Towards a Weighted-Tree Similarity Algorithm for RNA Secondary Structure Comparison

Jin, J.; Sarker, B.K.; Bhavsar, Virenda; Boley, Harold; Yang, L.

Publisher's version / Version de l'éditeur:

Proceedings of the IEEE Computer Society, 8th International Conference on High Performance Computing in Asia Pacific Region (HPC Asia 2005), 2005

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

*Towards a Weighted-Tree Similarity Algorithm for RNA Secondary Structure Comparison **

Jin, J., Sarker, B.K., Bhavsar, V.C., Boley, H., and Yang, L.
November 2005

* published in The Proceedings of the 8th International Conference on High Performance Computing in Asia Pacific Region (HPC Asia 2005) Beijing, China. IEEE Computer Society. November 30 – December 3, 2005. pp. 639-644. NRC 48536.

Canada

Towards a Weighted-Tree Similarity Algorithm for RNA Secondary Structure Comparison

Jing Jin[1], Biplab K. Sarker[1], Virendra C. Bhavsar[1], Harold Boley[2], Lu Yang[1]
[1] *Faculty of Computer Science, University of New Brunswick, Fredericton, Canada*
[2] *National Research Council of Canada, Fredericton, New Brunswick, Canada*
[1] *{jing.jin, sarker, bhavsar, lu.yang} AT unb.ca* [2] *harold.boley AT nrc-cnrc.gc.ca*

Abstract

A tree similarity algorithm for RNA (ribonucleic acid) secondary structure comparison is presented. The elements (nucleotides and nucleotide-pairs) of an RNA secondary structure are represented as normalized node-weighted trees. We show that our weighted tree representations of RNA secondary structures are informative and useful. Based on this unique representation for RNA secondary structure, we propose a weighted-tree similarity algorithm for computing the similarity between RNA secondary structures. The algorithm is justified by computing similarities among several well-known RNA secondary structures. For a given RNA secondary structure, the proposed algorithm provides a ranked list of RNA structures in a database according to their similarity values with the query RNA. Hence, our algorithm is helpful in predicting the functions and the class of a newly discovered RNA.

1. Introduction

The well known central dogma of molecular biology is "Information passes from DNA (deoxyribonucleic acid) to RNA to Protein" [9]. It is believed that DNA carries the genetic material, but there are exceptions of special viruses such as HIV where RNA contains the genetic material. Therefore, RNA is an important part of a cell that regulates the functions of such deadly viruses.

The comparison of RNA secondary structures is one of the basic computational problems appeared in literature [1, 4, 5, 7, 8, 10]. The objective is to find the relative common sections of known RNA structures, and consequently, find the overall similarities among the whole RNA structures. For a newly discovered RNA, it can give a clue to its functions and class to which it might belong in a database. This can also be considered as a structural motif discovery problem in RNA [1].

Basically, the primary structure of an RNA can be identified as a sequence of four nucleotides (bases): A (adenine), C (cytosine), G (Guanine) and U (uracil). These bases can form base-pairs (nucleotide-pairs), conventionally A pairs with U and C pairs with G. In addition, G pairing with U is also frequently observed. It can be represented as an edge between two complementary bases involved in the bonds. It is assumed that any base exists at most in one such pair and the edges of the bonded pairs are non-crossing. This implies a node-labeled tree-like structure [10].

A number of node-labeled tree models have been proposed in [7] and discussed in [1] for representing RNA secondary structures. The work on RNA comparisons using node-labeled trees [4, 5, 7, 8] does not take into account the base-paired nucleotides and unpaired nucleotides, and lacks in defining semantics of the process of transforming one RNA into another. To overcome these difficulties the work proposed in [10] uses paired and un-paired nucleotides and applies basic tree edit distance operations such as insertion, deletion and relabeling on them. However, Allali et al. [1] found limitations of this approach and introduced two additional operations, the so-called node-fusion and edge-fusion and then proposed a dynamic programming algorithm for comparing two RNA structures. Another method for comparing two RNA trees [2, 3] is based on aligning the trees and using the score of alignment as a measure of the distance between the trees.

However, trees only having node labels do not contain the information of RNA secondary structures such as: (a) how to find the importance of a nucleotide/base-pair in an RNA tree, and (b) how to guess and get information if some subtree(s) of an RNA tree are missing. Therefore, we propose a new

representation for RNA trees, where not only node labels but also node weights are incorporated to embody the semantics and importance of nodes in an RNA tree.

The contributions of the paper are two folds. We first give formal definitions of node-labeled, node-weighted and normalized node-weighted trees representing RNA secondary structures. We show that the normalized node-weighted tree is more informative, useful and contains most of the attributes of an RNA secondary structure. Second, we propose a tree similarity algorithm for RNA tree comparison and show that it is effective and useful for the comparison of RNA secondary structures represented as node-weighted trees. To our knowledge, in the field of computational biology, node-weighted tree representations and their similarity measures for RNA secondary structures have not been studied.

2. RNA secondary structures and representations

2.1. Tree representation

Fig. 1 shows a simple RNA secondary structure [1] and node-labeled tree. However, this kind of tree is inadequate in representing the semantic information of an RNA secondary structure. Here, we discuss the disadvantages of the node-labeled tree by examples and propose possible approaches to overcome them.

Case 1. The importance of a nucleotide or a base-pair in the node-labeled tree is not apparent.

Let us consider the base-pair *C-G* under the root node in the RNA tree (Fig. 1). There is no information showing how significant this base-pair is for the whole tree or for the subtree underneath. Similarly, its siblings (leaf nodes) do not reveal their importance compared to the *C-G* or the whole tree as well. We propose to add node weight to each node to indicate its relative importance to its siblings and the whole tree.

Case 2. It is hard to find that if some subtrees (substructures) of the tree are absent or missing.

For Fig. 1, let us consider the subtree that is rooted at the base-pair *U-A* at level 7. If the subtree is missing, it is difficult to guess that there is something missing from the whole tree. However, if we assign weights to the base-pair *U-A*, then the node-weight on *U-A* will be larger than siblings in its neighborhood (other leaf nodes) because it has a subtree underneath. Thus, when a subtree is missing from *U-A*, we can easily guess that it has a subtree underneath from its node-weight. We can also get clue even for a small part of a subtree such as a leaf node from the weight on the node if it is missing.

It is very much likely that these situations (case 1 and case 2) can also arise while comparing RNA secondary structures represented as trees. We propose a new representation of node-labeled RNA trees by assigning weight to every node (Fig. 2). The following subsection defines our new trees with examples.

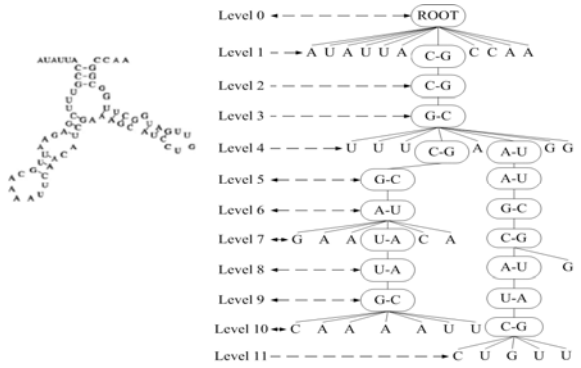

Figure 1. An RNA secondary structure [1] and its corresponding node-labeled tree

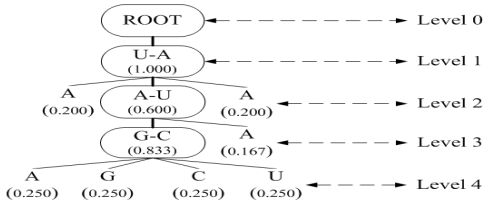

Figure 2. A normalized node-weighted tree representation of an RNA secondary structure

2.2. Definitions

Definition 1. Node-labeled tree. *A tree $T = (V, E, L_V)$ is a 3-tuple where V, E and L_V are sets of nodes, arcs and node labels, respectively, which satisfies the following conditions:*

1. One element in V is designated as the 'root'.

2. Each element in E connects a pair of elements in V.

3. There is a unique directed path, consisting of a sequence of elements in E, from the root to each of the other elements in V.

4. There is an $(n \rightarrow 1, n \geq 1)$ mapping from the elements in V to the elements in L_V (i.e. different nodes can carry the same labels).

Definition 2. Node-weighted tree. *A node-weighted tree is a 4-tuple $T = (V, E, L_V, V_W)$ of a set of nodes V, a set of arcs E, a set of node labels L_V, and a set of node weights such that (V, E, L_V) is a node-labeled tree and there is an $(n \rightarrow 1, n \geq 1)$ mapping from the*

elements in V to the elements in V_W (i.e. different nodes can carry the same weights).

Definition 3. Normalized Node-weighted tree. *A normalized node-weighted tree is a node-weighted tree $T = (V, E, L_V, V_W)$ having a fan-out-weight-normalized $(n \rightarrow 1,\ n \geq 1)$ mapping from the elements in V to the elements in V_W (i.e. the weights of every fan-out add up to 1.0).*

Such normalized trees are assumed in the rest of this paper. Node weights are assigned according to the importance of a nucleotide/base-pair in the tree and described in the next subsection.

2.3. Weight assignment

The weight of a leaf node is assigned according to

$$A_{WL} = 1/(N + \sum_{i=1}^{p}(N_{L_i} + N_{B_i}))$$

where,

A_{WL}: weight of the leaf node

N: no. of siblings at the same level in the same subtree

p: no. of non-leaf nodes at the same level in the same subtree

N_{L_i} : no. of leaf nodes that are the children of a non-leaf node

N_{B_i} : no. of non-leaf nodes in the subtree of a non-leaf node

Note that the sibling leaf nodes must have identical weights. The node weight of a non-leaf node is defined as follows.

$$A_{WB} = A_{WL}(N_L + N_B + 1)$$

where, A_{WB} represents the weight of a non-leaf node.

Note that the weights of siblings at any level must add up to 1.0, as given below:

$$\sum_{i=1}^{k}(A_{WL_i} + A_{WB_i}) = 1.0$$

where, k represents the number of siblings at the same level in a subtree.

If a node does not have siblings, then its weight is 1.0. The base-pairs tend to have larger weights than the nucleotides because they probably have bigger substructures below them. And thus they have a larger effect on the whole RNA secondary structure. Let us consider the level 2 of the tree in Fig. 2 as an example. The number of nodes at this level is 3. The number of leaves that are children of the only non-leaf node *A-U* at this level is 1, i.e. the leaf node *A* at level 3. And the number of non-leaf nodes in the subtree of *A-U* is 1, i.e. the non-leaf node *G-C* at level 3. Thus, using (1) the node weight of the leaf nodes at level 2 is A_{WL} = 1/(3+1+1) = 0.2, and using (2) the node weight of base-pair *A-U* is A_{WB} = 0.2 (1+1+1) = 0.6, therefore, according to (3), (0.2 + 0.6 + 0.2) = 1.0.

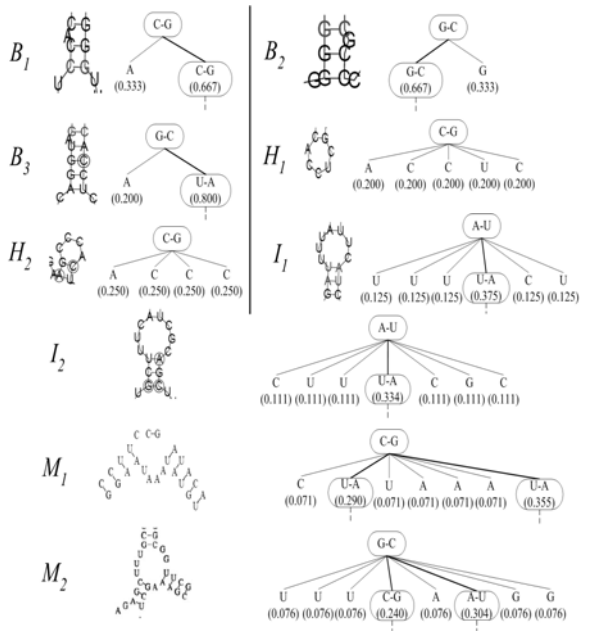

Figure 3. The Bulge (*B*), Interior (*I*), Hairpin (*H*) and Bifurcation (*M*) loops and corresponding node-weighted trees

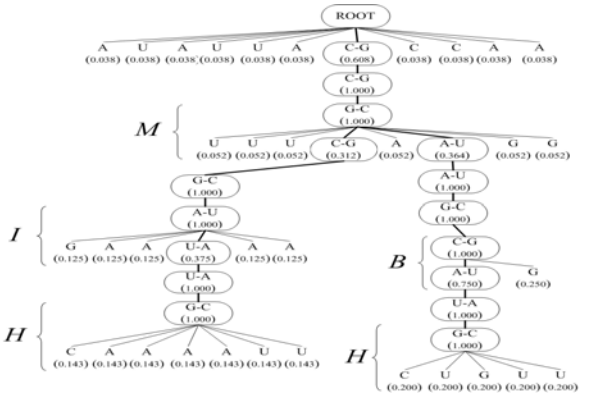

Figure 4. The node-weighted tree of the RNA [1] given in Figure 1

2.4. Tree representations for RNA loops

The RNA secondary structure consists of four important loops, the Bulge (*B*), Interior (*I*), Hairpin (*H*) and Bifurcation (*M*). All the RNA secondary structures can be represented as a tree (Fig. 3) from these loops, the base-pairs and unpaired nucleotides [4]. Fig. 4 represents the node-weighted tree of the RNA in Fig. 1, which contains all the four loops.

3. Comparison of RNA trees

After defining RNA secondary structures as node-weighted trees, we compare our RNA trees (i.e. node-weighted RNA trees) and calculate the similarities between them according to the algorithm given in Fig. 5.

3.1. RNA tree similarity algorithm

```
RNAtreesim(t, t')
BEGIN
    IF t and t' have subtree(s) THEN
        IF t and t' have identical root nodes THEN
            their root nodes' similarity is 1.0
            FOR i = 1 to Breath_of_t, j = 1 to Breath_of_ t'
                RNAtreesim(t_i, t'_j)
                IF i = Breath_of_t (j = Breath_of_ t') THEN
                    RNAtreeplicity(t'_j) (RNAtreeplicity(t_i))
                ENDIF
            ENDFOR
        ELSE
            the two trees' similarity is 0.0
            comparison is terminated
        ENDIF
    ENDIF
    IF only t (t') is a leaf node THEN
        RNAtreeplicity(t) (RNAtreeplicity(t'))
        // let t''=the tree rooted at the next (right) sibling of t
        RNAtreesim(t'', t')
    ENDIF
    IF t and t' both are leaf nodes THEN
        IF t and t' have identical root nodes THEN
            their root nodes' similarity is 1.0
        ELSE
            their root nodes' similarity is 0.0
            comparison is terminated
        ENDIF
    ENDIF
    RETURN the similarity value of t and t'
END
```

Figure 5. Algorithm for computing the similarity between RNA secondary structures

The pseudocode of our RNA tree similarity algorithm is given in Fig. 5. The main function *RNAtreesim*(t, t') recursively traverses the two trees t and t' in a depth-first strategy and then computes their similarity bottom-up. The equation embedded in this function that computes the similarity of two RNA trees is shown below.

$$S(t, t') = \sum_{i=1}^{n} ((w_i + w'_i)/2) S(t_i, t'_i) \quad (4)$$

where,

$S(t, t')$: similarity of two input trees t and t'

$S(t_i, t'_i)$: intermediate similarity of the i^{th} subtrees of the two trees t and t'

w_i *and* w'_i: node weights of the i^{th} child of the root node of tree t *and* t', respectively

When two trees t and t' both have subtrees underneath with different root nodes, the similarity of two trees is 0.0. However, if they have identical root nodes, i.e. the similarity of root nodes is 1.0, then the similarity of the two trees is computed via *RNAtreesim*(t_i, t'_j) by a recursive top-down traversal through the subtrees, t_i and t'_j. If branches of a tree (t)'s reach the end during the traversal at the same level, all the remaining branches in the other tree (t') are missing. The similarity of the missing subtrees is found by using their simplicity value multiplied by 0.5, using simplicity function *RNAtreeplicity*(t'_j). The equation embedded in the *RNAtreeplicity*(t) that computes the simplicity of the missing tree is shown below.

$$Simp(t) = \begin{cases} D_I \cdot (D_F)^d & \text{if } t \text{ is a leaf node,} \\ \frac{1}{m} \sum_{j=1}^{m} w_j \cdot Simp(t_j) & \text{otherwise.} \end{cases} \quad (5)$$

where,

Simp(t): simplicity value of a single tree t

D_I *and* D_F: depth degradation index and depth degradation factor

d: depth of a leaf node

m: root node degree of tree t that is not a leaf

w_j: node weight of the j^{th} child of the root node of tree t

t_j: subtree below the j^{th} node with node weight w_j

With a single tree t as input, this simplicity measure is defined recursively to map an arbitrary single tree t to a value from [0, 1], decreasing with both the tree breadth and depth. The recursion process terminates when t is a leaf node. For a (sub)tree, the simplicity is computed by a recursive top-down traversal through its subtrees. Basically, the simplicity value of a tree t is the sum of the simplicity values of its subtrees multiplied with node weights from [0, 1], a subtree depth degradation factor ($D_F \leq 0.5$), and a subtree breadth degradation factor from (0, 1].

The similarity of the missing subtree and the corresponding empty subtree in the other tree is computed by multiplying the simplicity value of the missing subtree with 0.5. If only one of t and t' is a leaf node, for example t is a leaf node, then this leaf node is missing in the other tree t'. We call the simplicity function *RNAtreeplicity*(t) and continue the comparison between the next (right) sibling of the leaf node t and t'. Finally, if t and t' are both leaf nodes, then their similarity is 1.0 when they have identical root node. Otherwise, their similarity is 0.0.

3.2. Illustrative examples

We give an illustrative example to show how the proposed algorithm works step by step (Fig. 6). The comparison starts at the identical ROOT nodes of the trees a and b (level 0), and traverses to level 1. After finding that the root node labels are both identical, i.e. both are *A*-*U*, it goes to level 2 and starts comparison from left to right. In Fig. 6, *U* at the first branch of the tree a matches *U* at the first branch of the tree b. When two *U*s at the second and third branches of the tree a are missing in the tree b, the simplicity function in

equation (5) is called. The simplicity value for one missing node is $0.9 \times (0.5)^0 = 0.9$. Here, we assume in our case, D_I = 0.9, D_F = 0.5, and d = 0 because they are leaf nodes. As we defined in Section 3.2, the similarity value of the missing *U* with the corresponding empty leaf node is $0.9 \times 0.5 = 0.45$, i.e. the simplicity value 0.9 is multiplied with 0.5. Next, the base-pair *U-A* at the fourth branch of the tree *a* matches *U-A* at the second branch of the tree *b*. Their similarity is 1.0. Since our algorithm traverses the two input trees in a left-right depth-first strategy, so the similarity function first recursively traverses the subtree under base-pair *U-A* at level 2, then continues the comparison among the remaining branches. The comparison traverses to level 3 and finds two identical base-pairs *C-G*, and then continues to level 4. At level 4, i.e. the bottom of the two trees, the comparison starts from left to right. The leaf nodes at the first, second and the third branches are identical. So, the similarity is 1.0. The *U* at the fourth branch of the tree *a* mismatches *C* at the fourth branch of the tree *b*. Therefore, their similarity is 0.0. And *C* at the fifth branch of the tree *a* is missing. Consequently, the similarity value is 0.45, i.e. $0.9 \times (0.5)^0 \times 0.5 = 0.45$.

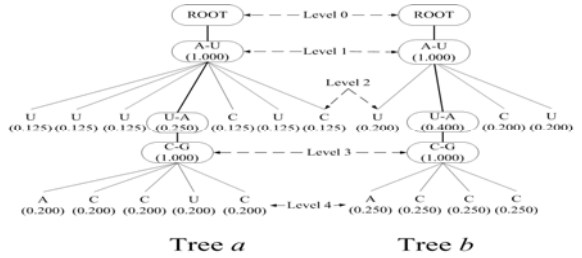

Figure 6. An example of computing the similarity between two RNA trees

Now, the algorithm starts to calculate the similarity bottom-up. The similarity of the subtree with root node *C-G* at level 3, according to (4), is 3(1.0((0.2+0.25)/2)) + 1(0.0((0.2+ 0.25)/2))+ 1(0.45((0.25+ 0.0)/2))= 0.731. The similarity of the subtree with the root node *U-A* at level 2 is 1(0.731((1.0+1.0)/2)) = 0.731 using (4). At level 2, the leaf nodes *C* and *U* at the fifth and sixth branches of the tree *a* match the leaf nodes *C* and *U* at the third and fourth branches of the tree *b*. Therefore, the similarity is 1.0. However, the leaf node *C* at the seventh branch of tree *a* is missing. Its similarity value is 0.45, i.e. $0.9 \times (0.5)^0 \times 0.5 = 0.45$. The similarity of the subtree with the root node *A-U* at level 1 is 3(1.0((0.125+0.2)/2))+3(0.45((0.125+0.0)/2))+1(0.731 ((0.25+0.4)/2)) = 0.809. Finally, the similarity of the two trees is 1(0.809((1.0+1.0)/2)) = 0.809 which is reasonable because the trees *a* and *b* are very similar.

Next, we provide more examples of computing similarity using all RNA trees presented in Fig. 3 which are calculated analogously. The similarity values of the four basic loops are given in Table 1.

Table 1. Computational results for the examples given in Figure 3

Tree	Tree	Similarity
B_1	B_2	0.816
B_1	B_3	0.267
B_2	B_3	0.120
H_1	H_2	0.720
I_1	I_2	0.733
M_1	M_2	0.446

4. Computational results

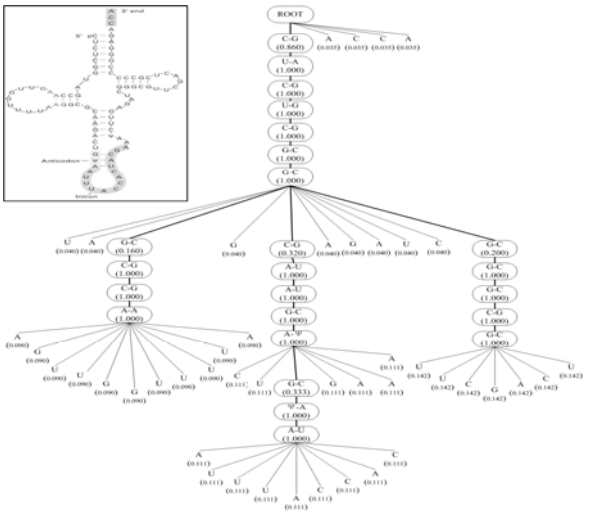

Figure 7. The node-weighted tree representation of the precursor tRNA.Tyr

In this section, we provide the computational results of several well-known RNA secondary structures and their corresponding node-weighted trees. Fig. 7 and 8 display the secondary structures and corresponding node-weighted trees of the precursor and mature yeast tRNA.Tyr [6]. The similarity value between them is 0.741, i.e. most of their substructures are identical. The similarity between the RNAs shown in Fig. 9 and the example in Fig. 1 is 0.086. In addition, we compare the similarity between the RNA in Fig. 10 and the mature tRNA.Tyr (Fig. 8), and we obtain the similarity value 0.16. Based on all of the examples described above, we can conclude that if two RNA secondary structures are

intuitively more similar, the similarity value should be higher. Otherwise, it should be closer to 0.

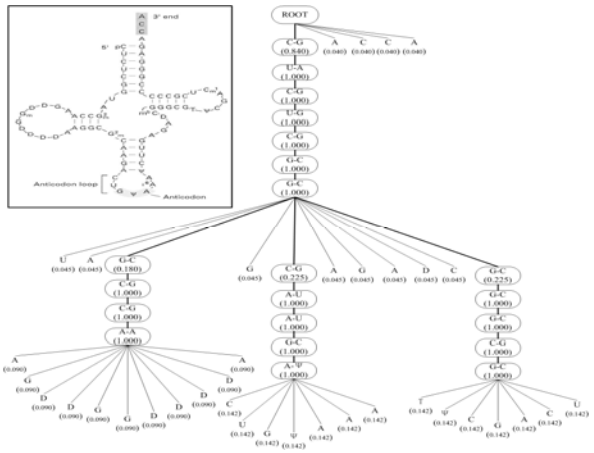

Figure 8. The node-weighted tree representation of the mature tRNA.Tyr

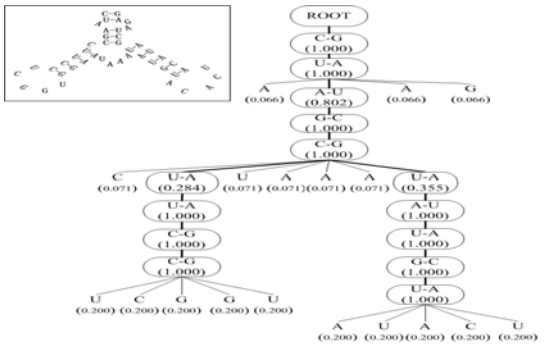

Figure 9. The RNA secondary structure [10] and the corresponding node-weighted tree representation

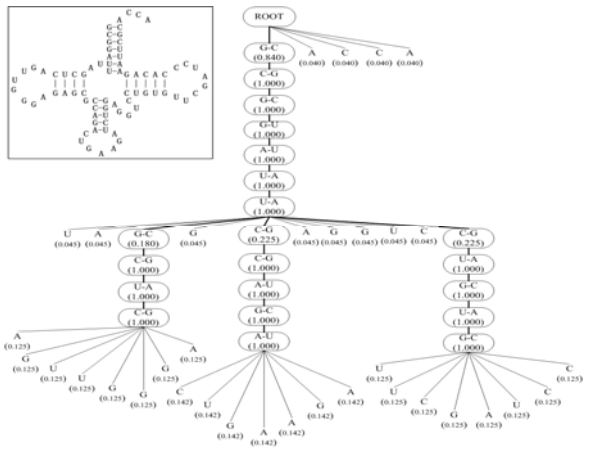

Figure 10. The RNA secondary structure [1] and the corresponding node-weighted tree representation

5. Conclusion

RNA databases are growing rapidly and therefore it is essential to develop appropriate representations for RNAs to incorporate all of their attributes. Further, efficient algorithms to find similarity of RNA secondary structures are required. In this paper, we have proposed a new representation (node-weighted tree) of RNA secondary structure. Our representation contains most of the information of an RNA secondary structure. We have also showed that our proposed tree similarity algorithm can find the similarity between RNA secondary structures and give similarity values to consistent with our intuitive understanding. At present, we plan to build a database with our RNA trees and cluster them according to their groups/functions. This would be used to compare and find similarities with newly discovered RNA structures and thus predict their functionality. Further, we plan to carry out extensive comparison with other similarity algorithms in terms of accuracy, efficiency and complexity.

References

[1] J. Allali and M.F. Sagot, "A New Distance for High Level RNA Secondary Structure Comparison", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2005, Vol. 2, No. 1, pp. 3-14.

[2] M. Höchsmann, T. Töller, R. Giegerich, and S. Kurtz, "Local Similarity in RNA Secondary Structures", *Proc. Com.. Sys.. Bioinformatics (CBS)*, 2003, pp. 159-168.

[3] T. Jiang, L. Wang, and K. Zhang, "Alignment of Trees-An Alternative to Tree Edit", *Proc. Fifth Ann. Symp. Comb. Pattern Matching (CPM)*, 1994, pp. 75-86.

[4] S.Y. Le, R. Nussinov, and J.V. Mazel, "Tree Graphs of RNA Secondary Structures and Their Comparisons", *Comput. Biomed. Res.*, 1989, Vol. 22, pp. 461-473.

[5] S.Y. Le, J. Owens, R.Nussinov, J.H. Chen, B. Shapiro, and J.V. Mazel, "RNA Secondary Structures: Comparisons and Determination of Frequently Recurring Substructures by Consensus", *Computer Applications in the Biosciences.*, 1989, Vol. 5, pp. 205-210.

[6] P.J. Russell., *Genetics, 3rd ed.*, Harper Collins Publishers Inc., New York, 1992.

[7] B. Shapiro, and K. Zhang, "Comparing Multiple RNA Secondary Structures Using Tree Comparisons", *Com. App. Biosci.*, 1990, Vol. 6, No. 4, pp. 309-318.

[8] B. Shapiro, "An Algorithm for Comparing Multiple RNA Secondary Structures", *Computer Applications in the Biosciences*, 1988, Vol. 4, No. 3, pp. 387-393.

[9] M.S. Waterman, *Introduction to Computational Biology: maps, sequences, and genomes*, Chapman and Hall publishers, London, 1995.

[10] K. Zhang, "Computing Similarity between RNA Secondary Structures", *Proc. IEEE Inter. Joint Symp. Intelligence and Systems*, 1998, pp. 126-132.