Efficient Parameterized Algorithms for Biopolymer Structure-Sequence Alignment

Yinglei Song, Chunmei Liu, Xiuzhen Huang, Russell L. Malmberg, Ying Xu, and Liming Cai

Abstract—Computational alignment of a biopolymer sequence (e.g., an RNA or a protein) to a structure is an effective approach to predict and search for the structure of new sequences. To identify the structure of remote homologs, the structure-sequence alignment has to consider not only sequence similarity, but also spatially conserved conformations caused by residue interactions and, consequently, is computationally intractable. It is difficult to cope with the inefficiency without compromising alignment accuracy, especially for structure search in genomes or large databases. This paper introduces a novel method and a parameterized algorithm for structure-sequence alignment. Both the structure and the sequence are represented as graphs, where, in general, the graph for a biopolymer structure has a naturally small tree width. The algorithm constructs an optimal alignment by finding in the sequence graph the maximum valued subgraph isomorphic to the structure graph. It has the computational time complexity $O(k^t N^2)$ for the structure of N residues and its tree decomposition of width t. Parameter k, small in nature, is determined by a statistical cutoff for the correspondence between the structure and the sequence. This paper demonstrates a successful application of the algorithm to RNA structure search used for noncoding RNA identification. An application to protein threading is also discussed.

Index Terms—Structure-sequence alignment, tree decomposition, parameterized algorithm, dynamic programming, RNA structure homology search, protein threading.

1 Introduction

Structure-sequence alignment plays a central role in a number of important computational biology methods. For instance, protein threading, an effective method to predict protein tertiary structure, is based on an alignment between the target sequence and structure templates in a template database [3], [5], [42], [20], [40]. Structure-sequence alignment is also essential to RNA structural homology search, a viable approach to annotating (and identifying new) noncoding RNAs [10], [13], [31], [24]. Structure-sequence alignment also finds applications in other bioinformatics tasks where structure plays an instrumental role, such as in the identification of the structure of intermolecular interactions [27], [29] and in the discovery of the structure of biological pathways through comparative genomics [8].

The structure-sequence alignment is to find an optimal way to "fit" the residues of a target sequence in the spatial positions of a structure template. To be able to identify the structure of remote homologs, the alignment has to consider not only sequence similarity but also spatially conserved conformations caused by sophisticated interactions between residues and, consequently, is computationally intractable. For example, it is both NP-hard for protein threading with amino acid interactions [19] and for thermodynamic determination of RNA secondary structure, including pseudoknots [25].

The alignment problem has often been formulated as integer programming that characterizes residue spatial interactions with (a large number of) linear inequality constraints [40], [22]. Commercial software packages for linear programming are usually used to approximate the integer programming and to reduce the computation time. More sophisticated techniques, such as branch-and-cut, can be used to dynamically include only needed linear constraints [22], [30]. Moreover, a divide-and-conquer method based on the notion of "open-links" has also been devised to address the residue-residue interaction issue [42]. For RNA structure-sequence alignment, dynamic programming has been extended to include crossing patterns of RNA nucleotide interactions [35], [7]. The above algorithmic techniques cope with the intractability of the structure-sequence alignment problem; however, most of them still require computation time polynomial of a high-degree.

In this paper, we introduce an efficient structure-sequence alignment algorithm. Both structure and sequence are represented as mixed graphs (containing both directed and undirected edges); the optimal alignment corresponds to finding the maximum valued (subgraph) isomorphism between the structure graph and a subgraph of the sequence graph. In addition, we introduce an integer parameter k to constrain the correspondence between the graphs. A dynamic programming algorithm is then developed over a tree decomposition of the structure graph. For each value of k, the optimal alignment can be found in time $O(k^t N^2)$ for each structure template containing N residues given a tree decomposition of tree width t for the structure graph.

- *Y. Song, C. Liu, and L. Cai are with the Department of Computer Science, 415 Boyd GSRC, University of Georgia, Athens, GA 30602. E-mail: {song, chunmei, cai}@cs.uga.edu.*
- *X. Huang is with the Department of Computer Science, Arkansas State University, State University, AR 72467. E-mail: xzhuang@csm.astate.edu.*
- *R.L. Malmberg is with the Department of Plant Biology, University of Georgia, Athens, GA 30602-7271. E-mail: russell@plantbio.uga.edu.*
- *Y. Xu is with the Department of Biochemistry and Molecular Biology, A110 Life Sciences Building, University of Georgia, 120 Green Street, Athens, GA 30602. E-mail: xyn@bmb.uga.edu.*

Manuscript received 14 Feb. 2006; revised 31 May 2006; accepted 15 June 2006; published online 31 Oct. 2006.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBBSI-0015-0206.

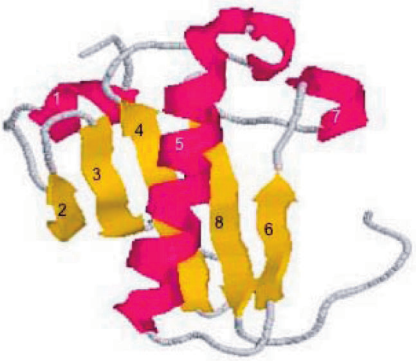


Fig. 1. Folded ChainB of Protein Kinase C interacting protein (the PDB-file corresponding to PDB-ID 1AV5) that contains eight cores.

Our algorithm is a parameterized algorithm that finds exact solutions [11] in which the naturally small parameter k determined by a statistical cutoff reflects the accuracy of the alignment. The new algorithm with the time complexity $O(k^t N^2)$ is more efficient than previous algorithms, for example, with time complexity $O(N^k)$ [42]. This is also because the tree width t of the graph for a biopolymer structure is small in nature. For example, the tree width is 2 for the graph of any pseudoknot-free RNA and the width can only increase slightly for all known pseudoknot structures. Our experiments also show that, among 3,890 protein tertiary structure templates compiled using PISCES [36], only 0.8 percent of them have tree width $t > 10$ and 92 percent have $t < 6$, when using a 7.5 Å $C_\beta - C_\beta$ distance cutoff for defining pair-wise interactions. Fig. 4 shows both percentages and cumulative percentages for tree widths in the range of 1 to 14 for these proteins.

The alignment algorithm has been applied to the development of a fast RNA structure homology search program [34]. With a significantly reduced amount of computation time, the new search method achieves the same accuracy as searches based on the widely used Covariance model (CM) [14]. The new algorithm yields about 24 to 50 times speed up for searches of pseudoknot-free RNAs with 90 to 150 nucleotides; it gains an even more significant advantage for larger RNAs or structures including pseudoknots. In addition, for all the conducted tests, including the searches of medium to large RNAs in bacteria and yeast genomes, parameter $k \leq 7$ has been sufficient for the accurate identification.

The alignment algorithm is also applicable to protein threading, one of the most successful techniques for protein tertiary structure prediction. A detailed discussion is given to show how an efficient algorithm for protein threading can be constructed based on the introduced alignment algorithm and related techniques.

2 Problem Formulation

We formulate structure-sequence alignment as a *generalized* subgraph isomorphism problem. Graphs used in this paper are *mixed* graphs contaning both undirected and directed edges. Let $V(G)$, $E(G)$, and $A(G)$ denote the vertex set, the undirected edge set, and the directed edge set of graph G, respectively.

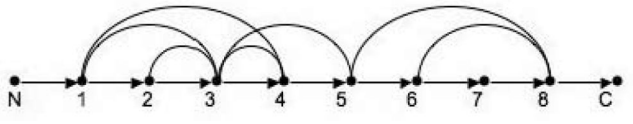


Fig. 2. The structure graph corresponding to the protein structure shown in Fig. 1, where undirected edges are spatial contact between cores and directed edges for neighboring cores from the N terminal to the C terminal.

2.1 Graph Representations for Structure and Sequence

Definition 1. *A* structural unit *in a biopolymer sequence is a stretch of contiguous residues (nucleotides or amino acids). A nonstructural stretch between two consecutive structural units is called a* loop. *A structure of the sequence is characterized by interactions among structural units. For example, structural units in a tertiary protein are α helices and β strands, called* cores. *Fig. 1 shows a protein structure with eight structural units. In RNA secondary structure, a structural unit is a stretch of nucleotides, one half of a stem formed by a stack of base pairings.*

The alignment between a structure template and a target sequence places residues of the sequence in the spatial positions of the template. Instead of placing individual residues to the spatial positions, the method we introduce in this paper allows us to put a stretch of residues as a whole in the position of some structural unit of the template.

Given a biopolymer structure template, a *structure graph*, H, can be defined such that each vertex in $V(H)$ represents a structural unit, each edge in $E(H)$ represents the interaction between two structural units, and each directed edge in $A(H)$ represents the loop ended by two consecutive structural units. Fig. 2 shows the structure graph for the folded protein in Fig. 1. Fig. 7 shows the structure graph for bacterial tmRNAs. Note that, for any structure graph of a biopolymer, the set of directed edges forms a directed path containing all the vertices in the graph.

The target sequence to be aligned to the structure is first preprocessed as follows: For every structure unit $u \in V(H)$ in the structure template graph H, we identify all *candidates* $\mathcal{C}(u)$ in the sequence so that each candidate $v \in \mathcal{C}(u)$ "matches well" structural unit u. Criteria for identifying the candidates are highly application-specific. For example, for RNA structure-sequence alignment, each structural unit u is either the left or right half of some stem template s. We find all stems in the sequence which can align well with the stem template s. If u is the left half of s, its candidates are the left halves of those found stems; otherwise, its candidates are the right halves of the found stems.

For example, in our application in RNA secondary structure modeling, each stem template will be modeled with the Covariance Model (CM) (i.e., a type of stochastic context-free grammar model [14]) without bifurcation rules. Alignment between the model and a candidate in the target sequence can be done with a simplified version of the dynamic programming Inside algorithm [14], which is to compute the full probability of the candidate to be generated by the model. Such an alignment should produce two halves

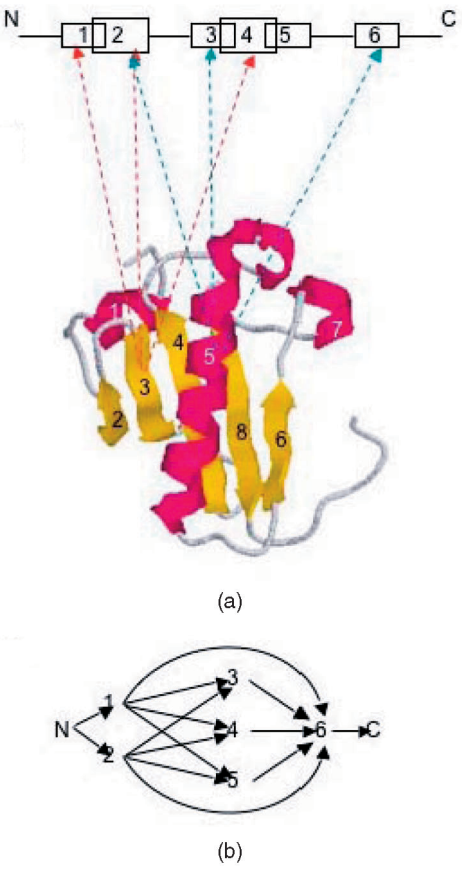


Fig. 3. (a) Six candidates found after preprocessing a protein sequence with cores in the protein structure given in Fig. 1. Core 3 has three candidates, 1, 2, and 4; core 5 has three candidates, 2, 3, and 6. Among these candidates, 1 and 2 overlap in their sequence positions and 3, 4, and 5 also overlap. (b) The corresponding sequence graph (interacting edges are not shown and directed edges incident on the N- or C-terminal end are not shown).

of a candidate stem in the target sequence. We refer the reader to the book by Durbin et al. [12] and to the application section of this paper for more detailed discussions on how such fine-grain stem-sequence alignment is done.

A *sequence graph* G can be constructed from the target sequence by representing each candidate as a vertex. In this graph, each edge in $E(G)$ connects a pair of candidates that may possibly interact but *do not overlap* in their sequence positions and a directed edge in $A(G)$ connects two candidates that do not overlap. Fig. 3a shows a simple example containing six candidates identified on a protein sequence after the preprocessing with the cores in the protein structure given in Fig. 1. In particular, core 3, a β-strand, has three candidates, numbered 1, 2, and 4, found on the target sequence; core 5, an α helix, has three candidates, numbered 2, 3, and 6, found on the target sequence. Some of these candidates overlap in their sequence positions. For example, candidates 1 and 2 overlap in their sequence positions. A part of the constructed sequence graph is given in Fig. 3b.

2.2 Alignment as Subgraph Isomorphism

Based on the graph representations, the structure-sequence alignment problem can be formulated as the problem of finding, in the sequence graph G, a subgraph isomorphic to the structure graph H such that the objective function based on the isomorphism achieves the optimum. For this, we first introduce a mechanism to parameterize (and to scrutinize) the mapping between H and G.

Definition 2. *A* map scheme M *between graphs* H *and* G *is a function:* $V(H) \rightarrow 2^{V(G)}$ *that maps every vertex in* H *to a subset of vertices in* G*. The maximum size of such a subset,* $k = \max_{v \in V(H)}\{|M(v)|\}$*, is called the* map width *of the map scheme.*

A map scheme can be obtained in the preprocessing step that finds all candidates of every structural unit. The qualifications of these candidates can usually be quantified by a statistical cutoff of the degree to which a candidate is aligned to a structural unit. One may simply choose the top k candidates for each structural unit which have the highest alignment scores with the structural unit. More sophisticated map schemes are possible (see Section 4) in which, ideally, the parameter k reflects the accuracy of alignment results.

We now define in the following the parameterized problem:

GENERALIZED SUBGRAPH ISOMORPHISM:
INPUT: mixed graphs H, G, and map scheme M of width k;
OUTPUT: a subgraph G' of G and an isomorphic mapping

$$f : V(H) \rightarrow V(G'),$$

constrained by $f(x) \in M(x)$ for every $x \in V(H)$ such that the objective function $score(f) =$

$$\sum_{u \in V(H)} S_1(u, f(u)) + \sum_{(u,v) \in E(H)} S_2((u,v),(f(u),f(v))) + \sum_{\langle u,v \rangle \in A(H)} S_3(\langle u,v \rangle, \langle f(u), f(v) \rangle) \quad (1)$$

achieves the optimum (i.e., maximum or minimum).

Functions S_1, S_2, and S_3 are application-specific, defining, respectively, three different alignment scores between the structure template and the target sequence. In particular, $S_1(u, f(u))$ defines the alignment score between a structural unit u and its image $f(u)$. $S_2((u,v),(f(u),f(v)))$ is the alignment score between the interaction (u,v) of two structural units u and v and the interaction $(f(u),f(v))$ of the corresponding images $f(u)$ and $f(v)$. $S_3(\langle u,v \rangle, \langle f(u),f(v) \rangle)$ represents the alignment score between the loop $\langle u,v \rangle$ (connecting two neighboring structural units u and v) and its correspondence loop $\langle f(u),f(v) \rangle$ in the sequence.

For example, for our application in RNA secondary structure search, $S_2((u,v),(f(u),f(v)))$ is defined as the alignment score between a stem, formed by the pairing regions u, v, and two candidate pairing regions, $f(u)$, $f(v)$, in the target sequence. The alignment is actually accomplished by aligning the given CM model of stem (u,v) to the two candidate regions $(f(u),f(v))$. The alignment algorithm is to compute the full probability that the model generates the candidate regions. Similarly, $S_3(\langle u,v \rangle, \langle f(u),f(v) \rangle)$ is the full probability that the given profile HMM of loop $\langle u,v \rangle$ generates candidate region $\langle f(u),f(v) \rangle$. For RNA secondary

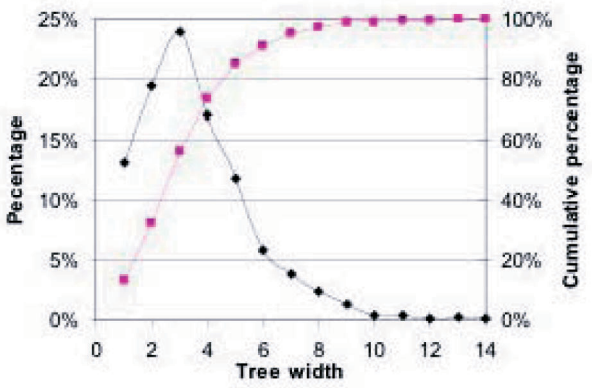


Fig. 4. The tree width distribution of the graphs for 3,890 protein structure templates compiled using PISCES [36], [37].

structure search, $S_1 = 0$. But, for protein threading, $S_1(u, f(u))$ gives the likelihood of a strand of sequence $f(u)$ matching the model u of an α helix or a β. Section 4 gives more discussions on these computations.

This problem generalizes the well-known NP-hard subgraph isomorphism decision problem. Efficient algorithms for subgraph isomorphism may be obtained on constrained instances. However, algorithms of this kind only exist for the cases where H is small, fixed, and G is planar or of a small tree width [1], [15], [26]. None of these conditions can be satisfied by the application in the structure-sequence alignment, where the structure can be large and the sequence graph is often arbitrary.

We conclude this section by noting that the parameterization introduced on the map width does not trivialize the problem under investigation. In particular, we have the following theorem:

Theorem 1. *GENERALIZED SUBGRAPH ISOMORPHISM remains NP-hard on map schemes of map width $k = 3$.*

Proof. We show that a decision version of problem GENERALIZED SUBGRAPH ISOMORPHISM is NP-hard. The decision problem is to decide, given graphs H, G, and map scheme M, if an isomorphic mapping exists between H and some subgraph of G. (Apparently, the decision version is not harder than the GENERALIZED SUBGRAPH ISOMORPHISM problem.) We prove the NP-hardness in the following by reducing problem 3-SAT to it. □

Given a Boolean formula ψ in the conjunctive normal form

$$\psi = (l_1^1, l_2^1, l_3^1) \wedge (l_1^2, l_2^2, l_3^2) \wedge \ldots \wedge (l_1^m, l_2^m, l_3^m),$$

we construct two graphs, H_ψ, G_ψ, and map scheme M_ψ as follows: H_ψ contains m vertices, $v_1, \ldots, v_m$, forming a clique. G_ψ contains $3m$ vertices, one for every literal occurrence in formula ψ in which two vertices u_i^s and u_j^t corresponding to l_i^s and l_j^t form an edge if $s \neq t$ and l_i^s, l_j^t are not complementary literals. The map scheme M_ψ is defined as $M_\psi(v_r) = \{u_1^r, u_2^r, u_3^r\}$, $r = 1, \ldots, m$.

It is not difficult to verify that formula ψ is satisfiable if and only if there is a clique subgraph in G_ψ which is isomorphic to H_ψ and the isomorphism is constrained by map scheme M_ψ.

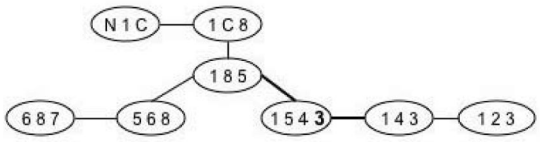


Fig. 5. A tree decomposition for the structure graph given in Fig. 2.

3 PARAMETERIZED ALIGNMENT ALGORITHM

Definition 3 [32]. *Pair (T, X) is a* tree decomposition *of an undirected graph H if*

1. *T is a tree,*
2. *$X = \{X_i | X_i \subseteq V(H), i \in V(T)\}$ and*

$$\bigcup_{X_i \in X} X_i = V(H),$$

3. *$\forall u, v, (u, v) \in E(H)$, $\exists i \in V(T)$ such that $u, v \in X_i$, and*
4. *$\forall i, j, k \in V(T)$, if k is on the path from i to j in tree T, then $X_i \cap X_j \subseteq X_k$.*

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

For the mixed graphs that we used to model biopolymer structures, their tree decompositions can be obtained by assuming all edges are undirected.

Tree decomposition presents a topological view on a graph and tree width tells how much the graph is tree-like. Intuitively, a tree decomposition groups locally related vertices (i.e., those connected by edges) into bags X_is, which are arranged into a tree topology. Fig. 5 illustrates a tree decomposition for the protein structure graph given in Fig. 2. In this tree decomposition, there are eight bags, each contains up to four vertices from the original structure graph. All vertices in the graph are contained in the bags, satisfying condition 2. It satisfies condition 3 since, for every undirected and directed edge, both end-points of the edge are contained in some bag. It can be seen that the tree decomposition also satisfies condition 4. Condition 4 in some sense is critical. It requires that bags containing the same vertex should form a connected subgraph tree; this allows efficient search of the graph via the tree topology. The smaller the tree width is, the more efficient the search will become.

In general, biopolymer structure graphs have small tree width. For instance, the tree width of the structure graphs for pseudoknot-free RNAs is 2 and it can only increase slightly for all known pseudoknots. The tree decomposition given in Fig. 5, which is for the pseudoknot structure in Fig. 2, has tree width 3, the maximum bag size -1. Fig. 4 gives statistics on the tree width of about 3,890 protein structure templates compiled using PISCES [36], [37].

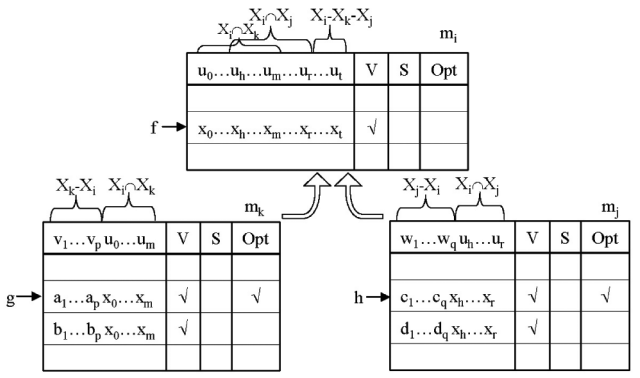


Fig. 6. Computing dynamic programming tables over a tree decomposition in which tree node i has two children k and j.

3.1 Parameterized Algorithm for Subgraph Isomorphism

We now describe a tree decomposition-based parameterized algorithm for the problem GENERALIZED SUBGRAPH ISOMORPHISM formulated in Section 2. Our algorithm assumes a given tree decomposition (T, X) of width t for structure graph H. Our algorithm follows the basic idea of the tree decomposition-based techniques in [1], [2].

To simplify our discussion, we assume that T for the tree decomposition is a binary tree. The following notations will also be useful: Let $U \subseteq V(H)$ and $Y \subseteq V(G)$ such that $|U| = |Y|$. Then, a mapping $f : U \to Y$ is a *valid mapping for* U if f is a subgraph isomorphism between the graph induced by U and the graph induced by Y. If $W \subseteq U$, then $f|_W$ is *f projected onto W*, therefore a valid mapping for W. A *partial isomorphism for H with respect to X_i* is a valid mapping f for $U = X_i \cup \bigcup_{k \in D(i)} X_k$, where $D(i)$ is the set of i's descendent nodes in the tree.

In a bottom up fashion, the algorithm establishes one table for each tree node. Let $X_i = \{u_0, u_1, \ldots, u_t\}$. Table m_i for tree node i consists of $|X_i| + 3$ columns, one for every vertex in X_i. Rows are all possible mappings for X_i restricted by the map scheme M; each row is of the form $\langle x_0, x_1, \ldots, x_t \rangle$, representing the mapping f, $f(u_l) = x_l$, $l = 0, 1, \ldots, t$. There are three additional columns in the table: V, S, and Opt (see Fig. 6). $V(f) =$ '$\surd$' if and only if mapping f is valid for X_i. $S(f)$ is the optimal score over all the partial isomorphism e for H with respect to X_i such that $f = e|_{X_i}$. $Opt(f)$ indicates whether $S(f)$ is the optimal overall valid mapping f' for X_i, where $f'|_{X_i \cap X_p} = f|_{X_i \cap X_p}$ for p, the parent node of i.

If i is a leaf node, the score $S(f)$ is simply the value computed based on (1) (see Section 2) for vertices in X_i only. If i is an internal node with children nodes k and j, $S(f)$ is the sum of the following three value:

1. the value computed for f with (1) for vertices in X_i only,
2. the maximum S value over all valid mappings g in table m_k such that $g|_{X_i \cap X_k} = f|_{X_i \cap X_k}$, and
3. the maximum S value over all valid mappings h in table m_j such that $h|_{X_i \cap X_j} = f|_{X_i \cap X_j}$.

Fig. 6 illustrates the computation for row f in table m_i of the internal node i that has two children nodes k and j. The formal algorithm, called GSGI, is outlined as a recursive process in the following. The algorithm assumes the input of a tree decomposition (T, X) and a map scheme M; it returns table m_i for every node i in T.

ALGORITHM GSGI (T, X_i, M, i, m_i)
1) If i has left child k, call GSGI(T, X_k, M, k, m_k);
2) If i has right child j, call GSGI(T, X_j, M, j, m_j);
3) For every mapping f for X_i, constrained by M, do
   a) If i has left child k in T, find in m_k a valid mapping g such that $g|_{X_i \cap X_k} = f|_{X_i \cap X_k}$ with $Opt(g)$ being '$\surd$';
   b) If i has right child j in T, find in m_j a valid mapping h, such that $h|_{X_i \cap X_j} = f|_{X_i \cap X_j}$ with $Opt(h)$ being '$\surd$';
   c) Compute score $score(f)$ with (1) for X_i only;
   d) Let $S(f) = score(f) + S(g) + S(h)$;
   e) If i has parent p in T and $S(f)$ maximizes over all f' with $f'|_{X_i \cap X_p} = f|_{X_i \cap X_p}$, then let $Opt(f) =$ '$\surd$';
4) Return (m_i);

The optimal score computed in the table for the root of the tree T is the best isomorphism score. A recursive routine can be used to trace back the corresponding optimal isomorphism. Because the trace back process is a typical one for dynamic programming, its details are omitted here.

To ensure that the score computed in the table for the root of the tree T is the best isomorphism score, we need to prove that the (bottom up) dynamic programming used by algorithm $GSGI$ always produces correct partial isomorphisms.

Theorem 2. *GSGI correctly solves the GENERALIZED SUBGRAPH ISOMORPHISM problem for every given tree decomposition and every given map scheme.*

Proof. Algorithm GSGI works in the bottom up fashion on the tree decomposition to build a dynamic programming table for each tree bag it encounters. Since the algorithm validates the isomorphism for locally involved vertices, it suffices to verify that the local solutions found by the algorithm for the current tree bag do not conflict with solutions constructed earlier (i.e., from the children of the current tree bag). That is, it suffices to show that, for every $u \in X_i$, the valid mapping $f(u) = x$ selected by the algorithm for some $x \in M(u)$ is consistent with any earlier valid mapping. We need to show that any valid earlier mapping should not contain mapping $f(v) = x$, for any vertex $v \in X_k$ and $v \neq u$, where k is any descendent of i. Interestingly enough, for mixed graphs H constructed from biopolymer structures, "consistency" can be guaranteed. The following is a justification for this claim. □

We define the following partial order relation $(V(H), \preceq)$ for graph H: Relationship $v \preceq u$ holds in H if

1. either $\langle u, v \rangle \in A(H)$ or
2. $\exists w, v \preceq w$ and $\langle u, w \rangle \in A(H)$.

Note that the partial order relation in graph H is a total order relation. The partial order relation for graph G can be defined similarly.

Assume that $v \preceq u$ in H (the case of $u \preceq v$ is similiar). We now prove the "consistency" property that any valid mapping f selected by the algorithm satisfies $f(v) \preceq f(u)$ in graph G. We do this by induction on how directed edges are involved in establishing the relationship $v \preceq u$.

1. If $\langle u, v\rangle \in A(H)$, then u, v belongs to the same bag and the mapping is directly validated by the algorithm, i.e., $\langle f(u), f(v)\rangle \in A(G)$, which implies $f(v) \preceq f(u)$.
2. If $\langle u, w\rangle \in A(H)$ and $v \preceq w$, we assume the "consistency" property for the relationship $v \preceq w : f(v) \preceq f(w)$ in graph G. Since $\langle u, w\rangle \in A(H)$, u, w belongs to the same bag; the mapping f should satisfy $\langle f(w), f(v)\rangle \in A(G)$, which is directly validated by the algorithm. According to the second rule for $\preceq$, we have $f(v) \preceq f(u)$ in graph G.

So, for any two vertices $u, v \in V(H)$, $u \neq v$, and any mapping f selected by the algorithm, $f(u) \neq f(v)$. Because the relation $\preceq$ in H is a total order, any mapping selected by the algorithm when it reaches the apex of the tree decomposition is a correct mapping. Also, since the algorithm searches the tree via dynamic programming, the produced result is the optimal.

Corollary 3. *Parameterized algorithm GSGI computes the optimal structure-sequence alignment for every given tree decomposition and every given map scheme.*

3.2 Approximating Tree Decomposition

Before the algorithm GSGI is applied, a tree decomposition needs to be found for the structure graph H. There exist theoretical algorithms [4] that, given a graph with tree width bounded by t, can find an optimal tree decomposition in time $O(c^t n)$. However, the constant c is so formidably large that such algorithms are too slow to be practically useful. For this work, faster heuristic or approximation algorithms can be used because the tree width of tree decompositions for the structure graph H will only affect the running time of the algorithm but not the accuracy of the alignment result. In the following, we introduce a simple greedy algorithm for tree decomposition that practically runs fast on structure graphs.

Given a structure graph H, undirected edges are selected such that removals of these edges from the graph result in an outerplanar graph. The removals of these edges are done by first removing an edge (but not the endpoints) that *crosses* with the maximum number of other edges and then repeating the same process until the resulting graph contains no crossing edges. Note that two edges (u, v) and (u', v') in H *cross* each other if either $v' \preceq v \preceq u' \preceq u$ or $v \preceq v' \preceq u \preceq u'$ (see Section 3.1 for the definition of the partial order $(V(H), \preceq)$.

A simple recursive process can find a tree decomposition of tree width 2 for the remaining outerplanar graph. Then, for each removed edge (u, v), in the tree we place v in every node on the (shortest) path from a node containing v to a node containing u. For example, Fig. 5 shows a tree decomposition for the structure graph given in Fig. 2. This tree decompositon is obtained by first removing crossing edge $(3, 5)$ in the graph (see Fig. 2). Then, a tree decomposition for the remaining outerplanar graph is built which is extended to the tree decomposition for the original graph by placing vertex 3 (in the bold font) in node {1, 5, 4} on the path from node $\{1, 4, 3\}$ to node $\{1, 8, 5\}$. This strategy produces a tree decomposition of size at most $2 + c$ if there are c crossing edges removed. In reality, the obtained tree decomposition has a much smaller tree width. For example, for the structure graph constructed from the bacterial *tmRNA* structure shown in Fig. 7, our greedy algorithm will yield a tree decomposition of tree width 4 instead of 9. This algorithm is of linear time $O(|E(H) + |A(H)| + |V(H)|)$.

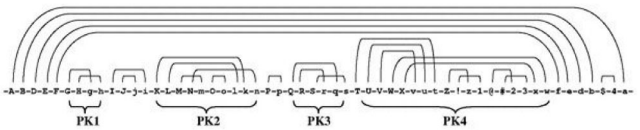


Fig. 7. Diagram of the pairing regions on the tmRNA gene. Upper case letters indicate base sequences that pair with the corresponding lower case letters. The four pseudoknots constitute the central part of the tmRNA gene and are called Pk1, Pk2, Pk3, and Pk4, respectively.

3.3 Computational Time Complexity

Let N be the length of the target sequence and n be the size $|V(H)|$ of the structure graph. Given a tree decomposition of tree width t for structure graph H, a mapping scheme of width k, the running time for algorithm GSGI is $O(k^t t^2 n)$. For each row in the table, the compliance with subgraph isomorphism needs to be validated, which takes $O(t^2)$ time, and a score is computed according to (1) (by looking up precomputed values of functions S_1, S_2, and S_3), which takes $O(t^2 + t \log_2 k)$ (note that the rows of a table can be ordered to facilitate binary search by the computation for its parent node).

It takes $O(knN)$ time to preprocess the target sequence of length N to construct the sequence graph. Simultaneously, this step precomputes the values of functions S_1 and S_2. The values of function S_3 can then be precomputed, in time $O(k \sum_{i=1}^{l} l_i^2) = O(knN)$, where l_i is the length of the ith loop and l is the number of loops in the structure. Summing up the times needed by the preprocessing, tree decomposition, and algorithm GISI, we have a loose upper bound $O(k^t nN)$, or $O(k^t N^2)$, for the total time for the structure-sequence alignment.

4 Application in RNA Structure Search

To evaluate the performance of our method and algorithm for structure-sequence alignment, we have applied them to the development of a fast program that can search for RNA structural homologs. We have also conducted extensive tests on finding medium to large RNA secondary structures (including pseudoknots) in both random sequences and biological genomes (bacteria and yeasts) [34]. We summarize our test results in the following.

4.1 Data Preparations

The tests on RNA structure searches that we conducted can be grouped into three categories:

1. on eight RNA pseudoknot-free structures, of medium size (61-112 nucleotides), inserted in random sequences of length 10^5,
2. on six RNA pseudoknot structures, of medium size (55-170 nucleotides), inserted in random sequences of length 10^5, and
3. on three RNA pseudoknot structures, of medium to large size (61-755), in a variety of genomes of lengths ranging from 2.7×10^4 to 1.1×10^7.

Each homologous RNA family is modeled with a structure graph. Each undirected edge in the graph represents a stem that is profiled with a simplified Covariance Model (CM) [14]. Each directed edge in the graph represents a loop (5′ to 3′) that is profiled with a profile Hidden Markov Model (HMM). In the first two categories of searches, for each family, we downloaded from the Rfam database [17] 30 RNA sequences with their mutual identities below 80 percent. We used them to train the CMs and profile HMMs in the model.

The alignment between a CM model and a pair of candidate base regions is accomplished with a simplified Inside algorithm (without bifurcation) [12]. This is to compute the function S_2 in (1). The alignment between a profile HMM and a candidate region is accomplished with the Forward algorithm [12]. This is to compute the function S_3 in (1). S_1 is set to zero for RNA structure-sequence alignment. All scores are probabilities before taking the negative logarithm.

For each family, we downloaded from Rfam another 30 sequences with their mutual identities below 80 percent and used them for search. They were inserted in a random background of 10^5 nucleotides generated with the same base compositions. Using a method similar to the one used in RSEARCH [18], we computed the statistical distribution for the alignment scores with a random sequence of 3,000 nucleotides generated with the same base composition as the sequences to be searched. An alignment score with a Z-score exceeding 5.0 was reported as a hit. Both random sequences and genomes were scanned through with a window of a size correlated with the structure model size. The segment of the sequence falling within the window was aligned to the model with the structure-sequence alignment algorithm presented in the earlier sections.

For the tests of the third category, we searched for three RNA pseudoknot structures: the pseudoknot structure in the 3′ UTR in the corona virus family [16], the bacterial tmRNA structure (see Fig. 7) that contains four pseudoknots [28], and yeast telomerase RNA consisting of up to 755 nucleotides [9]. The structures for these RNAs were trained with 14, 85, and 5 available sequences, respectively. The genomes searched for the 3′ UTR pseudoknot were Bovine corona virus, Murine hepatitus virus, Porcine diarrhea virus, and Human corona virus, with the average length 3×10^4. The two searched bacteria genomes for the tmRNA were *Haemophilus influenzae* and *Neisseria meningitidis*, with the average length 2×10^6. Yeast genomes, *Saccharomyces cerevisiae* and *Saccharomyces bayanus*, of average length 11×10^6, were used to search for the telomerase RNA.

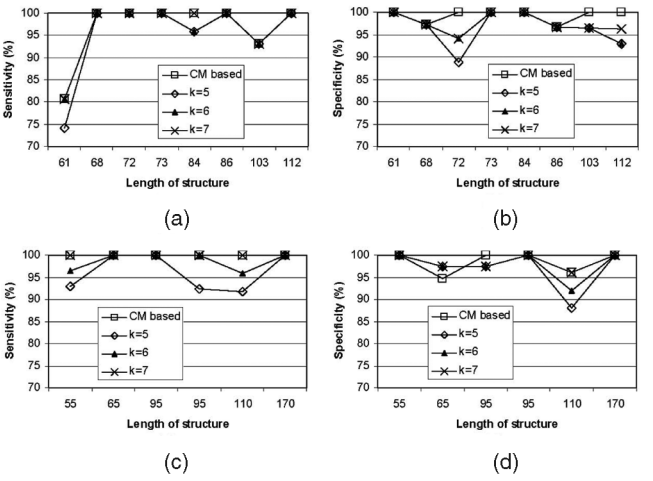


Fig. 8. Performance comparison between the tree decomposition-based method and the CM-based method on search for RNA structures (a) and (b) for pseudoknot-free structures and (c) and (d) for pseudoknots.

To obtain a reasonably small value for the parameter k, the map scheme between the structure and the sequence was designed with the constraint that candidates of a given stem were restricted in a certain region in the target sequence. For this, we assumed that, for homologous sequences, the distances from each pairing region of the given stem to the 3′ end follow a Gaussian distribution whose mean and standard deviation were computed based on the training sequences. For training sequences representing distant homologs of an RNA family, we could effectively divide data into groups so that a different but related structure model was built for each group and used for searches. This method ensures a small value for the parameter k in search models.

4.2 Performance Evaluations

We conducted the tests on the tree decomposition-based search program and on a Covariance Model (CM)-based search system[1] and compared the performances of the two. The tests results showed that, in all three categories, parameter $k = 7$ was sufficient for our new search program to achieve the same accuracy as the CM-based search system does. But, the computation time used by the new method was significantly reduced.

Fig. 8a and Fig. 8b, respectively, show the sensitivity comparison and specificity comparison between the two search methods for pseudoknot-free RNA structures. These structures were from eight RNA families: Entero_CRE, SECIS, Lin_4, Entero_OriR, Let_7, Tymo_tRNA-like, Purine, and S_box, in increasing order of their length. The tree decomposition-based algorithm performed quite well for $k = 6$ and larger values.

Fig. 8c and Fig. 8d, respectively, show the sensitivity comparison and specificity comparison between the two search methods on RNA pseudoknot structures. These were from six RNA families: Antizyme_FSE, corona_pk3,

1. We developed this CM-based system [23] in the same spirit of Brown and Wilson's work [6] that profiles pseudoknots with intersection of CMs. CM was first introduced by Eddy and Durbin [14] and has proved very accurate in profiling for search of pseudoknot-free RNA structures.

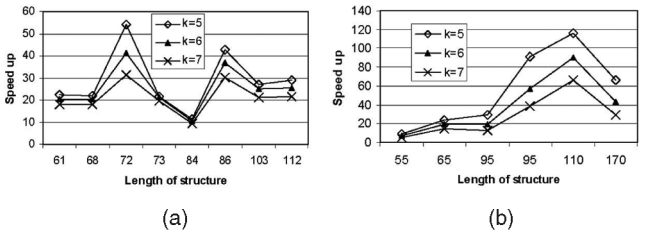


Fig. 9. The speed up of the tree decomposition-based method over the CM-based method: (a) on pseudoknot-free structures and (b) on pseudoknot structures.

	ncRNA	Real location		Tree decomposition based			CM based			Genome length
		Left	Right	Left offset	Right offset	Time	Left off	Right off	Time	
3'PK	BCV	30798	30859	0	0	0.053	0	0	1.24	3.1×10^4
	MHV	31792	31153	0	0	0.053	0	0	1.27	3.1×10^4
	PDV	27802	27882	0	0	0.048	0	0	1.17	2.8×10^4
	HCV	27063	27125	0	0	0.047	0	0	1.12	2.7×10^4
tmRNA	HI	472209	472574	-1	-1	44.0	0	0	1700	1.83×10^6
	NM	1241197	1241559	0	0	52.9	0	0	2044	2.2×10^5
TLRNA	SC	307688	308429	-3	-1	492.3	-	-	-	1.03×10^7
	SB	7121529	7122284	-3	2	550.2	-	-	-	1.15×10^7

Fig. 10. Performance comparison between the tree decomposition-based method and the CM-based method on RNA structure searches on genomes. The offset is between the annotated and the real positions. The time unit is hour.

HDV_ribozyme, Tombus_3_IV, Alpha_RBS, and IFN_gamma, in the increasing order of their lengths. As for pseudoknot-free structures, the tree decomposition-based searches for pseudoknots achieved the same performance as the CM-based method for parameter values $k \leq 7$.

Fig. 9 shows the speed up by the new method over the CM based method, for 1) pseudoknot-free and 2) pseudoknot structures. It is evident that, for $k = 7$, the new method was about 20 to 30 times faster than the other method on pseudoknot-free structures. On the pseudoknot structures, typically on Alpha_RBS and Tombus_3_IV containing more than 100 nucleotides, the new method was 66 and 38 times faster, suggesting its advantage in the search for larger and more complex structures.

Fig. 10 compares the search results obtained by the two methods on three types of RNA pseudoknots in virus, bacteria, and yeast genomes. Parameter $k = 7$ is used for the parameterized algorithm. Both methods achieve 100 percent sensitivity and specificity. It clearly shows that the new method had a speedup of about 30 to 40 times over the other method for searches in virus and bacteria genomes. With the new method, searching genomes of a moderate size for structures as complex as the tmRNA gene (see Fig. 7) only took days, instead of months. Searching a larger genome such as yeast for a larger structure like telomerase RNAs was also successful, a task not accomplishable by the CM-based system within a reasonable amount of time.

5 Conclusions and Discussions

We introduced a novel method and an efficient parameterized algorithm for the structure-sequence alignment problem by exploiting the small tree width of biopolymer structure graphs. The algorithm was applied to the development of a fast search program that is capable of accurately identifying a complex RNA secondary structure including pseudoknots in genomes [34].

Our method provides a new perspective on structure-sequence alignment that is important in a number of bioinformatics research areas where structure plays an instrumental role. For example, a similar approach has been used for protein side-chain packing when the backbone of the protein structure is known [21], [38]. In particular, we expect this method to yield very efficient algorithms for protein tertiary structure prediction via protein threading. The core portion of the protein threading technique is a structure-sequence alignment that aligns the target protein with every structure template in the fold database. In ongoing research, we are applying the algorithm GSGI to the development of an efficient protein threading program for protein tertiary structure prediction [33]. In the following, we briefly report this application of the GSGI algorithm. We note that an independent work for protein threading based on tree decomposition has also been proposed [39], which has yet to be implemented and tested.

In our application, the structural units in a protein structure template are alpha helices and beta strands, which are called *cores*. A structure graph H constructed from a protein structure template contains cores as vertices and interactions between cores as edges. In addition, the loop between every pair of consecutive cores is represented as a directed edge (from the N terminal to the C terminal).

The target protein is preprocessed in order to produce a sequence graph. We first use the profile of each core specified in the structure template to scan the target sequence to identify segments in the sequence that are aligned well with the core profile. Then, we construct a sequence graph G that contains all candidates as vertices and possible interactions between candidates as edges. In G, nonoverlapping candidates are also connected by directed edges. We define the parameter k to be such that, for each core, at most k of the segments with the top alignment scores are selected as the candidates of the core. The parameter k is determined by a statistical cut-off; our experiments show that k is small on real proteins.

To apply algorithm GSGI on the mixed graphs H and G to find an optimal subgraph isomorphism, we need to define the three scoring functions, S_1, S_2, and S_3 (see Section 2). For protein threading, the overall alignment score between the structure template and the target sequence is actually computed as the weighted sum of five types of energy terms [42], [41]:

$$E = w_m E_m + w_s E_s + w_p E_p + w_g E_g + w_{ss} E_{ss}.$$

These energy terms are explained as follows: Term E_m is the overall mutation energy of amino acids. Term E_s is the overall singleton energy that evaluates the preference of the placement of every amino acid in a residue location within a core of certain solvation accessibility. Term E_p is the overall pairwise interaction energy between amino acids. Term E_g is the overall gap penalty for the alignment. Term E_{ss} is the additional overall alignment score for those amino acids aligned to the cores.

Then, functions S_1, S_2, and S_3 can be defined with these energy terms. S_1 is the addition of the weighted terms E_m,

E_s, E_{ss}, and E_g for those amino acids aligned to cores. S_2 is the weighted E_p for those amino acids aligned to cores. S_3 is the sum of weighted E_m, E_s, and E_g for those amnio acids aligned to loops.

Based on the GSGI algorithm, we have implemented a protein threading program called PROTTD [33]. Tests of PROTTD on real protein structure templates and target sequences are being conducted.

REFERENCES

[1] S. Arnborg and A. Proskurowski, "Linear Time Algorithms for NP-Hard Problems Restricted to Partial k-Trees," *Discrete Applied Math.,* vol. 23, pp. 11-24, 1989.

[2] S. Arnborg, J. Lagergren, and D. Seese, "Easy Problems for Tree-Decomposable Graphs," *J. Algorithms,* vol. 12, pp. 308-340, 1991.

[3] J. Bowie, R. Luthy, and D. Eisenberg, "A Method to Identify Protein Sequences that Fold into a Known Three-Dimensional Structure," *Science,* vol. 253, pp. 164-170, 1991.

[4] H.L. Bodlaender, "A Linear Time Algorithm for Finding Tree-Decompositions of Small Treewidth," *SIAM J. Computing,* vol. 25, pp. 1305-1317, 1996.

[5] S.H. Bryant and S.F. Altschul, "Statistics of Sequence-Structure Threading," *Current Opinion Structural Biology,* vol. 5, pp. 236-244, 1995.

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

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

[8] T. Dandekar, S. Schuster, B. Snel, M. Huynen, and P. Bork, "Pathway Alignment: Application to the Comparative Analysis of Glycolytic Enzymes," *Biochemical J.,* vol. 1, pp. 115-24, 1999.

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

[10] J.A. Doudna, "Structural Genomics of RNA," *Nature Structural Biology,* vol. 7, no. 11 supp., pp. 954-956, 2000.

[11] R. Downey and M. Fellows, *Parameterized Complexity.* Springer, 1999.

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

[13] S.R. Eddy, "Computational Genomics of Non-Coding RNA Genes," *Cell,* vol. 109, pp. 137-140, 2002.

[14] S. Eddy and R. Durbin, "RNA Sequence Analysis Using Covariance Models," *Nucleic Acids Research,* vol. 22, pp. 2079-2088, 1994.

[15] D. Eppstein, "Subgraph Isomorphism in Planar Graphs and Related Problems," *J. Graph Algorithms and Applications,* vol. 3.3, pp. 1-27, 1999.

[16] S.J. Geobel, B. Hsue, T.F. Dombrowski, and P.S. Masters, "Characterization of the RNA Components of a Putative Molecular Switch in the 3′ Untranslated Region of the Murine Coronavirus Genome," *J. Virology,* vol. 78, pp. 669-682, 2004.

[17] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S.R. Eddy, "Rfam: An RNA Family Database," *Nucleic Acids Research,* vol. 31, pp. 439-441, 2003.

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

[19] R.H. Lathrop, "The Protein Threading Problem with Sequence Amino Acid Interaction Preferences is NP-Complete," *Protein Eng.,* vol. 7, pp. 1069-1068, 1994.

[20] R.H. Lathrop, R.G. Rogers Jr, J. Bienkowska, B.K.M. Bryant, L.J. Buturovic, C. Gaitatzes, R. Nambudripad, J.V. White, and T.F. Smith, "Analysis and Algorithms for Protein Sequence-Structure Alignment," *Computational Methods in Molecular Biology,* ?. Salzberg, ?. Searls, and ?. Kasif, eds., Elsevier, 1998.

[21] A. Leaver-Fay, B. Kuhlman, and J. Snoeyink, "An Adaptive Dynamic Programming Algorithm for the Side Chain Placement Problem," *Proc. Pacific Symp. Biocomputing 10,* pp. 16-27, 2005.

[22] H.-P. Lenhof, K. Reinert, and M. Vingron, "A Polyhedral Approach to RNA Sequence Structure Alignment," *J. Computational Biology,* vol. 5, no. 3, pp. 517-530, 1998.

[23] C. Liu, Y. Song, R. Malmberg, and L. Cai, "Profiling and Searching for RNA Pseudoknot Structures in Genomes," *Lecture Notes in Computer Science,* vol. 3515, pp. 968-975, 2005.

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

[25] S.B. Lyngso and C.N. Pedersen, "RNA Pseudoknot Prediction in Energy-Based Models," *J. Computational Biololgy,* vol. 7, no. 3, pp. 409-427, 2000.

[26] J. Matousek and R. Thomas, "On the Complexity of Finding Iso- and Other Morphisms for Partial k-Trees," *Discrete Math.,* vol. 108, pp. 343-364, 1992.

[27] E.M. Marcotte, P. Matteo, H.L. Ng, D.W. Rice, T.O. Yeates, and D. Eisenberg, "Detecting Protein Function and Protein-Protein Interactions from Genome Sequences," *Science,* vol. 285, pp. 751-753, year?

[28] N. Nameki, B. Felden, J.F. Atkins, R.F. Gesteland, H. Himeno, and A. Muto, "Functional and Structural Analysis of a Pseudoknot Upstream of the Tag-Encoded Sequence in E. coli tmRNA," *J. Molecular Biology,* vol. 286, no. 3, pp. 733-744, 1999.

[29] D.D. Pervouchine, "IRIS: Intermolecular RNA Interaction Search," *Genome Informatics,* vol. 15, no. 2, pp. 92-101, 2004.

[30] K. Reinert, H.-P. Lenhof, P. Mutzel, K. Mehlhorn, and J.D. Kececioglu, "A Branch-and-Cut Algorithm for Multiple Sequence Alignment," *Proc. First Ann. Int'l Conf. Computational Molecular Biology,* pp. 241-250, 1997.

[31] E. Rivas and S.R. Eddy, "Noncoding RNA Gene Detection Using Comparative Sequence Analysis," *BMC Bioinformatics,* vol. 2, p. 8, 2001.

[32] N. Robertson and P.D. Seymour, "Graph Minors II. Algorithmic Aspects of Tree-Width," *J. Algorithms,* vol. 7, pp. 309-322, 1986.

[33] Y. Song, K. Ellrott, C. Liu, J. Guo, Y. Xu, and L. Cai, "Efficient Protein Threading with Tree Decomposition," manuscript, year?

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

[35] Y. Uemura, A. Hasegawa, Y. Kobayashi, and T. Yokomori, "Tree Adjoining Grammars for RNA Structure Prediction," *Theoretical Computer Science,* vol. 210, pp. 277-303, 1999.

[36] G. Wang and R.L. Dunbrack Jr., "PISCES: A Protein Sequence Culling Server," *Bioinformatics,* vol. 19, pp. 1589-1591, 2003.

[37] D. Xu, M.A. Unseren, Y. Xu, and E.C. Uberbacher, "Sequence-Structure Specificity of a Knowledge Based Energy Function at the Secondary Structure Level," *Bioinformatics,* vol. 16, pp. 257-268, 2000.

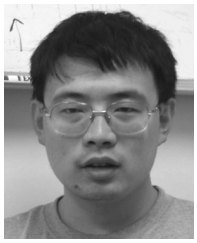
[38] J. Xu, "Rapid Side-Chain Packing via Tree Decomposition," *Proc. 2005 Int'l Conf. Research in Computational Biology,* pp. 423-439, 2005.

[39] J. Xu, F. Jiao, and B. Berger, "A Tree-Decomposition Approach to Protein Structure Prediction," *Proc. 2005 IEEE Computational Systems Bioinformatics Conf.,* pp. 247-256, 2005.

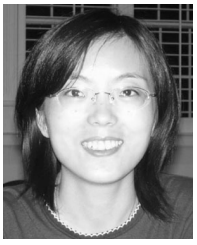
[40] J. Xu, M. Li, D. Kim, and Y. Xu, "RAPTOR: Optimal Protein Threading by Linear Programming," *J. Bioinformatics and Computational Biology,* vol. 1, no. 1, pp. 95-113, 2003.

[41] Y. Xu, Z. Liu, L. Cai, and D. Xu, "Protein Structure Prediction by Protein Threading," *Computational Methods for Protein Structure Prediction and Modeling,* ?. Xu, ?. Xu and ?. Liang, eds., Springer, 2006.

[42] Y. Xu, D. Xu, and E.C. Uberbacher, "An Efficient Computational Method for Globally Optimal Threading," *J. Computational Biology,* vol. 5, no. 3, pp. 597-614, year?



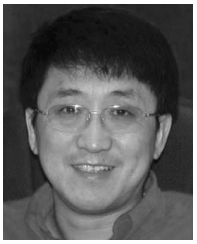
Yinglei Song received the BS degree in physics from Tsinghua University in 1998 and the MS degree in computer science from Ohio University in 2003. He is currently a PhD candidate in the Department of Computer Science at the University of Georgia. His research interests include structural bioinformatics, parameterized algorithms, and graph algorithms. He is a student member of the IEEE.



Chunmei Liu is a PhD candidate in the Department of Computer Science at the University of Georgia. Her research interests include algorithms, bioinformatics, and computational biology.

Xiuzhen Huang received the BS and MS degrees in computer science from Shandong University, China, in 1996 and 1999 and the PhD degree in computer science from Texas A&M University in 2004. She is an assistant professor in the Department of Computer Science at Arkansas State University. Her research interests include computational complexity and approximation, graph theory and algorithms, and bioinformatics.

Russell L. Malmberg received the PhD degree in genetics from the University of Wisconsin and did postdoctoral research at Michigan State University. He was a staff scientist at the Cold Spring Harbor Laboratory, then moved to the University of Georgia, where he is currently a professor of plant biology. He current research interests are in bioinformatics and evolutionary genetics.



Ying Xu received the undergraduate and graduate degrees in computer science from Jilin University and the PhD degree in theoretical computer science from the University of Colorado at Boulder in 1991. He is a chair professor of bioinformatics and computational biology in the Biochemistry and Molecular Biology Department and the director of the Institute of Bioinformatics, University of Georgia (UGA). Before joining UGA in September 2003, he was a senior staff scientist and group leader at Oak Ridge National Laboratory, where he still holds a joint position. He also holds guest or research professor positions at the University of Tennessee at Knoxville, Jilin Uiniversity and Zhejiang University of China, and an adjunct professor position in the Computer Science Department at UGA. He is interested in both bioinformatics tool development and the study of biological problems using in silicon approaches. His current research interests include computational inference and modeling of biological pathways and networks, protein structure prediction and modeling, large-scale biological data mining, and microbial and cancer bioinformatics.

Liming Cai received the BS and MS degrees in computer science from Tsinghua University, China, in 1984 and 1986, respectively, and the PhD degree in computer science from Texas A&M University in 1994. He is an associate professor in the Department of Computer Science at the University of Georgia. His current research interests include algorithms, computational biology, and the theory of computing.