448	Hairpin
1RLG	25	2.156	3.938	3.984	3.874	4.205	2.156	3.632	Hairpin, internal loop
2QUX	25	2.169	4.245	3.854	3.996	4.514	3.854	4.126	Hairpin
387D	26	6.002	6.442	6.762	8.708	7.986	6.002	7.180	Hairpin
1MSY	27	2.49	2.133	2.064	2.100	2.087	2.064	2.144	Hairpin
1L2X	28	2.433	3.956	4.817	4.842	4.614	3.750	4.396	Pseudoknot
2AP5	28	1.997	2.711	2.711	2.711	2.711	2.711	2.711	Pseudoknot
1JID	29	3.249	2.029	1.828	2.044	2.039	1.828	2.024	Hairpin, internal loop
1OOA	29	2.262	4.244	4.258	4.256	4.055	4.055	4.219	Hairpin, internal loop
430D	29	1.602	3.277	3.098	5.184	3.285	2.263	3.422	Hairpin, internal loop
3SNP	30	4.165	1.789	2.739	2.739	2.750	1.602	2.324	Hairpin, internal loop
2OZB	33	2.143	3.889	4.013	3.174	3.987	3.037	3.620	Hairpin, internal loop
1MJI	34	4.173	4.196	4.243	3.939	4.060	3.939	4.193	Hairpin, internal loop
1ET4	35	5.214	5.764	5.138	5.264	5.207	5.138	5.362	Pseudoknot
2HW8	36	4.255	3.742	3.667	3.836	3.635	3.579	3.692	Hairpin, internal loop
1I6U	37	3.051	3.432	3.414	3.486	3.273	3.273	3.448	Hairpin, internal loop
1F1T	38	4.094	3.912	3.622	4.039	3.934	3.622	3.912	Hairpin, internal loop
1ZHO	38	8.015	4.925	5.903	5.944	4.429	4.429	5.257	Hairpin, internal loop
1S03	47	2.88	2.670	2.562	2.542	2.401	2.401	2.545	Hairpin, internal loop
1XJR	47	2.707	3.113	3.035	2.626	3.197	2.267	2.848	Hairpin, internal loop
1U63	49	2.506	3.186	3.554	5.375	2.900	2.571	3.517	Hairpin, internal loop
2PXB	49	4.991	4.979	5.181	6.136	6.059	4.979	5.468	Hairpin, internal loop
3E5C	53	1.716	2.041	3.253	2.021	3.024	1.785	2.425	3-way junction (r: riboswitch)
1MZP	55	5.648	7.685	7.976	8.633	9.966	5.731	7.998	Hairpin internal
1DK1	57	1.686	1.994	1.982	2.228	3.092	1.685	2.196	3-way junction
1MMS	58	2.864	3.701	4.439	4.240	4.025	3.483	3.977	3-way junction
3EGZ	65	10.33	11.95	12.46	11.32	11.20	10.33	11.45	3-way junction (r)
2QUS	69	11.01	9.857	15.39	17.66	21.89	15.39	14.83	Pseudoknot, 3-way junction
1KXK	70	3.261	5.089	6.140	5.032	6.574	4.114	5.390	Hairpin, internal loop
2DU3	71	2.993	3.303	2.703	3.187	3.494	2.703	3.136	4-way junction (tRNA)
2OIU	71	5.9	6.832	10.24	6.127	8.771	10.24	7.582	3-way junction (r)
1SJ4	73	11.18	11.44	10.08	10.23	11.58	10.08	11.02	Pseudoknot, 4-way junction
1P5O	77	5.166	5.477	6.657	5.978	5.694	5.466	5.854	Hairpin, internal loop
3D2G	77	7.162	7.939	7.523	10.32	9.456	10.32	8.588	3-way junction (r)
2HOJ	79	2.564	2.750	3.709	3.162	3.461	2.566	3.130	3-way junction (r)
2GDI	80	3.22	3.149	4.142	3.189	4.025	3.112	3.523	3-way junction (r)
2GIS	94	14.13	15.32	17.45	15.80	16.51	14.13	15.84	Pseudoknot, 4-way junction (r)
1LNG	97	7.959	7.804	8.330	7.735	8.801	7.735	8.177	3-way junction (SRP)
1MFQ	128	18.3	24.42	24.92	23.37	24.19	22.37	23.85	3-way junction (SRP)

Figure A.12: List of performance values predicted using MC, Rosetta and BkTree3D on 4 representative RNAs chosen by (Laing and Schlick, 2010). The results generated by MC and Rosetta are obtained from the survey paper (Laing, 2014; Laing and Schlick, 2010). We display the results of each RNA in two categories, where the average and the best performances of up to five folds are shown in the upper and lower category respectively. STY, PPV and MCC values are calculated based on the interaction prediction results of the first stage of the pipeline; RMSD and DI are from the results of the 3D modeling stage. The highest values among the results of three methods are displayed in bold.

		MC				Rosetta				BkTree3D			
	PDB	1KXK	1XJR	2OIU	2QUS	1KXK	1XJR	2OIU	2QUS	1KXK	1XJR	2OIU	2QUS
	NT	70	47	71	69	70	47	71	69	70	47	71	69
Avg	STY	81	76	76	78	74	71	63	58	90	89	91	70
	PPV	89	87	92	86	85	83	87	86	90	94	89	76
	MCC	0.849	0.813	0.836	**0.819**	0.793	0.767	0.740	0.706	**0.908**	**0.916**	**0.905**	0.731
	RMSD	9.49	8.74	16.85	18.41	17.23	11.63	18.10	15.73	**3.26**	**2.88**	**5.90**	**11.01**
	DI	11.16	10.74	20.14	22.44	21.69	15.21	24.72	22.80	**3.59**	**3.14**	**6.51**	**15.06**
Best	STY	91	90	93	87	88	87	91	90	90	89	91	70
	PPV	84	82	80	79	76	78	75	73	90	94	89	76
	MCC	0.874	0.859	0.862	**0.829**	0.817	0.794	0.826	0.810	**0.908**	**0.916**	**0.905**	0.731
	RMSD	8.71	5.72	14.02	15.92	12.65	8.88	15.98	12.07	**3.26**	**2.88**	**5.90**	**11.01**
	DI	10.03	7.08	16.78	19.19	15.53	10.97	21.58	**15.03**	**3.59**	**3.14**	**6.51**	15.06

Bibliography

Mohammed Said Abual-Rub and Rosni Abdullah. A survey of protein fold recognition algorithms. *Journal of Computer Science*, 4(9):768–776, 2008.

R. Achawanantakun, S. Takyar, and Y. Sun. Grammar string: A novel ncRNA secondary structure representation. *lifesciences society org*, pages 2–13, 2010.

Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. The design and analysis of computer algorithms. *Addison-Wesley*, 1974.

A. Antoine Rozenknop. Gibbsian context-free grammar for parsing. In *Proceedings of Conference on Text, Speech and Dialogue*, pages 49–56, 2006.

S. Arnborg and A. Proskurowski. Linear time algorithms for np-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11–24, 1989.

S. Arnborg, A. Proskurowski, and DG. Corneil. Forbidden minors characterization of partial 3-trees. *Discrete Mathematics*, 80(1):1–19, 1990.

Stefan Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability-a survey. *BIT Numerical Mathematics*, 25(1):1–23, 1985.

Stefan Arnborg and Jens Lagergren. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):141–151, 1991.

S. Bakan, L. Meireles, and I Bahar. Prody: Protein dynamics inferred from theory and experiments. *Bioinformatics*, 27(11):1575–1577, 2011.

L. W. Beineke and R. E. Pippert. Properties and characterizations of k-trees. *Mathematika*, 18: 141–151, 1971.

Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Res*, 28: 235–242, 2000a.

H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. Prody: Protein dynamics inferred from theory and experiments. *Nucleic Acids Research*, 28:235–242, 2000b.

Marshall W. Bern, Eugene L. Lawler, and A. L. Wong. Linear-time computation of optimal subgraphs of decomposable graphs. *J. Algorithms*, 8(2):216–235, 1987.

Marshall Wayne Bern. *Network Design Problems: Steiner Trees and Spanning K-Trees*. PhD thesis, University of California, 1987.

J.P. Bida and L.J. III Maher. Improved prediction of rna 3d structure with insights into native state dynamics. *RNA*, 18:385–393, 2012.

Hans L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In *Proceedings of the 15th International Colloquium on Automata, Languages and Programming (ICALPâĂŹ88)*, pages 105–118. Springer, 1988.

Hans L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–23, 1993.

H.L. Bodlaender and A.M.C.A. Koster. Treewidth computations i. upper bounds. *Information and Computation*, 208(3):259–275, 2010.

Joseph K. Bradley and Carlos Guestrin. Learning tree conditional random fields. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

Leizhen Cai. Spanning 2-trees. *Algorithms, Concurrency and Knowledge*, 1023:10–22, 1995.

Leizhen Cai and Frédéric Maffray. On the spanning k-tree problem. *Discrete Applied Mathematics*, 44(1-3):139–156, 1993.

Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346âĂŞ1367, 2006.

D. Chiang, A. K. Joshi, and Searls D. B. Grammatical representations of macromolecular structure. *Journal of Computational Biology*, 13(5):1077–1100, 2006.

DM. Chickering, D. Geiger, and D Heckerman. Learning bayesian networks is np-hard. *Technical Rep MSR-TR-94-17, Microsoft Research Advanced Technology Division*, 1994.

CK. Chow and CN. Liu. Approximating discrete probability distribution with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Computation*, 85(1):12âĂŞ75, 1990.

P. Dagum and M. Luby. An optimal approximation algorithm for baysian inference. *Artificial Intelligence*, 60:141âĂŞ153, 1993.

Qiang-S. Daly, R and S. Aitken. Learning bayesian networks: approaches and issues. *Knowledge Engineering Review*, 26(2):99–127, 2011.

R. Das and D. Baker. Automated *de novo* prediction of native-like rna 3d structures. *Proc. Natl Acad Sci*, 104:14664–14669, 2007.

R. Das, J. Karanicolas, and D. Baker. Atomic accuracy in predicting and designing noncanonical rna structure. *Nature Methods*, 7:291–294, 2010.

S. Dasgupta. Learning polytree. In *In Laskey and Prade, ed. Proceedings Conference on Uncertainty in AI*, pages 134–141, 1999.

Erik D. Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. *The Computer Journal*, 51(3):292âĂŞ302, 2008.

K. A. Dill, A. Lucas, J. Hockenmaier, L. Huang, D. Chiang, and A. K. Josh. Computational linguistics: A new tool for exploring biopolymer structures and statistical mechanics. *Polymer*, 48:4289–4300, 2007.

F. Ding, S. Sharma, P. Chalasani, V.V. Demidov, N.E. Broude, and N.V. Dokholyan. *Ab initio* rna folding by discrete molecular dynamics: From structure prediction to folding mechanisms. *RNA*, 14:1164–1173, 2008.

L. Ding, A. Samad, G. Li, R. Robinson, X. Xue, R. Malmberg, and L. Cai. Finding maximum spanning k-trees on backbone graphs in polynomial time. *manuscript*, 2013.

L. Ding, A. Samad, X. Xue, X. Huang, R.L. Malmberg, and L. Cai. Stochastic k-tree grammar and its applications in biomolecular structure modeling. In *Lecture Notes in Computer Science Vol 8370 (LATA2014)*, pages 308–322, 2014a.

L. Ding, X. Xue, S. LaMarca, M. Mohebbi, A. Samad, R. Malmberg, and L. Cai. *Ab initio* prediction of rna nucleotide interactions with backbone k-tree model. In *Proceedings of ECCB'14 Workshop on Computational Methods for Structural RNAs*, pages 25–42, 2014b.

L. Ding, X. Xue, S. LaMarca, M. Mohebbi, A. Samad, R.L. Malmberg, and L. Cai. Accurate prediction of rna nucleotide interactions with backbone k-tree model. *Bioinformatics 2015*, (10.1093/bioinformatics/btv210), 2015.

L. Ding, A. Samad, G. Li, R.W. Robinson, X. Xue, R. Malmberg, and L. Cai. Polynomial-time algorithms for maximum spanning k-tree constrained by hamitonian path. *Submitted*, pages 1164–1173, 2016a.

L. Ding, X. Xue, S. LaMarca, R. L. Malmberg, C. Momany, and L. Cai. Accurate prediction of rna 3d structure based on backbone k-tree model. *Manuscript*, 2016b.

Liang Ding, Abdul Samad, Guojun Li, Robert W. Robinson, Xingran Xue, Russell L. Malmberg, and Liming Cai. Finding maximum spanning k-trees on backbone graphs in polynomial time. *Manuscript*, 2016c.

R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness ii: On completeness for W[1]. *Theoretical Computer Science*, 141(1-2):109–131, 1995.

R. Durbin, S. Eddy, A. Krogh, and Mitchison G. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms and Applications*, 3(3):1–27, 1999.

V. Fuchs-Tarlovsky. Role of antioxidants in cancer therapy. *Nutrition*, 29(1):15–21, 2013.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2007.

Z. Huang, Y. Wu, J. Robertson, L. Feng, R. Malmberg, and L. Cai. Fast and accurate search for non-coding RNA pseudoknot structures in genomes. *Bioinforamtics*, 24(20):2281–2287, 2008a.

Z. Huang, Mohebbi M., R. Malmberg, and L. Cai. RNAv: Non-coding RNA secondary structure variation search via graph homomorphism. In *Proceedings of Computational Systems Bioinformatics Conference (CSB2010)*, pages (9) 56–69, 2010.

Zhibin Huang, Yong Wu, Joseph Robertson, Liang Feng, Russell L. Malmberg, and Liming Cai. Fast and accurate search for non-coding rna pseudoknot structures in genomes. *Bioinforamtics*, 24(20):2281–2287, 2008b.

Sorin Istrail and Fumei Lam. Combinatorial algorithms for protein folding in lattice models: A survey of mathematical results. *Commun Inf Syst*, 9(4):303–346, 2009.

Frellsen. J., Moltke. I., M. Thiim, K.V. Mardia, J. Ferkinghoff-Borg, and T. Hamelryck. A probabilistic model of RNA conformational space. *PLoS Comput Biol*, 5(6), 2009.

H. L. Bodlaender J. H. P. Kwisthout and L. C. van der Gaag. The necessity of bounded treewidth for efficient inference in bayesian networks. In *In Proc. 19th European Conf. on AI*, page 237âĂŞ242, 2010.

M.A. Jonikas, R.J. Radmer, A. Laederach, R. Das, Pearlman S., D. Herschlag, and Altman R.B. Coarse-grained modeling of large rna molecules with knowledge-based potentials and structure filters. *RNA*, 15:189–199, 2009.

A. Joshi. How much context-sensitivity is necessary for characterizing structural descriptions. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Processing: Theoretical, Computational, and Psychological Perspectives*, pages 206–250. NY: Cambridge University Press, 1985.

A. Joshi and D. Vijay-Shanker, K. Weir. The convergence of mildly context-sensitive grammar formalisms. *Foundational Issues in Natural Language Processing, Cambridge, MA: MIT Press*, pages 31–81, 1991.

D. Jurafsky, C. Wooters, J. Segal, A. Stolcke, E. Fosler, G. Tajchaman, and N. Morgan. Using a stochastic context-free grammar as a language model for speech recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 189–192, 1995.

David Karger and Nathan Srebro. Learning markov networks: Maximum bounded tree-width graphs. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2001.

D. Klein and C. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430, 2003.

B. Knudsen and J. Hein. Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Res*, 31:3423–3428, 2003.

S. Kullback and RA. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

C. Laing and T. Schlick. Computational approaches to tertiary modeling of rna. *Journal of Physics: Condensed Matter*, 22(28), 2010.

K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

Lillian Lee. Fast context-free grammar parsing requires fast Boolean matrix multiplication. *Journal of the ACM*, 49(1):1–15, 2002.

N.B. Leontis and E. Westhof. Geometric nomenclature and classification of rna base pairs. *RNA*, 7(4):499–512, 2001.

N.B. Leontis and E. Westhof. Rna 3d structure analysis and prediction. *Springer*, 2012a.

N.B. Leontis, J. Stombaugh, and E. Westhof. The non-watson-crick base pairs and their associated isostericity matrices. *Nucleic Acids Res.*, 30(16):3497–3531, 2002.

Neocles Leontis and Eric Westhof. *RNA 3D Structure Analysis and Prediction*, volume 27. Nucleic Acids and Molecular Biology, 2012b.

II Lewis, PM. Approximating probabilistic distributions to reduce storage requirements. *Information and Control*, 2:214–225, 1959.

Chung-Shou Liao and Louxin Zhang. Approximating the spanning k-tree forest problem. In *Proceedings of the 3rd Frontiers in Algorithmics (FAW)*, pages 293–301, 2009.

D. Martin, R. Sigal, and E. J. Weyuker. *Computability, complexity, and languages: Fundamentals of theoretical computer science.* Morgan Kaufmann, 2 edition, 1994.

JirÃŋ Matousek and Robin Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1-3):343–364, 1992.

T. Mitchell. *Machine Learning.* McGraw Hill, New York, NY, USA, 1997.

A. G. Murzin, S. Brenner, T. Hubbard, and C. Chothia. Scop: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247(4): 536–540, 1995.

Friedman N, Linial M, and Pe'er D Nachman I. Using bayesian networks to analyze expression data. *J Comput Biol.*, 7(3-4):601–20, 2000.

R. Nagarajan, M. Scutari, and S. Lebre. *Bayesian Networks in R with Applications in Systems Biology*. Springer, 2013.

E. P. Nawrocki, D. L. Kolbe, and S. R. Eddy. Infernal 1.0: Inference of RNA alignments. *Bioinformatics*, 25:1335–1337, 2009.

H. F. Noller. Structure of ribosomal RNA. *Annual Review of Biochemistry*, 53:119–162, 1984.

M. Parisien and F. Major. The mc-fold and mc-sym pipeline infers rna structure from sequence data. *Nature*, 452:51–55, 2002.

H. P. Patil. On the structure of k-trees. *Journal of Combinatorics, Information and System Sciences*, 11(2-4):57–64, 1986.

Judea Pearl and Thomas S. Verma. A theory of inferred causation. *Studies in Logic and the Foundations of Mathematics*, 134:789–811, 1995.

M. Popenda, M. Szachniuk, Purzycka K.J. Lukasiak P. Bartol N. Blazewicz J. Antczak, M., and R.W. Adamiak. Automated 3d structure composition for large rnas. *Nucleic Acids Research*, 40 (14):e112, 2012.

Huzefa Rangwala and George Karypis. *Introduction to Protein Structure Prediction: Methods and Algorithms*, volume 14. Wiley. www.els.net, 2010.

V. Reinharz, F. Major, and J. Waldispühl. Towards 3d structure prediction of large rna molecules: an integer programming framework to insert local tertiary motifs in rna secondary structure. *Bioinformatics*, 28:i207–i214, 2013.

E. Rivas, R. Lang, and S. R. Eddy. A range of complex probabilistic models for RNA secondary structure prediction that include the nearest neighbor model and more. *RNA*, 18:193–212, 2012.

Neil Robertson and Paul D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

Donald J. Rose. On simple characterizations of k-trees. *Discrete Mathematics*, 7(3-4):317âĂŞ322, 1974.

Ambrish Roy and Yang Zhang. Protein structure prediction. *Wiley. www.els.net*, 2012.

Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjolander, R. C. Underwood, and D. Haussler. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research*, 22:5112–5120, 1994.

A. Salomaa. *Jewels of Formal Language Theory.* Computer Science Press, 1981.

I. A. Sánchez, J. M. Benedi, and D. Linares. Performance of a scfg-based language model with training data sets of increasing size. In *Proceedings of Conference on Pattern Recognition and Image Analysis*, pages 586–594, 2005.

M. Sarver, Stombaugh J. Zirbel, C.L., A. Mokdad, and N.B. Leontis. Fr3d: Finding local and composite recurrent structural motifs in rna 3d structures. *Journal of Mathematical Biology*, 56: 215–252, 2008.

D. B. Searls. The computational linguistics of biological sequences. *Artificial Intelligence and Molecular Biology*, pages 47–120, 1993.

D. B. Searls. Molecules, languages and automata. In *Proceedings of Conference on Grammatical Inference: Theoretical Results and Applications, Lecture Notes in Computer Science Volume 6339Proceedings of Conference on Grammatical Inference: Theoretical Results and Applications*, pages 5–10, 2010.

S. Sergio Caracciolo, G. Masbaum, A.D. Sokal, and A. Sportiello. A randomized polynomial-time algorithm for the spanning hypertree problem on 3-uniform hypergraphs. *CoRR abs/0812.3593*, 2008.

Pooya Shareghi, Yingfeng Wang, Russell Malmberg, and Liming Cai. Simultaneous prediction of rna secondary structure and helix coaxial stacking. *BMC Genomics*, 13(3):S7, 2012.

S. Sharma, F. Ding, and N.V. Dokholyan. ifoldrna: Three-dimensional rna structure prediction and folding. *Bioinformatics*, 24:1951–1952, 2008.

Y. Song, C. Liu, X. Huang, R. Malmberg, Y. Xu, and L. Cai. Efficient parameterized algorithms for biopolymer structure-sequence alignment. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):423–431, 2006.

Yinglei Song, Chunmei Liu, Xiuzhen Huang, Russell L. Malmberg, Ying Xu, and Liming Cai. Efficient parameterized algorithm for biopolymer structure-sequence alignment. In *Proceedings of Workshop on Algorithms for Bioinformatics*, pages 376–388, 2005.

Peter Spirtes, Clark N. Glymour, and Richard Scheines. Causality from probability. *Technical Report*, 1989.

Nathan Srebro. Maximum likelihood bounded tree-width markov networks. In *Artificial Intelligence*, pages 504–511, 2001.

j. Stombaugh, C.L. Zirbel, E. Westhof, and N.B. Leontis. Frequency and isostericity of rna base pairs. *Nucleic Acids Research*, 37(7):2294–2312, 2009.

J. Suzuki. Learning bayesian belief networks based on the mdl principle: An efficient algorithm using the branch and bound technique. *IEICE TRANSACTIONS on Information and Systems*, 82(2):356–367, 1999.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. *CoRR abs/1207.1429*, 2012.

K. Tomczak, P. Czerwinska, and M. Wiznerowicz. The cancer genome atlas (tcga): an immeasurable source of knowledge. *Contemp Oncol (Pozn)*, 19(1A):A68–77, 2015.

S.V. Torti and F.M. Torti. Annals of mathematical statistics. *Nat Rev Cancer*, 13(5):342–355, 2013.

S. Toyokuni. Role of iron in carcinogenesis: cancer as a ferrotoxic disease. *Cancer Sci*, 100(1):9–16, 2009.

Y. Uemura, A. Hasegawa, S. Kobayashi, and T. Yokomori. Tree adjoining grammars for RNA structure prediction. *Theoretical Computer Science*, 210:277–303, 1999.

J. Van Leeuwen. *Graph algorithms*. Handbook of Theoretical Computer Science, A: Algorithms and Complexity theory, North Halland, 1990.

K. Vijay-Shanker and D. Weir. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27(6):511–546, 1994.

C. J. Waters and B. A. MacDonald. Efficient word-graph parsing and search with a stochastic context-free grammar. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 311–318, 1997.

J. Xu and B. Berger. Fast and accurate algorithms for protein side-chain packing. *Journal of the ACM*, 53(4):533–557, 2006.

Jinbo Xu. Rapid protein side-chain packing via tree decomposition. In *Research in Computational Molecular Biology, Lecture Notes in Computer Science*, pages 423–439. Springer, 2005.

Jinbo Xu and Bonnie Berger. A tree-decomposition approach to protein structure prediction. In *Proceedings of 4th International IEEE Computer Society Computational Systems Bioinformatics Conference (CSB 2005)*, pages 247–256. IEEE Computer Society, 2005.

Y. Xu, J. Cui, and D. Puett. *Cancer bioinformatics*. Springer, 2014.

Ying Xu, Zhijie Liu, Liming Cai, and Dong Xu. Protein structure prediction by protein threading. *Computational Methods for Protein Structure Prediction and Modeling*, pages 1–42, 2007.

Zhang Y. Progress and challenges in protein structure prediction. *Current Opinions in Structural Biology*, 18(3):342–348, 2008.

C. Yuan and B. Malone. Learning optimal bayesian networks: A shortest path perspective. *J. Artificial Intelligence Research*, 48:23–65, 2013.

Weinberg Z. and Ruzzo L. Faster genome annotation of non-coding RNA families without loss of accuracy. In *Proceedings of Conference on Research in Computational Molecular Biology (RECOMB 2004)*, pages 243–251, 2004.

Yang Zhang. Progress and challenges in protein structure prediction. *Current Opinions in Structural Biology*, 18(3):342–348, 2008.

M. Zimand. The complexity of the optimal spanning hypertree problem. *Technical Report, University of Rochester. Computer Science Department*, 2004.

CL. Zirbel. Fr3d list of base-phosphate and base-ribose interactions in 1ehz. *http://rna.bgsu.edu/FR3D/ AnalyzedStructures/1EHZ/1EHZ_base_phosphate.html*, 2011.

C.L. Zirbel, Judit E., J.E. Sponer, J. Sponer, J. Stombaugh, and N.B Leontis. Classification and energetics of the base-phosphate interactions in rna. *Nucleic Acids Research*, 37(15):4898–4918, 2009.