A recursive, parallelizable, radial layout algorithm for interactive graph visualization and animation

Andrew Pavlo*
University of Wisconsin

Christopher Homan†
Rochester Institute of Technology

Jonathan Schull‡
Rochester Institute of Technology

Abstract

We have developed (1) an interactive graph visualization system that allows users to explore graphs by viewing them as a succession of spanning trees selected interactively, (2) a radial graph layout algorithm, and (3) an animation algorithm that generates meaningful visualizations and smooth transitions between graphs while minimizing edge crossings during transitions and in final layouts.

Our system is similar to the radial layout system of Yee et al. [25], but differs primarily in that each node is positioned on a coordinate system centered on its own parent rather than on a single coordinate system for all nodes. Our system is thus easy to define recursively and lends itself to parallelization. It also guarantees that layouts have many nice properties.

We compared the layouts and transitions produced by our algorithms to those produced by Yee et al. Results in several experiments indicate that our system produces fewer edge crossings during transitions between graph drawings, and that the transitions more often involve changes in local scaling rather than structure.

These findings suggest the system has promise as an interactive graph exploration tool in a variety of settings.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms; H.5.0 [Information Interfaces and Presentation]: General

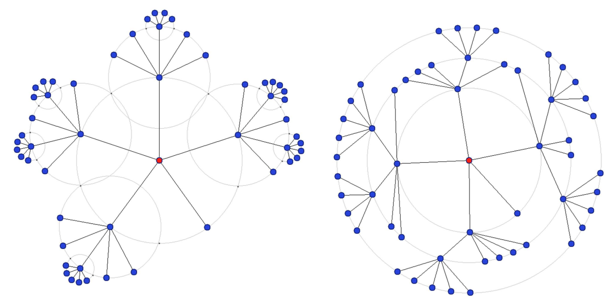
Keywords: Graph and network visualization, Interaction, Focus + Context Techniques, Animation, Hierarchy visualization

1 Introduction

Visualization can help make graph structures comprehensible [6, 11, 22]. However, edge crossings can challenge human perception of relationships between nodes [7, 18, 23]. Graphs often come to us as tangled webs that cannot be displayed in a two-dimensional viewing plane without edges crossing.

Because trees *can* be laid out on a plane without edge crossings, a common approach is to base graph visualizations on spanning trees extracted from graphs [8, 14, 17, 25]. Although the resulting drawings may discard some potentially significant edge information, a clearer mental picture of the full graph may nonetheless result if users can easily and intuitively explore multiple layouts based on different spanning trees.

Yee et al. [25] describe a tool that draws radial tree layouts [2, 12, 21, 24] of breadth-first spanning trees, given a graph and a node selected to be the root (see Figure 1(b)). A user may then select a new root node and the system transitions smoothly to a new layout based on the new root node. This transition is animated by a succession of linear interpolations of the polar coordinates (in a coordinate system centered on the root) of positions of the each node in the old and new layouts. Thus, a user can interactively explore a graph that would otherwise be too complex to visualize or comprehend as a single, static drawing.



(a) (b)

Figure 1: Two drawings for the same tree: Figure 1(a) is drawn by our algorithm, and Figure 1(b) is drawn by Yee et al.'s algorithm for Gnutellavision [25].

In drawings generated by Yee et al.'s radial layout method, each successive generation of nodes lies on a single circle centered on the root. Such layouts have the nice property that all nodes of a given generation are equidistant from the root. However, because each generation shares a single circle, two distantly related nodes may be positioned close to each other simply because they belong to the same generation. This can obscure symmetries in the tree. Furthermore, because the position of any node depends on the positions of all of its generation-mates, such drawing algorithms are not easily parallelized.

In this paper, we describe an approach that is similar to Yee et al.'s in that it bases its drawings on breadth-first rooted spanning trees extracted from graphs, allows users to interactively change views of each graph by selecting a new root, and smoothly transitions between successive layouts by moving nodes along radial paths.

However, unlike Yee et al., we place every subtree in the graph in a "parent-centric" circle surrounding its own subroot, instead of positioning each node on a "generation circle" centered on the root.

Because each node's position now depends only on its parent and siblings, not on its entire generation, the dependencies in our layout are thus very local. The drawings and animations in our system can be computed in multiple, potentially parallelizable, depth-first traversals for each subtree.

Other radial layout algorithms also use a parent-centric visualization scheme, but differ from our approach in that a child node's circle is placed entirely inside of its parents circle [21] or allow for any node's children to be positioned completely around its circle [12, 16].

In broad strokes, our algorithm works as follows. We place the root in the center of the display with its children evenly distributed along a containment circle centered on the root. We then draw circles around the root's children and evenly distribute their children along containment arcs that ensure that neither siblings nor cousins

*e-mail:pavlo@cs.wisc.edu
†e-mail:cmh@cs.rit.edu
‡e-mail:jis@it.rit.edu

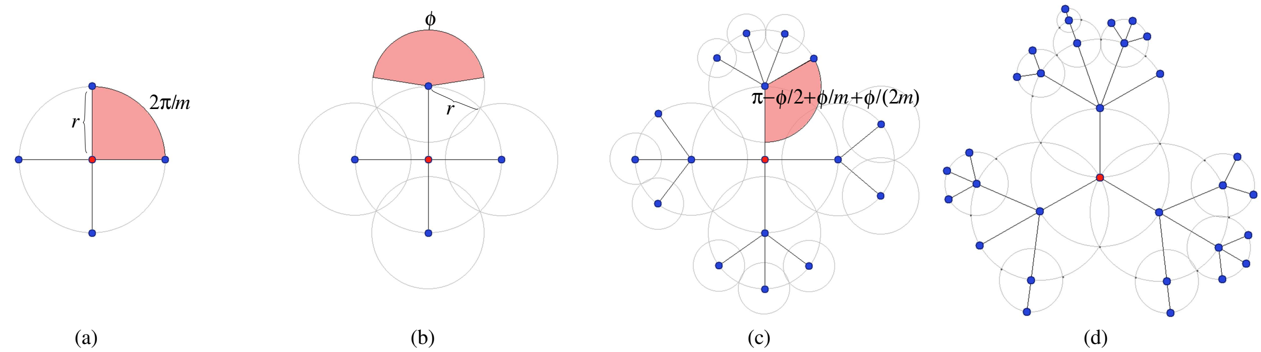


Figure 2: A visual overview of how our algorithm constructs a new layout for a graph. In Figure 2(a), the root is first placed at the center of the drawing surrounded by its children. Next, the root's children are allocated containment arcs (Figure 2(b)) where their descendants are positioned within (Figure 2(c)). The final static layout is shown in Figure 2(d).

overlap. This process proceeds recursively so that successively distant descendants of the root are positioned on successively smaller containment circles (Figure 2).

Our layouts have several aesthetic virtues: They have a flower-like, self-similar structure that differs from the "bulls-eye" appearance of Yee et al.'s layouts (compare Figure 1(a) and Figure 1(b)). And even though the distance between root and nonroot nodes is less directly represented than in Yee et al.'s system, there are powerful visual cues to compensate: Within a lineage, edge lengths decrease monotonically with distance from root, and all siblings within a family are arrayed along visually salient arcs equidistant from their common parent. With regard to animating transitions, in our algorithm sibling edges never cross when a new focal node is selected, and whenever the graph to be drawn is itself a tree.

We experimentally evaluated our animation algorithm, using Yee et al.'s algorithm as control. In our experiments, our algorithm produced fewer edge crossings during transitions than Yee et al.'s.

2 Data Model and Algorithms

We assume that all graphs are connected. We regard any drawing of a spanning tree of any graph as a drawing of the graph. Since all edges we draw are straight lines we need only describe how to map the nodes to points in the drawing plane. We use "node" to refer to both a vertex in a graph and the point on the drawing plane where the node is drawn. It is perhaps easiest to explain our algorithms in terms of a particular data model that completely describes a drawing in this restricted sense. Rather than represent the position of all nodes of some graph in terms of a single polar coordinate system centered at the origin of the drawing plane that all nodes share, we only use the standard drawing-plane-centered origin to represent the root node and its children. We represent every other node position in terms of polar coordinates *centered at the node's parent* [9].

2.1 Parent-centered data model

We now formally define this concept. Given a tree T and a drawing D of T we recursively define a *parent-centered model* of (D,T) as follows. For any node v of T, the polar coordinates of v are given in the coordinate system

(basis 1, i.e., if v is the root of T:) sharing the origin with the drawing plane and zero degrees with the positive direction of the drawing plane's x-axis,

(basis 2, i.e., if v is a child of the root of T:) having the root of T as the origin and the ray from the root having the same direction as the positive direction of the drawing plane's x-axis as zero degrees, or

(recursion, i.e., otherwise:) having v's parent in T as the origin and the ray from v's parent intersecting v's grandparent as zero degrees.

Thus nodes having the same parent share the same coordinate system and nodes having different parents have different coordinate systems.

Note that we can (and do) represent any straight-line graph drawing this way, not just those produced by Algorithm 1 below.

2.2 Static layout algorithm

We define our static layout algorithm recursively and in terms of our static layout model as follows (see also Figure 2). When we say that a nonroot node lies on a *containment circle*, we are refering to the circle centered at the node's parent that intersects the node. Note that if two siblings are the same distance from their parent (this is a property of the drawings Algorithm 1 produces) then they share the same containment circle.

Algorithm 1 *Given a spanning tree T, for each node v of T let the coordinates of the root node be $(0,0)$ and for each nonleaf node v let $v_1,\ldots,v_m$ be v's children. For each $i \in \{1,\ldots,m\}$ let the coordinates of v_i (in the parent-centered model) be*

(basis, i.e., if v is the root:) $(2\pi i/m, r)$, *where r is some user-defined value > 0,*

(recursion, i.e., otherwise:) $(\pi - \phi/2 + \phi i/m + \phi/(2m), r)$, *where ϕ is some user-defined value and r is*

- *the same as v's magnitude, if v's parent has fewer than three children.*
- *the radius of the circle centered at v that intersects the midway point between v and v's nearest siblings on their shared containment circle.*

Note that the value of r for any nonroot node depends only on the node's parent, so as claimed above all sibling nodes share the same value for r. This means they all lie on the same containment circle, which we call the containment circle *of* the parent node.

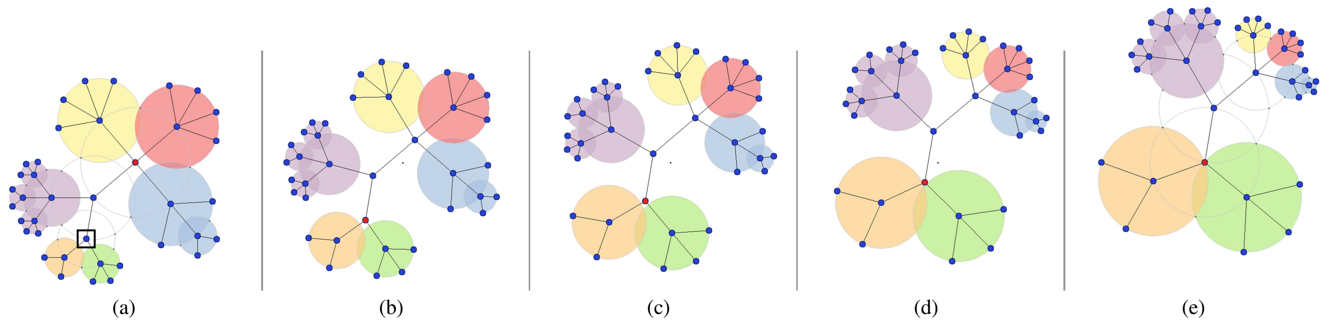


Figure 3: Isomorphic Tree Transition – Our visualization scheme transitions between two drawings of the same tree by scaling each parent node's containment circle with its children. In this example, the user selects a new root node from the initial drawing in Figure 3(a). The containment circles highlighted in the transition (Figures 3(b) – 3(d)) grow and shrink as the graph moves to the new drawing in Figure 3(e).

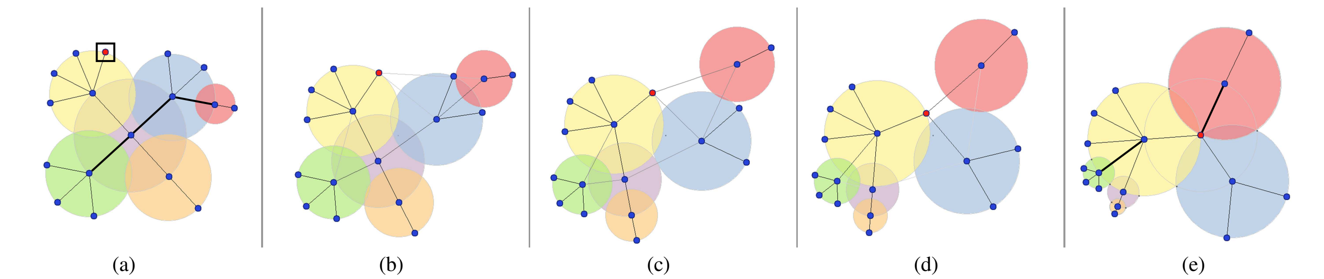


Figure 4: Spanning-Tree-to-Spanning-Tree Transition – The user's selection of a new node invokes a new spanning-tree-baseddrawing (edges that will fade out are highlighted in Figure 4(a); newly introduced edges are highlighted in Figure 4(e);)

2.3 Animation algorithm

Our static layout algorithm leads to a simple and intuitive algorithm for animating transitions from one layout to another that, for any graph G, drawing D_{old} and node v of G, interpolates between the parent-centered models of D_{old} and a drawing produced by Algorithm 1 of a spanning tree of G rooted at v (see Figures 4 and 3).

Algorithm 2 *1. Compute a breadth-first spanning tree T of G rooted at v.*

2. *Let D_{new} be a drawing produced by running Algorithm 1 on G and T.*

3. *Let M_{old} be a parent-centered model of (D_{old}, T) and M_{new} be a parent-centered model of (D_{new}, T)*

4. *For each t in an increasing sequence $0, t_1, \ldots, t_p, 1$, output a polar drawing D_t such that the model of (D_t, T) is described recursively as follows.*

   (basis, i.e., if v is the root of T_{new}): $(\theta, (1-t)r)$, *where (θ, r) are the polar coordinates of v in model of M_{old}.*

   (recursion, i.e., otherwise:) $(t\theta_{new} + (1-t)\theta_{old}, tr_{new} + (1-t)r_{old})$ *otherwise, where, for $x \in \{old, new\}$, (θ_x, r_x) are the coordinates of v in M_x.*

Thus, the new root node moves in a straight line to the center of the new drawing, and each nonroot node moves via a finite approximation of a smooth interpolation between its parent-centered polar coordinates in the new and old drawings. In the resulting animation, newly-central families expand and fan out as they move toward the center, while newly-peripheral families shrink as they arc toward the periphery. Neighboring family circles are guaranteed not to interpenetrate.

In practice, the algorithm is used to produce a succession of drawings (thus producing a temporal sort of focus+context [6, 20]) of G, each one based on a spanning tree rooted at whatever node the user chooses. Thus, D_{old} is typically a drawing produced by Algorithm 1 (though it need not be) and the drawing D_{new} produced as the output of Algorithm 2 is then used as the input drawing (i.e., it becomes D_{old}) the next time the algorithm is called.

There are different ways in which one can fix the times $t_1, \ldots, t_p$ when generating the intermediate drawings of an animation. We adopted the slow-in, slow-out technique of Yee et al. in our implementation so that the values of $t_1, \ldots, t_p$ are concentrated toward the boundary values 0 and 1.

2.4 Properties

Our algorithms have two main properties worth noting.

Aesthetics: Our layout algorithm (1) ensures that all siblings are equally distant from their parent, (2) ensures that containment arcs of siblings and cousins do not overlap, and (3) produces layouts that provide clear indications (via edge-length and family shape) of closeness to the root.

Also, for any graph G, any drawing D_{old} and a node v of G, there is a choice for ϕ such that for any time $t \in [0, 1]$, the edges corresponding to the spanning tree upon which D_t is based do not cross in D_t. This has a number of consequences, including:

1. if G is a tree then the drawing D_t has no edge crossings.
2. For any node v of G, the edges between v and its children do not cross.

Parallelizability: Note that all four steps of Algorithm 2 can be implemented as single traversal of T (i.e., during the breadth-first search that produces T).

Algorithm 2 thus lends itself easily to parallelization, as a new process can be forked whenever a node of T is traversed. The only data dependency in the algorithm, other than the ones between parent and child nodes, is when the algorithm finishes and each drawing needs to be rendered. This dependency, however, can be handled by having the algorithm create frame buffers for each drawing and writing its output directly into the buffers during the single traversal of T described above.

3 EXPERIMENTS

Our experiments compare our algorithms' layouts and animations to those produced by Yee et al.'s algorithms. In each experimental trial, a random graph was generated, two distinct root nodes within that graph were randomly selected, and the graph was then operated upon by both algorithms as they effected transitions from a spanning tree rooted at the first node to a spanning tree rooted at the second. Each experiment comprised 710 trials per algorithm, in which ten random graphs of order 30–100 (inclusive) were generated using the Erdös-Rényi model [3] with a 10% probability of an edge connecting any two nodes. In our first two experiments, we counted edge crossings during transitions by examining all edges present during the transition (whether derived from the new spanning tree or the old); a single crossing was counted during a trial if two edges crossed at any time during the transition, even if the edges crossed and uncrossed multiple times. In our last experiment, we measure the lengths of edges for sets of siblings nodes to their common parent in static layouts produced by the algorithms.

3.1 Isomorphic Tree Transitions

Because trees are by definition planar, transitional edge crossings are potentially avoidable in the special case where selection of a new root node does not change a tree's edge set. In this experiment, we first extract a spanning tree from a graph rooted at a randomly selected root node and construct a new drawing. We then transition from this drawing to a second drawing of the same tree but with a different node selected as the focal point.

Figure 5 shows that our algorithms successfully produce zero crossings while Yee et al.'s algorithms produce many for this particular transition scenario. As illustrated in Figure 3, our approach avoids crossings because "family circles" simply expand or contract as they move without interpenetrating. In contrast, Yee et al.'s algorithms maintain visual continuity by preserving the direction of the edge from the new root node to its parent in the previous drawing. This can produce dramatically different drawings of the same tree, and result in crossings during the transitions.

This visual effect of our animations is similar to that of rigid-body animation methods [4, 5] as the user can mentally group subgraphs as separate objects and follow the movements more easily [15].

3.2 Spanning-tree-to-spanning-tree transitions

In the second experiment, we counted edge crossings during transitions between two different spanning-tree-based drawings of the same graph. We first create a spanning-tree-based drawing for a graph rooted at a randomly selected node. We then select a second node for a new drawing based on a completely different spanning tree extracted from the graph. Unlike in the previous experiment, the edge sets of the two drawings are not the same in this experiment.

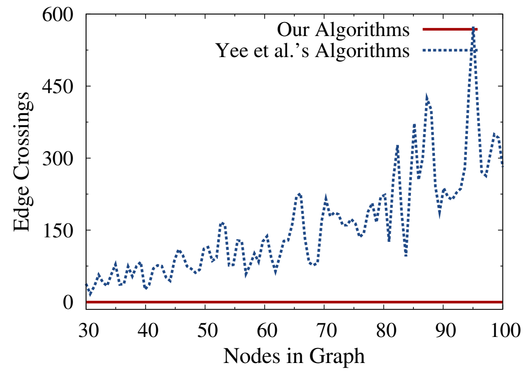


Figure 5: Isomorphic Tree Transitions
Our visualization scheme produces no edge crossings when transitioning between drawings of the same tree, while Yee et al.'s system produces many.

Our evaluation distinguishes "transient crossings involving fading-out edges" from "transient crossings involving final layout edges". A crossing is *transient but fading* if at least one of the edges fades from the viewing plane during the animation sequence. A *transient and non-fading* crossing occurs when both edges are part of the final drawing.

As shown in Figure 6(a), the two visualization schemes produce a comparable number of transient but fading crossings. But Figure 6(b) shows that our algorithms produce fewer transient and non-fading crossings than Yee et al.'s, and that this difference grows with graph order.

3.3 Spanning tree sibling edge lengths

Since our approach positions nodes on containment arcs around their parent whereas Yee et al. positions nodes on concentric circles around the root node, the two systems produce different patterns of regularities. In this experiment we quantify those regularities.

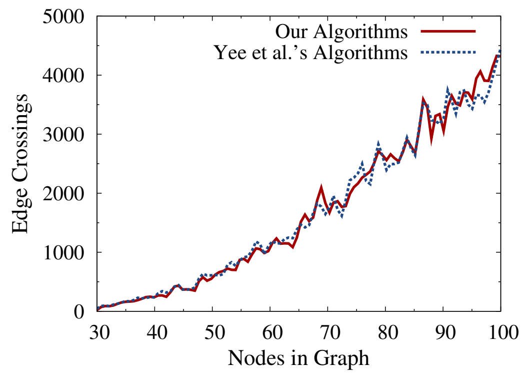
Figure 7 shows that as the generational distance increases from the root to nodes at a given depth in the tree, our system produces no variance among siblings in within-family distance from node to parent, whereas Yee et al.'s system produces substantial variance.

Conversely, in our system the distance from the root to nodes of a given generation can vary, whereas in Yee et al.'s system it does not. The variance arises in our case because containment arcs are adjusted to help prevent neighboring family circles from overlapping. Although this reduces the reliability of the length of edges as an indicator of distance from the root, the self-similar geometric pattern of family subsystems produces another cue that may well be more salient [13].

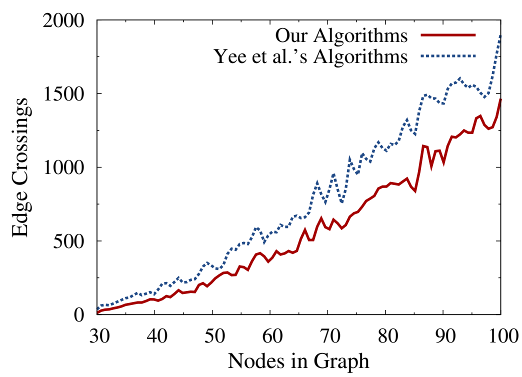
4 DISCUSSION & FUTURE WORK

Behavioral tests will be needed to determine whether these alternative layout and transition algorithms are psychologically significant. But our statistical experiments indicate that the drawings and animated transitions generated by our algorithms conform to many established aesthetics for graph drawings [1, 18].

Taken in the context of the prior research on graph drawing aesthetics, these results suggest that our system should reduce a user's mental effort and increase a user's capacity to make reliable judgments and develop useful intuitions about complicated graph structures [7, 18, 23]. Our research thus lays the groundwork for future study of the layout and animation algorithms, of the psychological significance of our metrics, and of the functional validity of the



(a) Transient Fading-Out Crossings



(b) Transient Final Layout Crossings

Figure 6: Spanning-Tree-to-Spanning-Tree Transitions
The results in Figure 6(a) shows that both visualization systems produced similar amounts of edge crossings during transitions between two different spanning-tree-based drawings. The results in Figure 6(a), however, clearly show that our algorithms produced fewer crossings than Yee et al.'s algorithms.

graph aesthetics themselves.

With regard to our algorithms, two areas are particularly ripe for further study. First, the drawings produced by both ours and Yee et al.'s graph drawing algorithm are not guaranteed to be planar; in our drawings, edge crossings can occur when long subtrees encroach on neighboring containment circles. An alternative method of allocating containment arcs might make it possible to guarantee planar drawings.

Second, with our approach, remote descendants of the root can become vanishingly small on the viewing plane. Our system does give users a natural solution to this problem: selecting a different root node so as to allocate more space to its descendants [19, 20]. However, future research could explore the algorithmic relation between our solution with the distortion of the viewing plane in hyperbolic visualizations [10, 14]. There are clearly differences: we position and move siblings by constraining them to circles on a parent-centered Euclidean plane, whereas hyperbolic layout algorithms position and move siblings through a non-Euclidean space. The relative computational and psychological merits of these approaches, however, remain to be determined.

5 Conclusion

We have presented a radial graph layout visualization scheme based on a parent-centered data model for spanning trees extracted from a graph. We introduced our static layout algorithm that produces

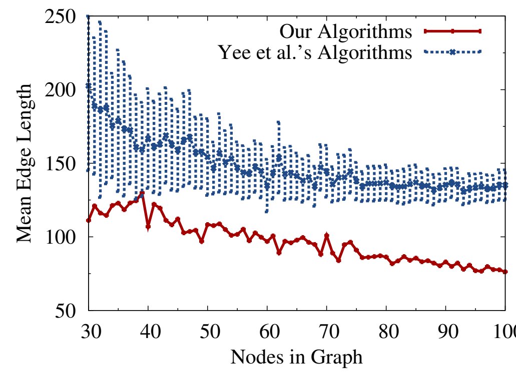


Figure 7: Spanning Tree Sibling Edge Lengths
This diagram shows the mean length of edges with the standard deviation for sets of siblings to their common parent. Sibling nodes are always equidistant to the parent node in the drawings produced by our algorithms. Yee et al.'s algorithms do not generate drawings with this property.

drawings of graphs where the root's children are evenly spaced on a circle centered at the root and the children of nonroot nodes are evenly spaced on a semicircle emanating from their parent. We also introduced an animation algorithm that smoothly transitions a graph from one spanning-tree-based layout to another. We conducted experiments to compare our experimental system with Yee et al.'s graph visualization system [25]. The results from these experiments suggest that our visualization and animation schemes have considerable promise in helping users understand and explore graphs.

References

[1] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Upper Saddle River, New Jersey, 1999.

[2] Peter Eades. Drawing free trees. *Bulletin of the Institute of Combinatorics and its Applications*, 5:10–36, 1992.

[3] P. Erdös and A. Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[4] Carsten Friedrich and Peter Eades. Graph drawing in motion. *J. Graph Algorithms Appl.*, 6(3):353–370, 2002.

[5] Carsten Friedrich and Michael E. Houle. Graph drawing in motion II. In *GD '01: Revised Papers from the 9th International Symposium on Graph Drawing*, pages 220–231, London, UK, 2002. Springer-Verlag.

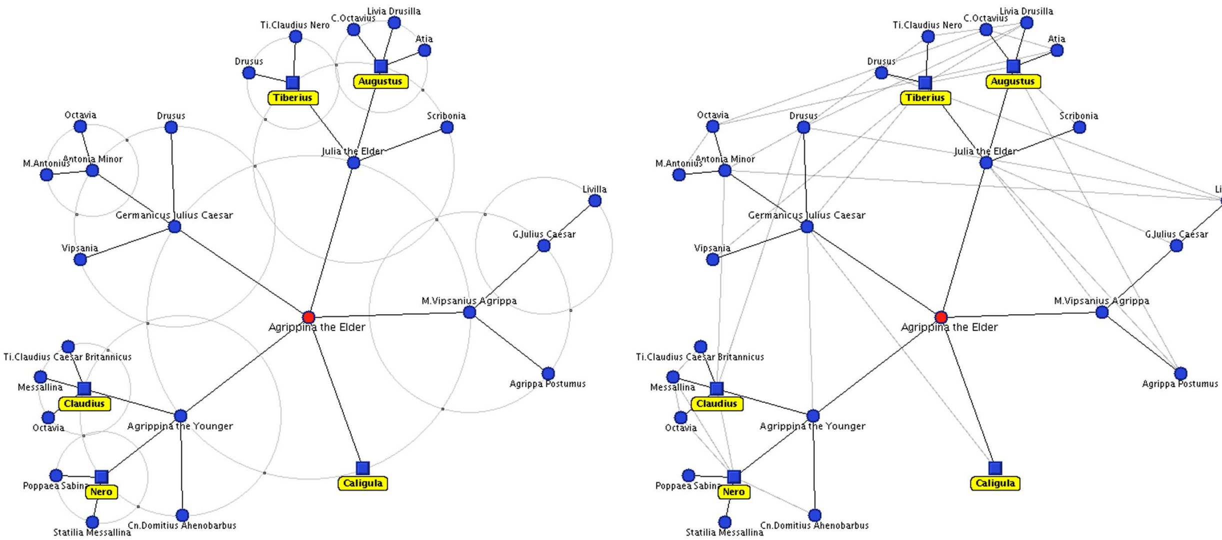
[6] Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[7] Weidong Huang and Peter Eades. How people read graphs. In *CRPIT '45: proceedings of the 2005 Asia-Pacific symposium on Information visualisation*, pages 51–58, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[8] T. J. Jankun-Kelly and Kwan-Liu Ma. Moiregraphs: Radial focus+context visualization and interaction for graphs with visual nodes. In *Proceedings of the IEEE Symposium on Information Visualization*, volume 00, pages 59–66, October 2003.

[9] Matthias Kreuseler, Norma Lopez, and Heidrun Schumann. A scalable framework for information visualization. In *INFOVIS '00: Proceedings of the IEEE Symposium on Information Vizualization 2000*, page 27, Washington, DC, USA, 2000. IEEE Computer Society.

[10] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.



(a) (b)

Figure 8: An example drawing of the Julio-Claudian Imperial family network generated by our algorithm. An edge between two entities in the graph denotes either a marital, parental, or adoption relationship. The Roman emperors in the graph are indicated by boxed labels. The graph is shown in Figure 8(a) with all the edges not a part of the spanning tree in Figure 8(b) revealed.

[11] Jill H. Larkin and Herbert A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11(1):65–100, 1987.

[12] Guy Melanon and Ivan Herman. Circular drawings of rooted trees. Technical report, Amsterdam, The Netherlands, 1998.

[13] Daniel R. Montello, Sara Irina Fabrikant, Marco Ruocco, and Richard S. Middleton. Testing the first law of cognitive geography on point-display spatializations. In Werner Kuhn, Michael F. Worboys, and Sabine Timpf, editors, *Spatial Information Theory. Foundations of Geographic Information Science, International Conference, COSIT Ittingen, Switzerland, September 24-28, 2003, Proceedings*, volume 2825 of *Lecture Notes in Computer Science*, pages 316–331. Springer, 2003.

[14] Tamara Munzner and Paul Burchard. Visualizing the structure of the World Wide Web in 3D hyperbolic space. In *Proc. 1st Symp. The VRML Modelling Language: Special issue of Computer Graphics*, pages 33–38. ACM Press, 14–15 1995.

[15] Keith V. Nesbitt and Carsten Friedrich. Applying gestalt principles to animated visualizations of network data. In *Sixth International Conference on Information Visualisation (IV'02)*, pages 737–743, 2002.

[16] Quang Vinh Nguyen and Mao Lin Huang. A space-optimized tree visualization. In *INFOVIS '02: Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, page 85, Washington, DC, USA, 2002. IEEE Computer Society.

[17] Steven Noel, Cheehung Henry Chu, and Vijay Raghavan. Visualization of document co-citation counts. In *In Proceedings of the 2002 Intl. Conf. on Data Mining*, pages 691–696, 2002.

[18] Helen C. Purchase. The effects of graph layout. In *OZCHI '98: Proceedings of the Australasian Conference on Computer Human Interaction*, page 80. IEEE Computer Society, 1998.

[19] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 83–91, New York, NY, USA, 1992. ACM Press.

[20] Doug Schaffer, Zhengping Zuo, Saul Greenberg, Lyn Bartram, John Dill, Shelli Dubs, and Mark Roseman. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Trans. Comput.-Hum. Interact.*, 3(2):162–188, 1996.

[21] Soon Tee Teoh and Kwan-Liu Ma. Rings: A technique for visualizing large hierarchies. In *GD '02: Revised Papers from the 10th International Symposium on Graph Drawing*, pages 268–275, London, UK, 2002. Springer-Verlag.

[22] Edward Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire CT, 1983.

[23] Colin Ware, Helen Purchase, Linda Colpoys, and Matthew McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002.

[24] Graham J. Wills. Nicheworks — interactive visualization of very large graphs. *Journal of Computational and Graphical Statistics*, 8(2):190–212, 1999.

[25] Ka-Ping Yee, Danyel Fisher, Rachna Dhamija, and Marti A. Hearst. Animated exploration of dynamic graphs with radial layout. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 43–50, 2001.