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

Liang Ding[1], Abdul Samad[1], Xingran Xue[1], Xiuzhen Huang[4], Russell L. Malmberg[2,3], and Liming Cai[1,2,⋆]

[1] Department of Computer Science
[2] Institute of Bioinformatics
[3] Department of Plant Biology
University of Georgia, GA 30602, USA
{lding,samad,xrxue,russell,cai}@uga.edu
[4] Dept. of Computer Science, Arkansas State University
Jonesboro, AR 72467, USA
xhuang@astate.edu

Abstract. Stochastic context-free grammar (SCFG) has been successful in modeling biomolecular structures, typically RNA secondary structure, for statistical analysis and structure prediction. Context-free grammar rules specify parallel and nested co-occurren-ces of terminals, and thus are ideal for modeling nucleotide canonical base pairs that constitute the RNA secondary structure. Stochastic grammars have been sought, which may adequately model biomolecular tertiary structures that are beyond context-free. Some of the existing linguistic grammars, developed mostly for natural language processing, appear insufficient to account for crossing relationships incurred by distant interactions of bio-residues, while others are overly powerful and cause excessive computational complexity.

This paper introduces a novel stochastic grammar, 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. The new grammar offers a viable approach to modeling context-sensitive interactions between bioresidues because such relationships are often constrained by k-trees, for small values of k, as demonstrated by earlier investigations. In this paper it is shown, for the first time, that probabilistic analysis of k-trees over strings are computable in polynomial time $n^{O(k)}$. Hence, SkTG permits not only modeling of biomolecular tertiary structures but also efficient analysis and prediction of such structures.

Keywords: stochastic grammar, context-sensitive language, k-tree, dynamic programming, biomolecule, RNA tertiary structure.

1 Introduction

Stochastic formal language systems, typically the stochastic context-free grammar (SCFG), have been significantly valuable to various applications. Such a

⋆ Correspondent author.

A.-H. Dediu et al. (Eds.): LATA 2014, LNCS 8370, pp. 308–322, 2014.
© Springer International Publishing Switzerland 2014

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 [26,9]. 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 [28,8]. 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 [18,15,35,16,27,2], SCFG has also been extensively adopted for statistical analysis of biomolecular structures [25,8,4,5,29]. 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 [21,39,17,1,24]. 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 [19,9]. Previous work in more constrained languages has studied mildly context-sensitive grammars, typically the Tree-Adjoining Grammar [13] and its equivalent variants [14,34], 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 [33,29,4]; 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 paper, 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 graph that does not contain cliques

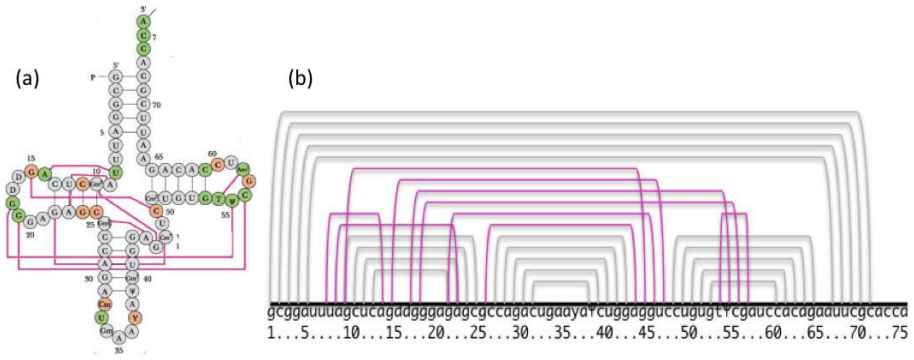

Fig. 1. A single RNA molecule can fold back on itself to form secondary and tertiary structures through bio-residue interactions. (a) The secondary structure of tRNA (Phe of yeast, PDB id: 1EHZ)) consists of parallel and nested canonical base parings (gray, lighter connections) between nucleotides, which 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.

of size more than $k+1$ as a graph minor [23,3]. 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 $k \leq 4$ [36,31,37,11,10]. 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 [32,40,30]. However, with the linear chain of vertices constrained on k-trees, we are able to show, for 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-tree structures 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 paper, we will also discuss the application in the prediction of biomolecular tertiary structures that has motivated this work.

2 k-Trees and the k-Tree Grammar

Definition 1. [23] Let integer $k \geq 1$. The class of k-trees are defined with the following inductive steps:

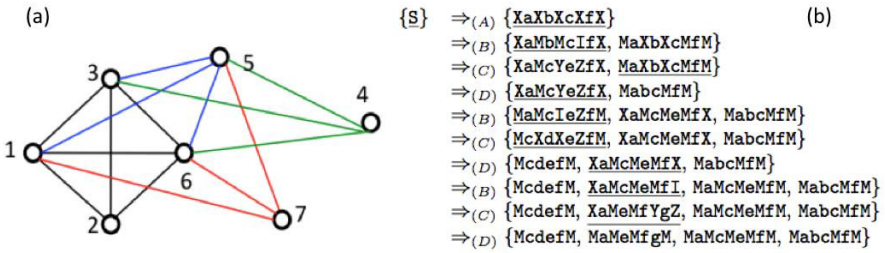

Fig. 2. (a) A generation of a 3-tree of 7 vertices by Definition 1. (b) A derivation of string `abcdefg` with 3-tree grammar rules introduced in Definition 3, with the types of applied grammar rules shown and the LHS of every applied rule underscored. The derivation also results in an induced 3-tree, the same graph shown in (a).

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 2 (a) shows of a 3-tree with seven vertices. By Definition 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.

2.1 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 $X_0 a_1 X_1 \cdots a_m X_m$ 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 2. Let $\alpha = X_0 a_1 X_1 \cdots a_m X_m$ be an m-alternating string for some $m \geq 0$. Let ω be the substring $X_i a_{i+1} \cdots X_j$ 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, $\alpha|_{Y_i}^{\forall i X_i}$ is the string obtained from α by replacing X_i with Y_i for every i

Definition 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 $\forall j = 0, \cdots, k+1$, $X_j \in \mathcal{N}$.)

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

(B) $\alpha \to \{\beta, \gamma\}$, 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 $Z_i = X_i$.

(C) $\alpha \to \{\beta\}$, 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 $a \in \Sigma$.

(D) $\alpha \to \{\beta\}$, 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 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 5. *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 6. 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.

2.2 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 7. 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 8. 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

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

Lemma 9. *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 8. 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 9, $G_{\mathcal{T}}$ is a k-tree of n vertices faithful to s. According to Definition 7, 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 10. *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 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 11. *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 12. *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. Each rule $\{S\} \rightarrow \{\beta\}$ 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 $\alpha \rightarrow \{\beta, \gamma\}$ of type (B) and each rule $\beta \rightarrow \rho$ 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 11, 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 Probability Computation with k-Tree Grammars

3.1 Stochastic k-Tree Grammars

Definition 13. 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 14. 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 15. 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 16. *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 17. 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 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 $1 \leq i \leq n$, be a given terminal string.

Definition 18. Let $\alpha = X_0 a_1 X_1 \cdots a_{k+1} X_{k+1} \in (\Sigma \cup \mathcal{N})^+$ be a symbolic string and $\kappa = (l_1, l_2, \cdots, l_{k+1})$ be $k+1$ ordered integers where $1 \leq l_1 < l_2, \cdots, l_{k+1} \leq n$. (α, κ) is *a consistent pair* if

(1) $a_i = s_{l_i}$, $1 \leq i \leq k+1$, 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 19. *$Maxp(s|\Gamma,\theta) = \max_{\kappa\in[n]^{k+1}} f(S,\kappa)$, 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.

4 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.

4.1 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 [22,20]. One of the most desirable computational biology tasks is to predict the tertiary structure from the sequence information only [12,38]. 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 nontrivial and may often be based on experience. Equipping a 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).

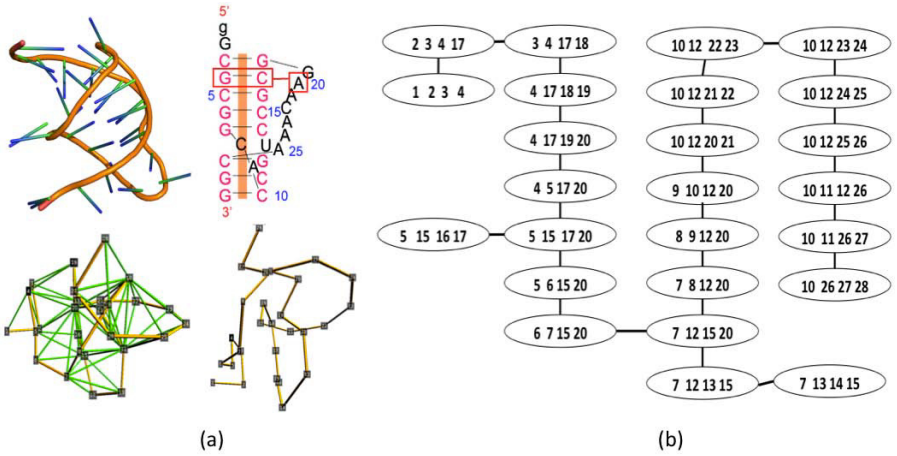

Fig. 3. Illustration of a tertiary structure prediction from BWYV (beet western yellows virus) RNA molecule sequence (PDB ID: 1L2X) that contains 28 nucleotides, coaxial helices connected with two loops, and an A-minor motif. Top of (a): Tertiary structure (drawn via `pymol`) and details of nucleotide 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.

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 illustrates this approach used in a tertiary structure prediction for a small RNA molecule.

4.2 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 [6], 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 [7] 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 parameter (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 paper. 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 [14]) for constrained context-sensitive languages.

Acknowledgments. This work was supported in part by NSF IIS grant (award No: 0916250).

References

1. Achawanantakun, R., Takyar, S., Sun, Y.: Grammar string: A novel ncRNA secondary structure representation. lifesciences society org, pp. 2–13 (2010)
2. Rozenknop, A.: Gibbsian context-free grammar for parsing. In: Sojka, P., Kopeček, I., Pala, K. (eds.) TSD 2002. LNCS (LNAI), vol. 2448, pp. 49–56. Springer, Heidelberg (2002)
3. Arnborg, S., Proskurowski, A.: Linear time algorithms for np-hard problems restricted to partial k-trees. Discrete Applied Mathematics 23(1), 11–24 (1989)
4. Chiang, D., Joshi, A.K., Searls, D.B.: Grammatical representations of macromolecular structure. Journal of Computational Biology 13(5), 1077–1100 (2006)
5. Dill, K.A., Lucas, A., Hockenmaier, J., Huang, L., Chiang, D., Josh, A.K.: Computational linguistics: A new tool for exploring biopolymer structures and statistical mechanics. Polymer 48, 4289–4300 (2007)
6. Ding, L., Samad, A., Li, G., Robinson, R., Xue, X., Malmberg, R., Cai, L.: Finding maximum spanning k-trees on backbone graphs in polynomial time (2013) (manuscript)

7. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer (1999)
8. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press (1998)
9. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (2007)
10. Huang, Z., Mohebbi, M., Malmberg, R., Cai, L.: RNAv: Non-coding RNA secondary structure variation search via graph homomorphism. In: Proceedings of Computational Systems Bioinformatics Conference (CSB 2010), vol. 9, pp. 56–69 (2010)
11. Huang, Z., Wu, Y., Robertson, J., Feng, L., Malmberg, R., Cai, L.: Fast and accurate search for non-coding RNA pseudoknot structures in genomes. Bioinforamtics 24(20), 2281–2287 (2008)
12. Thiim, J.F.I.M., Mardia, M., Ferkinghoff-Borg, K., Hamelryck, J.,, T.: A probabilistic model of RNA conformational space. PLoS Comput. Biol. 5(6) (2009)
13. Joshi, A.: How much context-sensitivity is necessary for characterizing structural descriptions. In: Dowty, D., Karttunen, L., Zwicky, A. (eds.) Natural Language Processing: Theoretical, Computational, and Psychological Perspectives, pp. 206–250. Cambridge University Press, NY (1985)
14. Joshi, A., Vijay-Shanker, K., Weir, D.: The convergence of mildly context-sensitive grammar formalisms. Issues in Natural Language Processing, pp. 31–81. MIT Press, Cambridge (1991)
15. Jurafsky, D., Wooters, C., Segal, J., Stolcke, A., Fosler, E., Tajchaman, G., Morgan, N.: Using a stochastic context-free grammar as a language model for speech recognition. In: Proceedings of International Conference on Acoustics, Speech and Signal Processing, pp. 189–192 (1995)
16. Klein, D., Manning, C.: Accurate unlexicalized parsing. In: Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423–430 (2003)
17. Knudsen, B., Hein, J.: Pfold: RNA secondary structure prediction using stochastic context-free grammars. Nucleic Acids Res. 31, 3423–3428 (2003)
18. Lari, K., Young, S.J.: The estimation of stochastic context-free grammars using the inside-outside algorithm. Computer Speech and Language 4, 35–56 (1990)
19. Martin, D., Sigal, R., Weyuker, E.J.: Computability, complexity, and languages: Fundamentals of theoretical computer science, 2nd edn. Morgan Kaufmann (1994)
20. Murzin, A.G., Brenner, S., Hubbard, T., Chothia, C.: Scop: A structural classification of proteins database for the investigation of sequences and structures. Journal of Molecular Biology 247(4), 536–540 (1995)
21. Nawrocki, E.P., Kolbe, D.L., Eddy, S.R.: Infernal 1.0: Inference of RNA alignments. Bioinformatics 25, 1335–1337 (2009)
22. Noller, H.F.: Structure of ribosomal RNA. Annual Review of Biochemistry 53, 119–162 (1984)
23. Patil, H.P.: On the structure of k-trees. Journal of Combinatorics, Information and System Sciences 11(2-4), 57–64 (1986)
24. Rivas, E., Lang, R., Eddy, S.R.: A range of complex probabilistic models for RNA secondary structure prediction that include the nearest neighbor model and more. RNA 18, 193–212 (2012)
25. Sakakibara, Y., Brown, M., Hughey, R., Mian, I.S., Sjolander, K., Underwood, R.C., Haussler, D.: Stochastic context-free grammars for tRNA modeling. Nucleic Acids Research 22, 5112–5120 (1994)
26. Salomaa, A.: Jewels of Formal Language Theory. Computer Science Press (1981)

27. Sánchez, I.A., Benedi, J.M., Linares, D.: Performance of a scfg-based language model with training data sets of increasing size. In: Proceedings of Conference on Pattern Recognition and Image Analysis, pp. 586–594 (2005)
28. Searls, D.B.: The computational linguistics of biological sequences. Artificial Intelligence and Molecular Biology, pp. 47–120 (1993)
29. Searls, D.B.: Molecules, languages and automata. In: Sempere, J.M., García, P. (eds.) ICGI 2010. LNCS, vol. 6339, pp. 5–10. Springer, Heidelberg (2010)
30. Sergio Caracciolo, S., Masbaum, G., Sokal, A., Sportiello, A.: A randomized polynomial-time algorithm for the spanning hypertree problem on 3-uniform hypergraphs. CoRR abs/0812.3593 (2008)
31. Song, Y., Liu, C., Huang, X., Malmberg, R., Xu, Y., Cai, L.: Efficient parameterized algorithms for biopolymer structure-sequence alignment. IEEE/ACM Transactions on Computational Biology and Bioinformatics 3(4), 423–431 (2006)
32. Srebro, N.: Maximum likelihood bounded tree-width Markov networks. Artificial Intelligence 143(2003), 123–138 (2003)
33. Uemura, Y., Hasegawa, A., Kobayashi, S., Yokomori, T.: Tree adjoining grammars for RNA structure prediction. Theoretical Computer Science 210, 277–303 (1999)
34. Vijay-Shanker, K., Weir, D.: The equivalence of four extensions of context-free grammars. Mathematical Systems Theory 27(6), 511–546 (1994)
35. Waters, C.J., MacDonald, B.A.: Efficient word-graph parsing and search with a stochastic context-free grammar. In: Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 311–318 (1997)
36. Xu, J., Berger, B.: Fast and accurate algorithms for protein side-chain packing. Journal of the ACM 53(4), 533–557 (2006)
37. Xu, Y., Liu, Z., Cai, L., Xu, D.: Protein structure prediction by protein threading. In: Computational Methods for Protein Structure Prediction and Modeling, pp. 389–430. Springer I&II (2006)
38. Progress, Y.Z.: challenges in protein structure prediction. Current Opinions in Structural Biology 18(3), 342–348 (2008)
39. Weinberg, Z., 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), pp. 243–251 (2004)
40. Zimand, M.: The complexity of the optimal spanning hypertree problem. Technical Report, University of Rochester. Computer Science Department (2004)