A Framework to Generate Arc-annotated Sequence Data for Evaluating RNA Analysis Algorithms

Sen Zhang [*] George Chang [†]

Abstract— **We present an algorithmic framework for generating synthetic RNA data conforming to the arc-tree model, a tree extension of the RNA arc annotated sequence model that has been under intensive research during the past decade. The arc-tree model aims to capture the secondary structures of RNA data: the parent-child and their derived ancestor-offspring relationships between tree nodes represent the nested arcs of RNA; the sibling relationships represent the parallel arcs. Furthermore, we extend the tree model to a rooted graph model for representing tertiary structures of RNAs. The synthetic arc-annotated RNA data such generated is expected to facilitate evaluating the effectiveness and completeness of the RNA analysis algorithms designed for the arc-annotated sequence model.** ***Keywords: Synthetic Data, Generator, RNA, Arc-annotated, Tree Model***

1 Introduction

Different from double-stranded deoxyribonucleic acid (DNA)s, Ribonucleic acid (RNA)s are usually single-stranded [4]. However, RNAs are still structurally interesting because these single stranded RNAs may present complex secondary and tertiary spatial structures. The combination of these superlinear RNA structures that appear in diverse RNA data including messenger RNA, transfer RNA and ribosomal RNA [4] has posed great challenges to the full spectrum of RNA analysis algorithms including comparison, alignment, distance calculating, prediction, pattern mining and motif discovery etc.

To facilitate the designs of various deterministic and heuristic RNA analysis algorithms, computational scientists have proposed several RNA data models such as tree model, arc model and loop model. Due to the fact that any abstract data model usually ignores certain constraints of the real world datasets to some extent, the space of the data of a simplifying model is actually much greater than that of the real world data. However, most bioinformatics practitioners tend to restrict the data used in their RNA analysis algorithm evaluations to the limited real world datasets only. This practice may suffer the following major limitations among many others. First, publicly available RNA data with clearly annotated secondary and tertiary structures is insufficient. Second, as pointed out earlier, the data model considers the real world data only as a subspace of its modeling space, therefore, the uncaptured characteristics of the real world data may bias the evaluations of RNA algorithms that are actually designed for an implicitly expanded data scope due to the simplifying model. Third, the user has very little control over the characteristics of the real RNA data.

These limitations can be considerably overcome by using synthetic data, especially when the major concern of the experiments is on evaluating the effectiveness and completeness of an RNA analysis algorithm. A synthetic data generator can produce not only large volumes of synthetic RNA data resembling real data, but also datasets covering all the possible subspaces of the model through varying the values of the different input parameters. These datasets can be used to test RNA algorithms more efficiently than the limited real world data can.

In this paper, we discuss an algorithmic framework for generating synthetic arc-annotated RNA data mainly for facilitating evaluation of research ideas and testing the performance (scalability, accuracy, efficiency etc) of various RNA algorithms.

The rest of the paper is organized as follows. Section 2 first discusses the tree model used in our generator. Then in section 3, we discuss the details of the framework step by step. Finally, we conclude the paper by reporting the preliminary implementation of the framework and discussing the possible improvements as our future work.

[*]Department of Mathematics, Computer Science and Statistics & State University of New York, College at Oneonta, Oneonta, NY USA 1380. Email: zhangs@oneonta.edu.

[†]Department of Computer Science, Kean University, Union, NJ 07083 Email: gchang@kean.edu

2 Tree Model

Conventionally, the primary sequence of an RNA data is treated as its linear structure, secondary structure as a tree, and tertiary structure a rooted ordered graph. It follows intuitively that in order to generate synthetic RNA data, we just need to generate the primary sequence first, then use the primary sequence to predict in tandem its secondary and tertiary structures through secondary structure prediction software [2] and tertiary structure prediction software[1] respectively. This seemingly straightforward approach is actually problematic for the following reason. A generated primary sequence tends to lack sufficient base pair support that is highly expected by most real world RNA sequences. Consequently, the chance for the random sequence to support pairwise segments of complementary subsequences that contribute to non-trivial higher level structures is very slim. As we will discuss in the next section, our generator addresses this issue by partially reversing the above misleading steps.

To facilitate the subsequent discussion on our framework, we first recapitulate two well-known RNA tree models: motif tree model and arc tree model. Figure 1 illustrates an RNA sequence composed of secondary structural motifs of stems, bulges, hairpins and loops etc. In the well-known motif tree model [7], this RNA is treated as a motif tree with each node representing a motif instance. Since there is no any specific biological meaning for a root, we treat the motif tree as unrooted. To help visualizing the higher level motif tree, we use Figure 2 to highlight only the secondary structures of an imaginary RNA. In the figure, the topleft subfigure shows the layout of the secondary structure by suppressing the details of the primary sequence, while the topright subfigure shows the unrooted tree structure formed by the nodes respectively representing one loop, three stems and two hairpins. The mapping relationships between the nodes of the tree model and the motifs of RNA data are represented by the double-arrowed dotted mapping curves.

From the figure, it can also be observed that the consecutive base pairs play an important role in forming RNA structures. Every RNA motif is actually a substructure delineated by several stems adjacent to it. Based on this observation, a more interesting arc tree model [6] has been proposed, where the constituent nodes represent not the motifs directly but individual arcs (stems in the arc model). Every node in the arc tree represents a possible arc starting a consecutive arc group (stem). An arc tree is rooted, where the root of the tree represents the whole sequence of RNA data covering the positions from the first base to the end, which usually is of a zero sized virtual stem for most cases. The children nodes of the root node represent the outmost parallel arcs of the RNA, each of which in turn has children/offspring representing the arcs cascadingly nesting in it at lower levels. The bottom-right subfigure of Figure 2. shows how such an arc tree can be constructed.

In terms of this arc-annotated model, the RNA data in Figure 1 can be represented by figure 3 in the arc annotated sequence format (the bottom portion of Figure 3). which is then represented by the higher level arc tree (the top portion of Figure 3). Notice that how the numbers 1 to 10 in Figure 1 are mapped to the corresponding arc feet in Figure 3 and how they are paired yet separated by the unpaired segments. The arc annotated tree an RNA data is ordered because the relative positions of the arcs are significant to the secondary and tertiary structures. We call this tree an arc skeleton instance tree or simply the arc tree whenever no confusion arises.

This arc annotated model has been used extensively in many RNA analysis algorithms[5]. It is also the base model for our generator framework, where the arc tree model is enhanced to a rooted graph model where the crossing arcs of tertiary structures that voilate the tree structure can be accomodated as graph components.

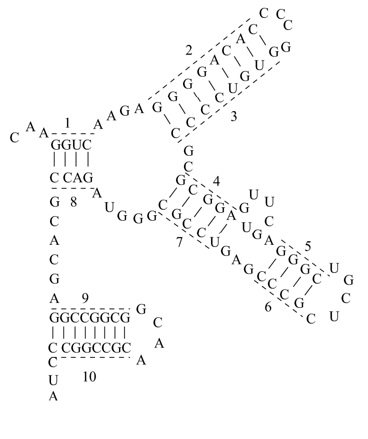


Figure 1: An illustration of an RNA structure

3 Generating the Random Arc-annoated RNA Data

Following the previous discussion on the arc tree/graph model, we were able to develop a generator framework which may appear to be a little anti-intuitive. Instead of generating primary, secondary and tertiary structures in that order, our generator produces the tree structure first, which corresponds the overall secondary structure; then we augment the tree to a rooted graph in order to define the cross arc groups (Arcs will be used whenever the context is clear); after that, we generate the feet positions and sizes of the arcs; finally, based on the higher level structures obtained from the previous stages, we generate the sequence symbols. The framework of our

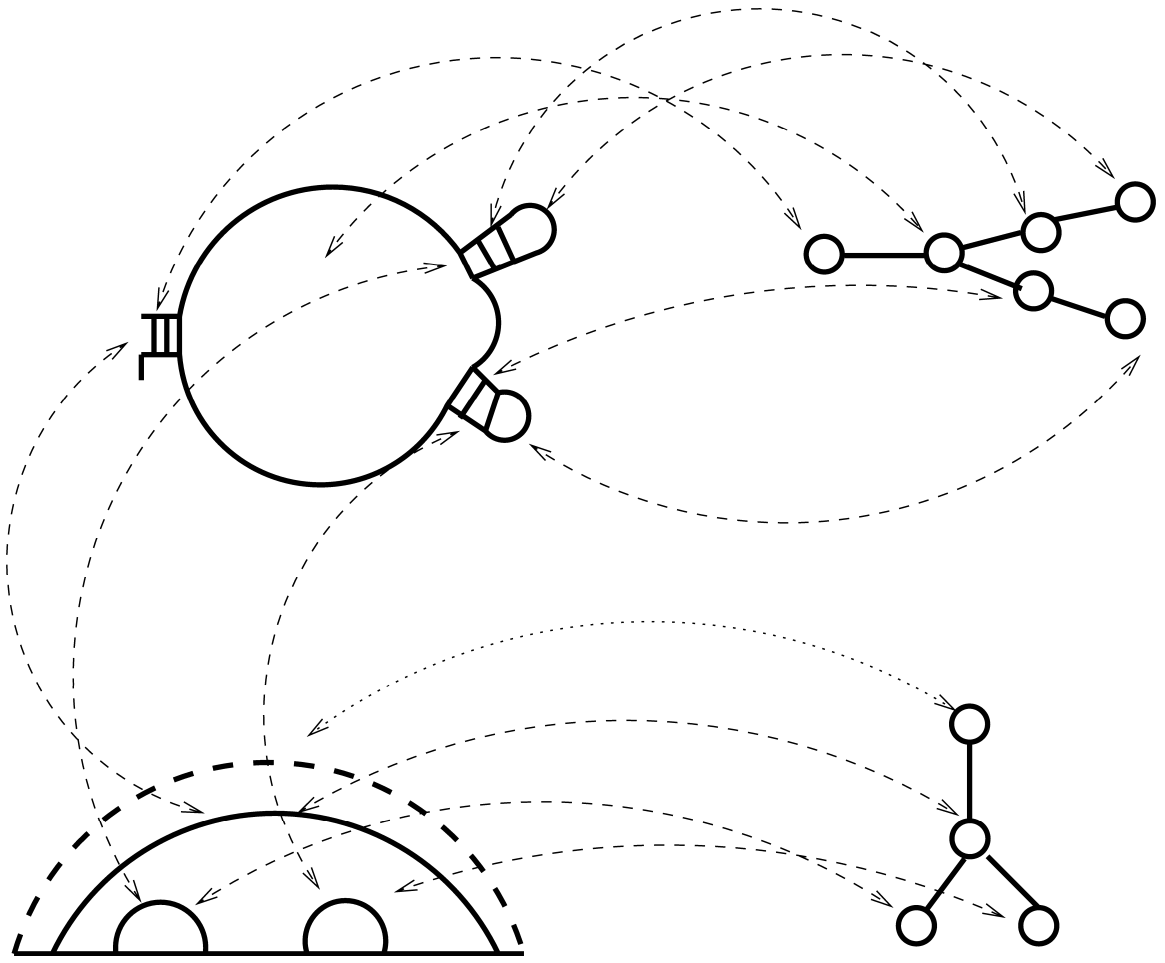


Figure 2: Two RNA tree models

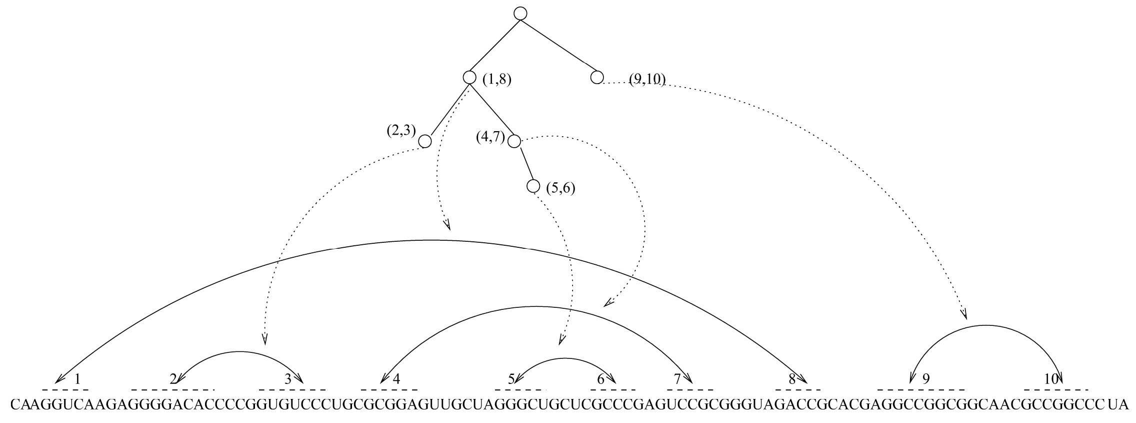

(1,8)
(9,10)
(2,3)
(4,7)
(5,6)
1
2
3
4
5
6
7
8
9
10
CAAGGUCAAGAGGGGACACCCCGGUGUCCCUGCGCGGAGUUGCUAGGGCUGCUCGCCCGAGUCCGCGGGUAGACCGCACGAGGCCGGCGGCAACGCCGGCCC UA

Figure 3: An arc-annotated model

generator is outlined in Figure of Algorithm 1 and relatively more details of the algorithm will be discussed phase by phase in the subsequent subsections.

```
/* Procedure:
   arcannotatedrna(controlparameterlist)      */
input  : Tree Control Parameters
output: Synthetic arc-annotated RNA data
1.1 Generate skeleton Tree;
1.2 Augment to Rooted Graph;
1.3 Populate the characteristics of stem and free
    segments ;
1.4 Fill out sequences of arcs and free segments;
1.5 return the arc-annotated RNA sequence
```

Algorithm 1: Acr annotated RNA Data Generation.

3.1 Generating the Tree Structure

Generating an arc tree skeleton is the first, also the core, phase of the whole process. In this phase we focus on the hierarchical structures of the tree without considering the details of the primary sequence of the RNA. In the tree, parallel arcs are represented by sibling nodes, while nested arcs are represented by parent-child relationships and their lineage derivations.

To help generate the tree, we consider the following input parameters the depth of tree D, the number of nodes N, the minimum and maximum fanouts of tree nodes: $minf$, $maxf$, respectively. Given reasonable values for the above parameters, our generator can generate complex trees of different shapes. The tree generator proceeds in the depth-first traversal manner, i.e., to expand a tree topdown from the root to leaves. The number of children of an internal node at any particular level of the tree is randomly chosen subjecting a uniform distribution over the range bounded by the input parameters $maxf$ and $minf$. To understand what the target trees generated at this stage look like, refer to the example in the top subfigure of Figure 3.

3.2 Augmenting Trees to Graphs

In this phase, we augment the RNA skeleton tree by using new nodes representing the crossing arcs to generate the placeholders for the tertiary structures. We consider two kinds of tertiary arc groups here. The first kind is the arc whose two feet fall into two different secondary arcs at the same level respectively. The second one is the tertiary arc whose two feet falling inbetween two secondary arc or unpaired segments at different levels.

To be semantically consistent with the nodes representing the secondary arcs, each tertiary arc is also represented by a node. However, since a tertiary arc crosses with other arcs, we need to link a tertiary node to the two parent nodes representing the two secondary arcs hosting its two feet. As a result, we enhance a tree to a rooted graph by admitting graph edges. The two anchoring feet of a tertiary arc should originally be either sibling nodes or ancestor-offspring seating on or off the same root-to-leaf path.

The augment is based on two assumptions: the least influential principle and simple tertiary structures. By the least influential principle, we mean among all the arcs, we would like to identify the secondary structures as many as possible first from left to right, and only the rest of the arcs that cross the claimed nested secondary or parallel secondary arcs will then be treated as the crossing arcs, representing tertiary structures. The simple tertiary means tertiary arcs do not host feet of other tertiary arcs.

To understand the idea refer to Figure 4, where the lowest node represents a stem connecting to two parenting nodes at different levels. The figure also shows that all the tertiary nodes will be at the lowest level of the rooted graph. This is consistent with the simple tertiary rule. Otherwise, we can always re-identify the secondary structures and to push all the tertiary nodes to the bottom level or delete them.

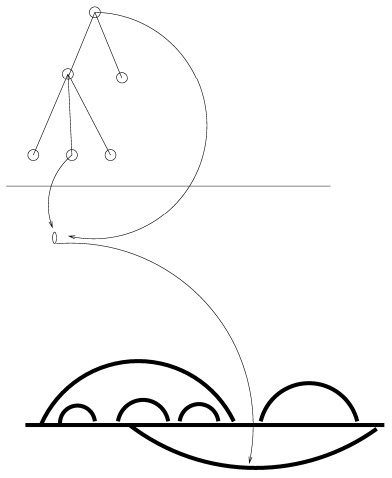


Figure 4: Graph model of sequence with crossing arcs

From the implementation point of view, there is a need to keep the forward and next sibling attributes of the node class consistent with that originally designed for tree nodes, we adopts a novel data structure which uses a pair of twin graph nodes (they are shadows to each other) to represent the two feet of a tertiary arc that falls in two separate tree nodes. The idea is illustrated in figure 5.

To differentiate all the tertiary nodes from the original secondary arc tree nodes, an arc type attribute is added to the node class. Once the tree of the secondary structures is generated, all the node have been ordered in the preoder traversal and typed as tree nodes; then when all the new graph nodes are grafted to the tree, they will be typed as graph nodes leaving the order system of the tree nodes intact. In the traversal stage at later stages, all the nodes will be traversed in BFT order but processed differently based on their types.

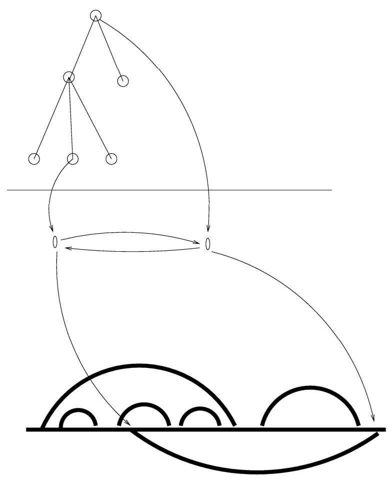


Figure 5: A modified graph model in our implementation

3.3 Preparing Stem Sizes and Positions

At this stage, we need to generate the length, starting position and ending position of each stem as well as that of each unpaired middle segment. The goal can be achieved by populating the corresponding preserved attributes of each instance of the unified tree/graph node class c.f. Figure 6. As the figure shows, each arc is defined by its left, middle, and right attributes representing the length of the left unpaired segment, the inner segment and the right unpaired one respectively. This stage traverses the tree/graph nodes in the breadth first traversal order and generates all the attributes for all the nodes level by level. At the end of this breath first traversal process, stem size, staring and ending positions of each arc must have been populated.

3.4 Filling Out Base Pairs and Single Base Segments

Once all the placeholders and their attributes of the tree have been well prepared, we need to use the base-pair probability distribution information to generate different symbols at all the base positions to complete the data generation process. In our current framework, we simply adopt the zero order Markov chain to randomly select base symbols based on the symbol distribution input parameters that can either follow the distributions extracted from the real world data or be supplied by users.

One particular concern at this phase is that a user can

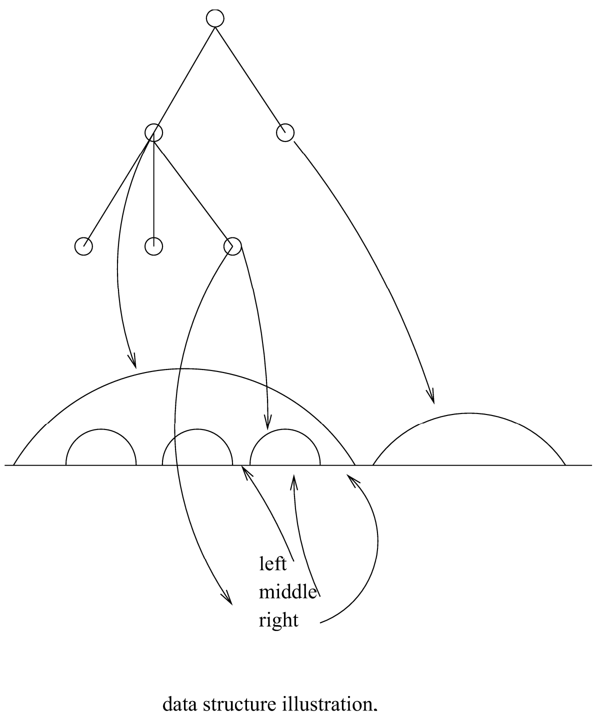
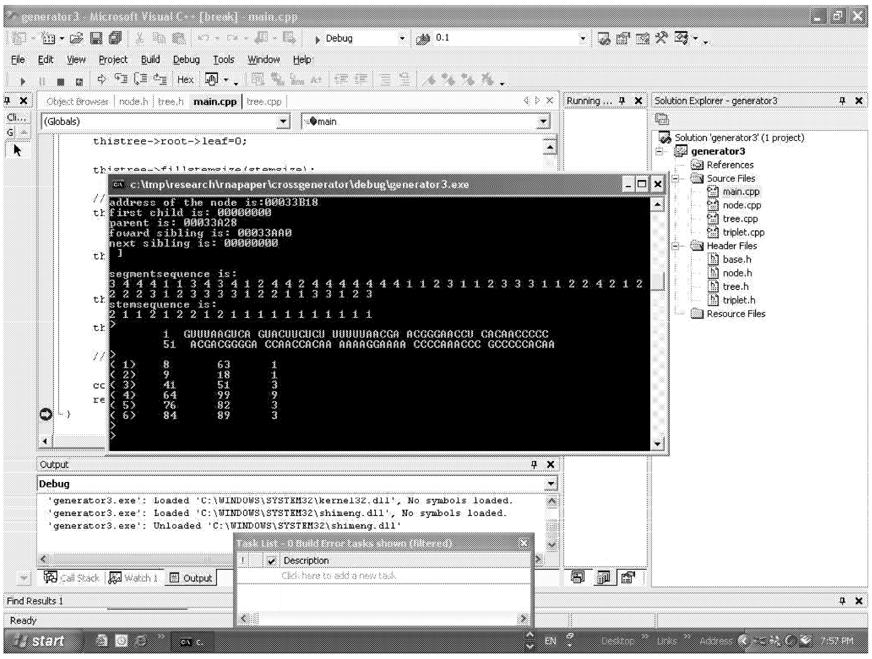


Figure 6: Data structure of generator tree.

only provide the input frequencies that govern the distributions of symbols at the whole sequence scope, due to unpredictable constraints posed by the base pairing rules of randomly generated tree shapes. We address this issue by generating the base pairs in all the arc segments first. Specifically, we apply the base generation process to the left foot of the arc data, then derive the counterpart half based on the pairing rules. After all the arc segments have been populated, the distribution of symbols, if inconsistent with the given frequencies, will be adjusted in the remaining unpaired segments to meet the global frequency distribution.

4 Implementation and Conclusions

We have discussed an algorithmic framework for generating synthetic complex arc-annotated RNA data mainly for evaluating the efficiency, correctness and completeness of RNA analysis and data mining algorithms. A preliminary implementation of the proposed generator has been coded in C++, runnable on all major platforms. Figure 7 shows a debugging session in the VC IDE on Windows system and its output windows displays a generated sequence which is annotated by several arcs. Each arc is defined by the starting position of the opening foot, the ending position of the closing foot as well as size of the arc. The implementation consists classes for generating trees, rooted graphs and sequences etc. Each class and its members and parameters can be independently im-



```
thistree->root->leaf=0;

thistree->fillstemsize(stemsize);
```

```
c:\tmp\research\rnapaper\crossgenerator\debug\generator3.exe
address of the node is:00033B18
first child is: 00000000
parent is: 00033A28
foward sibling is: 00033AA0
next sibling is: 00000000
 ]

segmentsequence is:
3 4 4 4 1 1 3 4 3 4 1 2 4 4 2 4 4 4 4 4 4 4 1 1 2 3 1 1 2 3 3 3 1 1 2 2 4 2 1 2
2 2 2 3 1 2 3 3 3 3 1 2 2 1 1 3 3 1 2 3
stemsequence is:
2 1 1 2 1 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1
>
       1  GUUUAAGUCA GUACUUCUCU UUUUUAACGA ACGGGAACCU CACAACCCCC
       51  ACGACGGGGA CCAACCACAA AAAAGGAAAA CCCCAAACCC GCCCCCACAA
>
( 1)    8       63      1
( 2)    9       18      1
( 3)    41      51      3
( 4)    64      99      9
( 5)    76      82      3
( 6)    84      89      3
>
>
```

```
'generator3.exe': Loaded 'C:\WINDOWS\SYSTEM32\kernel32.dll', No symbols loaded.
'generator3.exe': Loaded 'C:\WINDOWS\SYSTEM32\shimeng.dll', No symbols loaded.
'generator3.exe': Unloaded 'C:\WINDOWS\SYSTEM32\shimeng.dll'
```

Figure 7: A screenshot of our preliminary implementation of the framework

proved and enhanced. For example, the tree generation class may be modified to support other algorithms such as [3] etc.

In the future, we plan to extend, customize and refine the framework to produce more interesting data from the following perspectives. I. Produce possible data conforming to a given RNA family by accepting certain predefined RNA pattern seeds, possibly given in regular expressions. II. Replace the rooted graph structures with the loop representation. III. Use data mining techniques to extract from publicly available RNA databases the various statistic values which then can be used as the parameters to dictate our generator to produce data more similar to the real world data. For example, we can use the silhouette tree structure that has been mined from the real data to bypass the tree generate stage. Iv. Refine, improve and extend the framework and its implementation for generating more interesting higher level structures or arc-annotated RNA data.

5 Acknowledgements

The authors thank the referees and his colleagues for their helpful comments to improve the presentation of the paper.

References

[1] X. Z. Fu, H. Wang, W. Harrison, and R. Harrison. Rna pseudoknot prediction using term rewriting. In *Fifth IEEE Symposium on Bioinformatics and Bioengineering*, pages 169–176, 2005.

[2] Ivo L Hofacker, Walter Fontana, Peter F Stadler, L Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of RNA secondary structures. *Monatsh. Chem.*, 125:167–188, 1994.

[3] Hitoshi Iba. Random tree generation for genetic programming. In *The 4th International Conference on Parallel Problem Solving from Nature*, pages 144–153. Springer-Verlag, 1996.

[4] Barciszewski J, Frederic B, and Clark C. *RNA biochemistry and biotechnology*. Springer, 1999.

[5] T. Jiang, G. Lin, B. Ma, and K. Zhang. The longest common subsequence problem for arc-annotated sequences. *Journal of Discrete Algorithms*, 2(2):257–270, 2004.

[6] Bin Ma, Lusheng Wang, and Kaizhong Zhang. Computing similarity between rna structures. *Theoretical Computer Science*, 276(1-2):111–132, 2002.

[7] Bruce A. Shapiro and Kaizhong Zhang. Comparing multiple rna secondary structures using tree comparisons. *Computer Applications in the Biosciences*, 6(4):309–318, 1990.