International Journal of Biological, Life and Agricultural Sciences
ISSN: 2415-6612
Vol:11, No:6, 2017

CompPSA: A Component-Based Pairwise RNA Secondary Structure Alignment Algorithm

Ghada Badr, Arwa Alturki

***Abstract*—**The biological function of an RNA molecule depends on its structure. The objective of the alignment is finding the homology between two or more RNA secondary structures. Knowing the common functionalities between two RNA structures allows a better understanding and a discovery of other relationships between them. Besides, identifying non-coding RNAs -that is not translated into a protein- is a popular application in which RNA structural alignment is the first step A few methods for RNA structure-to-structure alignment have been developed. Most of these methods are partial structure-to-structure, sequence-to-structure, or structure-to-sequence alignment. Less attention is given in the literature to the use of efficient RNA structure representation and the structure-to-structure alignment methods are lacking. In this paper, we introduce an $O(N^2)$ Component-based Pairwise RNA Structure Alignment (CompPSA) algorithm, where structures are given as a component-based representation and where N is the maximum number of components in the two structures. The proposed algorithm compares the two RNA secondary structures based on their weighted component features rather than on their base-pair details. Extensive experiments are conducted illustrating the efficiency of the CompPSA algorithm when compared to other approaches and on different real and simulated datasets. The CompPSA algorithm shows an accurate similarity measure between components. The algorithm gives the flexibility for the user to align the two RNA structures based on their weighted features (position, full length, and/or stem length). Moreover, the algorithm proves scalability and efficiency in time and memory performance.

***Keywords*—**Alignment, RNA secondary structure, pairwise, component-based, data mining.

I. Introduction

IDENTIFYING the similarities and functionalities of two or more RNAs is an essential part of understanding biological processes. This is done by aligning the RNAs according to their structures. There are two ways to align RNA structures based on their representations, which can be: primary or secondary. For the primary structure, the sequence of the RNA is aligned, while for secondary structure, the RNA structure is aligned. Whether the alignment is sequential or structural, it can be classified as either global or local, and it can be either gapped or ungapped [1]. In this paper, we focus on structural alignment that is pairwise, global, and gapped. RNA sequence alignment is the process of matching similar regions of two or more RNA sequences. It depends on the assumption that similar sequences should have similar structures. The problem can be solved using different optimization techniques, such as dynamic programming [1], exact string matching and heuristic sequence alignment [2]. The Needleman and Wunsch algorithm [3] is one of the best-known dynamic programmingbased alignment algorithms for finding the global optimal alignment between two sequences. Smith and Waterman [4] updated Needlman-Wunsch so it can find the local optimal alignment. In general, the dynamic programming technique serves as a fundamental approach for other alignment techniques. RNA structure alignment algorithms whose approaches are developed using dynamic programming as a basic technique fall into three main categories: *sequence-to-structure*, *structure-to-sequence*, and *structure-to-structure*. FOLDALIGN [5]–[9] and LARA [10] are well-known examples of the sequence-to-structure approach. Many algorithms fall under the structure-to-sequence category: ExpaRNA [11], R-PASS [12], SCARNA [13], and RNASAlign [14]. The number of algorithms that do structure-to-structure alignment is limited. One of these algorithms is RNA-forester [15], which represents the structure of each sequence as rooted, ordered, and node-labeled trees; the collection of trees forms a forest. A tree contains both paired and unpaired bases; these bases are structured as parents and siblings based on how the base pairs, the order of $5'$ and $3'$ RNA molecular components, are nested. A pairwise local structure alignment is conducted using dynamic programming to nd the maximum similarity between two substructures. Some of these algorithms use special representations, such as the stem-representation algorithm introduced in [16]. The time and memory complexity of this work is relatively high, reaching S^4 , where S is the number of stems in the RNA structure pattern. Another tree-based algorithm developed for this purpose is RSmatch [17], which aligns the structures based on the details of their base pairs and does not utilize any other representation. ERA [18], a third complete structure-to-structure tree-based algorithm, which is also based on dynamic programming, has a complexity estimation of $O(N^3)$; it also aligns on the base-pair level.

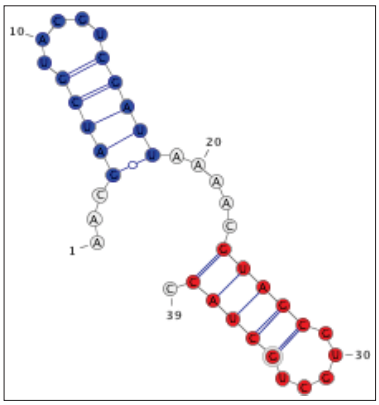
The paper outline is as follows: Section I contains an explanation of the fundamental background for components-based representation for RNA structures, a definition of the problem, and the proposed solution. The proposed similarity measures are introduced in Section II. The proposed CompPSA algorithm is introduced in details in Section III. Section IV presents the experiments that have been conducted in order to validate and check the performance of CompPSA. Finally, Section V concludes the paper.

Ghada Badr is with University of Ottawa, School of Electrical Engineering and Computer Science, Ottawa, Canada. She is also with RI, The City of Scientific Research and Technological Applications, University and Research District, P.O. 21934 New Borg Alarab, Alex, Egypt (e-mail: badrghada@hotmail.com).

Arwa Alturki is with King Saud University, College of Computer and Information Sciences, Saudi Arabia (e-mail: aralturki@ksu.edu.sa).

A. Component-Based Representation of RNA Secondary Structure

A component-based representation of RNA secondary structure was developed in 2011 [19] and was proven to perform well in secondary structures-localization algorithms [20]. The representation can be used to define interacting and non-interacting patterns for RNA secondary structures. A pattern $P = p_1, p_2, ..., p_m$ is uniquely defined by its sub-patterns $P_i, 0 < i > m$. Each sub-pattern is defined by its inter-molecular component (INTERM), its intra-molecular component (INTRAM), and its length. There are no INTERM components in non-interacting patterns. Each component is defined by its opening brackets (OB), closing brackets (CB), length, and relative locations within the sub-patterns. In the INTRAM component, OB and CB are located in the same sub-pattern. This means that there needs to be [$\pm$ at least] one sub-pattern for INTRAM to exist. OB and CB are located in p_i, where $1 \leq i \geq m$. OB and CB both are dened by their location and length; thus, INTRAM = {OB, CB, len}. In the INTERM component, OB and CB have to be in different sub-patterns. OB is located in p_i and CB is located p_j, where $j > i$ and $1 \leq j \geq m$. OB and CB are both defined by their distance from p_i the beginning of p_i and length, thus; INTERM = {OB, CB, j, len}. The representation extracts the main features for every RNA structure, showing the main components of each structure without going in details about the nucleotides that are found in each of them. Fig. 1 shows an example of an RNA structure with two non-interacting INRAM components in graphical [21] and dot-bracket representations [22].



(a) Graphic representation

...(((((.....)))))).....(((((.....))))).

(b) Dot-bracket representation

P = (39, INTERM = {}, INTRAM = {(4, 14, 5), (24, 34, 5)})

(c) Component-based representation

Fig. 1 An example of a component-based representation for a given graph and dot-bracket representations of the same RNA structure. The structure consists of two components: the first appears in blue, the second in red on the graphic representation

B. Problem Definition

In this paper, we propose a component-based pairwise structure-alignment (CompPSA) algorithm. Given two different RNA structures, a and b, the proposed algorithm has the following input:

- Two RNA secondary structures.
- Measure weights, gap penalty, and threshold (defined below).

The output is the pairwise gapped alignment for the two RNA structures. Alignment is given between components and is based on the similarity measure between single components.

C. Contributions

The efficiency of a structural alignment algorithm depends on its representation of the RNA structure. CompPSA is a novel pairwise structure alignment algorithm (for non-interacting structures). It compares the similarity between two RNA structurs based on their extracted features instead using their base pairs. In addition, we propose suitable similarity measures for the two RNA secondary structures to reflect the similarity between their components, and we introduce two ordering techniques during the alignment process.

II. The Proposed Similarity Measures

The proposed algorithm focuses on the alignment of pure RNA structures, where the RNA sequences may be not available. We propose that the similarity between any two given structures will depend on three similarity measures between any two components Ca_i and Cb_j:

- Component position.
- Component full length (henceforth "length").
- Component stem length.

Our measures are inspired by the R-PASS algorithm [12] equations, but we use the extracted components features. Our measures are different as follows:

- They do not distinguish between different types of components (motifs).
- There are no sequences; the sequence length is replaced by the pattern length (len_a).
- The start position is mapped with the opening bracket offset (OB).
- End position is equivalent to the closing bracket offset plus the length of the component minus one ($CB + len_{ai} - 1$).
- The end position of the first half of the stem can be represented by the opening bracket offset plus the length of the component minus one ($OB + len_{ai} - 1$).
- The start position of the second half is mapped to the closing bracket offset (CB).

A. Component Position Similarity

The similarity of the positions of the two components Ca_i and Cb_j is the maximum value of the opening bracket (OB_{ai}) position and the closing bracket position (CB_{ai}) position. Opening bracket position is calculated by considering the minimum value between the position of the first opening bracket (OB_{ai}) and the position of the last opening bracket ($OB_{ai} + len_{ai} - 1$), where len_{ai} is the stem length. In the same way, the closing bracket position is the minimum value between the first closing bracket (CB_{ai}) and the last closing bracket position ($CB_{ai} + len_{ai} - 1$). Since our algorithm seeks to maximize the similarity measure, we subtract the position distance from one after weighting it by W_p, with a value ranging from zero to one. The position weight here indicates the importance of position similarity between the two components during the alignment. For instance, setting the W_p value to zero means that the position similarity is not considered during the alignment process. By applying (1), we have the position distance between two components Ca_i and Cb_j:

$$Position(f_{ai}, f_{bj}) = 1 - W_p \times max\left(OpeningPosition, ClosingPosition\right), \textbf{Where}$$
$$OpeningPosition = min\left(|\frac{OB_{ai}}{len_{pa}} - \frac{OB_{bj}}{len_{pb}}|, |\frac{OB_{ai}+len_{ai}-1}{len_{pa}} - \frac{OB_{bj}+len_{bj}-1}{len_{pb}}|\right)$$
$$ClosingPosition = min\left(|\frac{CB_{ai}}{len_{pa}} - \frac{CB_{bj}}{len_{pb}}|, |\frac{CB_{ai}+len_{ai}-1}{len_{pa}} - \frac{CB_{bj}+len_{bj}-1}{len_{pb}}|\right) \quad (1)$$

B. Component Length Similarity

The component length similarity is the difference between the first opening bracket (OB_{ai}) and the first closing bracket (CB_{ai}) plus the stem length (len_{ai}). By multiplying this length by weight -W_{cl}- we show how important the similarity of component lengths is to the alignment process. Equation (2) gives us the component length similarity of two components Ca_i and Cb_j:

$$ComponentLength(f_{ai}, f_{bj}) = 1 - W_{cl} \times |\frac{(CB_{ai} - OB_{ai} + len_{ai}) - (CB_{bj} - OB_{bj} + len_{bj})}{(CB_{ai} - OB_{ai} + len_{ai}) + (CB_{bj} - OB_{bj} + len_{bj})}| \quad (2)$$

C. Component Stem Length Similarity

The stem length similarity can be found directly by using len_{ai}. Again, we are weighting the similarity between the stems by W_{sl} to indicate its contribution of it in the final similarity measure between the two components Ca_i and Cb_j:

$$StemLength(f_{ai}, f_{bj}) = 1 - W_{sl} \times |\frac{len_{ai} - len_{bj}}{len_{ai} + len_{bj}}| \quad (3)$$

D. Component Similarity

Based on the three metrics just introduced, the similarity measure between two components Ca_i and Cb_j can be calculated by (4):

$$d(f_{ai}, f_{bj}) = Position(f_{ai}, f_{bj}) \cdot ComponentLength(f_{ai}, f_{bj}) \cdot StemLength(f_{ai}, f_{bj}) \quad (4)$$

Because the proposed CompPSA algorithm is concerned with global gapped alignment and seeks to maximize the similarity measure between the components, it is important to introduce the cost of aligning a component with a gap. This *gap penalty*, to be determined through experiments, is denoted as λ and needs to be a negative value.

III. The Proposed CompPSA Algorithm

A. Feature Extraction

We extract the feature vector of each intra-molecular component from the component-based representation of that component. A component Ca_i in P_a is fully defined by its feature vector = $(OB_{ai}, CB_{ai}, len_{ai})$; the same holds true for Cb_j in P_b.

B. Dynamic Programming Application

The components of the two structures are arranged in a two-dimensional matrix F of size $(n+1)(m+1)$, where n is the number of intra-molecular components of the first RNA structure and m is the number of intra-molecular components of the second one. In addition to the dynamic programming matrix, the algorithm constructs another matrix, the *direction matrix*, to preserve the directions of computing the solution. By "directions" we mean the location of the neighbor who participates in the calculation of the similarity between each two components in the matrix. Two different ordering techniques are considered for the components in each RNA structure, one is based on the opening bracket offset, the other is based on the closing bracket offset.

The process of filling the dynamic programming matrix is based on the gapped-alignment algorithm [1] developed by Needleman and Wunsch. Entries in the matrix *F* are calculated based on the dynamic programming equation that was proposed by Needleman and Wunsch [3] with a modification. The modification is that if the similarity measure between two components Ca_i and Cb_j is less than a pre-defined threshold (the "pairwise component similarity threshold"), then the corresponding distance is set to the gap penalty. This is equivalent to inserting a gap in both structures. Thus, the dynamic programming formula is shown in (5).

Where $d(f_{ai}, f_{bj})$ is the distance (similarity) between two feature vectors f_{ai} and fbj calculated using (4), λ is the gap penalty, and ϵ is the threshold. Both λ and ϵ are parameters that need to be empirically adjusted. The final alignment score between the two structures a and b, $Score(a, b)$, is the summation of the similarities between the aligned components. If a component is aligned with a gap, then the similarity is considered as zero. The alignment score is calculated as in (6).

In order to measure the similarity score percentage of two RNA secondary structures, a and b, two metrics are used, $Score(a, b)$ and $Max(a, b)$ [16]. $Score(a, b)$ indicates the alignment of a and b. $Max(a, b)$ is the maximum score between $Score(a, a)$ and $Score(b, b)$, which is used for normalization. The similarity of the structures of the two RNAs is calculated as in (7).

$$F(i,j) = \begin{cases} F(i-1,j-1) + d(f_{ai}, f_{bj}), & if \;\; d(f_{ai}, f_{bj}) > \epsilon \\ F(i-1,j-1) + \lambda, & if \;\; d(f_{ai}, f_{bj}) < \epsilon \\ F(i-1,j) + \lambda \\ F(i,j-1) + \lambda \end{cases} \tag{5}$$

$$Score(a,b) = \begin{cases} \sum_{i=1}^{n} \sum_{j=1}^{m} d(f_{ai}, fbj), & if \;\; Ca_i \;\; is \;\; aligned \;\; with \;\; Cb_j \\ 0, & otherwise \end{cases} \tag{6}$$

$$ScorePercentage(a,b) = \frac{Score(a,b)}{Max(a,b)}, \textbf{\textit{Where}} \tag{7}$$
$$Max(a,b) = max\{Score(a,a), Score(b,b)\}$$

As the dynamic programming matrix is calculated, the direction matrix is also populated. The following notations are used to express the direction to the participating neighbor from which the cell value is calculated: **D** or **S** from a diagonal neighbor, **H** from the horizontal neighbor, and **V** from the vertical neighbor.

C. Construction of the Solution

The alignment with the highest score can be found by tracing the scores back from the bottom-right cell in the dynamic programming matrix to the top-left cell. Table I shows how this process works. Each step of the trace back can be one of the following:

- If the cell contains the letter **D**, align Ca_i with Cb_j and proceed to cell $(i-1, j-1)$.
- If the cell contains the letter **S**, skip both Ca_i and Cb_j and insert a gap in both structures. This is the case where the distance $d(f_{ai}, f_{bj})$ is equal to λ.
- If the cell contains the letter **H**, align Cb_j with a gap and step back to cell $(i, j-1)$.
- If the cell contains the letter **V**, align Ca_i with a gap and return to $(i-1, j)$.

TABLE I
FOUR CASES FOR DIRECTIONS. THE SHADED CELL REPRESENTS THE CELL THAT WE MOVE TO IN EACH DIRECTION

(TABLE I.A)
DIAGONAL NEIGHBOR

	-	Cb_1	Cb_2	Cb_3	Cb_4
-					
Ca_1					
Ca_2					
Ca_3					**D or S**

(TABLE I.B)
HORIZONTAL NEIGHBOR

	-	Cb_1	Cb_2	Cb_3	Cb_4
-					
Ca_1					
Ca_2					
Ca_3					**H**

(TABLE I.C)
VERTICAL NEIGHBOR

	-	Cb_1	Cb_2	Cb_3	Cb_4
-					
Ca_1					
Ca_2					
Ca_3					**V**

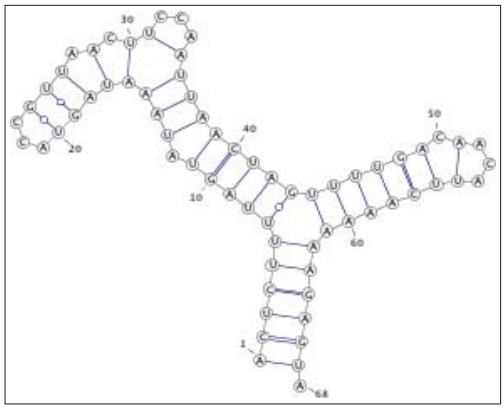


Fig. 2 First RNA structure (denoted as a)

IV. EXPERIMENTAL ANALYSIS

Extensive experiments are conducted to test and validate the CompPSA algorithm. For experimental purposes, a special datasets is collected from different sources. The datasets consist of: (1) a comparison dataset, (2) a performance accuracy dataset and (3) a scalability dataset.

A. Dataset

The experiments are conducted using three different datasets. Each dataset serves a specific purpose. In this section, we explain the three datasets in more detail. (Another real dataset, taken from Rfam dataset [23] along with its results, is omitted from the paper due to space limitations. However, the authors can be contacted for full results). Two of these datasets, the NCBI GenBank dataset [24] and the Genomic tRNA dataset [25], are real data. The third is a simulated dataset based on a structure that is selected from NCBI. The third dataset is used to test the performance of CompPSA with different RNA structure lengths.

1) NCBI GenBank Dataset: The NCBI GenBank dataset is used to validate the CompPSA similarity measure and compare it with the results of [16]. According to RNA secondary

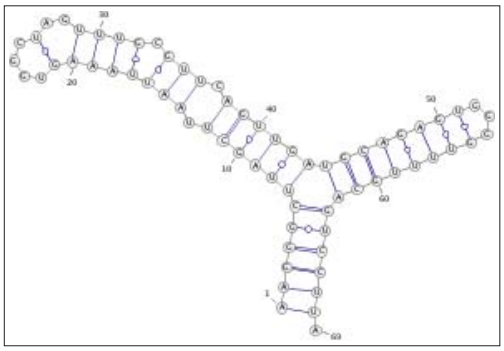


Fig. 3 Second RNA structure (denoted as b)

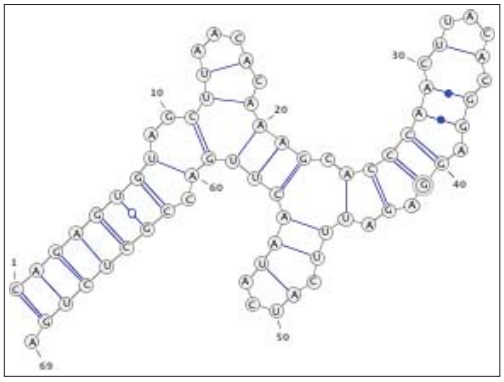


Fig. 4 Third RNA structure (denoted as c)

structure alignment based on stem representation [16], there are two cases in which RNA is similar:

- *Case 1:* the RNA structures are common, but the sequences are different.
- *Case 2:* the RNA sequences are similar, but the structures are dissimilar

The work done in [16] proves that case 1 has a higher similarity. Our objective is to validate the score of the measure we used in our algorithm for the same cases. For this purpose, in this dataset, we select three RNA structures from the NCBI GenBank [24]. The first RNA structure, *a*, has a length of 68 nucleotides and 9 components, as shown in Fig. 2, with vectors as shown in Table II. It is used for comparison in both cases.

TABLE II
STRUCTURE a FEATURE VECTORS

Component	*OB*	*CB*	*Len*
Ca_1	1	62	6
Ca_2	7	39	5
Ca_3	13	36	3
Ca_4	16	30	1
Ca_5	17	28	1
Ca_6	18	24	3
Ca_7	31	35	1
Ca_8	44	56	6
Ca_9	51	55	1

TABLE III
STRUCTURE b FEATURE VECTORS

Component	*OB*	*CB*	*Len*
Cb_1	1	63	6
Cb_2	7	38	6
Cb_3	14	34	3
Cb_4	17	29	4
Cb_5	21	26	1
Cb_6	44	55	8

TABLE IV
STRUCTURE c FEATURE VECTORS

Component	*OB*	*CB*	*Len*
Cc_1	1	62	7
Cc_2	8	60	1
Cc_3	11	59	1
Cc_4	12	19	1
Cc_5	13	17	1
Cc_6	20	56	3
Cc_7	24	45	1
Cc_8	25	43	1
Cc_9	26	40	2
Cc_10	28	37	2
Cc_11	31	35	1
Cc_12	46	54	2
Cc_13	49	53	1

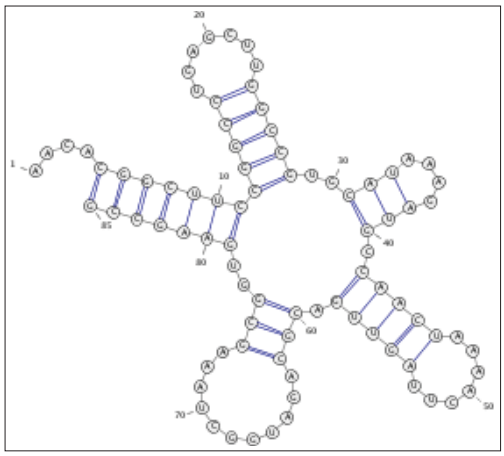


Fig. 5 E. coli tRNA for leucine

Case 1, Similar Structure and Dissimilar Sequences: The first RNA structure is compared with a is b (see Fig. 3 and Table III), which has a length of 69 nucleotides and 6 components.

Case 2, Similar Sequences and Dissimilar Structures: The second RNA structure is compared with a is c (see Fig. 4 and Table IV), which has a length of 69 nucleotides and 13 components.

2) Genomic tRNA Dataset: We also consider two real and simple RNA structures from the tRNA Genomic Database [26]. The first RNA structure is for *leucine E.coli tRNA* (see Fig. 5 and Table V), the second is for *alanine E.coli tRNA* (see Fig. 6 and Table VI). We denote them as a and b. As can be seen, structure a has a length of 85 and consists of 5 components; b has a length of 76 and consists of 4 components.

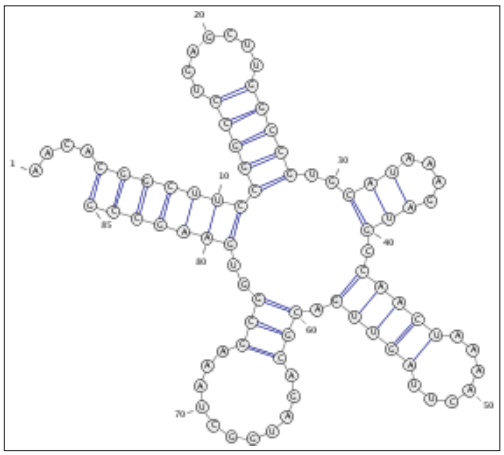


Fig. 6 E. coli tRNA for alanine

TABLE V
LEUCINE TRNA FEATURE VECTORS

Component	*OB*	*CB*	*Len*
Ca_1	5	79	7
Ca_2	12	24	5
Ca_3	31	38	3
Ca_4	42	54	5
Ca_5	60	74	3

3) Simulated Dataset: The purpose of the third simulated dataset is to test the scalability of the time and memory performance of the CompPSA algorithm against the base-pairs algorithm. This dataset contains 11 RNA structures based on RNA structure a (see Table II). The second RNA structure is a duplication of the first one, the third structure is a duplication of the second one, and so on. Table VII represents the lengths and component numbers for the 11 RNA structures in this dataset.

B. Setup of Experiments

Three different experiments are conducted:

1) Gap penalty and threshold adjustment are adjusted using the comparison dataset.
2) Four Weight variation experiments are conducted to compare the datasets using the adjusted parameters: (1) alignment based on component position, length, and stem length; (2) alignment based on component position only; (3) alignment based on component length only; and (4) alignment based on component stem length only. For each weight variation, the same experiment is conducted as in [16] and is used as a benchmark.
3) Time and memory performance experiments are conducted on all 11 RNA structures in the scalability dataset, as explained in Section IV.A. The results of the CompPSA algorithm are compared with the results of the base-pairs based alignment algorithm using the same experiments.

TABLE VI
ALANINE TRNA FEATURE VECTORS

Component	*OB*	*CB*	*Len*
Cb_1	5	70	7
Cb_2	12	24	5
Cb_3	34	46	5
Cb_4	52	64	4

All experiments are conducted using the two proposed ordering techniques for the components: (1) order based on opening brackets, and (2) order based on closing brackets. Due to similarity of results, only the results for the ordering by opening brackets are presented in this paper.

C. Results

This section summarizes the results for all experiments.

1) Gap Penalty λ and Pairwise Component Similarity Threshold ϵ Values: Special experiments are conducted [25] using the comparison dataset. We find that the gap value that fit best is 0.3, so the gap penalty λ is equal to-0.3. The results show that 0.5 is the best choice for the similarity threshold. By setting threshold value ϵ to 0.5 we are directing the proposed CompPSA algorithm to align only components with similarity more than or equal to 50%. In base pairs dynamic programming alignment, a match between any base pair is counted as 1, while the mismatch is counted by -1.

2) RNA-Forester Comparision Experiment: The RNA-forester comparison experiment is conducted on the second dataset, real genomic tRNA, which contains two simple RNA structures. This experiment is presented in details to clearly demonstrate the proposed algorithm. In order to combine the three similarity metrics (component position, component length, and component stem length), we need to adjust the three weights to one. Accordingly, the weighting adjustments are as follows: $W_p = 1$, $W_{cl} = 1$ and $W_{sl} = 1$. The result of this case is validated by comparing with the results from the RNA-forester tool [15].

3) Feature Extraction: The first step of the algorithm extracts the RNA structure features by representing them as a component-based. As we can see from Figs. 5 and 6, the first structure consists of 5 components while the second one has only 4 components. The resulting feature vectors are shown in Table V and VI, and the component-based representation is written as the follows:

- $P_a = (85,\ INTERM = \{\},\ INTRAM = \{Ca_1 = (5, 79, 7),\ Ca_2 = (12, 24, 5),\ Ca_3 = (31, 38, 3),\ Ca_4 = (42, 54, 5),\ Ca_5 = (60, 74, 3)\})$
- $P_b = (76,\ INTERM = \{\},\ INTRAM = \{Cb_1 = (5, 70, 7),\ Cb_2 = (12, 24, 5),\ Cb_3 = (34, 46, 5),\ Cb_4 = (52, 64, 4)\})$

4) Dynamic Programming Application: As mentioned previously, the first RNA structure in this example has 5 intra-molecular components; so $n = 5$; the second one has 4, so $m = 4$. On that basis we create two 6×5 matrices, one for dynamic programming scores and one for directions through the process. Since we are considering gapped alignment, there is one additional row and one additional column for the gap. The components can be ordered based either on their opening

TABLE VII
THE 11 RNA STRUCTURES WITH THEIR SIZES AND NUMBER OF COMPONENTS

RNA Structure #	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Size (length)	68	136	272	544	1088	2176	4352	8704	10336	20672	41344
Components #	9	18	36	72	144	288	576	1152	1368	2736	5472

brackets or on their closing brackets. The similarities between the two components are then calculated based on the proposed similarity measurement equations, with numbers (1)-(4). The resulting similarities are shown in Table VIII.
The score matrix has been calculated and lled in based on the dynamic programming technique in (5). At the same time, the direction matrix was also lled in according to the neighbor that contributed to the cell value, as described previously. Tables IX and X show the resulting score and direction matrices.

5) Construction of the Solution: Based on the direction matrix computed in Table X, we will trace back in the dynamic programming score matrix and the direction matrix from respective the bottom-right cells to the respective upper-left cells. For example, in the case of *b*, where the components are arranged according to their closing bracket offset, the bottom-right cell in the score matrix has the value 3.38; **D** is associated with the same cell in the direction matrix. This means Ca_1 and Cb_1 are aligned with each other, and we should move to cell $(i1, j1)$. There we nd 2.45 in the score matrix and **D** associated with it; that is, the component Ca_5 is aligned to Cb_4, to we move to (i_1, j_1). There we find the alignment score 1.63 and the direction indicator **D**; we then align Ca_4 with Ca_3 and move to cell (i_1, j_1). There we find **V**, which denotes that Ca_3 is aligned with a gap that has a score of 0.67. The result of the alignment from the completed trace back is shown in Fig. 7.
If an alignment, of this example, that is based on only the position similarity is desired, this can be achieved by setting the position weight W_p to 1 and the other weights W_{cl} and W_{sl} to 0. This will align the two structures based on their components full length only and discard other features. Similarly, setting W_{sl} to 1 and the other two weights to 0 results in an alignment of the structures that is based on component stem length. In our example, all of three cases produce an alignment similar to the one produced when all similarity features are included (i.e. $W_p = W_{cl} = W_{sl} = 1$; see Fig. 7). This leads us to believe that (1) all three features have similar effect on the alignment process, and (2) the two RNA structure are highly similar to each other.

We compared results of the alignment by CompPSA with those from RNA-forester, and they match. As a final summary of the alignment process and results, Table XI shows the aligned components, the measure of the similarity between them considered in the dynamic programming process, and the structure score percentages.

TABLE VIII
SIMILARITY FOR ORDERING BY OPENING BRACKET

	Cb_1	Cb_2	Cb_3	Cb_4
Ca_1	0.93	0.11	0.18	0.1
Ca_2	0.12	0.97	0.68	0.39
Ca_3	0.08	0.46	0.47	0.4
Ca_4	0.19	0.67	0.96	0.69
Ca_5	0.1	0.36	0.57	0.82

TABLE IX
SCORES FOR ORDERING BY OPENING BRACKET

	-	Cb_1	Cb_2	Cb_3	Cb_4
-	0.0	-0.3	-0.6	-0.9	-1.2
Ca_1	-0.3	0.93	0.63	0.33	0.03
Ca_2	-0.6	0.63	1.9	1.6	1.3
Ca_3	-0.9	0.33	1.6	1.6	1.3
Ca_4	-1.2	0.03	1.3	2.56	2.29
Ca_5	-1.5	-0.27	1.0	2.26	3.38

TABLE X
DIRECTION FOR ORDERING BY OPENING BRACKET

	-	Cb_1	Cb_2	Cb_3	Cb_4
-	*	H	H	H	H
Ca_1	V	D	H	H	H
Ca_2	V	V	D	H	H
Ca_3	V	V	V	S	H
Ca_4	V	V	V	D	D
Ca_5	V	V	V	V	D

6) Weighting Variation Experiments: The goal of the weighting variation experiments is to clarify and emphasize on the role of the three weights in our algorithm: component position weight W_p, component length weight W_{cl} and component stem length W_{sl}. According to [16], the alignment percentage of the structure pairs in case 1, where the RNA structures are the same but the sequences are different, should be higher than the alignment percentage in case 2, and that is what our results confirmed. Here we present the results of the alignment of the two structures based on the order of the opening brackets only.

Case 1, Similar Structure and Dissimilar Sequences: The alignment of the two structures is different when different weight variations are used. The results of this experiment are shown in Table XII. The higher the similarity between two components the more similar they are. When a component is aligned with a gap, or when two components are skipped, the similarity between them is set to the gap penalty λ, which is -0.3. In the same way, skipping two components instead of aligning them set the similarity to λ. According to the results in Table XII, the component position is the most important feature, more so than length and stem length, with a similarity score percentage of 66%. The second-most important feature is the component length. Because alignment based on component stem length gives results similar to those based on component position, alignment based on component stem length is the least important feature. Moreover, the following two situations reveal the results more clearly:

- When W_{sl} is set to 1, the alignment is the same when

TABLE XI
SUMMARY OF THE ALIGNMENT PROCESS FOR LEUCINE E. COLI TRNA AND ALANINE E. COLI TRNA

(TABLE XI.A)
ALIGNMENT BASED ON COMPONENT POSITIONS, COMPONENT LENGTH, AND COMPONENT STEM LENGTH

Weighting Setup	W_p **1**		W_{cl} **1**		W_{sl} **1**
Alignment	Ca_1	Ca_2	Ca_3	Ca_4	Ca_5
	Cb_1	Cb_2	-	Cb_3	Cb_4
Similarity	0.93	0.97	-0.3	0.96	0.82
$Score(a,b)$ = 3.68			$ScorePercentage = 74\%$		

(TABLE XI.B)
ALIGNMENT BASED ON SIMILARITY OF COMPONENT POSITION ONLY

Weighting Setup	W_p **1**		W_{cl} **0**		W_{sl} **0**
Alignment	Ca_1	Ca_2	Ca_3	Ca_4	Ca_5
	Cb_1	Cb_2	-	Cb_3	Cb_4
Similarity	0.99	0.97	-0.3	0.96	0.99
$Score(a,b)$ = 3.91			$ScorePercentage = 78\%$		

(TABLE XI.C)
ALIGNMENT BASED ON SIMILARITY OF COMPONENT LENGTH ONLY

Weighting Setup	W_p **1**		W_{cl} **0**		W_{sl} **0**
Alignment	Ca_1	Ca_2	Ca_3	Ca_4	Ca_5
	Cb_1	Cb_2	-	Cb_3	Cb_4
Similarity	0.94	1.0	-0.3	1.0	0.97
$Score(a,b)$ = 3.68			$ScorePercentage = 78\%$		

(TABLE XI.D)
ALIGNMENT BASED ON SIMILARITY OF COMPONENT LENGTH ONLY

Weighting Setup	W_p **1**		W_{cl} **0**		W_{sl} **0**
Alignment	Ca_1	Ca_2	Ca_3	Ca_4	Ca_5
	Cb_1	Cb_2	-	Cb_3	Cb_4
Similarity	1.0	1.0	-0.3	1.0	0.86
$Score(a,b)$ = 3.86			$ScorePercentage = 77\%$		

one or both of W_p and W_{cl} are set to 1.
- When W_p, W_{cl}, and W_{sl} are set to 1, 1, and 0, respectively, the alignment is different.

Case 2, Similar Sequences and Dissimilar Structures: The experiments conducted on case 1 RNA structures were repeated with case 2. The results are presented in Table XIII, which shows that component length and component position are the dominant similarities, 63% and 62%, respectively. As a result of this situation, the alignment that is based on component length is similar to the alignment that is based on component position:

- When W_{sl} is set to 0, the alignment of case 2 structures is the same when one or both of W_p and W_{cl} are set to 1.
- When the two dominant similarities, component position and length, are combined in one alignment and component stem length is discarded (i.e., W_{sl} is set to 0), the result is a new alignment.

According to the stem representation method in [16], the score percentage for case 1 alignment, where dissimilar sequences are considered, should be higher than that for case 2 alignment. Our results confirmed the results in [16]: When all three similarities are measured, the scores are 55% for case 1 (see Table XII) and 41% for case 2 (see Table XIII). The scores are almost the same when only component position or only component length is measured: the scores are 63% for case 1 and 55% for case 2. The different score for component stem length in case 2 is a result of similarities in component stem length that was not detected in the previous work.

7) Time and Memory Performance: In this experiment, we tested the empirical performance and scalability of the proposed CompPSA algorithm. Results from this set of experiments are shown in Table XIV. (Table XIV is plotted as a graph in Fig. 8.) As can be seen, time consumption in the proposed method is significantly less than in the base pairs method. This reduction in time comes because the number of components (N) is much lower than the structure length (P). This confirms our theoretical analysis. If the number of blocks is S, the time consumption of the stem representation method in [16] is $O(S^4)$, higher than $O(N^2)$ for our method. Also, the number of blocks S is larger than N. We thus conclude that the proposed CompPSA algorithm is more conservative in time than both base pairs and stem representation methods.

TABLE XII
CASE 1 ALIGNMENT PROCESS SUMMARY FOR THE FOUR WEIGHTING VARIATIONS

(TABLE XII.A)
ALIGNMENT BASED ON COMPONENT POSITION, COMPONENT LENGTH, AND COMPONENT STEM LENGTH

Weighting Setup		W_p **1**			W_{cl} **1**			W_{sl} **1**	
Alignment	Ca_1	Ca_2	Ca_3	Ca_4	Ca_5	Ca_6	Ca_7	Ca_8	Ca_9
	Cb_1	Cb_2	Cb_3	-	-	Cb_4	Cb_5	Cb_6	-
Similarity	0.99	0.9	0.9	-0.3	-0.3	0.58	0.77	0.83	-0.3
$Score(a,b) = 4.97$					$ScorePercentage = 55\%$				

(TABLE XII.B)
ALIGNMENT BASED ON COMPONENT POSITION SIMILARITY ONLY

Weighting Setup		W_p **1**			W_{cl} **0**			W_{sl} **0**	
Alignment	Ca_1	Ca_2	Ca_3	Ca_4	Ca_5	Ca_6	Ca_7	Ca_8	Ca_9
	Cb_1	Cb_2	Cb_3	-	Cb_4	Cb_5	-	Cb_6	-
Similarity	1	0.99	0.96	-0.3	0.99	0.99	-0.3	0.99	-0.3
$Score(a,b) = 5.92$					$ScorePercentage = 66\%$				

(TABLE XII.C)
ALIGNMENT BASED ON COMPONENT LENGTH SIMILARITY ONLY

Weighting Setup		W_p **0**			W_{cl} **1**			W_{sl} **0**	
Alignment	Ca_1	Ca_2	Ca_3	Ca_4	Ca_5	Ca_6	Ca_7	Ca_8	Ca_9
	Cb_1	Cb_2	Cb_3	Cb_4	-	-	Cb_5	Cb_6	-
Similarity	0.99	1	0.94	0.97	-0.3	-0.3	0.91	0.97	-0.3
$Score(a,b) = 5.78$					$ScorePercentage = 64\%$				

(TABLE XII.D)
ALIGNMENT BASED ON COMPONENT STEM LENGTH SIMILARITY ONLY

Weighting Setup		W_p **0**			W_{cl} **0**			W_{sl} **1**	
Alignment	Ca_1	Ca_2	Ca_3	Ca_4	Ca_5	Ca_6	Ca_7	Ca_8	Ca_9
	Cb_1	Cb_2	Cb_3	-	-	Cb_4	Cb_5	Cb_6	-
Similarity	1	0.91	1	-0.3	-0.3	0.86	1	0.86	-0.3
$Score(a,b) = 5.63$					$ScorePercentage = 64\%$				

We also compared the memory performance of our algorithm with base pairs and stem-representation methods. The proposed algorithm and base-pairs method utilize two matrices of size N^2 and P^2 respectively, one for the dynamic programming score and one for trace-back directions. This means the memory complexity for the proposed method is $O(2N^2) \sim O(N^2)$, and for the base pairs method it is $O(2P^2) \sim O(P^2)$. Recall that P is much larger than N; as a consequence, the memory requirement for base pairs method is much larger than that of the proposed method. Table XV presents the memory performance for both methods, and the information is plotted on the graph in Fig. 9. The stem representation method has a four-dimensional matrix; the maximum number of each dimension is S, so the memory complexity is $O(S^4)$, higher than the memory complexity of the CompPSA algorithm.

V. CONCLUSION

Comparing RNA sequences is a fundamental procedure in biology. Similar RNA sequences have similar structures and similar functionalities. Great efforts has been devoted to developing efficient RNA sequence alignment algorithms. The case is not the same for RNA structural alignment problems, where a limited number of algorithms have been developed. Most RNA structure alignment algorithms are not structure to structure; they also usually neglect the efficiency of representing the RNA structure. In this paper, we introduced a novel pairwise structure alignment algorithm that can efficiently calculate secondary structure similarities based on their components rather than just their nucleotide details. We also proposed three measures that enabled us to define the priority of similarities between two given RNA structures based on given weights. These measures are based on: component position, length, and stem length. This gives more flexibility to users of our algorithm in comparing RNA structures: they can compare them as components rather than just comparing them as wholes. Moreover, two ordering techniques were introduced that can be used in calculating the similarities between two RNA structures using dynamic programming: opening bracket offset ordering and closing bracket offset ordering.

In order to adjust the parameters and to test the validity, accuracy, and performance of the proposed algorithm, extensive experiments were conducted, and parameters were adjusted and to test accuracy and scalability in time and memory performance. The results were compared to those of other available approaches: a base pairs dynamic

TABLE XIII
CASE 2 ALIGNMENT PROCESS SUMMARY FOR THE FOUR WEIGHTING VARIATIONS

(TABLE XIII.A)
ALIGNMENT BASED ON COMPONENT POSITION, COMPONENT LENGTH, AND COMPONENT STEM LENGTH

Weighting Setup	W_p **1**					W_{cl} **1**					W_{sl} **1**		
Alignment	Ca_1	-	-	-	-	Ca_3	Ca_4	Ca_5	-	Ca_6	Ca_7	-	Ca_9
	Cc_1	-	Cc_3	Cc_4	Cc_5	Cc_6	Cc_7	Cc_8	Cc_9	Cc_{10}	Cc_{11}	-	Cc_{13}
Scalability	0.92	-0.3	-0.3	-0.3	-0.3	0.57	0.64	0.61	-0.3	0.6	0.99	-0.3	0.96
	$Score(a,b) = 5.29$								$ScorePercentage = 41\%$				

(TABLE XIII.B)
ALIGNMENT BASED ON COMPONENT POSITION SIMILARITY ONLY

Weighting Setup	W_p **1**					W_{cl} **0**					W_{sl} **0**		
Alignment	Ca_1	-	-	-	-	Ca_2	Ca_3	Ca_4	Ca_5	Ca_6	Ca_7	Ca_8	Ca_9
	Cc_1	Cc_2	Cc_3	Cc_4	Cc_5	Cc_6	Cc_7	Cc_8	Cc_9	Cc_{10}	Cc_{11}	Cc_{12}	Cc_{13}
Scalability	1	-0.3	-0.3	-0.3	-0.3	0.79	0.87	0.82	0.83	0.83	0.99	0.96	0.96
	$Score(a,b) = 8.5$								$ScorePercentage = 62\%$				

(TABLE XIII.C)
ALIGNMENT BASED ON COMPONENT LENGTH SIMILARITY ONLY

Weighting Setup	W_p **0**					W_{cl} **1**					W_{sl} **0**		
Alignment	Ca_1	-	-	-	-	Ca_2	Ca_3	Ca_4	Ca_5	Ca_6	Ca_7	Ca_8	Ca_9
	Cc_1	Cc_2	Cc_3	Cc_4	Cc_5	Cc_6	Cc_7	Cc_8	Cc_9	Cc_{10}	Cc_{11}	Cc_{12}	Cc_{13}
Scalability	0.99	-0.3	-0.3	-0.3	-0.3	0.97	0.92	0.88	0.86	0.9	1	0.71	1
	$Score(a,b) = 8.23$								$ScorePercentage = 62\%$				

(TABLE XIII.D)
ALIGNMENT BASED ON COMPONENT STEM LENGTH SIMILARITY ONLY

Weighting Setup	W_p **0**					W_{cl} **0**					W_{sl} **1**		
Alignment	Ca_1	-	-	-	-	Ca_2	Ca_3	Ca_4	Ca_5	Ca_6	Ca_7	Ca_8	Ca_9
	Cc_1	Cc_2	Cc_3	Cc_4	Cc_5	Cc_6	Cc_7	Cc_8	Cc_9	Cc_{10}	Cc_{11}	Cc_{12}	Cc_{13}
Scalability	0.92	-0.3	-0.3	-0.3	-0.3	0.75	0.5	1	0.67	0.8	1	0.5	1
	$Score(a,b) = 7.14$								$ScorePercentage = 55\%$				

TABLE XIV
TIME FOR BASE PAIRS ALIGNMENT AND COMPPSA ALIGNMENT FOR OPENING BRACKETS (OB)

Size	**Time (Seconds)**	
	Base Pair	***CompPSA***
68	0.13683509	0.018515662
136	0.182785867	0.032867564
272	0.286313291	0.080643647
544	0.6906022	0.155576236
1088	2.313445951	0.217882314
2176	6.623397349	0.358658094
4352	21.68938905	0.897526949
8704	81.4371613	2.983446382
10336	116.1859366	4.080809364
20672	472.4879804	11.44168366
41344	Out of memory after 1200	41.98851963

TABLE XV
MEMORY PERFORMANCE FOR BASE PAIRS ALIGNMENT AND COMPPSA ALIGNMENT FOR OPENING BRACKETS (OB)

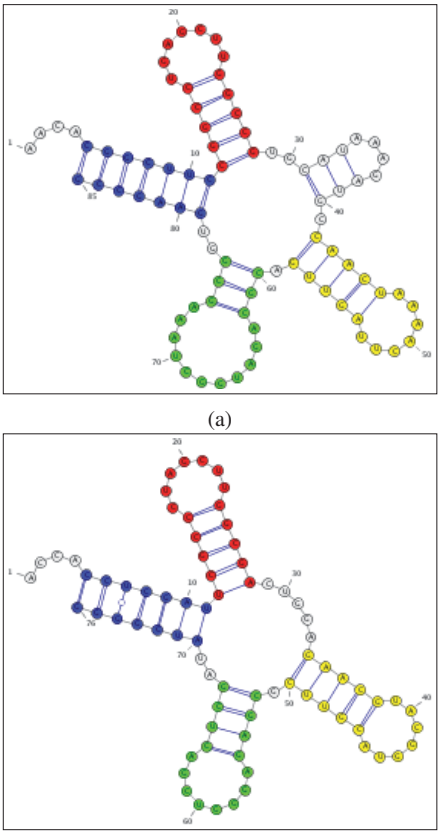
Size	**Memory Space (MB)**	
	Base Pair	***CompPSA***
68	0	0
136	0	0
272	61.44	1.28
544	245.76	5.75
1088	1013.76	24.28
2176	1022.43	1.73
4352	229.71	40.01
8704	2419.96	156.77
10336	1732.75	162.57
20672	1924.12	244.29
41344	Out of memory after 1200	1279.04

programming method and a stem-representation method. The results confirmed the results of the stem representation method. We found that our outperforms both algorithms in time and memory usage. Also, it provides high flexibility for users in comparing RNA structures with respect to component position, full length, and stem length.

In the future, we would like to extend the algorithm to interacting RNA structures and to generalize it so it can handle multiple RNA structure alignments. We anticipate that our technique is general enough to be applied to other structured data, such as chemical structures, where components can be extracted and compared. We would also like to investigate other component-ordering techniques.

REFERENCES

[1] D. Gusfield, *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge University Press, 1997.
[2] D. Sankoff, "Simultaneous solution of the rna folding, alignment and protosequence problems," *SIAM Journal on Applied Mathematics*, vol. 45, no. 5, pp. 810–825, 1985.



(a) (b)

Fig. 7 Alignment where the aligned components (regions) share the same color. In this example, where component position, component length, and component stem length are considered, the alignment results are the same whether order of the components is based on their opening bracket offset or on their closing bracket offset

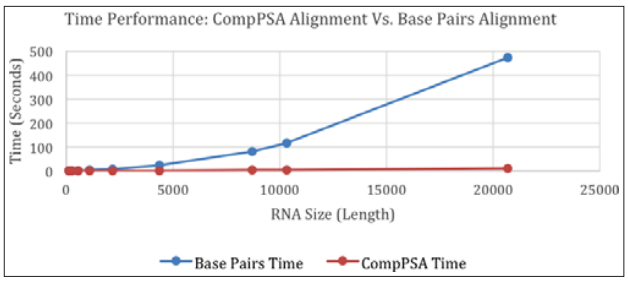


Fig. 8 Graphic representation of Table XIV

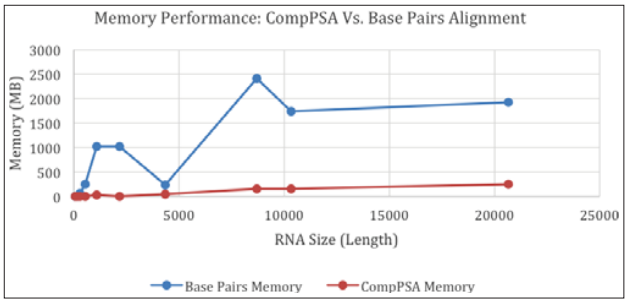


Fig. 9 Graphic representation of Table XV

[3] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.
[4] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.
[5] J. Gorodkin, L. J. Heyer, and G. D. Stormo, "Finding the most significant common sequence and structure motifs in a set of rna sequences," *Nucleic Acids Research*, vol. 25, no. 18, pp. 3724–3732, 1997.
[6] J. H. Havgaard, R. B. Lyngsø, G. D. Stormo, and J. Gorodkin, "Pairwise local structural alignment of rna sequences with sequence similarity less than 40%," *Bioinformatics*, vol. 21, no. 9, pp. 1815–1824, 2005.
[7] J. H. Havgaard, R. B. Lyngsø, and J. Gorodkin, "The foldalign web server for pairwise structural rna alignment and mutual motif search," *Nucleic acids research*, vol. 33, no. suppl 2, pp. W650–W653, 2005.
[8] J. H. Havgaard, E. Torarinsson, and J. Gorodkin, "Fast pairwise structural rna alignments by pruning of the dynamical programming matrix," *PLOS computational biology*, vol. 3, no. 10, p. e193, 2007.
[9] E. Torarinsson, J. H. Havgaard, and J. Gorodkin, "Multiple structural alignment and clustering of rna sequences," *Bioinformatics*, vol. 23, no. 8, pp. 926–932, 2007.
[10] M. Bauer, G. W. Klau, and K. Reinert, "Accurate multiple sequence-structure alignment of rna sequences using combinatorial optimization," *BMC bioinformatics*, vol. 8, no. 1, p. 271, 2007.
[11] S. Heyne, S. Will, M. Beckstette, and R. Backofen, "Lightweight comparison of rnas based on exact sequence-structure matches," *Bioinformatics*, p. btp065, 2009.
[12] Y. Jiang, W. Xu, L. P. Thompson, R. R. Gutell, and D. P. Miranker, "R-pass: A fast structure-based rna sequence alignment algorithm," in *Bioinformatics and Biomedicine (BIBM), 2011 IEEE International Conference on*. IEEE, 2011, pp. 618–622.
[13] Y. Tabei, K. Tsuda, T. Kin, and K. Asai, "Scarna: fast and accurate structural alignment of rna sequences by matching fixed-length stem fragments," *Bioinformatics*, vol. 22, no. 14, pp. 1723–1729, 2006.
[14] T. K. Wong, K.-L. Wan, B.-Y. Hsu, B. W. Cheung, W.-K. Hon, T.-W. Lam, and S.-M. Yiu, "Rnasalign: Rna structural alignment system," *Bioinformatics*, vol. 27, no. 15, pp. 2151–2152, 2011.
[15] M. Hochsmann, T. Toller, R. Giegerich, and S. Kurtz, "Local similarity in rna secondary structures," in *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*. IEEE, 2003, pp. 159–168.
[16] M.-Y. Wu, C.-B. Yang, and K.-S. Huang, "Rna secondary structure alignment based on stem representation," in *Proceedings of the 21st Workshop on Combinational Mathematics and Computation Theory*. Citeseer, pp. 60–69.
[17] J. Liu, J. T. Wang, J. Hu, and B. Tian, "A method for aligning rna secondary structures and its application to rna motif detection," *BMC bioinformatics*, vol. 6, no. 1, p. 89, 2005.
[18] C. Zhong and S. Zhang, "Efficient alignment of rna secondary structures using sparse dynamic programming," *BMC bioinformatics*, vol. 14, no. 1, p. 269, 2013.
[19] G. Badr and M. Turcotte, "Component-based matching for multiple interacting rna sequences," in *Bioinformatics Research and Applications*. Springer, 2011, pp. 73–86.
[20] H. Arraqibah and G. Badr, "Extended component-based motif localization for interacting rna structures," in *ISBRA 2013*, Charlotte, US, May 20 2013.
[21] B. Hunter, "Visualization of secondary rna structure prediction algorithms," 2006.
[22] I. L. Hofacker, W. Fontana, P. F. Stadler, L. S. Bonhoeffer, M. Tacker, and P. Schuster, "Fast folding and comparison of rna secondary structures," *Monatshefte für Chemie/Chemical Monthly*, vol. 125, no. 2, pp. 167–188, 1994.
[23] S. W. Burge, J. Daub, R. Eberhardt, J. Tate, L. Barquist, E. P. Nawrocki, S. R. Eddy, P. P. Gardner, and A. Bateman, "Rfam 11.0: 10 years of rna families," *Nucleic acids research*, p. gks1005, 2012.
[24] D. A. Benson, M. Cavanaugh, K. Clark, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers, "Genbank," *Nucleic acids research*, p. gks1195, 2012.
[25] A. Alturki, "Component based pair-wise rna secondary structure alignment algorithm," Master's thesis, King Saud University, 2014.
[26] 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, no. 5, pp. 0955–964, 1997.