Circular Tree Drawing by Simulating Network Synchronisation Dynamics and Scaling

Farshad Ghassemi Toosi and Nikola S. Nikolov
Department of Computer Science and Information Systems, University of Limerick, Ireland
{farshad.toosi,nikola.nikolov}@ul.ie

Summary

We present an algorithm which produces circular-shape layouts of rooted trees by simulating synchronisation dynamics on the network represented by the tree. Our approach consists of evolving scalar dynamical values at the nodes of the tree in a number of iterations and utilising dissimilarities between pairs of dynamical values in order to compute the tree layout at each iteration. Dissimilarities are employed for computing forces of attraction and repulsion between nodes and subsequently for finding the *x*- and *y*-coordinates of all nodes. We stop this process when all dissimilarities reach a given lower bound which we scale up at each iteration.

Algorithm

Input: a rooted tree *T(V, E)* with a node set *V* and an edge set *E*; a scalar value $L > 0$. Let the root v_r be a node with the highest betweenness centrality. (*L* is a very small value less than 1 at the begging which we scale up at each iteration).

Step 1. Initialisation.

Step 1.1. $t \leftarrow 0$. Assign a random scalar variable $w(v_i,t) \in [0, 2\pi)$ to each node $v_i \in V$. These are the *dynamical variables* at the nodes.

Step 1.2. $w(v_r,0) \leftarrow 0$.

Step 1.3. Assign random *x*- and *y*-coordinates to all nodes.

Step 2. Network Synchronisation.

Step 2.1. $t \leftarrow t+1$. Update the dynamical values for all non-root nodes, i.e. calculate $w(v_i, t)$ for each node $v_i \neq v_r$ according to a modified version of the **Kuramoto model** of network synchronisation [1, 4]. **N.B.** All dynamical values remain in the interval $[0, 2\pi)$. The term $\emptyset(v_i, t)$ is an adjustment which allows larger drawing space for subtrees rooted at nodes with a relatively high degree.

$$\frac{dw(v_i,t)}{dt} = sin(w(p(v_i),t) - w(v_i,t) + \phi(v_i,t))$$

Step. 2.2. Compute the dissimilarities $dis_{ij}(t)$ between the dynamical variables for each pair of nodes v_i and v_j as follows:

$$dis_{ij}(t) = max\{L, (\cos(w(v_i,t)) - \cos(w(v_j,t)))^2 + (\sin(w(v_i,t)) - \sin(w(v_j,t)))^2)^{1/2}\}$$

Step 2.3. Check if any dissimilarity (computed at Step 2.2) is less than the lower bound *L*. If so, set it equal to *L*.

Step 2.4. Use the dissimilarities computed at Step 2.2 to update the *x*- and *y*-coordinates of the nodes in force-directed manner. Force of attraction between each pair of adjacent nodes v_i and v_j:

$$q \times \sqrt[p]{(dis_{ij}(t))}$$

Force of repulsion between each pair of non-adjacent nodes v_i and v_j : $\quad -1 \times \sqrt{(dis_{ij}(t))}$

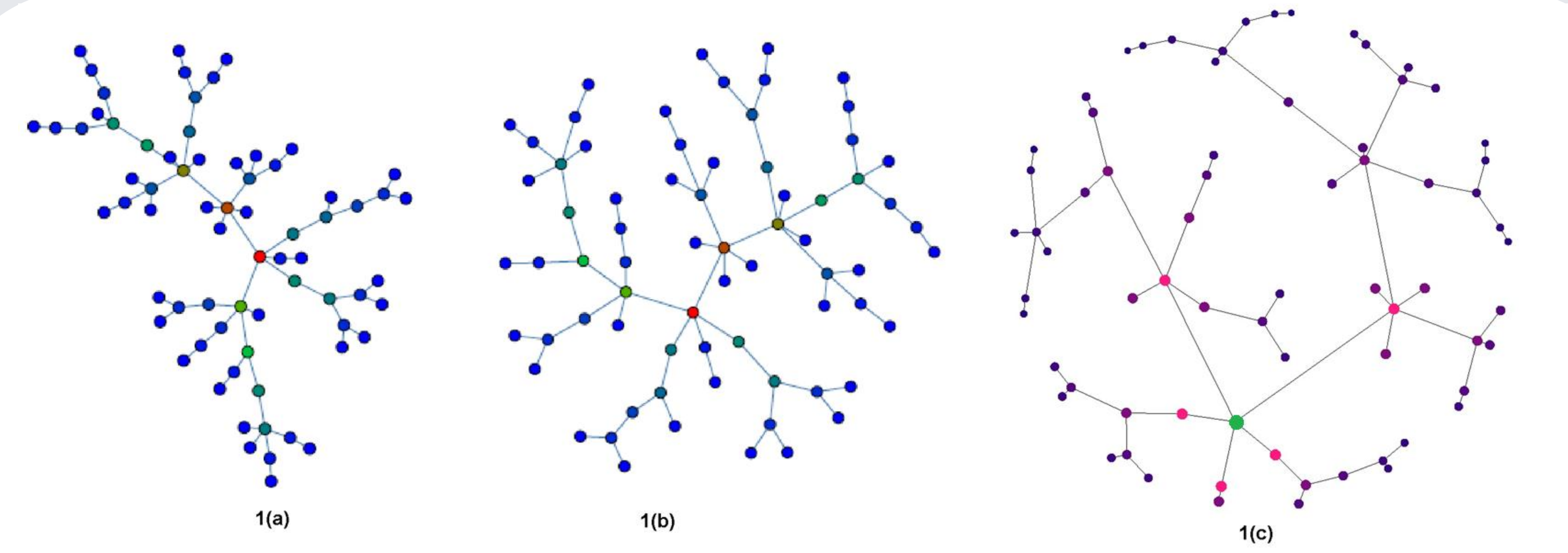
The parameters $\mathbf{0 < p \leq 1}$ and $\mathbf{q \geq 1}$ grow with the size of the tree;

Step 2.5. If all dissimilarities are equal to the lower bound *L* then stop. Otherwise, scale up *L* and go back to Step 2.1.

Discussion

Our layouts have either no or very few edge crossings. The simulation of network synchronisation dynamics leads nodes that belong to the same tree branch to be placed close to each other. However, complete synchronisation would tend to equalise all dynamic variables and thus cause large number of edge crossings and node cluttering [3]. To prevent this effect we use a lower bound *L* on the dissimilarities between dynamic variables at the nodes (Step 2.3) and we gradually scale up *L* at each iteration. At the very first step *L* is a very small number which allows nodes to *find* their relative positions in the layout. As time goes, *L* grows which helps to reduce edge crossings and node cluttering.

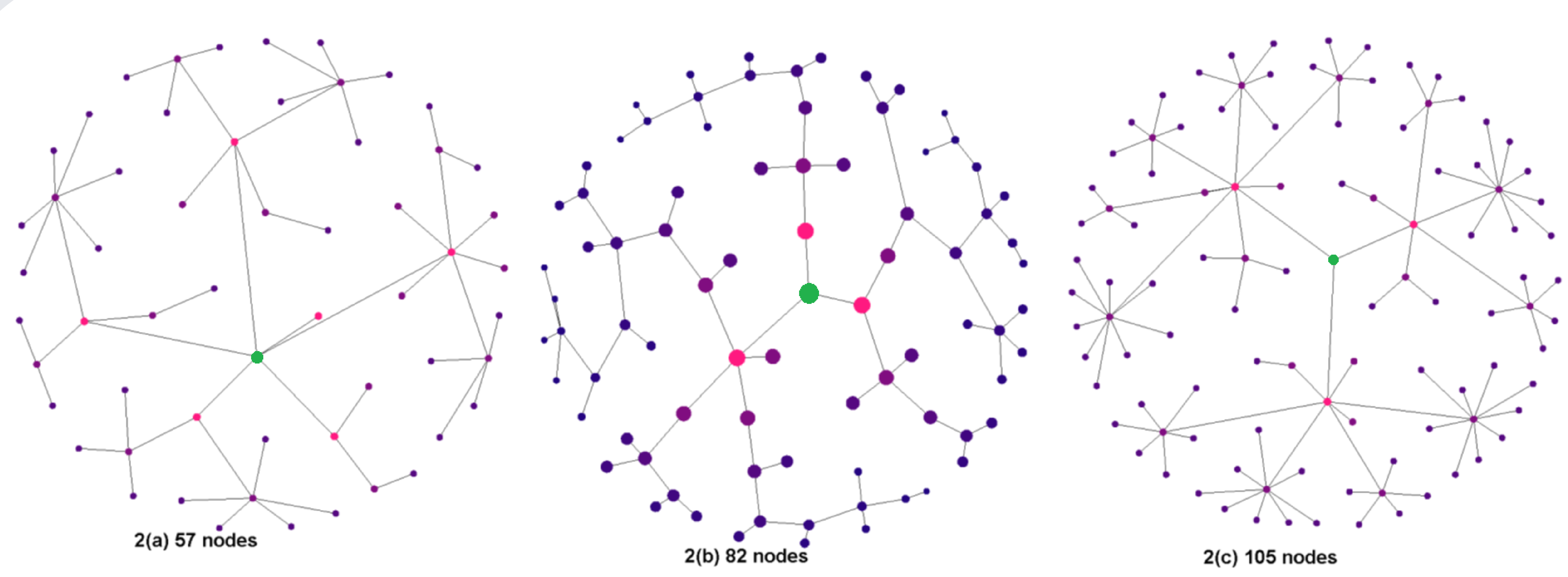
Fig. 1 shows three alternative layouts of the same tree: (a) a classical force-directed layout; (b) a gravitational force-directed embedding [2]; and (c) a layout produced by our algorithm.



(a) (b) (c)

Figure 1: Three different layouts of the same tree with 70 nodes: classical force-directed (a), gravitational force-directed (b) and a layout by our algorithm (c).

In Fig. 2 we show the layouts of another three *trees* by our algorithm.



(a) (b) (c)

Figure 2: Layouts of three different trees by our algorithm; (a) 57 nodes, (b) 82 nodes, and (c) 105 nodes.

Conclusion

The algorithm we propose produces layouts with an exact circular shape. This is both aesthetically pleasing and space efficient compared to the classical force-directed approach and to the result of applying a gravity force as demonstrated in Fig. 1. Next step in our research is to extend our algorithm to any undirected graph.

References

[1] Alex Arenas, Albert Daz-Guilera, and Conrad J. Prez-Vicente. *Synchronization reveals topological scales in complex networks*. Phys. Rev. Lett., 96(11):114102-1 – 114102-4, 2006.

[2] Michael J. Bannister, David Eppstein, Michael T. Goodrich, and Lowell Trott. *Force-directed graph drawing using social gravity and scaling*. In Proceedings of GD2013, LNCS 7704, pp. 414-425, Springer, 2013.

[3] Farshad Ghassemi Toosi, Fernando V. Paulovich, Marc-Thorsten Hütt, and Lars Linsen. *Projection-based visualization of dynamical processes on networks*. In Miriah Meyer and Tino Weinkauf (eds.): EuroVis - Short Papers, pp. 61 - 65, Vienna, Austria, 2012.

[4] Yoshiki Kuramoto. Chemical Oscillations, Waves, and Turbulence. Springer Series in Synergetics. Springer Berlin Heidelberg, 1984.