The Author(s) *BMC Bioinformatics* 2017, **18**(Suppl 12):427
DOI 10.1186/s12859-017-1830-6

BMC Bioinformatics

RESEARCH Open Access

DCJ-RNA - double cut and join for RNA secondary structures

Ghada H. Badr[1,2*†] and Haifa A. Al-aqel[3*†]

From 12th International Symposium on Bioinformatics Research and Applications (ISBRA 2016)
Minsk, Belarus. 5-8 June 2016

Abstract

Background: Genome rearrangements are essential processes for evolution and are responsible for existing varieties of genome architectures. Many studies have been conducted to obtain an algorithm that identifies the minimum number of inversions that are necessary to transform one genome into another; this allows for genome sequence representation in polynomial time. Studies have not been conducted on the topic of rearranging a genome when it is represented as a secondary structure. Unlike sequences, the secondary structure preserves the functionality of the genome. Sequences can be different, but they all share the same structure and, therefore, the same functionality.

Results: This paper proposes a double cut and join for RNA secondary structures (DCJ-RNA) algorithm. This algorithm allows for the description of evolutionary scenarios that are based on secondary structures rather than sequences. The main aim of this paper is to suggest an efficient algorithm that can help researchers compare two ribonucleic acid (RNA) secondary structures based on rearrangement operations. The results, which are based on real datasets, show that the algorithm is able to count the minimum number of rearrangement operations, as well as to report an optimum scenario that can increase the similarity between the two structures.

Conclusion: The algorithm calculates the distance between structures and reports a scenario based on the minimum rearrangement operations required to make the given structure similar to the other. DCJ-RNA can also be used to measure the distance between the two structures. This can help identify the common functionalities between different species.

Keywords: Genome Rearrangement, RNA Secondary Structure, DCJ, Similarity Measure, Sorting Scenario

Background

DNA is a biological blueprint that a living organism must have to exist and remain functional. RNA holds the guidelines for this blueprint. RNA is responsible for transferring the genetic code from the nucleus to the ribosome to build proteins. It is identified as a series of letters with bases {A, C, G, U}. RNA's secondary structure is required to define the functionality of RNA molecules. In contrast to representing the genome as a sequence, representing it as a secondary structure provides more insight into the genome's function. In this paper, RNA's secondary structure is presented using a component-based representation, which was recently proposed in 2011 [1]. In contrast to similarity between gene orders, identifying the similarity of functioning between two structures has a greater impact on comparing species. Comparing two species based on their secondary structures provides more information and reveals more accurate evolutionary scenarios [2]. Comparison of two species based on their secondary structures can also be combined with existing sequence-based algorithms to enhance sequence-based algorithms efficiency [3]. This helps create more accurate phylogenies [4].

* Correspondence: badrghada@hotmail.com; haagel@imamu.edu.sa
†Equal contributors
1IRI- The City of Scientific Research and Technological Applications, University and Research District, P. O. 21934, New Borg Alarab, Alexandria, Egypt
2University of Ottawa, Faculty of Engineering, Ottawa, Canada
3Imam Mohammad ibn Saud Islamic University, College of Computer and Information Sciences, Riyadh, Saudi Arabia

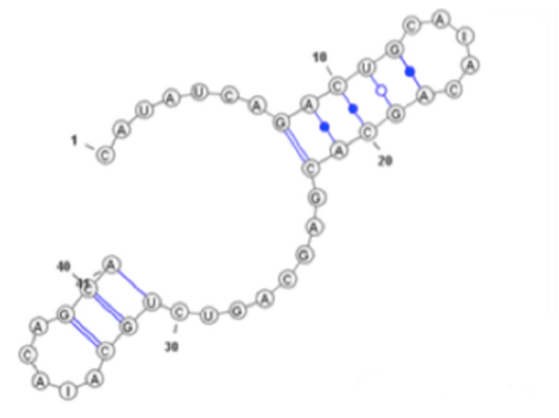
The paper outline is as follows - the RNA secondary structure is presented using a component-based representation. The researchers proceed to describe the measures that are used to determine the similarity between components of the given structures. Genome rearrangement in terms of sequences and its operations, sorting scenario, and distance measures are summarized. We then propose a DCJ-RNA rearrangement algorithm and explain it in detail. Two case studies using real data are presented, illustrating the detection and application of the proposed rearrangement operations for real RNA secondary structures. The results demonstrate that the proposed algorithm provides one evolutionary scenario that shows how to alter one structure to make it similar to the other or the same as the other. Preliminary work has been presented as a poster in [5].

RNA secondary structure component-based representation

Badr and Turcotte [1] propose a component-based structure to define interacting and non-interacting patterns as follows - 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 defined by its sub-patterns (P_i, $0 < i < m$). Each sub-pattern is defined by its length and intermolecular (INTERM) and intramolecular (INTRAM) components. For non-interacting patterns, there are no INTERM components. These components are defined by their opening bracket (OB), closing bracket (CB), length, and relative locations within the sub-patterns. In the INTERM component, OB and CB are located in two different sub-patterns. In the INTRAM component, OB and CB are located in the same sub-pattern. In the INTERM component, OB and CB must be in different sub-patterns, which suggests that there must be at least two sub-patterns to have INTERM components. OB is located in p_i, and CB is located in another sub-pattern (p_j), where $j > i$ and $1 \le j \le m$. OB and CB are defined by their lengths and locations relative to the beginning of p_i. Thus, INTERM = {OB, CB, j, len}. In INTRAM components, OB and CB have to be in the same sub-pattern, which indicates that there must be at least one sub-pattern to have INTRAM components. OB and CB are located in p_i, where $1 \le i \le m$. OB and CB both are defined by their location and length. Therefore, INTRAM = {OB, CB, len}. Figure 1 shows an example of a non-interacting pattern.

Similarities between two RNA secondary structures (Alignment distance)

Badr and AlTurki [6] propose a similarity measure based on aligning two secondary structures that are presented using a component-based representation. The algorithm extracts the features of each component, which are OB, CB, and length. The similarity between two structures depends on the component's position, full length, and stem length. These measures are used in the new proposed algorithm. The equations that are applied to calculate the similarity between two components, a_i in structure A and b_j in structure B, $d(f_{ai}, f_{bj})$, can be found in [6]. The similarity measure between two components is used to calculate the dynamic programming matrix using the method proposed by Needleman and Wunsch [7]. The alignment score between two structures is



```
.......(((((.....))))).........(((.....)))
```

P = { 41, INTERM = { }, INTRAM = { (8, 18, 5), (31, 39, 3) } }

Fig. 1 An example of a component-based representation

calculated using Eq. 1, while the percentage of the similarity between two structures is calculated using Eq. 2 [6].

$$Score(a,b) = \left\{ \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{m} d(fai, fbj) \quad \begin{matrix} \text{if } a_i \text{ is aligned with } b_j \\ 0 \text{ otherwise} \end{matrix} \right\} \tag{1}$$

$$\text{Score percentage }(a,b) = \frac{\text{Score}(a,b)}{\text{Max}(a,b)} \tag{2}$$

where $\text{Max}(a, b) = \text{Max}\ \{Score(a, a), Score(b, b.)\}$

RSmatch [8], which is another alignment distance, is a tool for aligning RNA secondary structures and is also used for motif detection. Determined with widely used algorithms for RNA folding, it decomposes the secondary structure of RNA into a set of atomic structural components. These components are further organized using a tree model to capture the structural particularities. RSmatch can find the optimal global or local alignment between two RNA secondary structures using two scoring matrices - one for single-stranded regions and the other for double-stranded regions. Jiang et al. [9] define the alignment of trees as a measure of similarity between two secondary structures in tree representation.

Sequence-based genome rearrangements

Genomes can be modeled using permutations. Each gene can be allocated once at the genome and assigned a unique number. A gene is modeled by a signed integer when the gene strand is known to biologists [10, 11].

Rearrangement operations

Two genomes can have the same number of genes but may have different orders. A sequence of operations can be applied to change one genome into another. The most common rearrangement events or operations are as follows [12, 13]:

- Inversion - This reverses the orientation of a gene (or a group of genes).
- Transposition - This changes the order of a gene (or a group of genes). In other words, if the gene is located in one index, it is moved to another index.
- Gain - This adds a gene (or a group of genes) to a genome.
- Loss - This removes a gene (or a group of genes) from a genome.
- Duplication - This duplicates a specific gene (or a group of genes) within a genome.

Distance measures

The distance between two genomes is the minimum number of events or operations that are required to transform one genome into the other. Yancopoulos et al. [14] first proposed double cut and join (DCJ) operations. A DCJ operation consists of cutting a genome at two distinct positions and joining the four resulting open ends in a different way. Since a gene (e.g., a) has an orientation, its two ends, namely the extremities, can be distinguished and denoted as at (tail) and ah (head). An adjacency in a genome is either the extremity of a gene that is adjacent to one of its telomeres or a pair of consecutive gene extremities in one of its chromosomes.

DCJ distance consists of two operations - cut, which cuts an adjacency in two telomeres, and join, which connect two telomeres to form an adjacency. A model in which any operation consists of two cuts followed by two joins on the extremities is considered a DCJ operation [15]. DCJ allows for multi-chromosomal genomes with both circular and linear chromosomes.

DCJ distance can be easily calculated with the assistance of an adjacency graph, which is a two-part multigraph in which each partition corresponds to the set of adjacencies of one of the two input genomes. An edge connects the same extremities of genes in both genomes. In other words, a one-to-one correspondence exists between the set of edges in an adjacency graph and the set of gene extremities. Vertices have degree one or two. Therefore, an adjacency graph is a collection of paths and cycles. DCJ distance can be define as follows:

$$\text{dDCJ}\ (G_1, G_2) = N - (c(G_1, G_2) + p(G_1, G_2)/2) \tag{3}$$

In this equation, c (G_1, G_2) is the number of cycles, and p (G_1, G_2) is the number of odd paths in the adjacency graph.

Sorting scenario

One related issue is identifying a sorting scenario for the given distance, which provides the operations themselves. A single or number of possible solutions or sorting sequences can be found.

Bergeron et al. [11] provide an algorithm to obtain the DCJ operation in O(n) time (Algorithm 1). Mathematically, sorting using DCJ operations is simple. As with DCJ distance, DCJ operations take two adjacencies or telomeres, cut the adjacencies/telomeres, and create new adjacencies or telomeres. There are several DCJ operation types. A DCJ operation may create two adjacencies by cutting two adjacencies. A DCJ operation may also create an adjacency and telomere by cutting an adjacency and removing a telomere. In addition, a DCJ operation can consist of forming two telomeres by cutting an adjacency. Finally, DCJ operations may create an adjacency by removing two telomeres.

Algorithm 1 (Greedy sorting by DCJ)

```
for each adjacency {p, q} in genome B do
    let u be the element of genome A that contains p
    let v be the element of genome A that contains q
    if u ≠ v then
        replace u and v in A by {p, q} and $(u \backslash \{p\}) \cup (v \backslash \{q\})$
    end if
end for
for each telomere {p} in genome B do
    let u be the element of genome A that contains p
    if u is an adjacency then
        replace u in A by {p} and $(u \backslash \{p\})$
    end if
end for
```

Method: DCJ-RNA algorithm

The RNA component-based rearrangement algorithm uses a component-based representation [2] that allows for the unique description of any RNA pattern and shows the main features of the pattern efficiently. The proposed algorithm also uses the DCJ algorithm to describe rearrangement operations. It uses classical operations (inversions, translocations, fissions, fusions, transposition, and block interchanges) with a single operation and provides multi-chromosomal genomes. The DCJ-RNA algorithm (Algorithm 2) is described next.

Algorithm 2 (DCJ-RNA)

Input: Two RNA structures in component-based representation $A = (a_1, a_2, \dots, a_n)$ and $B = (b_1, b_2, \dots, b_m)$, measure weights (position W_p, component length W_{cl} and stem length W_{sl}) and threshold enhancement ε.

Output: The minimum number of rearrangement operations that are required to transform one structure into the other with one evolutionary scenario.

```
Begin
    Align similar components based on their components lengths and stem lengths.
    Generate the permutations.
    Apply DCJ on the generated permutation.
End
```

The DCJ-RNA algorithm completes three main steps:

Step 1 - Alignment of similar components based on their component lengths and stem lengths.

In this step, calculate the similarity between components in terms of their component lengths and stem lengths [6]. Similar components are assigned together, beginning with those with the greatest similarity. The similarity measure that is used in this step is as follows -

$$d_1(f_{ai}, f_{bj}) = \text{ComponentLength}(f_{ai}, f_{bi}).\text{StemLength}(f_{ai}, f_{bi}) \quad (4)$$

Then, a matrix (m × n) is built; the entries are the component similarities in terms of component length and stem length. The rows represent the components of the first structure, and the columns represent the components of the second structure. We then search for the maximum entry (greedy) in the matrix. If it is greater than the threshold enhancement (ε) (the minimum similarity score between two components), the components are assigned together, and the corresponding row and column are deleted. If maximum similarity appears in more than one entry, the position similarity is compared between those components only and the assigned components with the greatest similarity in position. Table 1 shows the matrix structure.

Table 1 Component length and stem length similarity

	a_1	a_2	a_3	..	a_n
b_1					
b_2					
b_3					
b_m					

Step 2 - Permutation generation

In this step, a corresponding permutation is generated for each of the two structures. This is completed by determining the components to be inserted or deleted, as well as the order of the similar components using the alignment that is generated from step 1. A two-dimensional array of 3 X in size (the maximum number of components in A or B + 1) is constructed and identified as SortArray. The first row contains the desired structure, the second row contains the deleted components from the actual structure, and the third row contains the inserted components from the desired structure. An index value of zero for the first row is reserved for the number of components in the actual structure. An index value of zero for the second row is

Table 2 The structure of SortArray

Index	0	1 2 3 4 5 … Max + 1
SortArray[0]	# of components in actual structure	Desired Structure Components
SortArray[1]	# of deleted components	Deleted Components
SortArray[2]	# of inserted components	Inserted Components

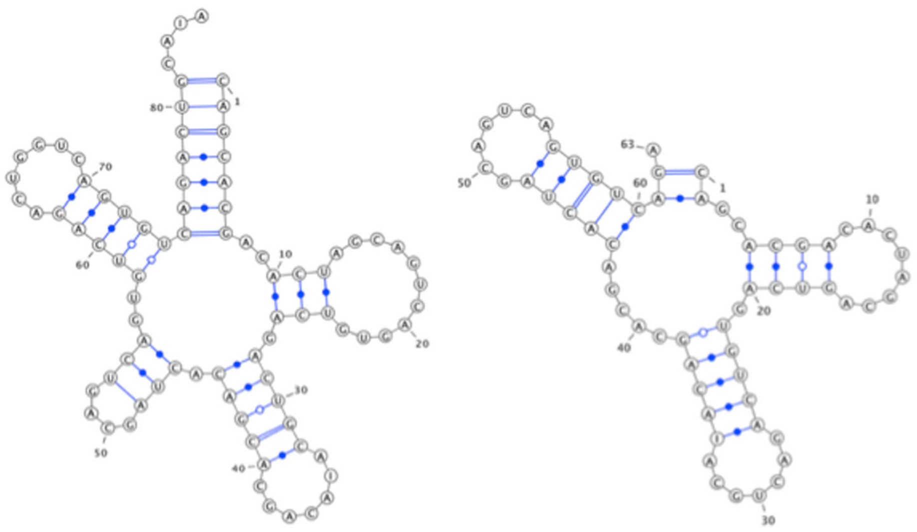


Fig. 2 Structure A (left) and structure B (right)

reserved for the number of deleted components. For third row, an index of zero is reserved for the number of components. Table 2 shows the SortArray structure.

Step 3 - Applying the DCJ algorithm.

The component numbers are used to determine the permutations in the DCJ algorithm [16]. Two permutations are provided. The first is for the given or actual permutation, and the second permutation is for the desired one.

Each permutation has two chromosomes -

For the first permutation - The first chromosome is the actual structure of the components, and the second chromosome is the inserted components.

For the second permutation - The first chromosome is the desired structure, and the second chromosome consists of the deleted components.

Each permutation is represented by its adjacencies and telomeres. Finally, the DCJ algorithm is applied to the first and second permutations as input.

The DCJ algorithm [17] is modified in the way that it is applied to sort the first chromosome from the second permutation; this changes the first chromosome of the first permutation. The second chromosome of the second permutation consists of the deleted components, which do not need to be sorted.

Example

In order to clarify the steps of the algorithm, real RNA secondary structures from the Genomic tRNA Database [18] are used as examples. The first structure is for *E. coli* tRNA for leucine (A), while the other structure is for *E. coli* tRNA for alanine (B) (see Fig. 2).

The two structures are presented using a component-based representation -

- A = (85, INTERM = {}, INRAM = {a_1 = (1, 75, 7), a_2 = (10, 24, 3), a_3 = (28, 40, 5), a_4 = (46, 53, 3), a_5 = (58, 70, 5)})
- B = (76, INTERM = {}, INTRAM = {b_1 = (1, 66, 7), b_2 = (10, 22, 4), b_3 = (27, 39, 5), b_4 = (49, 61, 5)})
- The measure weights are equal to one, and threshold enhancement (ε) is equal to 0.5.

Step 1 - Alignment of similar components based on their component lengths and stem lengths.

In this step, the similarity between components is calculated in terms of their component lengths and stem lengths. Similar components are assigned together, beginning with those with the greatest similarity (greedy).

In this example, the similarity between components is shown in the matrix in Table 3. First, the maximum

Table 3 Similarity between components based on component length and stem length

	b_1	b_2	b_3	b_4
a_1	0.39	0.24	0.29	0.29
a_2	0.34	0.83	0.75	0.75
a_3	0.25	0.86	1	1
a_4	0.22	0.66	0.56	0.56
a_5	0.25	0.86	1	1

Table 4 SortArray for the example

Index	0	1	2	3	4	5
SortArray[0]	5	6(b_1)	2(a_2)	3(a_3)	5(a_5)	
SortArray[1]	2	1(a_1)	4(a_4)			
SortArray[2]	1	6(b_1)				

number is one. The components are assigned together, and the row and column are removed. In this case, d_1 (a_3, b_3) and d_1 (a_3, b_4) are at the same position, so the nearest components are assigned in terms of their position (a_3 and b_3). The same case applies for d_1 (a_5, b_3) and d_1 (a_5, b_4). The maximum value, which is 0.83, is searched for once again. Then, a_2 and b_2 are assigned, and the row and column are deleted. The next value is 0.39, which is less than the threshold enhancement (ε) value, suggesting that b_1 must be inserted and that a_1 must be deleted. Then, a_4 is deleted because no other components remain from the second structure.

Step 2 - Permutation generation

In this step, similar components are mapped according to the process outlined in the previous step. The inserted components and deleted components are then identified (Table 4).

Step 3 - Applying the DCJ algorithm.

The permutations are constructed to apply the DCJ algorithm. The first permutation is chr_1 = {1, 2, 3, 4, 5} and chr_2 = {6}. The permutations are represented as a sequence of numbers. To differentiate between the components of the first structure and the second one, the researchers represent the second structure's component i as i + N, where N equals the number of components in the first structure. The second permutation is chr_1 = {6, 2, 3, 5} and chr_2 = {1, 4}.

Then, each genome is represented with its adjacencies and telomeres to ensure that the DCJ algorithm can be applied; the first and second permutations are as follows:

The first permutation is: {{1 t}, {1 h, 2 t}, {2 h, 3 t}, {3 h, 4 t}, {4 h, 5 t}, {5 h}, {6 t}, {6 h}}
The Second permutation is: {{6 t}, {6 h, 2 t}, {2 h, 3 t}, {3 h, 4 t}, {4 h, 5 t}, {5 h}, {1 t}, {1 h, 4 t}, {4 h}}

In addition, {1 t}, {1 h, 4 t}, and {4 h} will not be sorted because they are included in the second chromosome. After applying the DCJ algorithm, the number of DCJ operations (3) is retrieved, as well as the sorting scenario is:

{{{6 t}, {1 h, 2 t}, {1 t}, {2 h, 3 t}, {3 h, 4 t}, {4 h, 5 t}, {5 h}, {6 h}},
{{6 t}, {6 h, 2 t}, {1 h}, {1 t}, {2 h, 3 t}, {3 h, 4 t}, {4 h, 5 t}, {5 h}},
{{6 t}, {6 h, 2 t}, {1 h}, {1 t}, {2 h, 3 t}, {3 h, 5 t}, {4 h, 4 t}, {5 h}}}.

Figure 3 shows the given structures following each rearrangement operation, as well as the similarity score with the original structure after applying each rearrangement operation. It also shows the final desired operation.

To demonstrate the effect of the DCJ-RNA on increasing the similarity between the structures, the CompPSA algorithm [6] is used to calculate the similarity between the structures before and after applying the algorithm. The similarity between the structures is 42% before applying any changes and increases to 94% after applying the DCJ-RNA algorithm (Fig. 4).

Results and discussion

To test and validate the DCJ-RNA algorithm, extensive experiments are conducted, three experiments are applied to three different datasets.

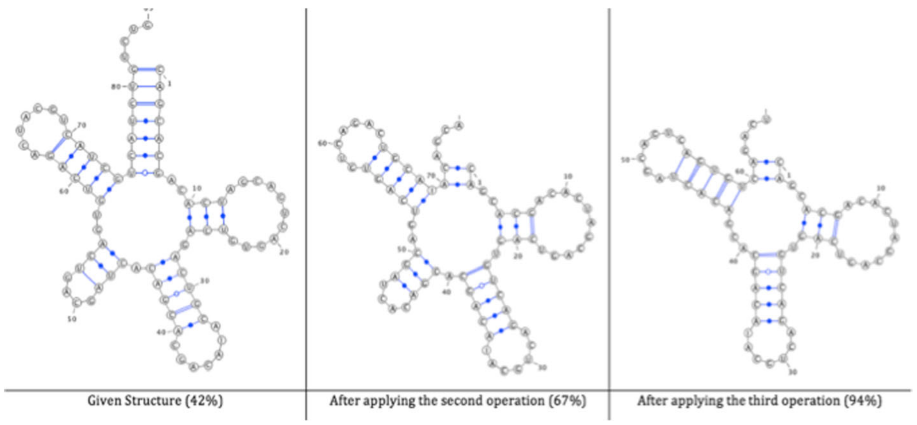

Given Structure (42%)
After applying the second operation (67%)
After applying the third operation (94%)

Fig. 3 The given structures following each operation

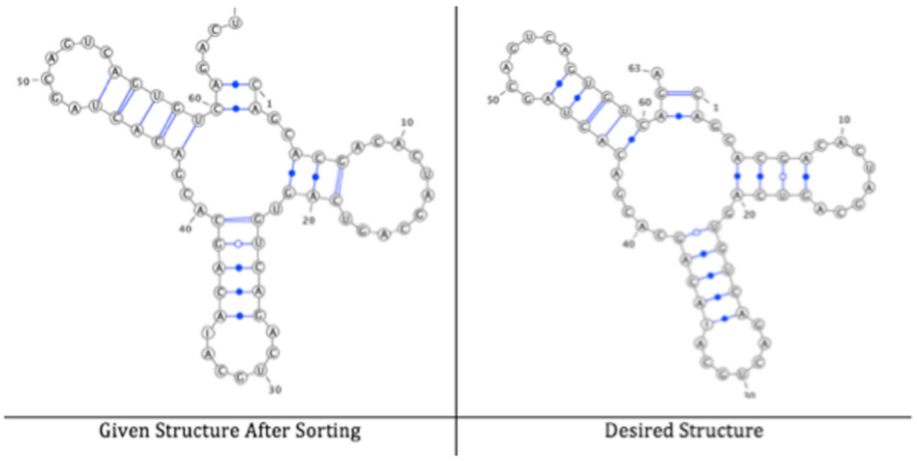
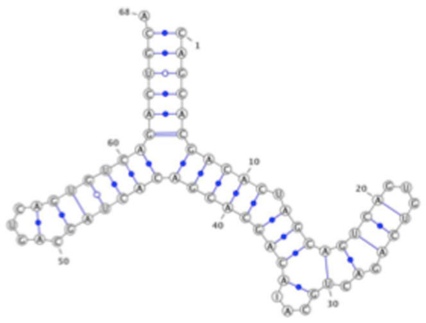


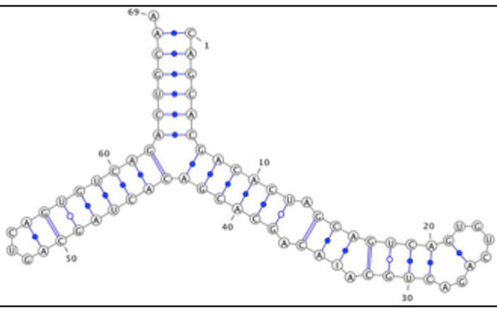
Fig. 4 Structure A after applying the DCJ-RNA algorithm

a



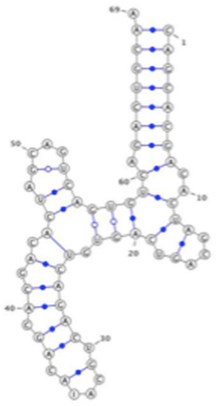
Structure (A) Feature Vectors			
Component	OB	CB	Length
A_1	1	62	6
A_2	7	39	5
A_3	13	36	3
A_4	16	30	1
A_5	17	28	1
A_6	18	24	3
A_7	31	35	1
A_8	44	56	6
A_9	51	55	1

b



Structure (B) Feature Vectors			
Component	OB	CB	Length
B_1	1	63	6
B_2	7	38	6
B_3	14	34	3
B_4	17	29	4
B_5	21	26	1
B_6	44	55	8

c



Structure (C) Feature Vectors			
Component	OB	CB	Length
C_1	1	62	7
C_2	8	60	1
C_3	11	59	1
C_4	12	19	1
C_5	13	17	1
C_6	20	56	3
C_7	24	45	1
C_8	25	43	1
C_9	26	40	2
C_10	28	37	2
C_11	31	35	1
C_12	46	54	2
C_13	49	53	1

Fig. 5 Structures A, B, and C, respectively, with their features listed as follows (ComponentID, opening bracket, closing bracket, component length)

Datasets

There are three different datasets - adjust dataset, accuracy dataset and scalability dataset. In this section, each dataset is described in detail.

Adjust dataset

This dataset consists of three real RNA structures named A, B and C shown in Fig. 5 where selected from the NCBI GenBank [16]. it is used to determine the best threshold enhancement (ε) value. There are two cases for RNA similarities. Dissimilar sequences and exact/approximate similar structures, structures A and B are used. In other case, dissimilar structures and exact/approximate similar sequences, structures A and C are used.

Accuracy dataset

The accuracy dataset is used to calculate the performance and accuracy of the DCJ-RNA algorithm using different RNA structure sizes. This dataset consists of three pairs of RNA structures that are chosen from the GenBank [19] and Rfam database [20] and differ in size. The first pair of RNA structures consists of two small RNA structures; named D and E, as shown in Fig. 6.

The second pair consists of two medium RNA structures; named F and G, as shown in Fig. 7.

The third pair consists of two large RNA structures; named H and I, as shown in Fig. 8.

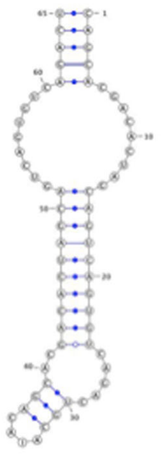
Scalability dataset

The scalability dataset is used to calculate the scalability of the time and memory performance of the DCJ-RNA algorithm using different RNA structure sizes. This dataset consists of 11 RNA structures based on the first RNA structure, A, in the adjust dataset. Then the second structure is a duplicate of the first one, the third structure is a duplicate of the second one, and so on. The RNA structures' numbers, names, sizes, and number of components are shown in Table 5. The first six RNA structures (J, K, L, M, N, and O) are shown in Fig. 9.

Experiments

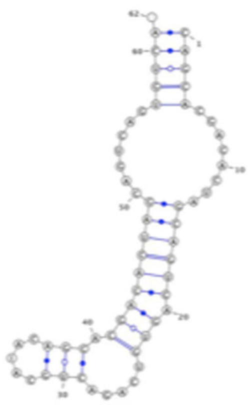
Three experiments are conducted - threshold adjustment, performance accuracy, and time and memory

a



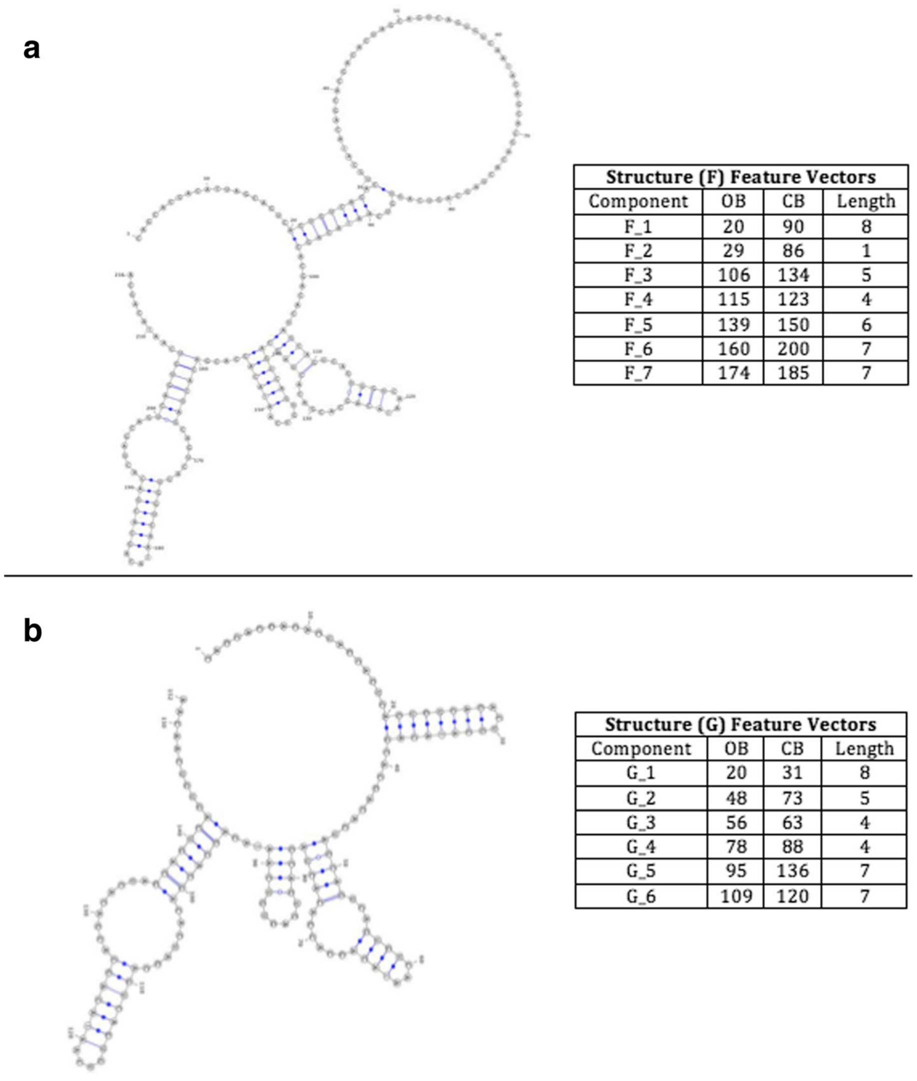
Structure (D) Feature Vectors			
Component	OB	CB	Length
D_1	1	61	5
D_2	15	42	10
D_3	30	37	3

b



Structure (E) Feature Vectors			
Component	OB	CB	Length
E_1	1	57	5
E_2	14	44	6
E_3	21	41	3
E_4	29	37	3

Fig. 6 Structures D and E, respectively, with their features listed as follows (ComponentID, opening bracket, closing bracket, component length)



Structure (F) Feature Vectors			
Component	OB	CB	Length
F_1	20	90	8
F_2	29	86	1
F_3	106	134	5
F_4	115	123	4
F_5	139	150	6
F_6	160	200	7
F_7	174	185	7

Structure (G) Feature Vectors			
Component	OB	CB	Length
G_1	20	31	8
G_2	48	73	5
G_3	56	63	4
G_4	78	88	4
G_5	95	136	7
G_6	109	120	7

Fig. 7 Structures F and G, respectively, with their features listed as follows (ComponentID, opening bracket, closing bracket, component length)

performance experiments, the experiments are obtained using real and simulated data in [19].

Threshold adjustment experiment

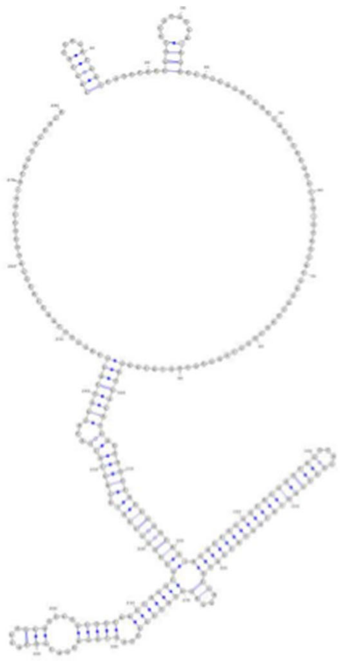
Threshold adjustment experiments are conducted to determine the best threshold enhancement (ε) value that gives the minimum number of rearrangement operations to make the RNA structures exactly the same or approximately similar.

Experiment setup The used dataset is the adjust dataset, while fixed parameters are W_P equals 0 and W_{cl} and W_{sl} equal 1. Experiments are conducted for 10 values of threshold enhancement (ε) from 0 to 1.

Experiment results We change the value of the threshold enhancement (ε) from 0.0, 0.1, 0.2, … 1.0 and obtain the result shown in Table 6 for both cases - similar structures with dissimilar sequences and similar structures with dissimilar sequences. As illustrated in Table 7, when the threshold enhancement (ε) equals 1.0, it means that the RNA structures are exactly similar but the number of the rearrangement operations is greater than the other values. On the other side, when threshold enhancement (ε) equals 0.0, it means that when the desired structure has less than or equal number of components as compared to the given structure, the order of the components is changed, and no components are added or deleted.

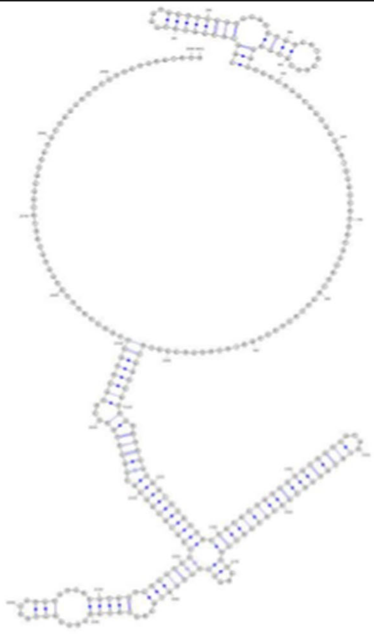
From results, it can be seen that when the structures are similar, the best threshold enhancement (ε) equals 0.6, because of the similarity between structures and the number of rearrangement operations is reasonable; the structures after sorting for each threshold enhancement

a



Structure (H) Feature Vectors			
Component	OB	CB	Length
H_1	1	10	5
H_2	22	34	4
H_3	97	237	8
H_4	105	232	2
H_5	108	227	5
H_6	113	216	10
H_7	123	145	18
H_8	163	168	2
H_9	170	209	7
H_10	180	203	5
H_11	189	196	3

b



Structure (I) Feature Vectors			
Component	OB	CB	Length
I_1	1	44	3
I_2	4	16	8
I_3	28	40	4
I_4	103	243	8
I_5	111	238	2
I_6	114	233	5
I_7	119	222	10
I_8	129	151	18
I_9	169	174	2
I_10	176	215	7
I_11	186	209	5
I_12	195	202	3

Fig. 8 Structures H and I, respectively, with their features listed as follows (ComponentID, opening bracket, closing bracket, component length)

Table 5 RNA structures with their features

RNA structure #	1	2	3	4	5	6	7	8	9	10	11
RNA Structure Name	J	K	L	M	N	O	P	Q	R	S	T
Size (length)	68	136	272	544	1088	2176	4352	8704	10,336	20,672	41,344
Components Number	9	18	36	72	144	288	576	1152	1368	2736	5472

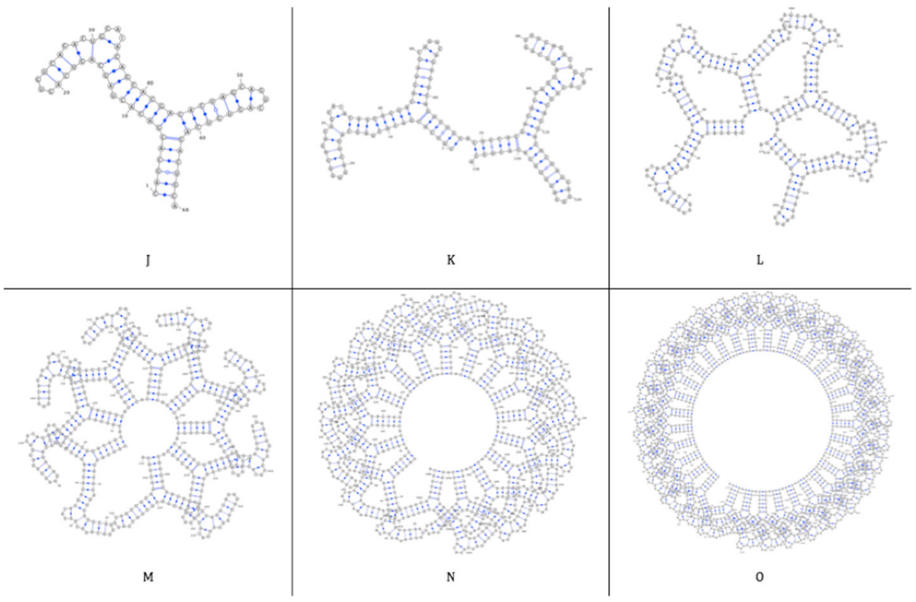


Fig. 9 Scalability dataset with six RNA structures

(ε) are shown in Fig. 10. For the same reason, when the structures are dissimilar, the best threshold enhancement (ε) equals 0.8.

Performance accuracy experiment

The performance accuracy experiment is conducted to show the accuracy of the DCJ-RNA algorithm with different RNA sizes. To test the effect of the DCJ-RNA algorithm and calculate the similarity between structures, the CompPSA algorithm [6] is used.

Experiment setup The dataset used is accuracy dataset. Since all three RNA structures pairs are similar in their structures and dissimilar in their sequences, the threshold enhancement (ε) equals 0.6 and fixed parameters are W_P equals 0 and W_{cl} and W_{sl} are equal to 1.

Table 6 Different threshold enhancement (ε) values with algorithm accuracy

	Similar structures and dissimilar sequences (35%)		Similar sequences and dissimilar structures (20%)	
Threshold enhancement (ε)	CompPSA	Rearrangement operations	CompPSA	Rearrangement operations
0.0	64%	2	59%	13
0.1	64%	2	71%	14
0.2	64%	2	71%	14
0.3	64%	2	71%	14
0.4	64%	2	71%	14
0.5	64%	2	94%	14
0.6	64%	2	94%	14
0.7	69%	3	94%	14
0.8	69%	3	97%	14
0.9	71%	4	100%	14
1.0	100%	7	100%	14

Experiment results DCJ-RNA was applied to three pairs of RNA structures - small, medium, and large RNA structures. Each experiment is discussed in detail in the following.

Small pairs of RNA structures

Step 1 - Alignment of Similar Components Based on Component Lengths and Stem Lengths

Calculate the similarity between components as shown in Table 8. Then assign similar components together whenever the similarity between them is greater than or equal to threshold enhancement (ε), which is 0.6. Here, assign D_1 with E_1, E_4 with D_3, E_2 with D_2, and add E_3.

Step 2 - Permutation Generation

Table 7 Length similarity of small pairs of RNA structures in terms of component length and stem length

	E_1	E_2	E_3	E_4
D_1	0.97	0.65	0.39	0.22
D_2	0.5	0.74	0.35	0.21
D_3	0.21	0.29	0.61	0.95

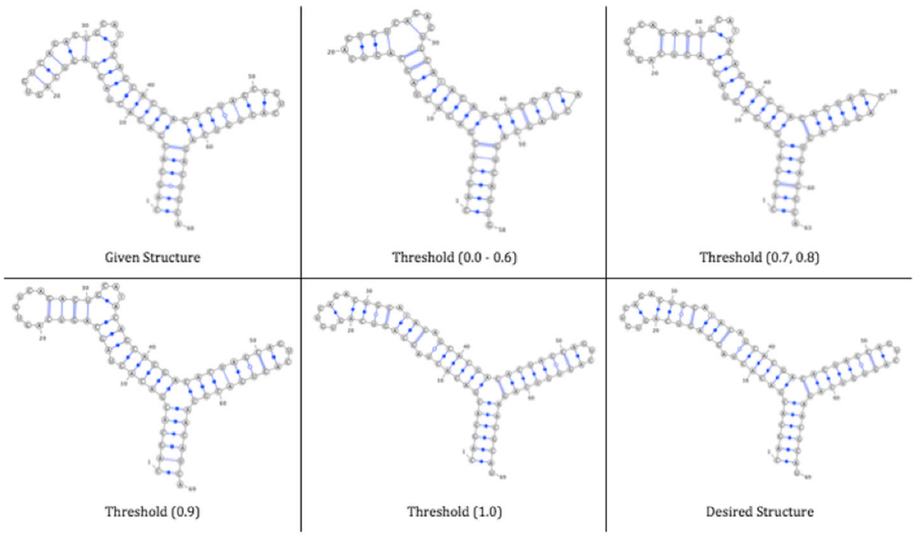


Fig. 10 RNA structures after sorting for each threshold enhancement (ε)

Construct SortArray, fill it as shown in Table 9. After that, construct the permutations to apply the DCJ algorithm.

Step 3 - Apply the Double Cut and Join Algorithm

Construct the permutations to apply the DCJ algorithm. First permutation is (chr_1 = {1,2,3} and chr_2 = {6}). (Note - permutation represented as a sequence of numbers, to differentiate between the first structure's components and the second structure's components, we represent the second structure's component i as i + N, where N equals the number of components in the first structure.) The second permutation is - (chr_1 = {1,2,6,3} and chr_2 = {}). Represent each genome with its adjacencies and telomeres to apply the DCJ algorithm, the first and second permutations are as follows:

The first permutation is: {{1 t}, {1 h, 2 t}, {2 h, 3 t}, {3 h}, {6 t}, {6 h}}
The second permutation is: {{1 t}, {1 h, 2 t}, {2 h, 6 t}, {6 h, 3 t}, {3 h}}

After applying the DCJ algorithm, we obtain the number of the DCJ operations, which is 2, and the sorting scenario is:

Table 8 SortArray for small pairs of RNA structures

Index	0	1	2	3	4
SortArray[0]	3	1(D_1)	2(D_2)	6(E_3)	3(D_3)
SortArray[1]	0				
SortArray[2]	1	6(E_3)			

{{{1 t}, {1 h, 2 t}, {2 h, 3 t}, {3 h}, {6 t}, {6 h}}, {{1 t}, {1 h, 2 t}, {2 h, 6 t}, {6 h, 3 t}, {3 h}}}

The similarity between the given structures D and E is 58% before applying any changes, while it increases to 85% after applying the DCJ-RNA algorithm; see Fig. 11.

Medium pairs of RNA structures

Step 1 - Alignment of Similar Components Based on Component Lengths and Stem Lengths

Calculate the similarity between components as shown in Table 10, then, assign F_7 with G_6, F_6 with G_5, F_4 with G_3, F_3 with G_2, F_5 with G_1, delete F_1, delete $F_{2,}$ and add G_4.

Step 2 - Permutation Generation

Table 9 Length similarity of medium pairs of RNA structures in terms of component length and stem length

	G_1	G_2	G_3	G_4	G_5	G_6
F_1	0.39	0.43	0.16	0.2	0.71	0.35
F_2	0.11	0.23	0.13	0.16	0.23	0.12
F_3	0.56	0.95	0.44	0.53	0.68	0.59
F_4	0.52	0.51	0.96	0.92	0.29	0.58
F_5	0.81	0.66	0.63	0.72	0.48	0.9
F_6	0.54	0.65	0.28	0.33	0.99	0.55
F_7	0.91	0.62	0.55	0.64	0.55	1.0

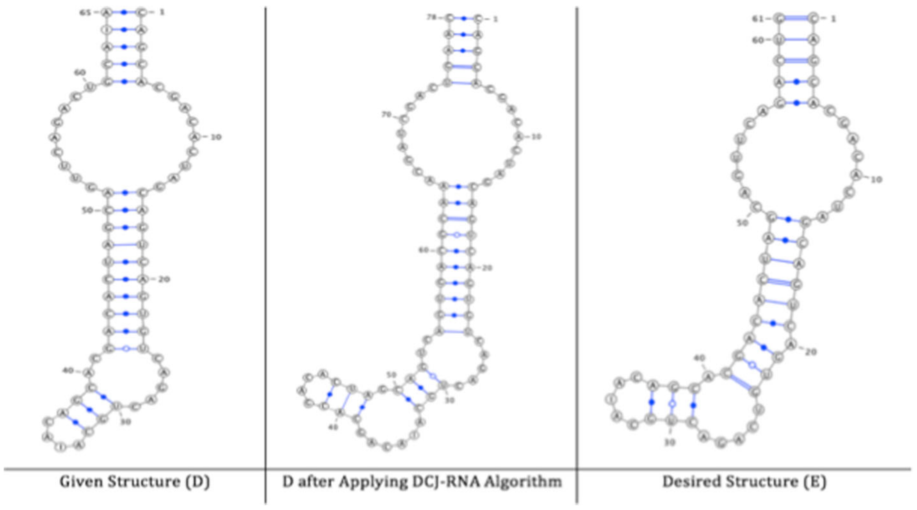


Fig. 11 Given, sorted, and desired structures for small pairs of RNA structures

Construct SortArray, fill it as shown in Table 11. After that, construct the permutations to apply the DCJ algorithm.

Step 3 - Apply the Double Cut and Join Algorithm

Construct the permutations to apply the DCJ algorithm. The first permutation is (chr_1 = {1, 2, 3, 4, 5, 6, 7} and chr_2 = {11}). The second permutation is - (chr_1 = {5, 3, 4, 11, 6, 7} and chr_2 = {1, 2}). Represent each genome with its adjacencies and telomeres as:

The first permutation is: {{1 t}, {1 h, 2 t}, {2 h, 3 t}, {3 h, 4 t}, {4 h}, {5 t}, {5 h, 6 t}, {6 h, 7 t}, {7 h}, {11 t}, {11 h}}
The second permutation is: {{5 t}, {5 h, 3 t}, {3 h, 4 t}, {4 h, 11 t}, {11 h, 6 t}, {6 h, 7 t}, {7 h}, {1 t}, {1 h, 2 t}, {2 h}}

After applying the DCJ algorithm, we obtain the number of the DCJ operations, which is 4, and the sorting scenario is:

{{{1 t}, {1 h, 2 t}, {2 h, 3 t}, {3 h, 4 t}, {4 h}, {5 t}, {5 h, 6 t}, {6 h, 7 t}, {7 h}, {11 t}, {11 h}},
{{1 t}, {1 h, 2 t}, {2 h, 6 t}, {3 h, 4 t}, {4 h}, {5 t}, {5 h, 3 t}, {6 h, 7 t}, {7 h}, {11 t}, {11 h}}

Table 10 SortArray for medium pairs of RNA structures

Index	0	1	2	3	4	5	6	7
SortArray[0]	7	5(F_5)	3(F_3)	4(F_4)	11(G_4)	6(F_6)	7(F_7)	
SortArray[1]	2	1(F_1)	2(F_2)					
SortArray[2]	1	11(G_4)						

{{1 t}, {1 h, 2 t}, {2 h, 6 t}, {3 h, 4 t}, {4 h, 11 t}, {5 t}, {5 h, 3 t}, {6 h, 7 t}, {7 h}, {11 h}}
{{1 t}, {1 h, 2 t}, {2 h}, {3 h, 4 t}, {4 h, 11 t}, {5 t}, {5 h, 3 t}, {6 h, 7 t}, {7 h}, {11 h, 6 t}}}

The similarity between the given structures F and G is 49% before applying any changes, while it increases to 94% after applying the DCJ-RNA algorithm; see Fig. 12.

Large pairs of RNA structures

Step 1 - Alignment of Similar Components Based on Component Lengths and Stem Lengths

Calculate the similarity between components as shown in Table 4.7, then, assign H_1 with I_2, H_2 with I_3, H_3 with I_4, H_4 with I_5, H_5 with I_6, H_6 with I_7, H_7 with I_8, H_8 with I_9, H with I_{10}, H_{10} with I_{11}, H_{11} with $I_{12,}$ and insert I_1.

Step 2 - Permutation Generation

Construct SortArray fill it as shown in Table 12. After that, construct the permutations to apply the DCJ algorithm.

Step 3 - Apply the Double Cut and Join Algorithm

Construct the permutations to apply the DCJ algorithm. The first permutation is (chr_1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11} and chr_2 = {12}). The second permutation is - (chr_1 = {12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11} and chr_2 = {}). Represent each genome with its

Table 11 Length similarity of large pairs of RNA structures in terms of component length and stem length

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}
H_1	0.35	0.63	0.83	0.13	0.11	0.2	0.15	0.23	0.38	0.39	0.67	0.62
H_2	0.44	0.59	1.0	0.13	0.15	0.2	0.14	0.21	0.41	0.38	0.65	0.66
H_3	0.26	0.24	0.13	1.0	0.37	0.7	0.77	0.26	0.04	0.44	0.24	0.07
H_4	0.42	0.11	0.15	0.37	1.0	0.56	0.31	0.09	0.1	0.23	0.2	0.12
H_5	0.41	0.21	0.2	0.7	0.56	1.0	0.64	0.21	0.06	0.45	0.37	0.11
H_6	0.27	0.27	0.14	0.77	0.31	0.64	1.0	0.37	0.04	0.48	0.26	0.08
H_7	0.27	0.41	0.21	0.26	0.09	0.21	0.37	1.0	0.06	0.52	0.36	0.11
H_8	0.21	0.21	0.41	0.04	0.1	0.06	0.04	0.06	1.0	0.12	0.23	0.66
H_9	0.6	0.57	0.38	0.44	0.23	0.45	0.48	0.52	0.12	1.0	0.63	0.21
H_{10}	0.57	0.64	0.65	0.24	0.2	0.37	0.26	0.36	0.23	0.63	1.0	0.39
H_{11}	0.36	0.36	0.66	0.07	0.12	0.11	0.08	0.11	0.66	0.21	0.39	1.0

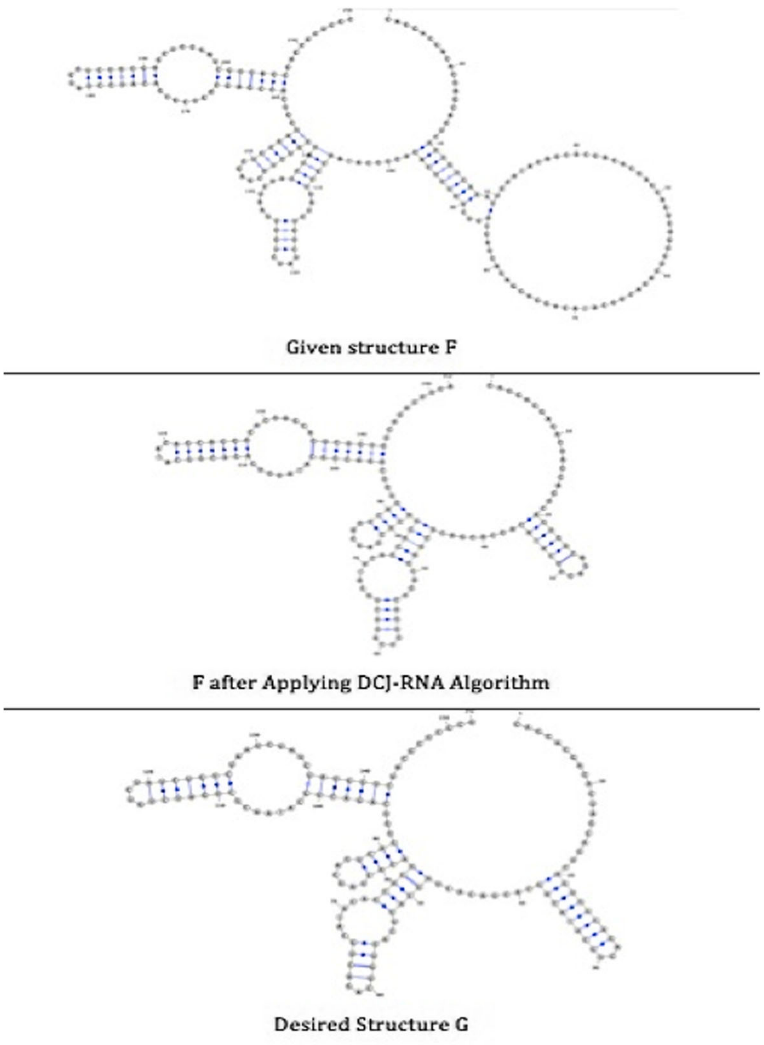


Fig. 12 Given, sorted, and desired structures for medium pairs of RNA structures

Table 12 SortArray for large pairs of RNA structures

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
SortArray[0]	12	12(I_1)	1(H_1)	2(H_2)	3(H_3)	4(H_4)	5(H_5)	6(H_6)	7(H_7)	8(H_8)	9(H_9)	10(H_{10})	11(H_{11})
SortArray[1]	0												
SortArray[2]	1	12(I_1)											

adjacencies and telomeres to apply the DCJ algorithm, as the following:

The first permutation is: {{1 t}, {1 h, 2 t}, {2 h, 3 t}, {3 h, 4 t}, {4 h, 5 t}, {5 h, 6 t}, {6 h, 7 t}, {7 h, 8 t}, {8 h, 9 t}, {9 h, 10 t}, {10 h, 11 t}, {11 h}, {12 t}, {12 h}}
The second permutation is: {{12 t}, {12 h, 1 t}, {1 h, 2 t}, {2 h,3 t}, {3 h, 4 t}, {4 h, 5 t}, {5 h, 6 t}, {6 h, 7 t}, {7 h, 8 t}, {8 h, 9 t}, {9 h, 10 t}, {10 h, 11 t}, {11 h}}

After applying the DCJ operations, we get the number of the DCJ algorithm, which is 2, and the sorting scenario is:

{{{12 t}, {1 t}, {1 h, 2 t}, {2 h, 3 t}, {3 h, 4 t}, {4 h, 5 t}, {5 h, 6 t}, {6 h, 7 t}, {7 h, 8 t}, {8 h, 9 t}, {9 h, 10 t}, {10 h, 11 t}, {11 h},{12 h}},
{{12 t}, {12 h, 1 t}, {1 h, 2 t}, {2 h,3 t}, {3 h, 4 t}, {4 h, 5 t}, {5 h, 6 t}, {6 h, 7 t}, {7 h, 8 t}, {8 h, 9 t}, {9 h, 10 t}, {10 h, 11 t}, {11 h}}}

The similarity between the given structures H and I is 84% before applying any changes, while it increases to 91% after applying the DCJ-RNA algorithm; see Fig. 13.

Time & Memory performance experiment

The time and memory performance experiment is conducted to test the performance of the DCJ-RNA algorithm using different RNA structure sizes.

Experiment setup The scalability dataset is used, while fixed parameters W_P equals 0 and W_{cl} and W_{sl} are equal to 1. Threshold enhancement (ε) equals 0.6 since

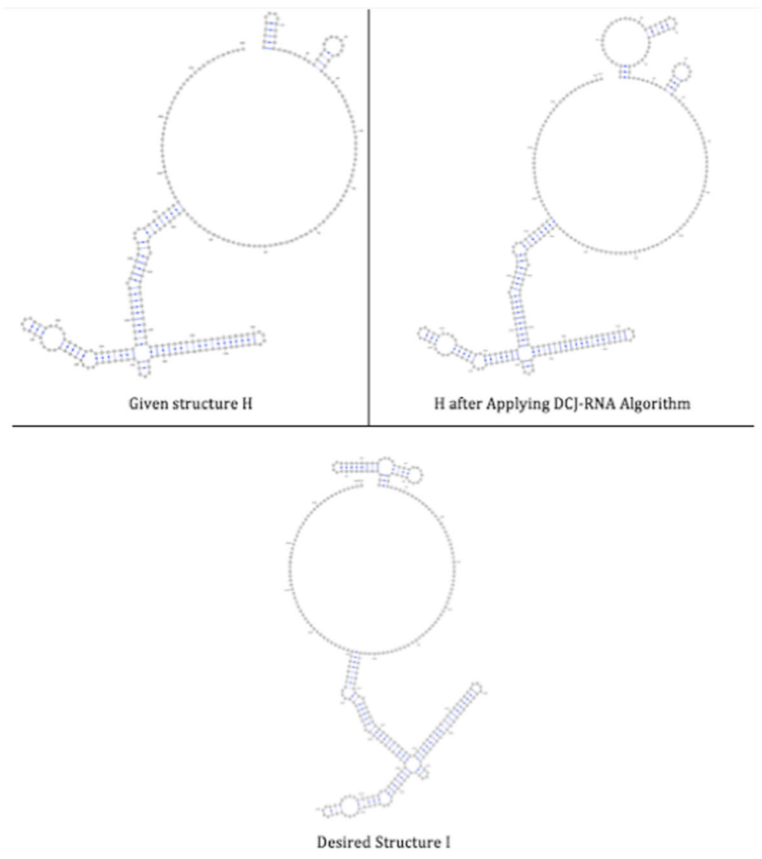


Fig. 13 Given, sorted, and desired structures for large pairs of RNA structures

Table 13 Time and memory performance results of the DCJ-RNA algorithm

Length	Number of components	Time in seconds	Memory usage in MB
68	9	0.010739	1.11
136	18	0.020159	1.11
272	36	0.026246	1.78
544	72	0.039157	3.44
1088	144	0.130200	9.38
2176	288	0.208723	1.50
4352	576	0.502496	4.43
8704	1152	2.657500	17.50

structures are similar. The two structures in each experiment are identical which means the similarity between them is 100%.

Experiment results *Consider the maximum number of components to be N; the time complexity of step 1 is O(N log N) for the worst case. Each time we have to search for the maximum value for N values then discard the row and column related to maximum value, as a result, the next search is applied to (N-1) components and so on. The time complexity of the second step is O(N), since this step determines the inserted components and the deleted components. The algorithm moves through the entries only once to fill SortArray in which they are all of size N. For step three, the time complexity is O(N) since the DCJ algorithm is used. Therefore, the worst time for the entire algorithm is O(N log N).* Table 13 *and* Fig. 14 *confirm the time performance analysis empirically using the scalability dataset. The space requirement for the first step is* $O(N^2)$ *when the same number of components are present. For the second step, the memory takes O(3 N) for SortArray. For the third step, the space of memory is O(2 N). Hence, the total space requirement for DCJ-RNA algorithm is* $O(N^2)$*.* Table 13 *shows time and memory performance results from this experiment and the corresponding graph representation* (Fig. 14).

Conclusion

The DCJ-RNA algorithm is proposed and is able to describe the evolutionary scenarios that are based on

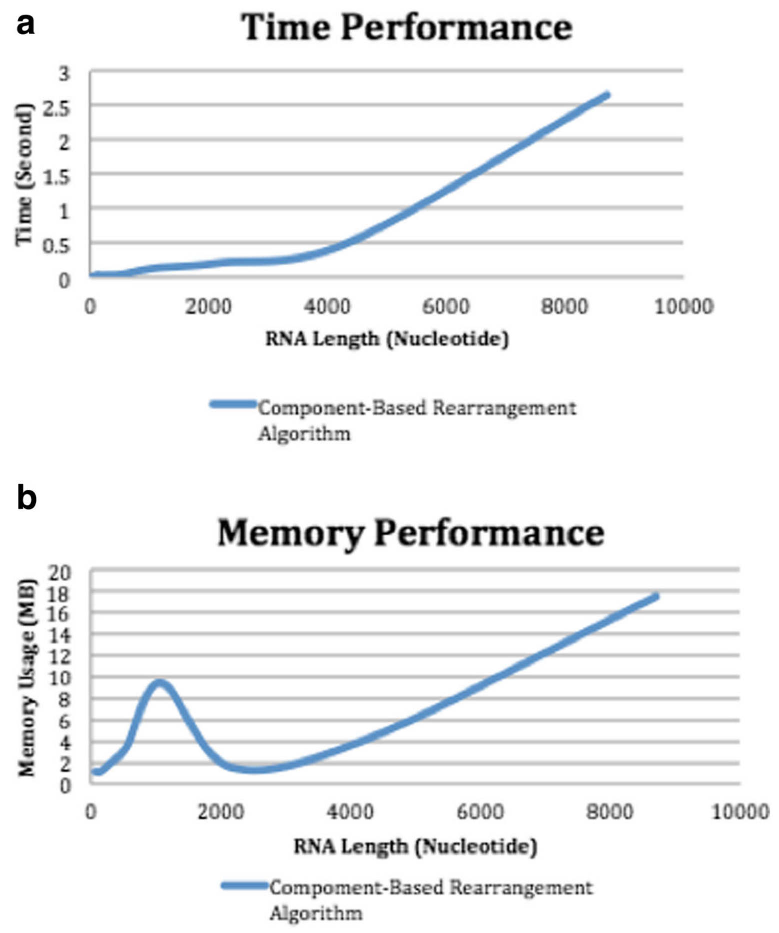


Fig. 14 The performance results for time (**a**) and memory (**b**)

rearrangements of secondary structures rather than sequences. The DCJ-RNA algorithm is optimal. Since RNA secondary structures reveal more functionality, this algorithm can help in the comparison between the functionality of structures. Real data is used to illustrate the details of the proposed algorithm. It demonstrates that the algorithm is able to detect the minimum number of rearrangement operations in order to make one structure more similar to the other. A rearrangement scenario increases similarity between the first structure and any other structure. This creates an ideal framework for applying rearrangement operations to secondary structures rather than sequences.

The algorithm is applied to non-interacting patterns only. Therefore, future work should extend the algorithm to consider interacting RNA patterns. In addition, the researchers would like to explore other well-defined structures, such as chemical structures, and investigate the application of a similar approach that can define a scenario for changing one structure into another structure. Using the DCJ-RNA approach, we would also like to develop a tool that can help biologists compare RNA structures to folded RNA structures that are based on the corresponding RNA sequence. This tool, which is unavailable, would be ideal for biologists, as suggested at the RECOMB-CG conference in 2014.

Acknowledgements
A 2-page abstract has been published in Lecture notes in computer science: Bioinformatics research and applications.

Funding
This research has been supported by the National Plan for Sciences and Technology, King Saud University, Riyadh, Saudi Arabia (Project No. 12-BIO2605–02). The Funding institute did not play any role in design and conclusions. The publication costs were covered by the authors.

Availability of data and materials
Data can be available upon request.

About this supplement
This article has been published as part of BMC Bioinformatics Volume 18 Supplement 12, 2017: Selected articles from the 12th International Symposium on Bioinformatics Research and Applications (ISBRA-16): bioinformatics. The full contents of the supplement are available online at https://bmcbioinformatics.biomedcentral.com/articles/supplements/volume-18-supplement-12.

Authors' contributions
GB proposed, conceived, designed, and coordinated the study, helped in drafting of the manuscript, and critically revised the final manuscript. HA designed the benchmark, developed the DCJ-RNA steps, carried out testing and validation, and helped in drafting of the manuscript. All authors participated in analysis and interpretation of results. Both authors read and approved the final manuscript.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Published: 16 October 2017

References

1. Badr G, Turcotte M. Component-based matching for multiple interacting RNA sequences. In: 7th International Conference on Bioinformatics Research and Application. Berlin, Heidelberg; 2011. p. 73–86.
2. Gesell T, Schuster P. Phylogeny and evolution of RNA structure. Methods Mol Biol. 2014;1097:319–78.
3. Shang L, Gardner D, Xu W, Cannone J, Miranker D, Ozer S, Gutell R. Two accurate sequence, structure, and phylogenetic template-based RNA alignment systems. BMC Syst Biol. 2013;7(4):1–15.
4. Keller A, Förster F, Müller T, Dandekar T, Schultz J, Wolf M. Including RNA secondary structures improves accuracy and robustness in reconstruction of phylogenetic trees. Biol Direct. 2010;5:1–12.
5. Badr G, Alaqel H. Genome rearrangement for RNA secondary structure using a component-based representation - An initial framework. New York: Poster presentation at RECOMB-CG; 2014.
6. Alturki A, Badr G, Benhidour H. Component-based pair-wise RNA secondary structure alignment algorithm, Master Project. Riyadh: King Saud University; 2013.
7. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol. 1970; 48(3):443–53.
8. Liu J et al. A method for aligning RNA secondary structures and its application to RNA motif detection. BMC Bioinformatics. 2005;6–89. doi:10.1186/1471-2105-6-89.
9. Jiang T, Wang L, Zhang K. Alignment of trees - An alternative to tree edit. In: Crochemore M, Gusfield D, editors. Combinatorial Pattern Matching. Berlin, Heidelberg: Springer; 1994. p. 75–86.
10. Hannenhelli S, Pevzner PA. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals. In: 27th Annual ACM Symposium on the Theory of Computing; 1995. p. 178–89.
11. Bergeron A, Mixtacki J, Stoye J. A unifying view of genome rearrangements. In: B√°cher P, Moret BE, editors. Algorithms in Bioinformatics. vol. 4175. Berlin, Heidelberg: Springer; 2006. p. 163–73.
12. Hannenhalli S, Pevzner PA. Transforming men into mice (polynomial algorithm for genomic distance problem). In: Foundations of Computer Science, 1995 Proceedings, 36th Annual Symposium on Foundations of Computer Science; 1995. p. 581–92.
13. Dias Z, Meidanis J. Genome rearrangements distance by fusion, fission, and transposition is easy. In - String Processing and Information Retrieval, SPIRE 2001 Proceedings, 8th International Symposium on 13–15 Nov 2001. p. 250–3.
14. Yancopoulos S, Attie O, Friedberg R. Efficient sorting of genomic permutations by translocation, inversion, and block interchange. Bioinformatics. 2005;21:3340–6.
15. Christie - Genome rearrangement problems, Ph.D. Dissertation. Glasgow: Department of Computer Science, Glasgow University; 1998.
16. Chan PP, Lowe TM. GtRNAdb - A database of transfer RNA genes detected in genomic sequence. Nucleic Acids Res. 2009;37(Database):D93–D97.
17. Zhang K, Shasha D. Simple fast algorithms for the editing distance between trees and related problems. SIAM J Comput. 1989;18:1245–62.
18. Alaqel H, Badr G. Genome rearrangement for RNA secondary structure using a component-based representation: Master Project. Riyadh: King Saud University; 2015.
19. Benson DA, Cavanaugh M, Clark K, Karsch-Mizrachi I, Lipman DJ, Ostell J, Sayers EW. GenBank. Nucleic Acids Res. 2013;41(Database issue):D36-42.
20. Burge SW, Daub J, Eberhardt R, Tate J, Barquist L, Nawrocki EP, et al. Rfam 11.0–10 years of RNA families. Nucleic Acids Research. 2012:1–7.