BMC Bioinformatics

RESEARCH ARTICLE Open Access

Numerical integration methods and layout improvements in the context of dynamic RNA visualization

Boris Shabash and Kay C. Wiese*

Abstract

Background: RNA visualization software tools have traditionally presented a static visualization of RNA molecules with limited ability for users to interact with the resulting image once it is complete. Only a few tools allowed for dynamic structures. One such tool is jViz.RNA. Currently, jViz.RNA employs a unique method for the creation of the RNA molecule layout by mapping the RNA nucleotides into vertexes in a graph, which we call the detailed graph, and then utilizes a Newtonian mechanics inspired system of forces to calculate a layout for the RNA molecule. The work presented here focuses on improvements to jViz.RNA that allow the drawing of RNA secondary structures according to common drawing conventions, as well as dramatic run-time performance improvements. This is done first by presenting an alternative method for mapping the RNA molecule into a graph, which we call the compressed graph, and then employing advanced numerical integration methods for the compressed graph representation.

Results: Comparing the compressed graph and detailed graph implementations, we find that the compressed graph produces results more consistent with RNA drawing conventions. However, we also find that employing the compressed graph method requires a more sophisticated initial layout to produce visualizations that would require minimal user interference. Comparing the two numerical integration methods demonstrates the higher stability of the Backward Euler method, and its resulting ability to handle much larger time steps, a high priority feature for any software which entails user interaction.

Conclusion: The work in this manuscript presents the preferred use of compressed graphs to detailed ones, as well as the advantages of employing the Backward Euler method over the Forward Euler method. These improvements produce more stable as well as visually aesthetic representations of the RNA secondary structures. The results presented demonstrate that both the compressed graph representation, as well as the Backward Euler integrator, greatly enhance the run-time performance and usability. The newest iteration of jViz.RNA is available at https://jviz.cs.sfu.ca/download/download.html.

Keywords: RNA, Visualization, Graph layout, Numerical integration

Background

RNA and its structure

Ribo-nucleic Acid (RNA) is a polymer of nitrogenous bases, composed mainly of Adenine, Cytosine, Guanine, and Uracil (denoted as A, C, G and U, respectively). RNA is very similar to Deoxyribo-nucleic Acid (DNA) in its basic composition, but while DNA is regularly found as two complimentary strands, RNA can be found as a single strand of nucleotides. This primary structure (the string of nucleotides) then folds over itself into a secondary structure when the bases in the RNA strands pair up via hydrogen bonding. The RNA molecule can then twist, fold, or otherwise change its conformation in 3D space, giving it a functional three dimensional form, known as the tertiary structure.

Single stranded, functional RNA is an important agent in many biological processes. From humans to bacteria and viruses, there are many examples of RNA molecules that are important to understand, classify, and research.

*Correspondence: wiese@sfu.ca
School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, BC, Canada

Some notable examples include RNA motifs that allow viruses to manipulate host replication machinery [1–5], bacterial RNA motifs that give rise to antibiotic resistance [6, 7], and man-made RNA molecules designed for therapeutics [8].

RNA secondary structure visualization

There are many RNA visualization tools that have been developed, and two excellent reviews of them can be found in [9, 10], with notable examples that are still available including VARNA [11], jViz.RNA [12–16], Forna [17], PseudoViewer [18–22], 4SALE [23, 24], Assemble2 [25, 26], RNA2DMap [27], R2R [28], and R-Chie [29]. All Visualization software developed for RNA have as their goal to display an informative structure of the RNA molecule, usually focusing on its secondary structure, that can be annotated and used to convey information in presentations, publications, and any other two-dimensional media.

However, the majority of RNA visualization software designed produce a static layout of the RNA molecule that may not be ideal for the user. While for small RNA molecules such as transfer RNA (tRNA), this problem almost never arises, for large RNA molecules, this layout can be such that sections of the RNA overlap each other, making annotation of certain regions problematic and uninformative. There are only three notable examples that create dynamic layouts which are responsive to user interactions: jViz.RNA, PseudoViewer, and VARNA.

The designers of VARNA do not explicitly state how they construct the RNA molecule and how the algorithm responsible for user response behaves, but one can estimate how the algorithm operates from interacting with the software as a web applet [30]. The RNA structure is translated into a graph where loops make up the vertexes and stems make up the edges. Thus, by dragging the stems, the user can arrange the layout of the RNA molecule. While this allows users to fully customize the RNA layout to their needs. The high degree of user involvement might make the task seem very daunting for large RNA molecules.

PseudoViewer is another application that allows its users to manipulate the RNA structure. Originally an application designed focusing on RNA pseudoknots, PseudoViewer puts a great deal of effort into creating the initial layout of the RNA molecule. Additionally, the user can manipulate the RNA structural motifs. However, certain manipulations of the RNA conformation destabilize the system and cause the RNA model to break apart. Furthermore, since PseudoViewer's focus is primarily on pseudoknotted RNA structures, it tends to arrange the RNA structure layout focusing very heavily on clear display of pseudoknot types, at the expense of the aesthetics related to the rest of the structure.

jViz.RNA, a third software designed for dynamic RNA visualization, employs a different approach to creating the RNA model. jViz.RNA translates the RNA molecule into a graph as well, but maps each nucleotide to a vertex and each bond between nucleotides (hydrogen or covalent) to an edge. jViz.RNA then uses repulsion and attraction forces between all nucleotides to calculate the position and movement. This process continues until the forces reach an equilibrium for all nucleotides. Users can still move the individual nucleotides, and by doing so interact with the RNA structure. Mapping each nucleotide into a vertex creates a graph with a high number of vertexes and edges. As such, we denote this method as constructing a *detailed graph*.

A software similar to jViz.RNA, Forna, also uses a detailed graph representation in order to construct a dynamic RNA model. All nucleotides and chemical bonds are translated into vertexes and edges, and an automatic layout is constructed utilizing attraction and repulsion forces, as well as invisible "helper vertexes" which aid in the improvement of loop appearance. Like the previous three software tools, Forna allows for users to interact with the constructed dynamic model.

Of the four software tools mentioned, the approach employed by jViz.RNA and Forna relies mostly on the program to produce the layout, rather than require user intervention. This approach makes the production of RNA images less involved for the user, due to very little, if any, overlap even for large molecules. However, both Forna and jViz.RNA require more computation time for structure layout than PseudoViewer or VARNA. For very large structures of over 900nt, waiting for the automatic layout can be very inconvenient, and interaction with the structure becomes problematic as the structure's movement becomes delayed. In addition, current jViz.RNA output contains some inconsistencies with regards to current accepted RNA visualization guidelines. The following section discusses these shortcomings in more detail.

Motivation

While the current implementation for jViz.RNA produces dynamic RNA models that display secondary structure elements very well, the simple graph representation described presents some visual shortcomings. Most RNA images found in the literature follow several visualization norms, and two such norms that are very important are that stems are drawn such that the distance between base-pairs is consistent across the stem, and that loops are drawn as circular elements wherever they occur in the structure. Figure 1 demonstrates such limitations of jViz.RNA. While Fig. 1a contains stems with consistent base-pair distances and loops which are clearly distinguishable by their circular nature, Fig. 1b contains stems where the base pair distance varies across the stem, and

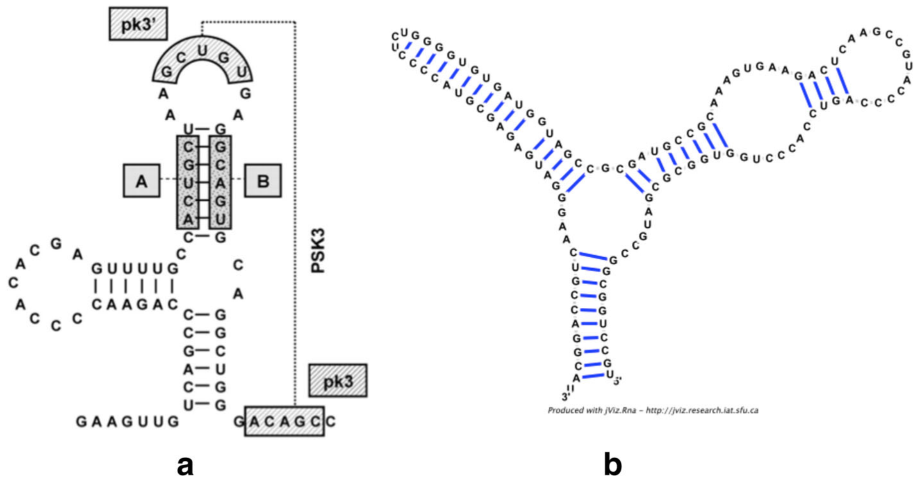

Fig. 1 The visualization differences observed in jViz.RNA compared to an RNA image which highlights RNA visualization norms in literature. **a** The Yellow Fever Virus 3′ Untranslated Region (UTR). Image taken from [39] and used with permission, (**b**) An example produced by jViz.RNA of the *S. cervisiae* 5s ribosomal RNA (rRNA) (accession X67579)

some loops that are not immediately visible (such as the loop at the end of the left stem, and the bulge loop near the beginning of the right stem substructure).

The main motivation for the work presented in this manuscript was to address the visual shortcomings presented by the current algorithm employed in jViz.RNA, as well as improvements in the run-time performance. The objectives of this work were:

- To create an RNA representation which was more consistent with existing RNA visual conventions, such as round loops and equidistant base pairs.
- To design an automatic layout algorithm which works with the new representation, while introducing noticeable speed-ups to that automatic layout algorithm, and reduce overlap of structural elements.

Both objectives address important areas for improvement in jViz.RNA. While currently the resulting RNA molecule is laid out very clearly, personal feedback from RNA researchers, as well as visual comparison with other software, demonstrates that it ignores several visualization conventions such as the shape of loops and stems. This visualization pattern makes it more difficult to share informative images and diagrams about RNA, as the resulting visualization is not in the format most RNA scientists expect.

Another challenge the current setup faces is its time complexity. Calculating the attraction force for each nucleotide n_i is in $O(1)$ since there are at most three nucleotides bonded to it, so the total calculation time for all the nucleotides' attraction forces is in $O(N)$ at each iteration (where N is the number of nucleotides in the RNA molecule). However, calculating the repulsion forces for each nucleotide is in $O(N)$ since it must account for the repulsion of all other nucleotides, so the total calculation time for all the nucleotides' repulsing forces is in $O(N^2)$. This run-time can make the automatic layout algorithm perform slowly for large RNA structures such as 16S ribosomal RNA. As such, the work presented in this manuscript was set to implement the improvements while reducing the run-time required for the RNA structures to stabilize. In order to achieve this two-pronged effect, the method by which jViz.RNA builds the graph to represent the RNA structure had to be modified; the simple graphs built were replaced by a more compressed representation of the RNA molecule, named *compressed graphs*.

The work in this manuscript is divided into three main sections. First, we discuss how the RNA molecule can be mapped into a graph with a smaller number of vertexes, a compressed graph, similar to the one used by VARNA. Secondly, we profile the performance of the original version of jViz.RNA against the performance obtained by employing the reduced graph. Third and finally, a superior method for calculating movement is presented and profiled against the original movement calculation method.

Methods

Language, implementation, and system

The code presented in this manuscript was implemented using Java 6.0. The Swing library was used for the graphical component of the code. Measurements were all taken on a PC with an Intel Core i7-4790 3.60 GHz processor, running Ubuntu 14.04.5 LTS.

Constructing and manipulating the new graph

One intuitive method of decreasing the run-time for the algorithm mentioned is to decrease the number of vertexes simulated. Since the run-time for the algorithm at every iteration is in order of $O(N^2)$, decreasing the number of vertexes would theoretically have a tremendous effect on reducing the run-time. However, the mapping of the structure into vertexes and edges must be done in a way that still produces visually pleasing layouts and RNA diagrams. Inspired by VARNA, we have employed a representation which maps each RNA loop, as well as every stem base pair, to a vertex, and connects those via edges. (Fig. 2).

Furthermore, we have implemented the system in such a way that repulsion only occurs between loops, and not base pairs. Since the repulsion step is the main time consuming step of each iteration (having run time in order of $O(N^2)$), decreasing the number of participating vertexes in the repulsion interaction should greatly reduce the run time of the algorithm. Constructing the system in such a way that only loops experience repulsion ensures that loops will be pushed away from each other, thus not intersecting each other. In theory, this should aid the structure in adopting a final layout that has minimal or no intersection of any structural elements.

At the initial step of the simulation, the RNA graphs are placed in a naive initial layout which is inspired from a circular representation of the RNA molecule (Fig. 3). Then, an iterative process begins in which the structure is slowly brought to a stable position by Newtonian inspired spring and repulsion forces.

jViz.RNA and the Newtonian model

Originally jViz.RNA mapped the RNA structure into a detailed graph, $G = \{V, E\}$. In the detailed graph representation, each nucleotide is a vertex $v \in V$, and each chemical bond corresponds to an edge $e \in E$. The entire structure is initially laid out in a circle, and an iterative process designed to move the structure into a stable layout begins. In this paper, we employ the ***compressed graph*** representation, where each structural element (loops and base pairs) is a vertex $v \in V$, and the edges are graph elements which connect the different structure elements (Fig. 2b).

For the purposes of the following sections, the notation $\vec{P}$ will be used to denote the positions of the different vertexes which represent the RNA structures, and $\vec{P}^i_{n,k}$ will be used to denote the position of vertex i (since there are many vertexes), at time-step n (since the simulation operates in discrete time-steps), during the k-th Newton iteration (since a portion of the experiments in this paper use Newton's method for converging on the position of vertex i at time-step n).

The most basic computation done at each iteration is the unit vector function, $\vec{U}(\vec{P}^i, \vec{P}^j)$. This function calculates the unit vector pointing from point $\vec{P}^j$ (the position

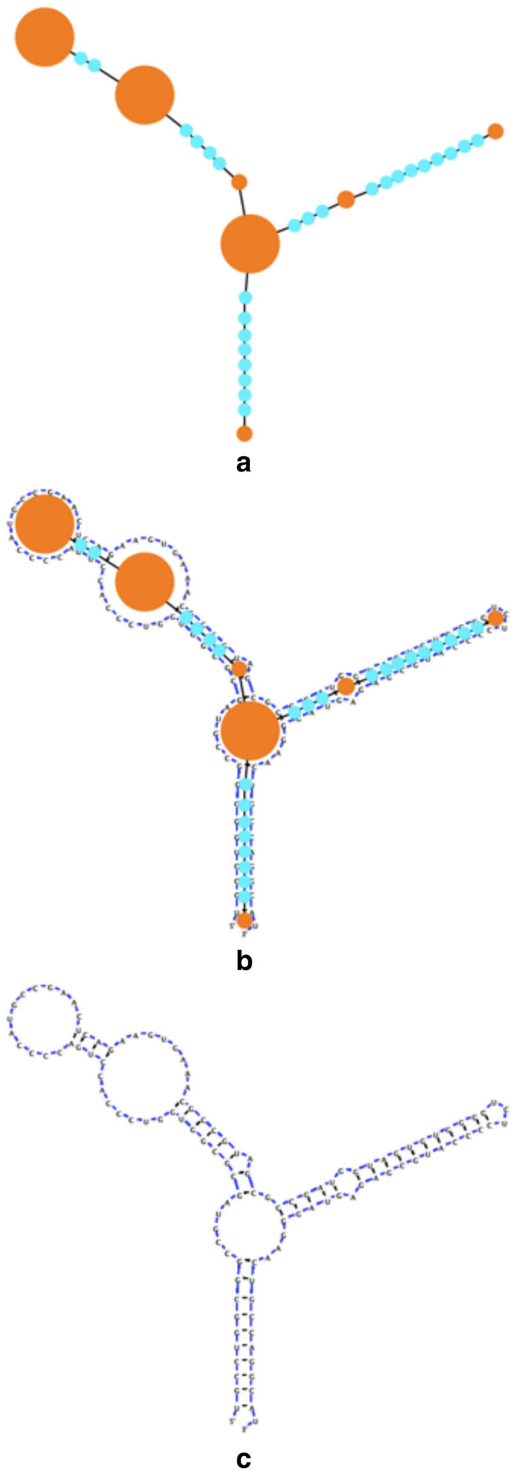

Fig. 2 The compressed graph mapped to an RNA structure. **a** The main RNA elements are compressed into vertexes where each vertex represents an RNA loop element, or a stem base pair. **b** The nucleotides belonging to each RNA element are drawn on top of the underlying RNA compressed graph. **c** The resulting RNA representation contains less vertexes than there are nucleotides (in this case 120 nucleotides versus 34 vertexes), and a more familiar visual layout

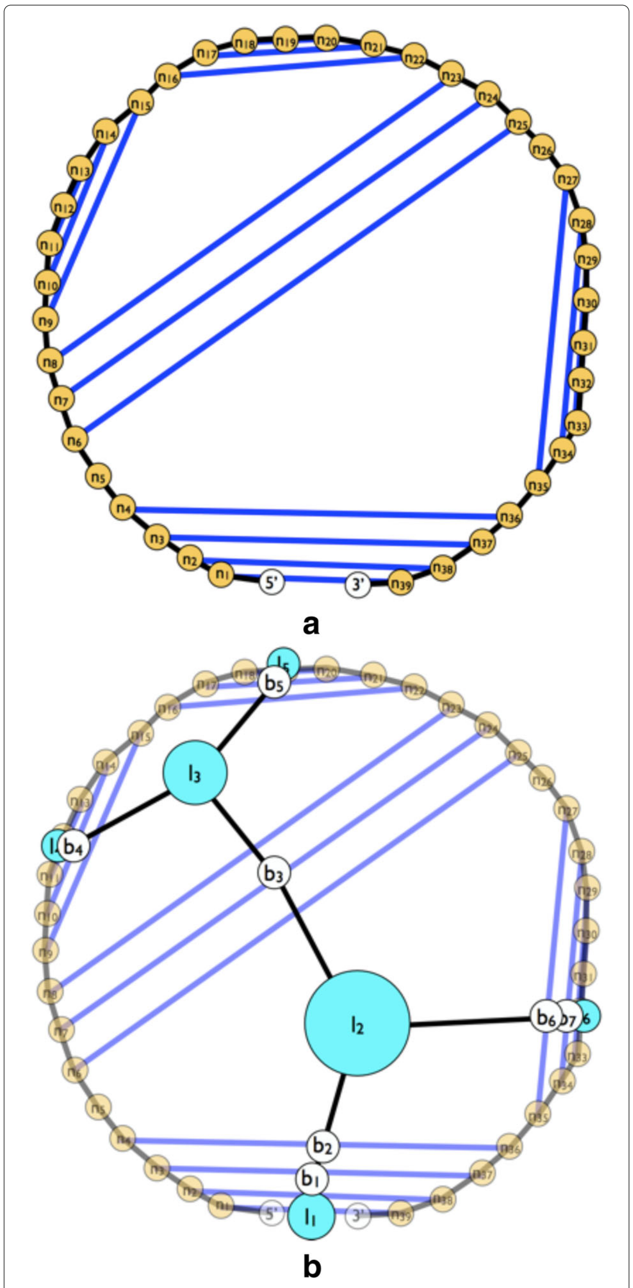

Fig. 3 The initial vertex layout process demonstrated using a sample theoretical RNA molecule. l_1, l_2, l_3, l_4, l_5, and l_6 represent loops while b_1, b_2, b_3, b_4, b_5, b_6, and b_7 represent base pairs. The loops and base pair vertexes are connected via *black* edges. **a** The RNA nucleotides are first laid out in a *circle*. **b** Each set of nucleotides has its average position calculated, and the vertex corresponding to that set is placed in that average position. Following this step, the iterative process of stabilization begins

of vertex j, v^j) to point $\vec{P}^i$ (the position of vertex i, v^i) as follows:

$$\vec{U}\left(\vec{P}^i,\vec{P}^j\right) = \frac{\vec{P}^i - \vec{P}^j}{\left|\vec{P}^i - \vec{P}^j\right|} \quad (1)$$

In each iteration, each vertex moves based on two forces: repulsion and attraction. The repulsion forces each vertex v^i experiences from vertex v^j can be described as:

$$\vec{R}\left(\vec{P}^i,\vec{P}^j\right) = \frac{G}{\left|\vec{P}^i - \vec{P}^j\right|^2} \times \vec{U}\left(\vec{P}^i,\vec{P}^j\right) \quad (2)$$

where $\vec{U}(\vec{P}^i,\vec{P}^j)$ is the unit vector function showing the direction from vertex v^j to vertex v^i, and G is a coefficient to control the size of the force experienced. The attraction forces each vertex v^i experiences from vertex v^j can be described as:

$$\vec{A}\left(\vec{P}^i,\vec{P}^j\right) = K \times \left[\vec{P}^j - \vec{P}^i + \left(r_{ideal} \times \vec{U}\left(\vec{P}^i,\vec{P}^j\right)\right)\right] \quad (3)$$

where $\vec{U}(\vec{P}^i,\vec{P}^j)$ is again the unit vector function, K is an attraction coefficient to control for the size of the force, and r_{ideal} is the ideal desired distance between the vertexes v^i and v^j.

The iterative process stops when the forces for all vertexes have reached equilibrium, or when for all vertexes $\{v^i | 1 \le i \le N\}$ the following holds:

$$\forall v^i, \left| \sum_{v^j \in L, j \neq i} \vec{R}\left(\vec{P}^i,\vec{P}^j\right) + \sum_{v^j \in C^i} \vec{A}\left(\vec{P}^i,\vec{P}^j\right) \right| \le \epsilon \quad (4)$$

Where C^i is the set of all vertexes connected to vertex v^i (in other words, $v^j \in C^i$ iff there is an edge between v^j and v^i), and L is the set of all loops (that is, $v^j \in L$ iff v^j is a vertex representing a loop).

That is to say, the iterative process stops when the sum of the forces acting on the vertexes is smaller than ϵ. Setting $\epsilon = 0$ will force the simulation to continue to calculate until the forces are perfectly at odds, but setting a small value for ϵ allows the layout algorithm to stop sooner when achieving a stable structure. ϵ as such, controls the degree of stability required before simulation of the structure's movement stops. In this work, we chose to explore two methods of implementing the physics based RNA model: The Forward Euler method, and the Backward Euler method. The two methods make it possible to evaluate the movement of the vertexes. However, the latter is more numerically stable than the former, and allows for greater time steps and faster visualizations, as well as a more stable user interaction experience.

The forward Euler method

The Euler method is a first-order integration method which belongs to a larger class called the Runge-Kutta methods (most famous being the fourth-order method [31]). The simplest version is called the Forward Euler Method [32]. This method of calculating each time step can be expressed in the following manner:

$$\begin{aligned} t_{n+1} &= t_n + \Delta t \\ \vec{P}^i_{n+1} &= \vec{P}^i_n + \Delta t f\left(t_n, \vec{P}^i_n\right) \end{aligned} \quad (5)$$

which means the time t_n is advanced by the time-step Δt and then the position of the vertex v^i is updated based on the size of the time-step, and the current behaviour of the particle, f (which is usually a function which depends on the particle's current state and/or the time).

When applied to the movement of the RNA vertexes, the Forward Euler method can be written as:

$$\begin{aligned} t_{n+1} &= t_n + \Delta t \\ f(t_n,\vec{P}^i_n) &= \textstyle\sum_{v^j\in L, j\neq i} \vec{R}\left(\vec{P}^i_n,\vec{P}^j_n\right) + \sum\limits_{v^j\in C^i} \vec{A}\left(\vec{P}^i_n,\vec{P}^j_n\right) \\ \vec{P}^i_{n+1} &= \vec{P}^i_n + \Delta t f\left(t_n,\vec{P}^i_n\right) \end{aligned} \tag{6}$$

where in this case, the behaviour of the particle with regard to its position is the sum of repulsion ($\vec{R}$) forces (over all loops $v^j \in L$, where L is the set of all loops) and attraction ($\vec{A}$) forces acting on the particle.

Since base pair vertexes do no participate in the repulsion step, the expression for a base pair vertex's Forward Euler implementation will be:

$$\begin{aligned} t_{n+1} &= t_n + \Delta t \\ f\left(t_n,\vec{P}^i_n\right) &= \textstyle\sum_{v^j\in C^i} \vec{A}\left(\vec{P}^i_n,\vec{P}^j_n\right) \\ \vec{P}^i_{n+1} &= \vec{P}^i_n + \Delta t f\left(t_n,\vec{P}^i_n\right) \end{aligned} \tag{7}$$

However, since the implementation of this expression is trivially similar to the expression in (6), the remainder of this text will focus on that expression, with the implications for the expression in (7) being omitted for brevity.

The main drawback presented by the Forward Euler method is its numerical instability. Simply put, when the time-step Δt is too long, or the coefficients which control the simulation become too large, the simulation does not stabilize into an equilibrium. In fact, it can become increasingly unstable. The solution to this drawback lies in the implementation of the Backward Euler method, which takes this instability into account.

The backward Euler method

Much like the Forward Euler method is described as explicit, there is an implicit Euler method; the Backward Euler method. Generally, it is defined as:

$$\begin{aligned} t_{n+1} &= t_n + \Delta t \\ \vec{P}^i_{n+1} &= \vec{P}^i_n + \Delta t f\left(t_n,\vec{P}^i_{n+1}\right) \end{aligned} \tag{8}$$

where $f(t_n,\vec{P}^i_{n+1})$ is again a function that describes the movement of the object. Notice it is very similar to the explicit method, but the term $\vec{P}^i_{n+1}$ appears on both sides of the equation. As a result, finding $\vec{P}^i_{n+1}$ is no longer a simple issue of updating the timestep, but it is that of solving for it algebraically.

In the case of the current simulation, the Backward Euler method would yield the following expression:

$$\begin{aligned} t_{n+1} &= t_n + \Delta t \\ f\left(t_n,\vec{P}^i_{n+1}\right) &= \textstyle\sum_{v^j\in L, j\neq i} \vec{R}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) + \sum\limits_{v^j\in C^i} \vec{A}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) \\ \vec{P}^i_{n+1} &= \vec{P}^i_n + \Delta t f\left(t_n,\vec{P}^i_{n+1}\right) \end{aligned} \tag{9}$$

which becomes a fairly difficult equation to solve for $\vec{P}^i_{n+1}$ directly. Instead, an approximation is used to solve for $\vec{P}^i_{n+1}$.

Applying Newton's method to solve the Backward Euler expression

The expression in (9) can be rearranged to produce the following equation:

$$\vec{P}^i_{n+1} = \vec{P}^i_n + \Delta t \left[\sum_{v^j\in L, j\neq i} \vec{R}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) + \sum_{v^j\in C^i} \vec{A}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) \right] \tag{10}$$

which can be rewritten as:

$$\begin{aligned} \vec{F}\left(\vec{P}^i_{n+1}\right) = \vec{0} = -\vec{P}^i_{n+1} + \vec{P}^i_n \\ + \Delta t \left[\sum_{v^j\in L, j\neq i} \vec{R}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) + \sum_{v^j\in C^i} \vec{A}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) \right] \end{aligned} \tag{11}$$

meaning the solution for $\vec{P}^i_{n+1}$ is the root of the function $\vec{F}(\vec{P}^i_{n+1})$. While it may be difficult to solve for the root directly, Newton's method offers an approach for approximating the root of the vector function $\vec{F}\left(\vec{P}^i_{n+1}\right)$ [33].

Defining the vector function's components

As outlined in [33], it is necessary to define each of the components in $\vec{F}$ individually so that their derivatives can then be found with respect to each of the variables. In the case of the RNA simulation, the function $\vec{F}$ contains only two components; f_x and f_y which are each defined as:

$$\begin{aligned} f_x\left(\vec{P}^i_{n+1}\right) &= -x^i_{n+1} + x^i_n + \Delta t \left[\textstyle\sum_{v^j\in L, j\neq i} R_x\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) \right. \\ &\qquad \left. + \textstyle\sum_{v^j\in C^i} A_x\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) \right] \\ f_y\left(\vec{P}^i_{n+1}\right) &= -y^i_{n+1} + y^i_n + \Delta t \left[\textstyle\sum_{v^j\in L, j\neq i} R_y\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) \right. \\ &\qquad \left. + \textstyle\sum_{v^j\in C^i} A_y\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) \right] \end{aligned} \tag{12}$$

This definition requires both $\vec{R}$ and $\vec{A}$ (as well as $\vec{U}$) to be defined in terms of their x and y components as:

$$\begin{aligned}
R_x\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= \frac{G}{\left(x^i_{n+1}-x^j_n\right)^2+\left(y^i_{n+1}-y^j_n\right)^2} \times U_x\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\\
R_y\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= \frac{G}{\left(x^i_{n+1}-x^j_n\right)^2+\left(y^i_{n+1}-y^j_n\right)^2} \times U_y\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\\
A_x\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= K \times \left[x^j_n - x^i_{n+1} + \left(r_{ideal} \times U_x\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right)\right]\\
A_y\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= K \times \left[y^j_n - y^i_{n+1} + \left(r_{ideal} \times U_y\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right)\right]\\
U_x\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= \frac{x^i_{n+1}-x^j_n}{\sqrt{\left(x^i_{n+1}-x^j_n\right)^2+\left(y^i_{n+1}-y^j_n\right)^2}}\\
U_y\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= \frac{y^i_{n+1}-y^j_n}{\sqrt{\left(x^i_{n+1}-x^j_n\right)^2+\left(y^i_{n+1}-y^j_n\right)^2}}
\end{aligned} \tag{13}$$

Finding the components' derivatives

In order to apply Newton's method to the RNA model, the Jacobian matrix D of the vector function $\vec{F}$ needs to be defined. In order to do so, expressions for all partial derivatives of the components in Eqs. (12) - (13) need to be defined, where each component has two partial derivatives; with respect to x^i_{n+1} and with respect to y^i_{n+1}. The derivation of each component's partial derivatives is quite long and is not the main focus of this article. Therefore, for brevity purposes, the individual derivatives are outlined in the set of Eqs. (14)-(17):

$$\begin{aligned}
\frac{\delta f_x}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1}\right) &= -1 + \Delta t\left[\sum\nolimits_{v^j\in L, j\neq i}\frac{\delta R_x}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right.\\
&\qquad \left. + \sum\nolimits_{v^j\in C^i}\frac{\delta A_x}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right]\\
\frac{\delta f_x}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1}\right) &= \Delta t\left[\sum\nolimits_{v^j\in L, j\neq i}\frac{\delta R_x}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right.\\
&\qquad \left. + \sum\nolimits_{v^j\in C^i}\frac{\delta A_x}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right]\\
\frac{\delta f_y}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1}\right) &= \Delta t\left[\sum\nolimits_{v^j\in L, j\neq i}\frac{\delta R_y}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right.\\
&\qquad \left. + \sum\nolimits_{v^j\in C^i}\frac{\delta A_y}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right]\\
\frac{\delta f_y}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1}\right) &= -1 + \Delta t\left[\sum\nolimits_{v^j\in L, j\neq i}\frac{\delta R_y}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right.\\
&\qquad \left. + \sum\nolimits_{v^j\in C^i}\frac{\delta A_y}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right]
\end{aligned} \tag{14}$$

$$\begin{aligned}
r &= \left(x^i_{n+1} - x^j_n\right)^2 + \left(y^i_{n+1} - y^j_n\right)^2\\
\frac{\delta R_x}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= \left\{\left[\frac{\delta U_x}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) \times \frac{G}{r}\right]\right.\\
&\qquad \left. - \left[\frac{2G\left(x^i_{n+1}-x^j_n\right)}{r^2} \times U_x\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right]\right\}\\
\frac{\delta R_x}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= \left\{\left[\frac{\delta U_x}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) \times \frac{G}{r}\right]\right.\\
&\qquad \left. - \left[\frac{2G\left(y^i_{n+1}-y^j_n\right)}{r^2} \times U_x\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right]\right\}\\
\frac{\delta R_y}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= \left\{\left[\frac{\delta U_y}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) \times \frac{G}{r}\right]\right.\\
&\qquad \left. - \left[\frac{2G\left(x^i_{n+1}-x^j_n\right)}{r^2} \times U_y\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right]\right\}\\
\frac{\delta R_y}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= \left\{\left[\frac{\delta U_y}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) \times \frac{G}{r}\right]\right.\\
&\qquad \left. - \left[\frac{2G\left(y^i_{n+1}-y^j_n\right)}{r^2} \times U_y\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right]\right\}
\end{aligned} \tag{15}$$

$$\begin{aligned}
\frac{\delta A_x}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= K \times \left[-1 + \left(r_{ideal} \times \frac{\delta U_x}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right)\right]\\
\frac{\delta A_x}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= K \times \left(r_{ideal} \times \frac{\delta U_x}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right)\\
\frac{\delta A_y}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= K \times \left(r_{ideal} \times \frac{\delta U_y}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right)\\
\frac{\delta A_y}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= K \times \left[-1 + \left(r_{ideal} \times \frac{\delta U_y}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right)\right)\right]
\end{aligned} \tag{16}$$

$$\begin{aligned}
r &= \sqrt{\left(x^i_{n+1} - x^j_n\right)^2 + \left(y^i_{n+1} - y^j_n\right)^2}\\
\frac{\delta U_x}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= \frac{\left(y^i_{n+1}-y^j_n\right)^2}{r^3}\\
\frac{\delta U_x}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= \frac{-\left(y^i_{n+1}-y^j_n\right)\left(x^i_{n+1}-x^j_n\right)}{r^3}\\
\frac{\delta U_y}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= \frac{-\left(y^i_{n+1}-y^j_n\right)\left(x^i_{n+1}-x^j_n\right)}{r^3}\\
\frac{\delta U_y}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1},\vec{P}^j_n\right) &= \frac{\left(x^i_{n+1}-x^j_n\right)^2}{r^3}
\end{aligned} \tag{17}$$

and the matrix D is defined as:

$$D\left(\vec{P}^i_{n+1}\right) = \begin{bmatrix} \frac{\delta f_x}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1}\right) & \frac{\delta f_x}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1}\right)\\ \frac{\delta f_y}{\delta x^i_{n+1}}\left(\vec{P}^i_{n+1}\right) & \frac{\delta f_y}{\delta y^i_{n+1}}\left(\vec{P}^i_{n+1}\right) \end{bmatrix}.$$

Constructing the Newton step

Given the function $\vec{F}$ and the matrix D, progressively better estimates for the value of $\vec{P}^i_{n+1}$ can be found by applying the following Newton step:

$$\vec{P}^i_{n+1,k+1} = \vec{P}^i_{n+1,k} - \vec{F}\left(\vec{P}^i_{n+1,k}\right) \times D^{-1}\left(\vec{P}^i_{n+1,k}\right) \quad (18)$$

where $D^{-1}\left(\vec{P}^i_{n+1,k}\right)$ is the inverse matrix of $D\left(\vec{P}^i_{n+1,k}\right)$. That is, at every Newton step $k+1$, the value of both the function $\vec{F}$ and its components' derivatives, encapsulated in the matrix D^{-1}, are evaluated at the point $\vec{P}^i_{n+1,k}$, that is, the point $\vec{P}^i_{n+1}$ from the previous Newton step. The initial estimate, $\vec{P}^i_{n+1,0}$ can be obtained by applying the Forward Euler. As more Newton steps are repeated, a better and better estimate for $\vec{P}^i_{n+1}$ emerges. However, each Newton step increases the run-time of each iteration of the algorithm. In general, each additional Newton step increases the run time of the physics based simulation by $O(L^2)$ where L is the number of loops in the simulation.

Experimental parameters and test-bed structures

For the purposes of these experiments, 17 RNA molecules were chosen from the RNA STRAND v2.0 database [34], and were run under two different configurations. The configurations and their parameters can be found in Table 1, while the structure details can be found in Table 2[1]. The structure lengths are given in "nt," which stands for "nucleotides."

Different time-steps were chosen for the different configurations (Table 1). Configuration 1 was assigned the highest time-steps it can support without losing stability. Configuration 2 can handle larger time steps, but the choice of time-step influences the choice for the number of Newton iterations (such that larger time steps required more Newton iterations to reach convergence). Therefore, a value of 3.0 was chosen to support satisfactory convergence within 5 Newton iterations.

Each structure was run 20 times and the CPU time of the run was measured until the structure stabilized (that is, until the large movement of any of its components was less than ϵ). The average run-time was calculated and plotted. If a structure's stabilization process took more than 30 mins (1800 s) it was terminated and its stabilization time was taken as 1800 s.

Table 1 The parameters for the two experimental configurations

Configuration #	1	2
Movement update	Forward Euler	Backward Euler
K	10.0	10.0
G	0.01	0.01
Time-step (Δt)	0.01	3.0
Minimal stablization movement (ϵ)	0.0001	0.3
Newton iterations	N/A	5

Table 2 The RNA structures chosen for comparison between the forward and backward Euler methods

#	RNA STRAND ID	Original ID	Length (nt)	Reference
1	NDB_00051	PDB: 1VTQ	75	[40]
2	PDB_01255	PDB: 2R8S	159	[41]
3	PDB_01076	PDB: 2GO5	217	[42]
4	PDB_00985	PDB: 2CZJ	248	[43]
5	PDB_00528	PDB: 1KOG	304	[44]
6	PDB_00398	PDB:1FCW	380	[45]
7	PDB_01144	PDB: 2J37	408	[46]
8	SRP_00288	SRPDB: Sacc.cere._M28116	522	[47]
9	RFA_00829	Rfam: RF00551	551	[48]
10	CRW_00736	CRW: a.I2.c.N.tabacum.B.ND2	696	[35]
11	CRW_00731	CRW: a.I2.c.N.tabacum.A.trnI.i1	772	[35]
12	CRW_00757	CRW: a.I2.m.Z.mays.A.OX2.i1	912	[35]
13	CRW_00533	CRW: d.233.m.C.elegans	953	[35]
14	CRW_00540	CRW: d.233.m.L.terrestris	1279	[35]
15	CRW_00539	CRW: d.233.m.L.bleekeri	1333	[35]
16	CRW_00742	CRW: a.I2.m.A.aegerita.B.LSU.2059	1857	[35]
17	CRW_00534	CRW: d.233.m.C.eugametos	1915	[35]

Improving the attraction force calculations

The system of forces described in the previous section allowed the RNA structure simulation to stabilize and present the RNA structural elements much better than the former jViz.RNA implementation (Fig. 4a). However, the resulting stable layouts were not satisfactory due to the overlap artefacts created (Fig. 4b–c). Stems would often overlap loops and would not stabilize into their correct position based on their connectivity to the loops. While a user could, in theory, address such a problem manually, we felt there is room for further improvements. In order to correct the overlap artefacts, a slight modification to the attraction force calculation was implemented.

Originally, the attraction forces would apply attraction between the centres of two vertexes (Fig. 5a–b). However, with a slight modification, each vertex can store the ideal positions for each stem protruding from it (Fig. 5c). Using these ideal positions in the equation for $\vec{A}(P^i, P^j)$ to move each vertex to its ideal position and orientation (Fig. 5d). The resulting layouts prove to be much more visually appealing and containing much less overlap, especially for smaller RNA structures (Figs. 8, 9, 10, 11, 12 and 13).

Results

Comparison of jViz.RNA's performance employing the forward and backward Euler methods

Figure 6 shows the run times of jViz.RNA when employing the Forward and Backward Euler method. As expected,

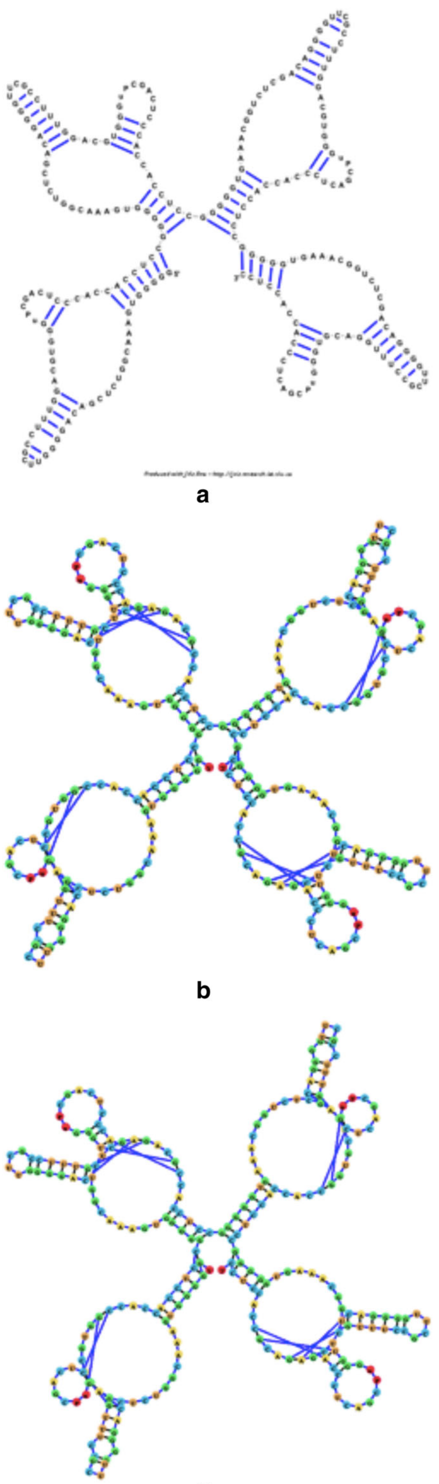

Fig. 4 The visualization result obtained for the 248 nt RNA (RNA STRAND ID PDB_00985). **a** The visualization obtained with jViz's detailed graph representation (employing the Forward Euler method). **b** The visualization obtained with jViz's compressed graph representation and the Forward Euler method. **c** The visualization obtained with jViz's compressed graph representation and the Backward Euler method

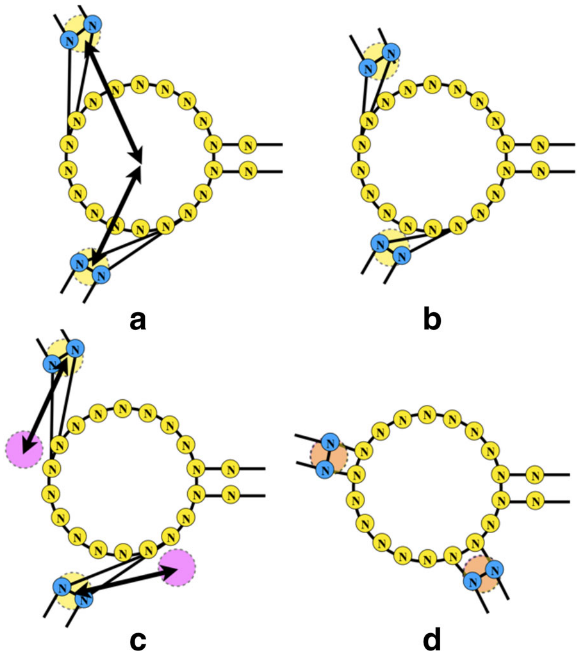

Fig. 5 Implementing the ideal position attraction forces causes the stems to align with their ideal layout. **a** Originally, attraction forces were acting between base pairs, and the loops, attracting the centres of the vertexes directly. **b** The resulting layout contained artefacts of distorted stems, since base-pairs were unaware of their positions relative to loops. **c** The idealized attraction forces employ the ideal positions (*purple circles*) of the stems to attract the base-pair vertexes. **d** The resulting layout when employing the ideal positions is aware of the position stems should take relative to their parent loops

since the Backward Euler method takes a much larger time step, the structures subject to the Backward Euler simulation converge to a stable layout much more rapidly than when subject to the Forward Euler. In fact, to truly appreciate the difference, the log_{10} of the run times was taken and plotted in Fig. 7. As can be seen, the run times of the Forward Euler method are often $\approx$ 100 times longer than the Backward Euler run times. Considering the fact that no structure was allowed longer than 1800 seconds to stabilize, it is fair to assume that under the current parameters of K and G, the difference in run time could have been even greater for some structures.

One would expect that the run-times would increase in a quadratic order to the number of nucleotides. However, while there is a general increase in run time with structure size, some small structures take longer than larger ones to stabilize. This observation points to the fact that the connectivity of the structure plays a very important role in its stabilization time. Overall, a structure X composed of 3 times as many nucleotides as structure Y would take longer to stabilize, but it may not be straightforward to deduce exactly how much longer. Even the number of vertexes in a given structure does not provide a good heuristic

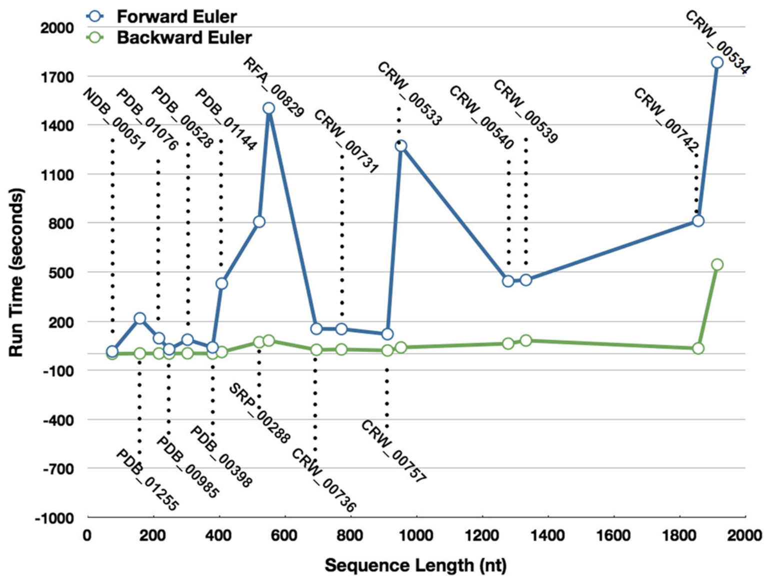

Fig. 6 The Run-times (expressed in seconds) of jViz.RNA's compressed graph representations employing both the Forward and Backward Euler methods

to calculating the difference in stabilization time for both the Forward and Backward Euler.

Despite the relative uncertainty in the relationship between a given structure's run time and its size, there is a great deal of certainty that the Backward Euler proved superior when compared to the Forward Euler. First, it can produce stable layouts employing a time step 300 times larger than the Forward Euler method without losing stability. Second, it exhibits much faster run-time performance. As demonstrated in this work, some large structures may pose a challenge to a system which takes smaller time step since the topology of the structure itself dictates how long it will take to stabilize

Visual comparison of the different algorithms

In order to get a full appreciation of the advantages of the different methods explored in this work, as well as potential future improvements, it is necessary to look at

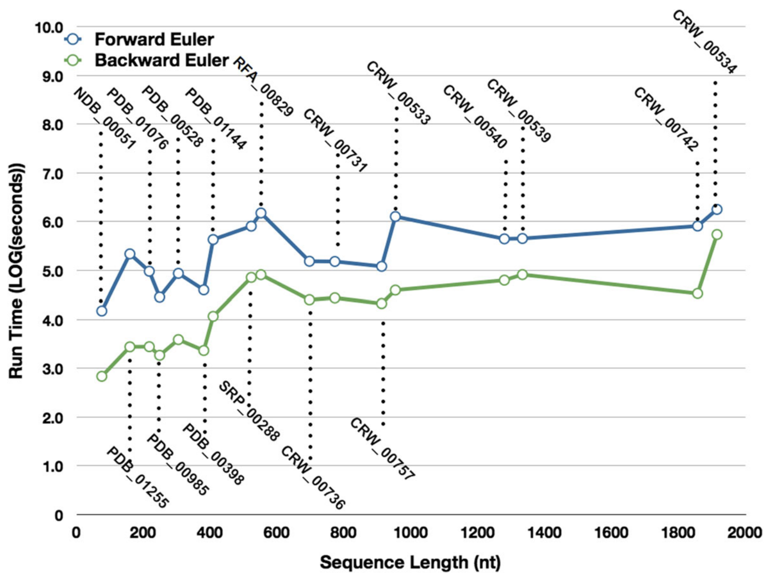

Fig. 7 The Run-times (expressed in log_{10}(seconds)) of jViz.RNA's compressed graph representations employing both the Forward and Backward Euler methods

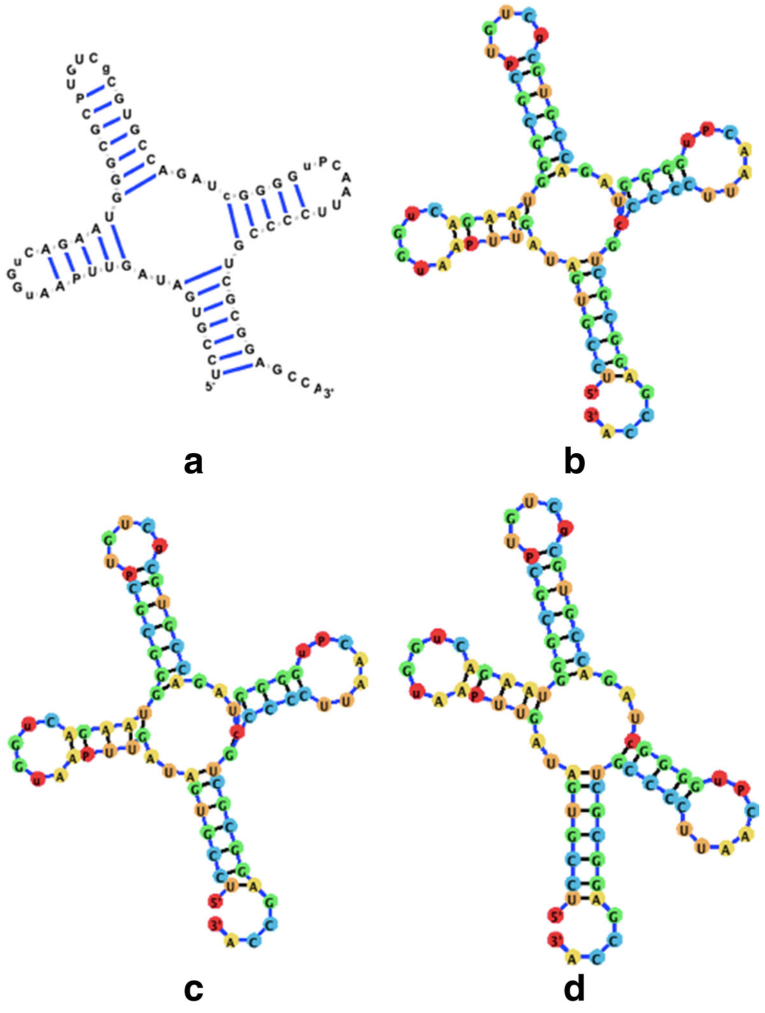

Fig. 8 The visualization result obtained for the 75 nt RNA (RNA STRAND ID NDB_00051) utilizing: **a** The detailed graph representation (employing the Forward Euler method). **b** The compressed graph representation and the Forward Euler method. **c** The compressed graph representation and the Backward Euler method. **d** The compressed graph representation and the Backward Euler method while employing ideal positions attraction forces

both the run-times reported in Figs. 6 and 7, as well as at the resulting visualizations each method produced. Figures 8, 9, 10, 11, 12 and 13 demonstrate the visualizations produced when employing the four different methods explored in this paper (all Euler implementation of the compressed graph as well as their detailed graph counter-part).

Comparing the images produced by the compressed and detailed graphs reveals additional differences between the methods. It is immediately evident that the figures produced by employing the compressed graph adhere more strictly to RNA visualization conventions; namely, the circular loops, and the constant distance between base pairs. However, at the same time, the detailed graph representation demonstrates some advantages over the compressed graph images. First, there are no cases of stems that intersect, which contributes to less user intervention being required to "untangle" the structure. The compressed graph representations, on the other hand, occasionally have stems that intersect and would require the user to explore the structure to resolve such conflicts. Though this drawback is addressed to a large extent by substituting the ideal positions as the attraction points for the vertexes, the need to manually untangle the structure may persist in certain cases.

Figure 14 demonstrates the layout algorithm as it is applied to a few instances of related RNAs. All three RNA molecules shown in Fig. 14 are tRNA molecules for different amino acids. The utilization of the ideal layout algorithm allows the related RNA molecules to be laid out in a similar conformation. Though different users may wish to align RNA structures differently, related RNA

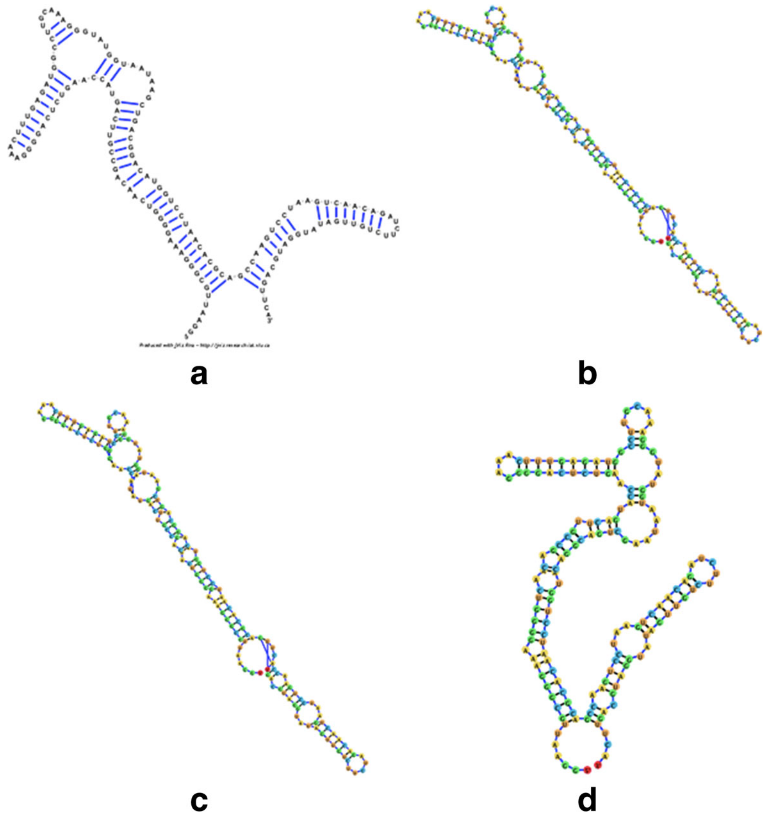

Fig. 9 The visualization result obtained for the 159 nt RNA (RNA STRAND ID PDB_01255) utilizing: **a** The detailed graph representation (employing the Forward Euler method). **b** The compressed graph representation and the Forward Euler method. **c** The compressed graph representation and the Backward Euler method. **d** The compressed graph representation and the Backward Euler method while employing ideal positions attraction forces

molecules should be drawn in a similar fashion to easily highlight homologous structural regions, which may share functional roles (such as the anticodons located on the tRNA middle stem, and the binding site for the amino acid located at the 3' end).

Discussion

Two major objectives have been set out for this work: The first was to improve jViz.RNA's visualization through a new representation, the second was to design an enhanced automatic layout algorithm in light of this new representation and to improve its run-time performance. Both objectives have been achieved through our employment of the compressed graph representation and the Backward Euler method.

Improving jViz.RNA's visualization

Comparing the detailed graphs and compressed graph visualization demonstrated that employing the compressed graphs produces visualizations more consistent with current RNA visualization methods, and does so at a fraction of the time. However, for large molecules that gain in time may be offset by the time required by the user to examine and untangle the RNA structure in the case of intersecting stems. This can be addressed by the modification of the attraction force to act as both an attraction and rotation force. However, the existence of some overlap may still be present for certain structures (such as those seen in Fig. 11d). Future work to address this limitation would focus on configuring the system to find an equilibrium between the correct positioning of stems and repulsion between the various RNA components to prevent them from overlaying each other.

Extending the automatic layout algorithm

Comparing the two methods of calculating movement for the RNA components of the compressed graph revealed that employing the Backward Euler method produces more stable simulations of the vertexes' movement, and does so in less time than the Forward Euler method (Figs. 6 and 7). Though it is evident the connectivity of the structure determines the degree of run-time improvement, it is evident that all structures we have tested stabilize faster using the Backward Euler method. Given that the Backward Euler implementation is much

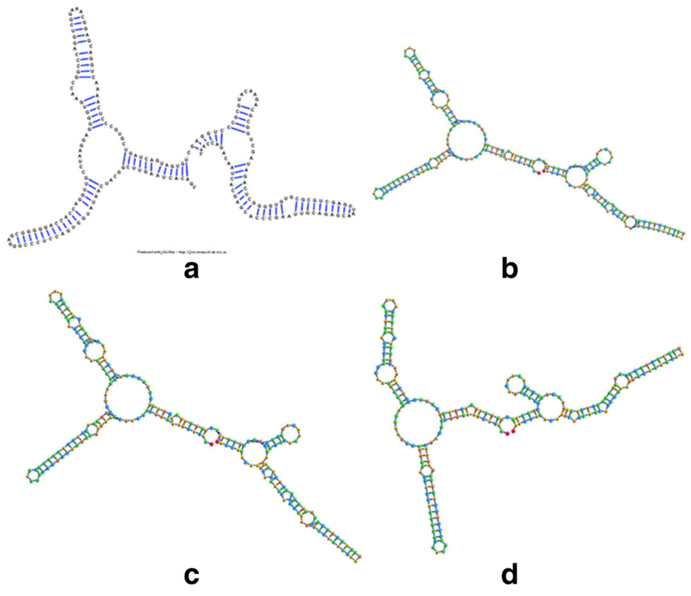

Fig. 10 The visualization result obtained for the 217 nt RNA (RNA STRAND ID PDB_01076) utilizing: **a** The detailed graph representation (employing the Forward Euler method. **b** The compressed graph representation and the Forward Euler method. **c** The compressed graph representation and the Backward Euler method. **d** The compressed graph representation and the Backward Euler method while employing ideal positions attraction forces

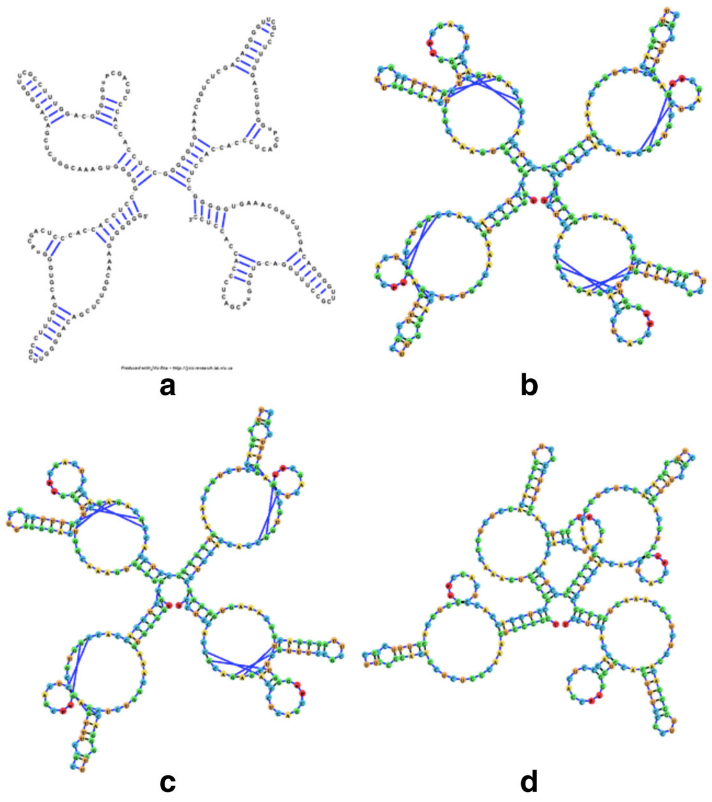

Fig. 11 The visualization result obtained for the 248 nt RNA (RNA STRAND ID PDB_00985) utilizing: **a** The detailed graph representation (employing the Forward Euler method). **b** The compressed graph representation and the Forward Euler method. **c** The compressed graph representation and the Backward Euler method. **d** The compressed graph representation and the Backward Euler method while employing ideal positions attraction forces

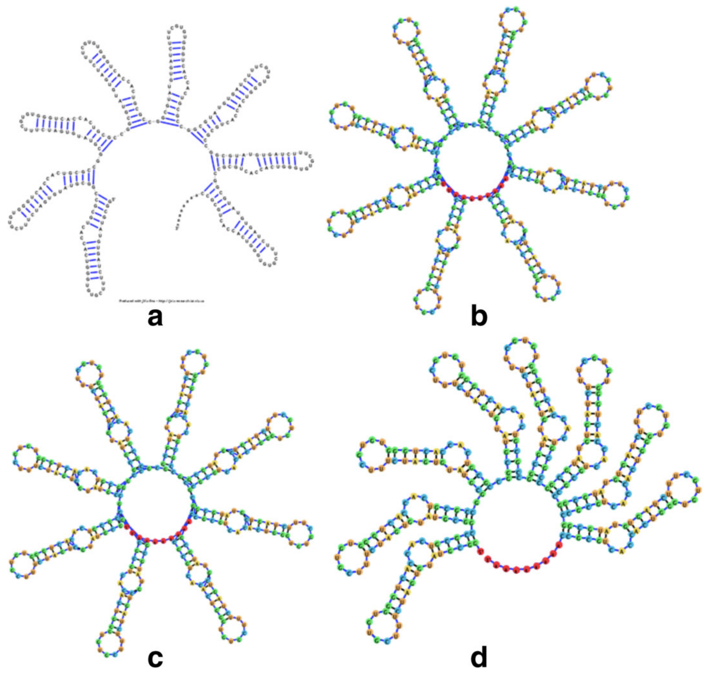

Fig. 12 The visualization result obtained for the 304 nt RNA (RNA STRAND ID PDB_00528) utilizing: **a** The detailed graph representation (employing the Forward Euler method). **b** The compressed graph representation and the Forward Euler method. **c** The compressed graph representation and the Backward Euler method. **d** The compressed graph representation and the Backward Euler method while employing ideal positions attraction forces

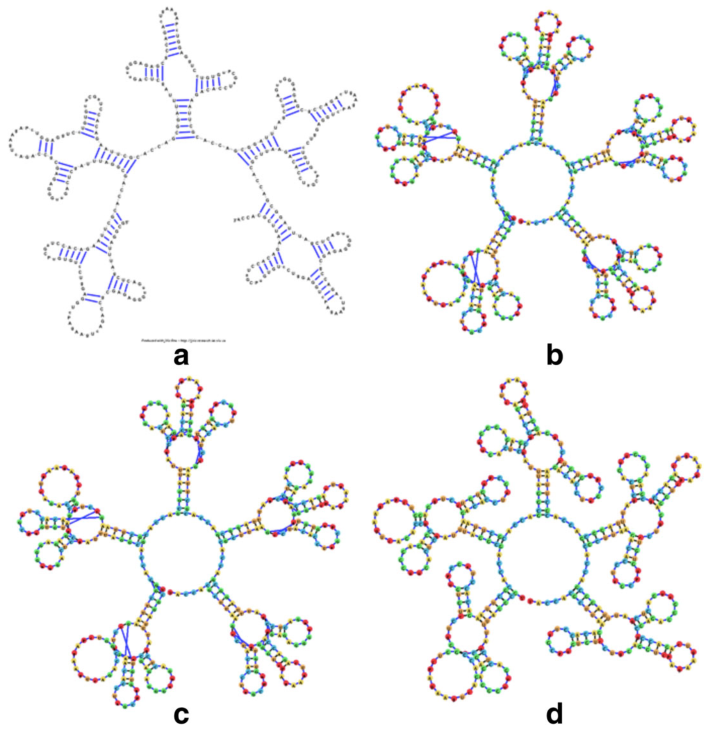

Fig. 13 The visualization result obtained for the 380 nt RNA (RNA STRAND ID PDB_00398) utilizing: **a** The detailed graph representation (employing the Forward Euler method). **b** The compressed graph representation and the Forward Euler method. **c** The compressed graph representation and the Backward Euler method. **d** The compressed graph representation and the Backward Euler method while employing ideal positions attraction forces

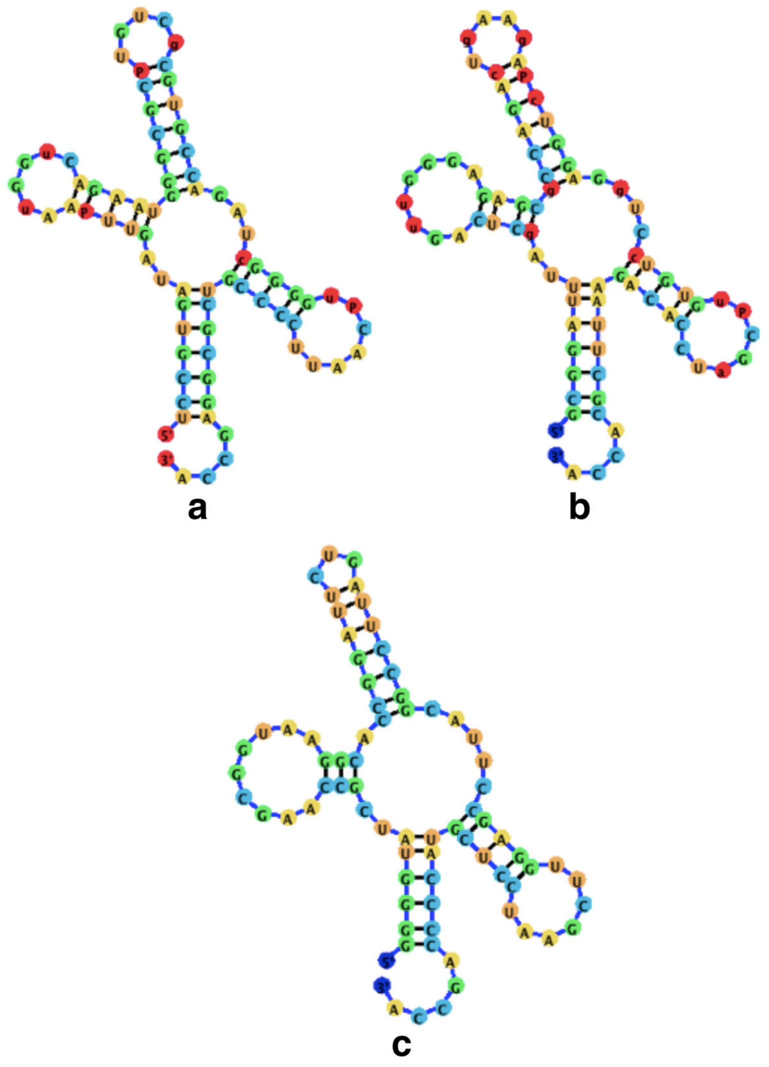

Fig. 14 A visualization comparison of tRNA molecules employing the Backward Euler method and the ideal positions attraction forces. **a** The visualization for a Yeast tRNA utilizing jViz.RNA (RNA STRAND ID: NDB_00051, PDB ID: 1VTQ). **b** The visualization for a Yeast tRNA utilizing jViz.RNA (RNA STRAND ID: PDB_00045, PDB ID: 1EHZ). **c** The visualization for an E.coli tRNA utilizing jViz.RNA (RNA STRAND ID: PDB_00426, NDB ID: 1GTS)

more stable than its Forward counter-part, it is the Backward Euler method that becomes the more desirable choice.

Conclusion

In this article we described advances made in the representation and layout algorithms for dynamic RNA secondary structure visualization. We reviewed existing tools and algorithms and showed that only few allow for dynamic visualization. One such tool is jViz.RNA. We discussed its shortcomings in terms of layout, stability and run-time performance and proposed several improvements based on a compressed graph representation and advanced numerical integration methods. We presented two graph based representations for RNA visualization, as well as two methods to create dynamic RNA structures that lay themselves out automatically and respond to user interaction.

The utilization of compressed graphs as a model of the RNA structure, the experiments shown here profiling their performance, and examining the underlying physics, demonstrate a substantial improvement to the original representation and layout algorithms employed in jViz.RNA. The work on layout conventions was greatly influenced by feedback from life science collaborators. The new algorithms have increased stability for automatic layout, reduced overlap of structural elements that decreases the need for user intervention, and increased run time performance to allow for the handling of larger RNA structures. Having examined the basic properties of compressed graph behaviour, we can verify it is a better tool to produce dynamic and responsive RNA models than its detailed counter-part. We have also discovered a few important areas of improvement such as stem orientation relative to loops, and prevention of structural element intersection, and provided improvements to the layout algorithm to handle these issues. By comparing the Forward and Backward Euler methods we demonstrated the superiority of the Backward Euler method in its ability to support larger time steps without losing stability, and consequently, allowing for faster simulations of the RNA compressed graph.

Our future work will focus on incorporating pseudoknot representation into the compressed graph model, to allow for the visualization of pseudoknotted structures. Additionally, constructing and modifying RNA structures will also be explored.

Overall, this manuscript has explored the use of compressed graphs to improve the layout of RNA secondary structures, as well as the best physics based simulation method for such an implementation. The results presented demonstrate that both the compressed graph representation, as well as the Backward Euler integrator, greatly enhance the run-time performance and usability.

We anticipate these findings to benefit other researchers in RNA structure visualization, or more generally, biological structure visualization, as the underlying ideas are transferrable. We also provide a tool that is platform independent, easy to use, and can quickly render publication quality structure images. We would anticipate that many researchers in RNA structure will find this tool useful and that it will find wide acceptance. We believe the technical details discussed in this manuscript will impact how other visualization researchers think about dynamic structure visualization and we see the impact of this work in both providing a useful software tool as well as presenting a methodology for other visualization tools to adopt.

Endnote

[1] The original IDs of the structures were obtained from the Comparative RNA Website (CRW) [35], the Protein Data Bank (PDB) [36], the Signal Recognition Particle Database (SRPDB) [37], and the RNA Families Database (Rfam) [38].

Additional file

Additional file 1: The file jViz3.0_Complete is a zip file containing a Java executable file (jViz3.0.jar), a User Manual in PDF format (jViz3.0 User Manual), and a subfolder containing RNA secondary structure files in .ct format for the RNA structures visualized in this manuscript (RNA_Structures). (ZIP 843 kb)

Abbreviations

CRW: Comparative RNA Website; DNA: Deoxyribo-nucleic Acid; nt: nucleotide; PDB: Protein Data Bank; Rfam: RNA Families Database; RNA: Ribo-nucleic Acid; rRNA: Ribosomal RNA; SRPDB: Signal Recognition Particle Database; tRNA: Transfer RNA; UTR: Untranslated Region

Acknowledgements

The authors would like to acknowledge feedback on the visualization aspects of the software received from Peter Unrau, Professor of Molecular Biology and Biochemistry, Simon Fraser University, his lab members, and the many international users of earlier versions of the software who have provided their feedback.

Funding

We would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for their support under grant number 611362. The first author would also like to thank NSERC for providing funding in the form of a CGS-D scholarship.

Availability of data and materials

The .ct files for the RNA secondary structures analysed in this study are available in the RNA STRAND v2.0 repository, http://www.rnasoft.ca/strand/. The Java executable, a User Manual, and the structures visualized in this article are also available in Additional file 1 as a zip file.

Authors' contributions

Both authors have contributed equally to the writing of this manuscript. Both authors read and approved the final manuscript.

About the Authors

Boris Shabash received a Bachelor of Health Sciences (BHSc), specializing in bioinformatics, at the University of Calgary (2009), a Master of Science (MSc), specializing in Computing Science, at Simon Fraser University (2011), and is currently working on his PhD at Simon Fraser University. His areas of expertise lie in Evolutionary Computation techniques for solving bioinformatics based problems, as well as bioinformatics related visualization. He has also extensive experience working on synthetic biology based applications, both in sillico and in vitro.
Kay C. Wiese is an Associate Professor in the School of Computing Science at Simon Fraser University, BC, Canada, where he is the Director of the Bioinformatics Research Lab. His research interests are in computational intelligence and bioinformatics, particularly algorithms for RNA secondary structure prediction and visualization. Dr. Wiese has served as a member of the organizing committee of Research in Computational Molecular Biology (RECOMB) and the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB) in various roles including Program Chair. Dr. Wiese was the Chair of the IEEE Computational Intelligence Society Bioinformatics and Bioengineering Technical Committee for 2006-2007 and served as the Vice President for Technical Activities of the IEEE Computational Intelligence Society during 2008-2009. He also serves on the steering committee of 3 IEEE journals and on the editorial board of 2 international journals in the area of Computational Intelligence and Bioinformatics.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Publisher's Note

Received: 9 February 2017 Accepted: 11 May 2017
Published online: 30 May 2017

References

1. Rietveld K, Poelgeest RV, Pleij A, Boom JHV, Bosch L. The tRNA-like structure at the 3′ terminus of turnip yellow mosaic virus RNA. Differences and similarities with canonical tRNA. Nucleic Acids Res. 1982;10(6): 1929–46.
2. Joshi RL, Joshi S, Chapeville F, Haenni AL. tRNA-like structures of plant viral RNAs: conformational requirements for adenylation and aminoacylation. EMBO J. 1983;2(7):1123–27.
3. Rietveld K, Pleij CWA, Bosch L. Three-dimensional models of the tRNA-like 3′ termini of some plant viral RNAs. EMBO J. 1983;2(7):1079–85.
4. Rao ALN, Dreher TW, Marsh LE, Hall TC. Telomeric function of the tRNA-like structure of brome mosaic virus RNA. In: Proceedings of the National Academy of Science of the United States of America, vol. 86. Washington: National Academy of Sciences; 1989. p. 5335–39.
5. Matsuda D, Dreher TW. The tRNA-like structure of Turnip yellow mosaic virus RNA is a 3′-translational enhancer. Virology. 2003;321:36–46.
6. Powers T, Noller HF. A functional pseudoknot in 16S ribosomal RNA. EMBO J. 1991;10(8):2203–14.
7. Finken M, Kirschner P, Meier A, Wrede A, Bottger EC. Molecular basis of streptomycin resistance in *Mycobacterium tuberculosis*: alterations of the ribosomal protein S12 gene and point mutations within a functional 16S ribosomal RNA pseudoknot. Mol Microbiol. 1993;9(6):1239–46.
8. Chaloin L, Lehmann MJ, Sczakiel G, Restle T. Endogenous expression of a high-affinity pseudoknot rna aptamer suppresses replication of hiv-1. Nucleic Acids Res. 2002;30(18):4001–008. doi:10.1093/nar/gkf522. http://nar.oxfordjournals.org/content/30/18/4001.full.pdf+html.
9. Shabash B, Wiese KC. RNA Visualization: Relevance and the Current State-of-the-art Focusing on Pseudoknots. IEEE/ACM Trans Comput Biol Bioinforma. 2016;PP(99):1–1. doi:10.1109/TCBB.2016.2522421.
10. Ponty Y, Leclerc F. Drawing and editing the secondary structure(s) of RNA In: Picardi E, editor. RNA Bioinformatics. New York: Springer; 2015. p. 63–100.
11. Darty K, Denise A, Ponty Y. VARNA: Interactive drawing and editing of the RNA secondary structure. Bioinformatics. 2009;25(15):1974–5.
12. Wiese KC, Glen E, Vasudevan A. jViz.RNA - A Java Tool for RNA Secondary Structure Visualization. IEEE Trans Nanobioscience. 2005;4(3): 212–8.
13. Wiese KC, Glen E. jViz.RNA - An Interactive Graphical Tool for Visualizing RNA Secondary Structures Including Pseudoknots. In: The 19th Symposium on Computer-Based Medical Systems. New York: Institute of Electrical and Electronics Engineers; 2006.
14. Glen E. JVIZ.RNA - A TOOL FOR VISUAL COMPARISON AND ANALYSIS OF RNA SECONDARY STRUCTURES Master's thesis, Simon Fraser University. 2007.
15. Shabash B. Improving the Portability and Performance of jViz.RNA - A Dynamic RNA Visualization Software. Master's thesis, Simon Fraser University. 2009.
16. Shabash B, Wiese KC, Glen E. Improving the Portability and Performance of jViz.RNA - A Dynamic RNA Visualization Software In: Shibuya T, Kashima H, Sese J, Ahmad S, editors. Pattern Recognition in Bioinformatics. Lecture Notes in Computer Science. Cham: Springer; 2012. p. 82–93.
17. Kerpedjiev P, Hammer S, Hofacker IL. Forna (force-directed RNA): Simple and effective online RNA secondary structure diagrams. Bioinformatics. 2015;31(20):3377. doi:10.1093/bioinformatics/btv372.
18. Kim W, Lee Y, Han K. Visualization of RNA Pseudoknot Structure In: Palma JLM, Sousa AA, Dongarra J, Hernández V, editors. High Performance Computing for Computational Science — VECPAR 2002. Lecture Notes in Computer Science. Berlin: Springer; 2003. p. 181–94.
19. Han K, Byun Y. PseudoViewer2: visualization of RNA pseudoknots of any type. Nucleic Acids Res. 2003;31(13):3432–40.
20. Byun Y, Han K. PseudoViewer3: generating planar drawings of large scale RNA structures with pseudoknots. Bioinformatics. 2009;25(11):1435–7.
21. Byun YA, Han KS. Visualization method of RNA Pseudoknot structures. Google Patents. EP Patent App. EP20,040,012,477. 2004. http://www.google.com/patents/EP1482439A2?cl=en. Accessed 9 Feb 2017.
22. Byun Y, Han K. PseudoViewer: web application and web service for visualizing RNA pseudoknots and secondary structure. Nucleic Acids Res. 2006;34:416–22.
23. Seibel PN, Müller T, Dandekar T, Schultz J, Wolf M. 4SALE - A tool for synchronous RNA sequences and secondary structure alignment and editing. BMC Bioinforma. 2006;7(498):498–504.
24. Seibel PN, Müller T, Dandekar T, Wolf M. Synchronous visual analysis and editing of RNA sequences and secondary structure alignments using 4SALE. BMC Res Notes. 2008;1(91):91–7.
25. Jossinet F, Westhof E. Sequence to Structure (S2S): display, manipulate and interconnect RNA data from sequence to structure. Bioinformatics. 2005;21(15):3320–21.
26. Jossinet F, Westhof E. S2S-Assemble2: a Semi-Automatic Bioinformatics Framework to Study and Model RNA 3D Architectures. Hoboken: Wiley-VCH Verlog GmbH & Co. KGaA; 2014, pp. 667–86.
27. Xu W, Wongsa A, ad Lei Shang JL, Cannone JJ, Gutell RR. RNA2DMap: A Visual Exploration Tool of the Information in RNA's Higher-Order Structure. In: Proceedings of the 2011 IEEE International Conference on Bioinformatics and Biomedicine. New York: Institute of Electrical and Electronics Engineers; 2011. p. 613–7.
28. Weinberg Z, Breaker RR. R2R - software to speed the depiction of aesthetic consensus RNA secondary structures. BMC Bioinforma. 2011;12(1):3–11.
29. Lai D, Proctor JR, Zhu JYA, Meyer IM. R-chie: a web server and R package for visualizing RNA secondary structures. Nucleic Acids Res. 2012; 40(12):95.
30. Ponty Y. VARNA: Visualization Applet for RNA. http://varna.lri.fr/index.php?lang=en&page=demo&css=varna. Accessed 9 Feb 2017.
31. Kutta W. Beitrag zur näherungweisen integration totaler differentialgleichungen. 1901.
32. Weisstein EW. Euler Forward Method. http://mathworld.wolfram.com/EulerForwardMethod.html. Accessed 9 Feb 2017.
33. University O. Lecture 13 Nonlinear Systems - Newton's Method. https://www.math.ohiou.edu/courses/math3600/lecture13.pdf. Accessed 9 Feb 2017.
34. Andronescu M, Bereg V, Hoos H, Condon A. RNA STRAND: The RNA Secondary Structure and Statistical Analysis Database. BMC Bioinforma. 2008;9(1):340. doi:10.1186/1471-2105-9-340.
35. Cannone JJ, Subramanian S, Schnare MN, Collett JR, D'Souza LM, Du Y, Feng B, Lin N, Madabusi LV, Müller KM, Pande N, Shang Z, Yu N, Gutell RR. The Comparative RNA Web (CRW) Site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. BMC Bioinforma. 2002;3(1):1–31. doi:10.1186/1471-2105-3-2.
36. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IM, Bourne PE. The protein data bank. Nucleic Acids Res. 2000;28:235–42.
37. Zweib C. SRPDB (Signal Recognition Particle Database). 2017. http://rth.dk/resources/rnp/SRPDB/SRPDB.html. Accessed 9 Feb 2017.
38. Nawrocki EP, Burge SW, Bateman A, Daub J, Eberhardt RY, Eddy SR, Floden EW, Gardner PP, Jones TA, Tate J, Finn RD. Rfam 12.0: updates to the RNA families database. Nucleic Acids Res. 2015;43(D1):130. doi:10.1093/nar/gku1063.
39. Silva PAGC, Pereira CF, Dalebout TJ, Spaan WJM, Bredenbeek PJ. An RNA Pseudoknot Is Required for Production of Yellow Fever Virus Subgenomic RNA by the Host Nuclease XRN1. J Virol. 2010;84(21):11395–406.
40. Comarmond MB, Giege R, Thierry JC, Morast D. Three-dimensional structure of yeast tRNA^{Asp}.I,structure determination. Acta Crystallogr. 1986;B(42):272–80.
41. Ye JD, Tereshko V, Frederiksen JK, Koide A, Fellouse FA, Sidhu SS, Koide S, Kossiakoff AA, Piccirilli JA. Synthetic antibodies for specific recognition and crystallization of structured RNA. Proc Natl Acad Sci. 2008;105(1):82–7.
42. Halic M, Gartmann M, Schlenker O, Mielke T, Pool MR, Sinning I, Beckmann R. Signal recognition particle receptor exposes the ribosomal translocon binding site. Science. 2006;312(5774):745–7. doi:10.1126/science.1124864. http://science.sciencemag.org/content/312/5774/745.full.pdf.
43. Bessho Y, Shibata R, Sekine S-i, Murayama K, Higashijima K, Hori-Takemoto C, Shirouzu M, Kuramitsu S, Yokoyama S. Structural basis for functional mimicry of long-variable-arm tRNA by transfer-messenger RNA. Proc Natl Acad Sci. 2007;104(20):8293–98. doi:10.1073/pnas.0700402104. http://www.pnas.org/content/104/20/8293.full.pdf.
44. Torres-Larios A, Anne-Catherine, Dock-Bregeon, Romby P, Rees B, Sankaranarayanan R, Caillet J, Springer M, Ehresmann C, Ehresmann B, Moras D. Structural basis of translational control by Escherichia coli threonyl tRNA synthetase. Nat Struct Biol. 2002;9:343–7.

45. Agrawal RK, Spahn CMT, Penczek P, Grassucci RA, Nierhaus KH, Frank J. Visualization of tRNA movements on the escherichia coli 70s ribosome during the elongation cycle. J Cell Biol. 2000;150(3):447–60.
46. Halic M, Blau M, Becker T, Mielke T, Pool MR, Wild K, Sinning I, Beckmann R. Following the signal sequence from ribosomal tunnel exit to signal recognition particle. Nature. 2006;444(7118):507–11.
47. Dallas A, Moore PB. The loop E–loop D region of Escherichia coli 5S rRNA: the solution structure reveals an unusual loop that may be important for binding ribosomal proteins. Structure. 1997;5(12):1639–53. doi:10.1016/S0969-2126(97)00311-0.
48. Griffiths-Jones S, Moxon S, Marshall M, Khanna A, Eddy SR, Bateman A. Rfam: annotating non-coding RNAs in complete genomes. Nucleic Acids Res. 2005;33(suppl 1):121–4. doi:10.1093/nar/gki081. http://nar.oxfordjournals.org/content/33/suppl_1/D121.full.pdf+html.