Journal of Bioinformatics and Computational Biology
Vol. 10, No. 4 (2012) 1250001 (15 pages)
© Imperial College Press
DOI: 10.1142/S0219720012500011

A METHOD FOR DISCOVERING COMMON PATTERNS FROM TWO RNA SECONDARY STRUCTURES AND ITS APPLICATION TO STRUCTURAL REPEAT DETECTION

LEI HUA[*,‡], JASON T. L. WANG[*,†,§] and XIANG JI[*,¶]

[*]*Department of Computer Science*
New Jersey Institute of Technology
Newark, New Jersey 07102, USA

[†]*Bioinformatics Program*
New Jersey Institute of Technology, Newark
New Jersey 07102, USA
[‡]*lh56@njit.edu*
[§]*wangj@njit.edu*
[¶]*xj25@njit.edu*

ANKUR MALHOTRA

Division of Biological Sciences
University of California, San Diego
La Jolla, California 92093, USA
anmalhotra@ucsd.edu

MUGDHA KHALADKAR

Biology Department, University of Pennsylvania
Philadelphia, Pennsylvania 19104, USA
mugdhak@pcbi.upenn.edu

BRUCE A. SHAPIRO

Center for Cancer Research Nanobiology Program
National Cancer Institute, Frederick
Maryland 21702, USA
shapirbr@mail.nih.gov

KAIZHONG ZHANG

Department of Computer Science
University of Western Ontario
London, Ontario N6A, 5B7, Canada
kzhang@csd.uwo.ca

Received 29 March 2011
Revised 6 July 2011
Accepted 5 November 2011
Published 12 January 2012

We propose an *ab initio* method, named DiscoverR, for finding common patterns from two RNA secondary structures. The method works by representing RNA secondary structures as ordered labeled trees and performs tree pattern discovery using an efficient dynamic programming algorithm. DiscoverR is able to identify and extract the largest common substructures from two RNA molecules having different sizes without prior knowledge of the locations and topologies of these substructures. We also extend DiscoverR to find repeated regions in an RNA secondary structure, and apply this extended method to detect structural repeats in the 3′-untranslated region of a protein kinase gene. We describe the biological significance of a repeated hairpin found by our method, demonstrating the usefulness of the method. DiscoverR is implemented in Java; a jar file including the source code of the program is available for download at http://bioinformatics.njit.edu/DiscoverR.

Keywords: RNA secondary structure; pattern discovery; repeat finding.

1. Introduction

Many functional RNAs exhibit a highly conserved secondary structure although their nucleotide sequences share little similarity. Thus, in developing effective tools for comparing and detecting the functional RNAs as well as important evolutionary divergences, researchers often consider the secondary structures of the RNA molecules.[1–3]

We propose here a novel method, named DiscoverR, for detecting common patterns of two RNA secondary structures. DiscoverR builds upon our previous work in tree pattern finding.[4] It works by representing RNA secondary structures as ordered labeled trees, and then performs tree pattern discovery by allowing certain subtrees to be removed at no cost. The method is capable of identifying and extracting the largest common substructures from two RNA molecules having different sizes without prior knowledge of the locations and topologies of these substructures. The method is faster than the existing algorithm for general approximate tree pattern discovery.[4]

We then present an application of DiscoverR by using it to find repeated regions in an RNA secondary structure. Repeat finding has been an important subject in bioinformatics and computational biology. Past work has mainly focused on detecting repeats in sequences.[5,6] In contrast, DiscoverR is capable of locating structural repeats or repeated regions in an RNA secondary structure. We show how our method can be used to find structural repeats in the 3′-untranslated region of a protein kinase gene, and describe the biological significance of a repeated hairpin detected by our method. Finally we compare our method with some closely related methods and conclude the paper.

2. Method

2.1. *Representing RNA secondary structures by trees*

Let RS be an RNA sequence containing nucleotides or bases A, U, C, G. $RS[i]$ denotes the base at position i of RS and $RS[i, j]$ is the subsequence starting at position i and ending at position j in RS. Let R be the secondary structure of RS. A base pair connecting position i and position j in R is denoted by (i, j) and its enclosed sequence is $RS[i, j]$. A loop in R refers to a hairpin, a bulge, an internal loop or a multibranched loop.[7,8] Given a loop L in the secondary structure R, the base pair $(i*, j*)$ in L is called the *exterior pair* of L if position i^* (j^*, respectively) is closest to the 5′ (3′, respectively) end of R among all positions in L. All other non-exterior base pairs in L are called *interior pairs* of L.[9]

As in previous work[10–16] we model the RNA secondary structure R of the sequence RS by an ordered labeled tree RT in which each node has a label and the left to right order of siblings is significant (Fig. 1). With this model, pseudoknots are not allowed. Each node in RT corresponds to a base pair in R and vice versa. Base pairs are numbered according to the order from the 5′ end to the 3′ end of R. Except for the exterior pairs of loops, the kth base pair of R corresponds to the node labeled "Pk" in RT and vice versa. For example, the node labeled "P3" in the tree RT

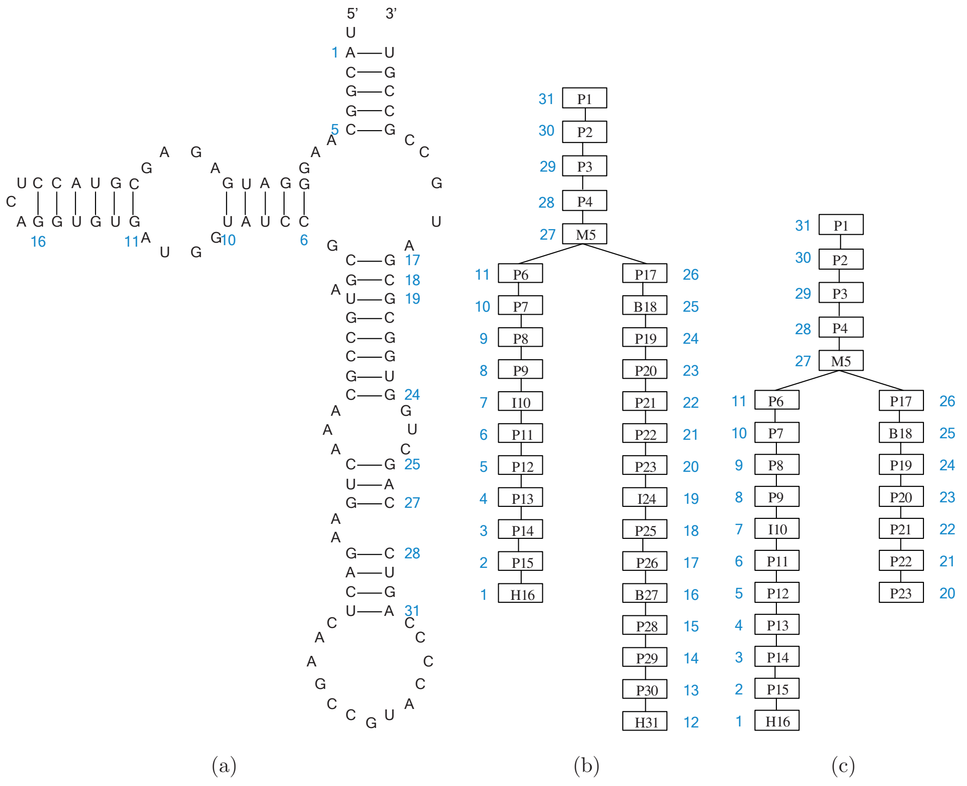


Fig. 1. (a) An RNA secondary structure is composed of base pairs, which are numbered according to the order from the 5′-end to the 3′-end of the secondary structure. (b) The base pairs are organized into an ordered labeled tree. Each node in the tree corresponds to a base pair in the secondary structure and vice versa. The numeric value next to each node in the tree is the position of that node in the left-to-right postorder traversal of the tree. (c) Cutting at the node labeled I24 means removing the subtree rooted at the node labeled I24.

shown in Fig. 1(b) corresponds to the third base pair in the RNA secondary structure R shown in Fig. 1(a).

The exterior pair of a multibranched loop containing n interior pairs in R corresponds to a node v with n children in RT with each child corresponding to one of the n interior pairs. Assuming the exterior pair is the kth base pair in R, the node label of v in RT is "Mk." The exterior pair of a bulge loop (internal loop, hairpin loop, respectively) in R corresponds to the node labeled "Bk" ("Ik," "Hk," respectively) in RT if the exterior pair is the kth base pair in R. For example, the node labeled "M5" ("B18," "I24," "H31," respectively) in the tree RT shown in Fig. 1(b) corresponds to the exterior pair of the multibranched loop (bulge loop, internal loop, hairpin loop, respectively) where the exterior pair is the fifth (18th, 24th, 31st, respectively) base pair in the RNA secondary structure R shown in Fig. 1(a). For each node v in the tree RT, we use $NB(v)$ to represent the number of bases v has. If the node label of v is "Pi" for some i, i.e. v corresponds to a base pair, $NB(v) = 2$. If v corresponds to the exterior pair of a loop, $NB(v)$ equals the number of bases in that loop.

Our algorithm uses a postorder numbering of nodes in the tree RT representing the RNA secondary structure R. Let $rt[i]$ be the node of RT whose position in the left-to-right postorder traversal of RT is i. Referring to the tree RT shown in Fig. 1(b), the numeric value next to each node is the position of that node in the left-to-right postorder traversal of RT. Let $RT[i]$ represent the subtree rooted at $rt[i]$. We introduce a cut operation on nodes in a tree.[17] Cutting at node $rt[i]$ means removing $RT[i]$ from the tree RT (*cf.* Fig. 1(c)). A set S of nodes of $RT[k]$ is said to be a set of consistent subtree cuts in $RT[k]$ if (i) $rt[i] \in S$ implies that $rt[i]$ is a node in $RT[k]$, and (ii) $rt[i]$, $rt[j] \in S$ implies that neither is an ancestor of the other in $RT[k]$. Intuitively, S is the set of all roots of the removed subtrees in $RT[k]$. For example, consider the nodes labeled P9 and P23 in the tree shown in Fig. 1(b). Neither node is an ancestor of the other. Thus, the set containing these two nodes is a set of consistent subtree cuts. We use $Cut(RT, S)$ to represent the substructure of RT resulted from cutting at all nodes in S. Notice that the substructure $Cut(RT, S)$ is connected at the structure level; that is, if two nodes in RT are contained in the substructure such that one node is an ancestor of the other node, then all nodes in between the two nodes are also contained in the substructure. For example, cutting at the nodes labeled P9 and P23 in the secondary structure in Fig. 1 yields the substructure shown in Fig. 2, which is connected at the structure level. We use $Subtrees(RT)$ to represent the set of all possible sets of consistent subtree cuts in RT.

2.2. *Common patterns of two trees*

Let R_1 and R_2 be two RNA secondary structures. Let RT_1 (RT_2, respectively) be the tree representing R_1(R_2, respectively). Let rt_1 be a node in RT_1 and let rt_2 be a node in RT_2. The dissimilarity between the two nodes rt_1 and rt_2, denoted $\delta(rt_1, rt_2)$, is

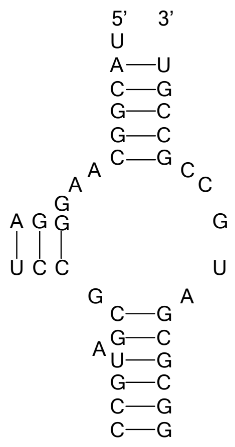


Fig. 2. The substructure obtained by cutting at the nodes labeled P9 and P23 in the secondary structure in Fig. 1.

calculated by Eq. (1):

$$\delta(rt_1, rt_2) = \frac{|NB(rt_1) - NB(rt_2)|}{NB(rt_1) + NB(rt_2)}. \tag{1}$$

Thus, $\delta(rt_1, rt_2)$ equals 0 if rt_1 and rt_2 have the same number of bases. We say node rt_1 matches node rt_2, denoted $rt_1 \approx rt_2$, if $\delta(rt_1, rt_2) \leq \varepsilon$ where ε is an adjustable non-negative threshold value. (In the present study, we use the default threshold value, which is set to 0.1.) When rt_1 (rt_2, respectively) corresponds to a base pair, rt_1 always matches rt_2, since $\delta(rt_1, rt_2)$ equals 0. We say tree RT_1 matches tree RT_2, denoted $RT_1 \approx RT_2$, if the two trees are isomorphic and each node in RT_1 matches its corresponding node in RT_2.

The size of the largest common substructures or *common patterns* of $RT_1[i]$ and $RT_2[j]$, denoted $\Psi(RT_1[i], RT_2[j])$ (or simply $\Psi(i, j)$ when the context is clear), is $\max\{|Cut(RT_1[i],\ S_1)|\}$ or $\max\{|Cut(RT_2[j],\ S_2)|\}$ subject to $Cut(RT_1[i],\ S_1) \approx Cut(RT_2[j], S_2)$, $S_1 \in Subtrees(RT_1[i])$, $S_2 \in Subtrees(RT_2[j])$, where $|\cdot|$ is the number of nodes in the indicated substructure. It should be pointed out that $Cut(RT_1[i], S_1)$ is isomorphic to $Cut(RT_2[j], S_2)$, and therefore $|Cut(RT_1[i], S_1)| = |Cut(RT_2[j], S_2)|$. Our goal is to compute $\max_{1 \leq i \leq |RT_1|, 1 \leq j \leq |RT_2|}\{\Psi(RT_1[i], RT_2[j])\}$ and locate the $Cut(RT_1[i], S_i)$ and $Cut(RT_2[j], S_j)$, where $S_i \in Subtrees(RT_1[i])$ and $S_j \in Subtrees(RT_2[j])$, that achieve the maximum size. We will focus on calculating the maximum size. By memorizing the size information during the computation and by a backtracking technique, one can find the maximum size and a substructure pair yielding the size with the same time complexity.

2.3. *Common patterns of two forests*

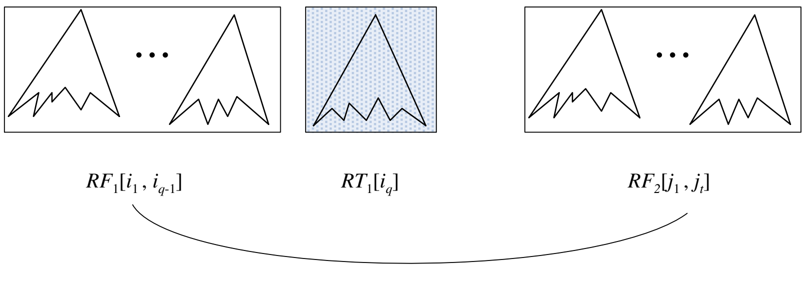
The degree of a node v is defined as the number of children of v. Suppose the degree of the node $rt_1[i]$ ($rt_2[j]$, respectively) in the tree RT_1 (RT_2, respectively) is

m_i (n_j, respectively). Denote the children of $rt_1[i]$ as $rt_1[i_1], rt_1[i_2], \ldots, rt_1[i_{m_i}]$, and the children of $rt_2[j]$ as $rt_2[j_1], rt_2[j_2], \ldots, rt_2[j_{n_j}]$. For any p, q, $1 \leq p \leq q \leq m_i$, let $RF_1[i_p, i_q]$ represent the forest containing the subtrees $RT_1[i_p], RT_1[i_{p+1}], \ldots, RT_1[i_q]$. $RF_1[i_p, i_q] = \phi$ if $p > q$, and $RF_1[i_p, i_q] = RT_1[i_p]$ if $p = q$. $RF_1[i] = RF_1[i_1, i_{m_i}]$. $RF_2[j_s, j_t]$, $1 \leq s \leq t \leq n_j$, and $RF_2[j]$ are defined similarly. We say forest RF_1 matches forest RF_2, denoted $RF_1 \approx RF_2$, if the two forests are isomorphic and each node in RF_1 matches its corresponding node in RF_2.

A set S of nodes of forest RF is said to be a set of consistent subtree cuts in RF if (i) $rt[i] \in S$ implies that $rt[i]$ is a node in RF, and (ii) $rt[i]$, $rt[j] \in S$ implies that neither is an ancestor of the other in RF. We use $Cut(RF, S)$ to represent the subforest of RF resulted from cutting at all nodes in S. Let $Subtrees(RF)$ be the set of all possible sets of consistent subtree cuts in RF. Define the size of the largest common substructures or common patterns of forest RF_1 and forest RF_2, denoted $\Phi(RF_1, RF_2)$, to be $\max\{|Cut(RF_1, S_1)|\}$ or $\max\{|Cut(RF_2, S_2)|\}$ subject to $Cut(RF_1, S_1) \approx Cut(RF_2, S_2)$, $S_1 \in Subtrees(RF_1)$, $S_2 \in Subtrees(RF_2)$. When $RF_1 = RF_1[i_p, i_q]$ and $RF_2 = RF_2[j_s, j_t]$, we also represent $\Phi(RF_1, RF_2)$ by $\Phi(i_p \ldots i_q,\ j_s \ldots j_t)$ if there is no confusion.

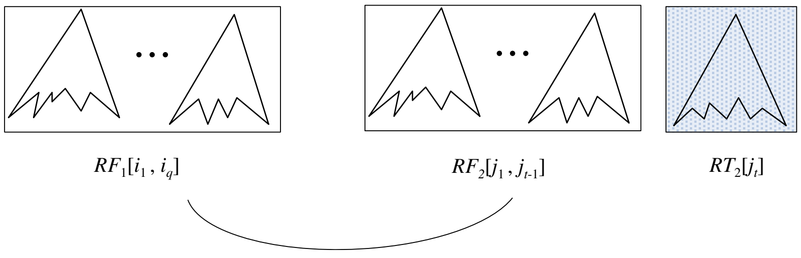
2.4. *Filling in the maximum size table*

It is clear that $\Psi(\phi, \phi) = 0$, $\Phi(\phi, \phi) = 0$, $\Psi(RT_1[i], \phi) = 0$, $\Psi(\phi, RT_2[j]) = 0$, $\Phi(RF_1[i], \phi) = \Phi(RF_1[i_1, i_{m_i}],\ \phi) = 0$ and $\Phi(\phi, RF_2[j]) = \Phi(\phi, RF_2[j_1, j_{n_j}]) = 0$, i.e. the size of the common patterns of two trees (forests, respectively) is 0 if one of the trees (forests, respectively) is empty. In general, there are two cases to be considered. In case 1, we compute $\Phi(RF_1[i_1, i_q],\ RF_2[j_1,\ j_t])$, where $1 \leq q \leq m_i$ and $1 \leq t \leq n_j$. There are three subcases to be considered (Fig. 3): (1) The subtree $RT_1[i_q]$ is removed, hence $\Phi(i_1 \ldots i_q,\ j_1 \ldots j_t) = \Phi(i_1 \ldots i_{q-1},\ j_1 \ldots j_t)$; (2) The subtree

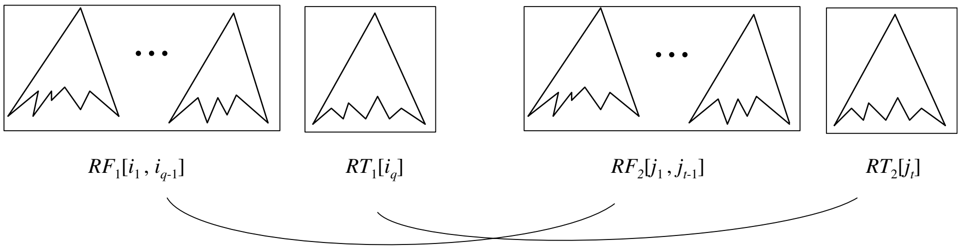


(a)

Fig. 3. (a) The shaded subtree $RT_1[i_q]$ is removed. The size of the common patterns of forest $RF_1[i_1, i_q]$ and forest $RF_2[j_1, j_t]$ is the same as the size of the common patterns of forest $RF_1[i_1, i_{q-1}]$ and forest $RF_2[j_1, j_t]$. (b) The shaded subtree $RT_2[j_t]$ is removed. The size of the common patterns of forest $RF_1[i_1, i_q]$ and forest $RF_2[j_1, j_t]$ is the same as the size of the common patterns of forest $RF_1[i_1, i_q]$ and forest $RF_2[j_1, j_{t-1}]$. (c) Neither $RT_1[i_q]$ nor $RT_2[j_t]$ is removed. The size of the common patterns of forest $RF_1[i_1, i_q]$ and forest $RF_2[j_1, j_t]$ equals the size of the common patterns of $RF_1[i_1, i_{q-1}]$ and forest $RF_2[j_1, j_{t-1}]$ plus the size of the common patterns of tree $RT_1[i_q]$ and tree $RT_2[j_t]$.



(b)



(c)

Fig. 3. (*Continued*)

$RT_2[j_t]$ is removed, hence $\Phi(i_1 \mathinner{..} i_q, j_1 \mathinner{..} j_t) = \Phi(i_1 \mathinner{..} i_q, j_1 \mathinner{..} j_{t-1})$; (3) Neither $RT_1[i_q]$ nor $RT_2[j_t]$ is removed. Hence, the size of the common patterns of $RF_1[i_1, i_q]$ and $RF_2[j_1, j_t]$ equals the size of the common patterns of $RF_1[i_1, i_{q-1}]$ and $RF_2[j_1, j_{t-1}]$ plus the size of the common patterns of $RT_1[i_q]$ and $RT_2[j_t]$. We take the maximum of the three subcases, obtaining the following recurrence equation:

$$\Phi(i_1 \mathinner{..} i_q, j_1 \mathinner{..} j_t) = \max\begin{cases}\Phi(i_1 \mathinner{..} i_{q-1}, j_1 \mathinner{..} j_t)\\ \Phi(i_1 \mathinner{..} i_q, j_1 \mathinner{..} j_{t-1})\\ \Phi(i_1 \mathinner{..} i_{q-1}, j_1 \mathinner{..} j_{t-1}) + \Psi(i_q, j_t).\end{cases} \tag{2}$$

In case 2, we compute $\Psi(RT_1[i], RT_2[j])$, $1 \le i \le |RT_1|$, $1 \le j \le |RT_2|$. There are two subcases to be considered: (1) The node $rt_1[i]$ matches the node $rt_2[j]$, hence $\Psi(i,j) = \Phi(i_1 \mathinner{..} i_{m_i}, j_1 \mathinner{..} j_{n_j}) + 1$ (Fig. 4). (2) The node $rt_1[i]$ does not match the node $rt_2[j]$, hence $\Psi(i,j) = 0$. Therefore,

$$\Psi(i,j) = \begin{cases}\Phi(i_1 \mathinner{..} i_{m_i}, j_1 \mathinner{..} j_{n_j}) + 1 & \text{if } rt_1[i] \approx rt_2[j]\\ 0 & \text{otherwise.}\end{cases} \tag{3}$$

2.5. *Algorithm and complexity*

DiscoverR employs a dynamic programming algorithm that maintains a two-dimensional table in which $c(i,j)$ represents the cell located at the intersection of the ith row and the jth column of the table. The value stored in the cell $c(i,j)$,

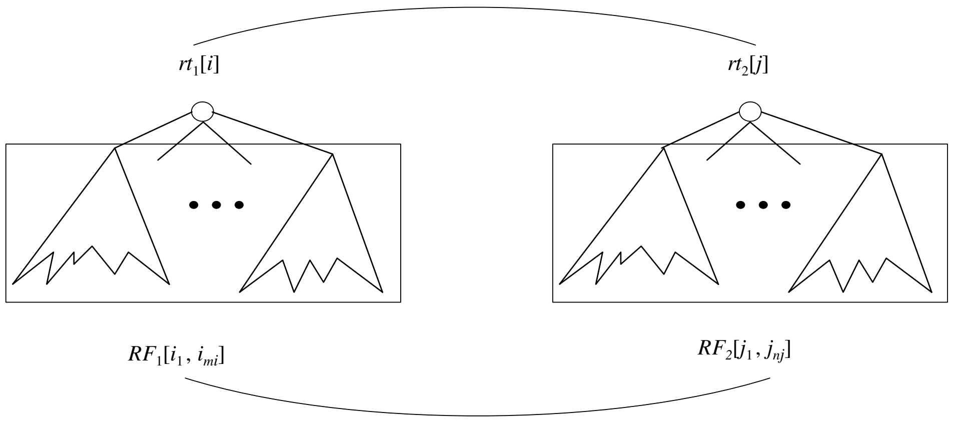


Fig. 4. The node $rt_1[i]$ matches the node $rt_2[j]$. Thus, the size of the common patterns of tree $RT_1[i]$ and tree $RT_2[j]$ equals the size of the common patterns of forest $RF_1[i_1, i_{m_i}]$ and forest $RF_2[j_1, j_{n_j}]$ plus 1.

$1 \le i \le |RT_1|$, $1 \le j \le |RT_2|$, is $\Psi(RT_1[i], RT_2[j])$. The algorithm calculates the values in the table by traversing the trees RT_1 and RT_2 in a bottom-up manner. Figures 5 and 6 present the main procedures employed by the algorithm. For each input $RF_1[i_1, i_{m_i}]$ and $RF_2[j_1, j_{n_j}]$, the running time of Procedure 1 is $O(m_i \times n_j)$. Thus, the time complexity of the algorithm is

$$\sum_{i=1}^{|RT_1|} \sum_{j=1}^{|RT_2|} O(m_i \times n_j) = O(|RT_1| \times |RT_2|),$$

```
Procedure 1: Computing $\Phi(i_1 .. i_{m_i}, j_1 .. j_{n_j})$
Input: $RF_1[i_1, i_{m_i}]$ and $RF_2[j_1, j_{n_j}]$
Output: $\Phi(i_1 .. i_{m_i}, j_1 .. j_{n_j})$
1. $\Phi(\phi, \phi) \leftarrow 0$
2. for $q := 1$ to $m_i$
3.     $\Phi(i_1 .. i_q, \phi) \leftarrow 0$
4. for $t := 1$ to $n_j$
5.     $\Phi(\phi, j_1 .. j_t) \leftarrow 0$
6. for $q := 1$ to $m_i$
7.     for $t := 1$ to $n_j$
8.         Compute $\Phi(i_1 .. i_q, j_1 .. j_t)$ as in Eq. (2)
```

Fig. 5. Procedure for computing $\Phi(i_1 .. i_{m_i}, j_1 .. j_{n_j})$.

```
Procedure 2: Computing $\Psi(i,j)$ for all $1 \le i \le |RT_1|, 1 \le j \le |RT_2|$
Input: $RT_1$ and $RT_2$
Output: $\Psi(i,j)$ for all $1 \le i \le |RT_1|, 1 \le j \le |RT_2|$
1.  $\Psi(\phi,\phi) \leftarrow 0$
2.  for $i := 1$ to $|RT_1|$
3.      $\Psi(RT_1[i],\phi) \leftarrow 0$
4.  for $j := 1$ to $|RT_2|$
5.      $\Psi(\phi, RT_2[j]) \leftarrow 0$
6.  for $i := 1$ to $|RT_1|$
7.      for $j := 1$ to $|RT_2|$
8.          Compute $\Psi(i,j)$ as in Eq. (3)
            by calling Procedure 1 on $RF_1[i_1, i_{m_i}]$ and $RF_2[j_1, j_{n_j}]$
```

Fig. 6. Procedure for computing $\Psi(i,j)$ for all $1 \le i \le |RT_1|, 1 \le j \le |RT_2|$.

which is much faster than the existing algorithm for general approximate tree pattern discovery.[4] We can calculate $\max_{1\le i\le |RT_1|,\, 1\le j\le |RT_2|}\{\Psi(RT_1[i], RT_2[j])\}$ in the same time. Since the number of nodes in the tree RT_1 (RT_2, respectively) equals the number of base pairs in the RNA secondary structure R_1 (R_2, respectively), and 54% of the nucleotides on average in an RNA sequence are involved in the base pairs of its secondary structure,[18] the time complexity of the DiscoverR method is $O(|R_1| \times |R_2|)$, where $|.|$ is the number of nucleotides in the indicated RNA secondary structure. After calculating $\max_{1\le i\le |RT_1|,\, 1\le j\le |RT_2|}\{\Psi(RT_1[i], RT_2[j])\}$ and locating the largest common substructures or common patterns of RT_1 and RT_2 that yield the maximum size, we print out the common patterns, which constitute the output of DiscoverR.

3. Application

In this section, we present an application of DiscoverR in locating structural repeats or repeated regions in an RNA secondary structure. Structural repeats involving trinucleotides such as CUG are present in many genomes and their expansion in specific genes causes neurological disorder or disease.[19] Repeated hairpin structures containing CAG play regulatory roles mediated by their interactions with RNA-binding proteins.[20] These structural repeats are also involved in RNA splicing.

DiscoverR can be easily extended to detect repeated regions in a given RNA secondary structure R. The logic is explained as follows. We use DiscoverR to compare R with R itself. Thus, both RT_1 and RT_2 in Fig. 6 correspond to the same structure R. As before, the algorithm maintains a two-dimensional table in which

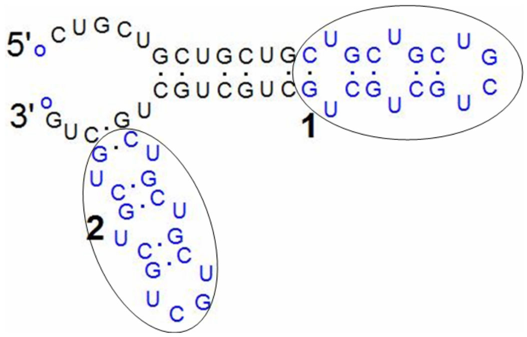


Fig. 7. Illustration of a structural repeat, circled with solid lines and highlighted in blue, detected in an RNA secondary structure in the 3′-UTR of the DMPK gene.

$c(i, j)$ represents the cell located at the intersection of the ith row and the jth column of the table. The value stored in the cell $c(i, j)$, $1 \leq i \leq |RT_1|$, $1 \leq j \leq |RT_2|$, is $\Psi(RT_1[i], RT_2[j])$. The algorithm calculates the values in the table by traversing the trees RT_1 and RT_2 in a bottom-up manner. If the value in the cell $c(i, j)$, $i \neq j$, is greater than or equal to a user-determined size threshold, the two substructures rooted at $rt_1[i]$ and $rt_2[j]$ which, respectively, are common patterns of tree $RT_1[i]$ and tree $RT_2[j]$, give rise to a repeated region or structural repeat in R. (In the study presented here, the size threshold is set to 2.) Figure 7 shows a structural repeat, highlighted in blue, DiscoverR detects in an RNA secondary structure in the 3′-untranslated region (UTR) of the DM protein kinase (DMPK) gene.[21] The repeated hairpin structure in Fig. 7 forms the genetic basis of myotonic dystrophy.[22] This example shows that the proposed method is able to detect biologically significant structural repeats in RNA molecules, thereby demonstrating the usefulness of the method.

4. Related Work

Backofen and Siebert[23] presented a method for finding common sequence structure patterns between two RNAs. These common patterns share the same local sequential and structural properties. Like the patterns found by DiscoverR, the patterns found by Backofen and Siebert are connected at the structure level (whose definition is given in Sec. 2.1). In addition, the patterns found by Backofen and Siebert are also connected at the sequence level, meaning that for any two nodes in a common substructure, there is a matched path via backbone or structure bonds that connects the two nodes. The method of Backofen and Siebert is useful in detecting local regions of large RNAs that do not share global similarities. The time complexity of their method is $O(m \times n)$, where m and n are the lengths of the two input RNAs, respectively.

Höchsmann et al.[24] developed another approach for detecting local similarities in RNA secondary structures. The authors represented RNA secondary structures as forests and devised a dynamic programming algorithm to calculate local forest alignments. These alignments gave rise to local similar regions in RNA secondary structures. The time complexity of their algorithm is $O(|F_1| \times |F_2| \times deg(F_1) \times deg(F_2) \times (deg(F_1) + deg(F_2)))$, where $|F_i|$ is the number of nodes in forest F_i and $deg(F_i)$ is the degree of F_i. Höchsmann *et al.* showed that the algorithm can discover potential regulatory motifs solely by their structural preservation, independent of their sequence conservation and position.

Mauri and Pavesi[25] employed affix trees to locate patterns in an RNA sequence (secondary structure). The time complexity of their approach is asymptotically $O(n)$ where n is the length of the sequence. The authors described in detail how to locate hairpins in the input sequence. For more complex RNA motifs, these motifs are first decomposed into single hairpins. Their approach then locates in the sequence all the single hairpins, and through post-processing, determines and identifies the complex motifs comprising the hairpins. Due to the use of affix trees, the patterns found by their approach contain contiguous bases in the RNA sequence.

Our method differs from the above-mentioned methods in two ways: (i) the discovered patterns and (ii) the algorithms used to find the patterns. Unlike the patterns found by Backofen and Siebert, which are connected both at the structure level and at the sequence level, the patterns found by our method are connected at the structure level only. For example, consider the hypothetical RNA secondary structure in Fig. 8(a). The substructure in Fig. 8(b), obtained by cutting at the two C−G base pairs as shown in Fig. 8(a), is a potential pattern that can be found by our method. However, since this pattern is not connected at the sequence level (e.g. there is no path via backbone or structure bonds connecting the two A−U base pairs

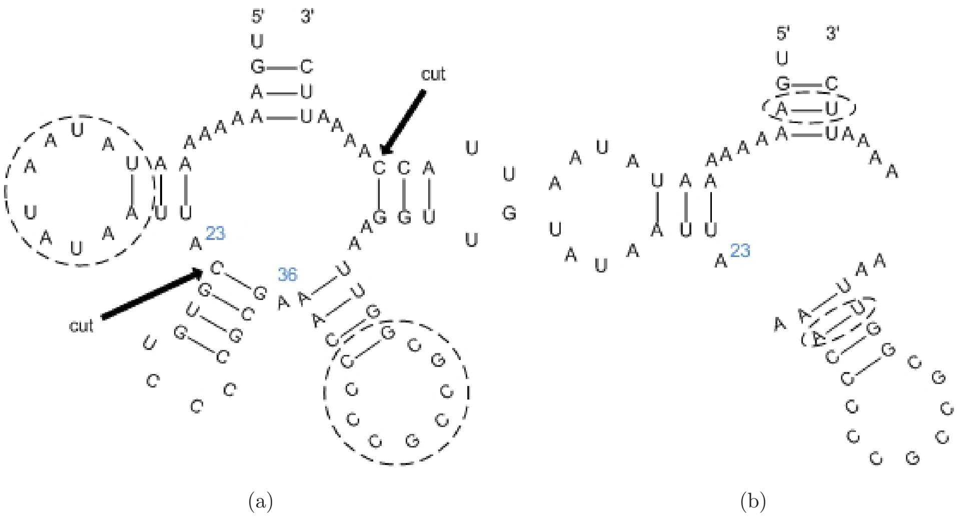


Fig. 8. Examples illustrating the differences between our method and related methods.

circled by dashed lines), the pattern in Fig. 8(b) cannot be found by Backofen and Siebert's method. Furthermore, since this pattern contains non-contiguous bases (bases between position 24 and position 35 are removed), the pattern in Fig. 8(b) cannot be located by Mauri and Pavesi's method either.

In contrast to the local alignment algorithm developed by Höchsmann *et al.*,[24] which seeks small local regions with high similarity where bases are close to each other, our method looks at the entire RNA molecules to extract their largest common substructures possibly with distant bases on the respective molecules. For example, consider again the structure in Fig. 8(a) and the structure in Fig. 8(b). When comparing these two structures, our method can find their common patterns containing the two hairpin loops circled by dashed lines by freely cutting at the two C−G base pairs as shown in Fig. 8(a). However, the local alignment algorithm would not identify these patterns due to the penalty incurred in aligning the bases on the stem-loop between position 24 and position 35 in Fig. 8(a) with gaps.

To locate the patterns with distant bases, our method employs cost-free cut operations, which do not exist in the above-mentioned methods, and hence our algorithm is totally different from the algorithms employed in the other methods. The only method that also uses cut operations for tree pattern discovery is the algorithm developed in our previous work.[4] That algorithm finds the largest approximately common substructures U_1 and U_2 of two given ordered labeled trees T_1 and T_2, where the substructure U_1 of T_1 is within edit distance d of the substructure U_2 of T_2. The time complexity of that algorithm is $O(d^2 \times |T_1| \times |T_2| \times \min(H_1, L_1) \times \min(H_2, L_2))$, where H_i, $i = 1, 2$, is the height of T_i and L_i is the number of leaves in T_i. In contrast, DiscoverR is a faster algorithm with a time complexity of $O(|T_1| \times |T_2|)$.

5. Conclusions

We proposed a new approach (DiscoverR) to finding common patterns in two RNA secondary structures. DiscoverR is an *ab initio* method, capable of identifying and extracting the largest common substructures from two RNA molecules having different sizes without prior knowledge of the locations and topologies of these substructures. To demonstrate the usefulness of the proposed method, we applied it to locating structural repeats in the 3′-untranslated region of a protein kinase gene, and described the biological significance of a repeated hairpin found by our method. In practice, DiscoverR is computationally efficient, mainly based on a quadratic-time dynamic programming algorithm, making it a suitable tool for pattern mining in RNAs. In this paper, our method is presented as the algorithm tailored to a particular class of trees, namely those used to represent RNA secondary structures. In general, our method can find the patterns induced by cost-free cut operations with the same complexity on all kinds of ordered labeled trees.

Acknowledgments

We thank the anonymous reviewers for their thoughtful comments and constructive suggestions. This research was supported, in part, by the Intramural Research Program of the NIH, National Cancer Institute, Center for Cancer Research.

References

1. Thurner C, Witwer C, Hofacker IL, Stadler PF, Conserved RNA secondary structures in Flaviviridae genomes, *J Gen Virol* **85**:1113–1124, 2004.
2. Washietl S, Hofacker IL, Lukasser M, Huttenhofer A, Stadler PF, Mapping of conserved RNA secondary structures predicts thousands of functional noncoding RNAs in the human genome, *Nat Biotechnol* **23**:1383–1390, 2005.
3. Khaladkar M, Liu J, Wen D, Wang JTL, Tian B, Mining small RNA structure elements in untranslated regions of human and mouse mRNAs using structure-based alignment, *BMC Genomics* **9**:189, 2008.
4. Wang JTL, Shapiro BA, Shasha D, Zhang K, Currey KM, An algorithm for finding the largest approximately common substructures of two trees, *IEEE Trans Pattern Analysis and Machine Intelligence* **20**:889–895, 1998.
5. Sokol D, Benson G, Tojeira J, Tandem repeats over the edit distance, *Bioinformatics* **23**:e30–35, 2007.
6. Wexler Y, Yakhini Z, Kashi Y, Geiger D, Finding approximate tandem repeats in genomic sequences, *J Comput Biol* **12**:928–942, 2005.
7. Hofacker IL, Vienna RNA secondary structure server, *Nucleic Acids Res* **31**:3429–3431, 2003.
8. Zuker M, Mfold web server for nucleic acid folding and hybridization prediction, *Nucleic Acids Res* **31**:3406–3415, 2003.
9. Spirollari J, Wang JTL, Zhang K, Bellofatto V, Park Y, Shapiro BA, Predicting consensus structures for RNA alignments via pseudo-energy minimization, *Bioinf Biol Insights* **3**:51–69, 2009.
10. Höchsmann M, Voss B, Giegerich R, Pure multiple RNA secondary structure alignments: A progressive profile approach, *IEEE/ACM Trans Comput Biol Bioinform* **1**:53–62, 2004.
11. Jiang T, Lin G, Ma B, Zhang K, A general edit distance between RNA structures, *J Comput Biol* **9**:371–388, 2002.
12. Liu J, Wang JTL, Hu J, Tian B, A method for aligning RNA secondary structures and its application to RNA motif detection, *BMC Bioinformatics* **6**:89, 2005.
13. Patel V, Wang JTL, Setia S, Verma A, Warden CD, Zhang K, On comparing two structured RNA multiple alignments, *J Bioinform Comput Biol* **8**:967–980, 2010.
14. Shapiro BA, Zhang K, Comparing multiple RNA secondary structures using tree comparisons, *Comput Appl Biosci* **6**:309–318, 1990.
15. Wang L, Zhao J, Parametric alignment of ordered trees, *Bioinformatics* **19**:2237–2245, 2003.
16. Wang Z, Zhang K, Multiple RNA structure alignment, *J Bioinform Comput Biol* **3**:609–626, 2005.
17. Zhang K, Shasha D, Simple fast algorithms for the editing distance between trees and related problems, *SIAM J Comput* **18**:1245–1262, 1989.
18. Mathews DH, Banerjee AR, Luan DD, Eickbush TH, Turner DH, Secondary structure model of the RNA recognized by the reverse transcriptase from the R2 retrotransposable element, *RNA* **3**:1–16, 1997.

19. Mankodi A, Takahashi MP, Jiang H, Beck CL, Bowers WJ, Moxley RT, Cannon SC, Thornton CA, Expanded CUG repeats trigger aberrant splicing of ClC-1 chloride channel pre-mRNA and hyperexcitability of skeletal muscle in myotonic dystrophy, *Mol Cell* **10**:35–44, 2002.
20. McLaughlin BA, Spencer C, Eberwine J, CAG trinucleotide RNA repeats interact with RNA-binding proteins, *Am J Hum Genet* **59**:561–569, 1996.
21. Brook JD, McCurrach ME, Harley HG *et al.*, Molecular basis of myotonic dystrophy: Expansion of a trinucleotide (CTG) repeat at the 39 end of a transcript encoding a protein kinase family member, *Cell* **68**:799–808, 1992.
22. Tian B, White RJ, Xia T, Welle S, Turner DH, Mathews MB, Thornton CA, Expanded CUG repeat RNAs form hairpins that activate the double-stranded RNA-dependent protein kinase PKR, *RNA* **6**:79–87, 2000.
23. Backofen R, Siebert S, Fast detection of common sequence structure patterns in RNAs, *J Discrete Algorithms* **5**:212–228, 2007.
24. Höchsmann M, Töller T, Giegerich R, Kurtz S, Local similarity in RNA secondary structures, *Proc 2nd IEEE Comput Soc Bioinform Conf*, 159–168, 2003.
25. Mauri G, Pavesi G, Algorithms for pattern matching and discovery in RNA secondary structure, *Theor Comput Sci* **335**:29–51, 2005.

Lei Hua is a Ph.D. candidate in the Computer Science Department at the New Jersey Institute of Technology. Her research interests include data mining and bioinformatics.

Jason T. L. Wang received a B.S. in Mathematics from National Taiwan University, Taipei, Taiwan, and a Ph.D. in Computer Science from the Courant Institute of Mathematical Sciences at New York University in 1991. He is a Professor of Computer Science and Bioinformatics and Director of the Data and Knowledge Engineering Laboratory at the New Jersey Institute of Technology. His research interests include data mining, databases, computational biology and bioinformatics. He has published 6 books and over 120 papers in these areas.

Xiang Ji is a graduate student in the Computer Science Department at the New Jersey Institute of Technology. His research interests include data and web mining and biomedical informatics.

Ankur Malhotra received an M.S. in Bioinformatics from the New Jersey Institute of Technology. He works as a bioinformatics researcher in the Division of Biological Sciences at the University of California, San Diego. His research interests include genomics and bioinformatics.

Mugdha Khaladkar received a Ph.D. in Computer Science from the New Jersey Institute of Technology and currently works in the Biology Department at the University of Pennsylvania. Her research interests include computational genomics and bioinformatics.

Bruce A. Shapiro received a Ph.D. in Computer Science from the University of Maryland, College Park, in 1978, with undergraduate work in mathematics and physics. He is a Principal Investigator at the Center for Cancer Research Nanobiology Program, National Cancer Institute, National Institutes of Health. His research interests include RNA nanobiology, RNA structure-function and the use of parallel high-performance computer architectures to solve problems related to RNA computational biology and molecular modeling.

Kaizhong Zhang received an M.S. degree in Mathematics from Peking University, Beijing, China, in 1981, and M.S. and Ph.D. degrees in Computer Science from the Courant Institute of Mathematical Sciences, New York University, New York, USA, in 1986 and 1989, respectively. He is currently a full professor in the Department of Computer Science, University of Western Ontario, London, Ontario, Canada. His research interests include bioinformatics, algorithms, image processing and databases.