Family Tree Visualization

Kerstin Keller University of California, Berkeley Email: k_keller@berkeley.edu Prahalika Reddy University of California, Berkeley Email: prali@berkeley.edu Shimul Sachdeva University of California, Berkeley Email: shimul@berkeley.edu

Abstract—This paper discusses the layout of a family tree that emphasizes temporal data. The ancestors and descendants are laid out radially around a centered person. The layout also supports dynamic interaction with the family tree.

I. INTRODUCTION

A. Motivation

Family trees are one of the most common ways to trace the genealogy of a certain person. Family trees have existed for a long time, and the interest in them has not waned. Many people work on constructing their family tree as a common pastime or hobby. In addition, family trees can be very useful in medical and anthropological studies.

B. Problems

Many family trees fail to properly encode all the necessary and useful information. There are very few family trees that effectively visualize temporal data. Being able to easily see the differences in ages among different nodes is very useful in family trees. In addition, encoding ages into a family tree also provides the ability to extract more information, such as the age ranges within a single generation or among multiple generations. Unfortunately, there are not many ways to do so in a clean and effective manner.

Another problem that family trees do not effectively solve is the issue of scalability and clutter. As a family tree gets bigger, visualizing it becomes a more difficult task because the chances of clutter get higher. In addition, easily searching for a node in family trees is also difficult because there is no easy path to follow unless a user has more information about that person.

C. Solution

In this paper, we discuss our solution to these problems and how they are implemented in the visualization. First of all, the conical-shaped family tree layout helps immensely with the problems listed above. The family tree has ancestors above and descendents below a central root node. The root node is the center of the tree and the upper and lower halves of the tree fan out from it. In addition, the family tree visualization has a circular axis that highlights the age differences between certain nodes. The circular axis is centered on the root node, and allows the user to see which nodes are how far in age from the root node. Each ring in the circular axis represents a certain number of years from the root node. Finally, by implementing many interactive features in this family tree, we have been able to effectively show temporal data and reduce unnecessary clutter in the tree.

II. RELATED WORK

Family trees have been a topic of interest for researchers. Various approaches for visualizing tree-like structures have been researched.

A. Research

1) Tree Panel - A study on multiple views for tree visualization: In his paper, Soon Tee Teoh, talks about multiple views for tree visualization [1]. The paper discusses Tree-panels, a tree visualization system that provides four simultaneous visualizations of a tree. These four layouts are:

- 1) RINGS Layout
- 2) Radial Tree Layout
- 3) TreeMap Layout
- 4) Hierarchical Layout

In figure 1, the RINGS and hierarchical layouts can be seen [1]. The paper discusses results from user studies conducted to get responses on effectiveness of the four different visualizations.

2) Interactive Fan Charts: In [2], Draper and Riesenfeld talk about interactive fan charts as a space-saving technique for genealogical graph exploration. A fan chart is a radial structure where a person's name is drawn in the center of the graphic, surrounded by concentric rings containing the names of the person's ancestors (see Figure 2). Fan charts are a popular method for visualizing family trees, due perhaps in part to their aesthetic appeal as well as their compact appearance relative to the more commmon tree-based pedigree chart. We leverages this idea of a radial layout in our implementation.

B. Existing family tree software

There exists a number of commercial as well as free software applications for generating family trees. The focus of most of these trees is to allow users to dynamically build a tree - by interactively adding nodes.

Some examples include:



Fig. 1: Horizontal Hierarchical Layout (left) and RINGS layout (right) of an actual file directory tree. The similarity between the sub-directories is more obvious in RINGS, while the hierarchical layout uncovers many immediate files (leaves) in this directory that are obscured in RINGS.



Fig. 2: A sample fan chart.

1) PAF Companion 5.4: PAF (Personal Ancestral File) helps users organize their family history records. It can produce, either on screen or on paper, pedigree charts, family group records, and other reports to help users in their search for missing ancestors [3].

PAF Companion - Pros

- The colors make it easy to identify nodes at different levels as the tree gets huge
- Each node has a label with information
- Birthyears convey some temporal information

PAF Companion - Cons

- As can be seen in figure 3, this is a one-sided tree, showing only ancestors
- The tree is a static image without much interaction
- Temporal data cannot be easily compared or analyzed

2) *MyHeritage.com:* Their vision has been to make it easier for people around the world to use the power of the Internet to discover their heritage and strengthen



Fig. 3: PAF Companion 5.4



Fig. 4: MyHeritage.com

their bonds with family and friends [4].

MyHeritage.com - Pros

- The pink/blue colors help to identify gender
- The labels contain qualitative information about the nodes
- Presents some form of temporal information birth and death years

MyHeritage.com - Cons

- Temporal data, even though present on each node, is not easily comparable across nodes
- Scalability with the structure is hard

3) *GenoPro:* GenoPro is a user-friendly comprehensive software that allows one to draw family trees and genograms that can be shared with family. [5]

GenoPro - Pros

- The tree edges encode emotional aspects of relationships well
- The labels contain birth information



Fig. 5: GenoPro



Fig. 6: Online Social Community Representation using Vizster

GenoPro - Cons

- Temporal data, even though present on each node, is not easily comparable across nodes
- The different symbols make the tree look confusing
- Excessive use of color also contributes to the cluttered look

4) Vizster: Vizster is a tool built upon Prefuse that is aimed at visualizing online social networks [6]. Even though the structure of the graph is similar to what we hope to achieve through our visualization, it still does not convey temporal information. Figure 6 shows a sample visualization.

C. Existing Problems

From the above examples, some common problems can be identified (enumerated below):

- Most software applications create static family trees, without much interaction
- Scalability of trees poses problems
- Temporal or geographical data is often not depicted well, or at all
- Rare relationships like second marriages and incest are not well depicted
- Searching over tree nodes has not received much attention

D. Existing graph layout algorithms

1) Radial layout algorithm: Draper and Riesenfeld's paper gave the idea to center a person in the middle of the family tree layout and display all ancestors and descendants around it [7]. In general, radial graph layouts specify a root node and place all other nodes circularly around that root node [8] and [9].

2) Force directed layout algorithms: One of the properties of a nice graph is that nodes that are connected by an edge should be close to each other (or a certain distance apart when the edges are weighted). On the other hand, two nodes should not be so close to each other that they overlap [10]. A way to produce a graph that meets these properties is to model the graph as a "mass-springsystem". For mass-spring systems, each node is given a mass, and is connected by springs to other nodes [11]. Differential equations for each node have to be solved to update its position.

Force directed layout algorithms produce visually appealing results for smaller graphs, but they are usually not suited for large graphs due to time-constrained interactive visualizations [12].

III. DISPLAYING TEMPORAL DATA

The main goal of the visualization is to display temporal information in the context of family trees. This entails showing the age differences within the same generation of a family. In order to encode temporal data in the visualization, we took two approaches.

A. Edge Lengths

Each visualization, when rendered, is centered around a root node. The length of the edge between a node in the tree and the root represents the age difference between the two nodes. Hence, length is used to represent relative ages. For instance, a person who is two years older to the root will be closer to the root, than a person who is ten years older. This invariant of edge length representing age gap, is however, limited to the edges of the root node. The length between other nodes in the graph does



Fig. 7: Radial Family Tree Layout

not represent age difference. This applies to nodes that do not have a direct edge to the root node.

In Figure 7, the center is the root. Edges joining other nodes to the root represent the age difference from the root node in terms of their length. The nodes on the outermost concentric circle are older than the ones on the inner circles. Non-rooted edges do not respect this principle. Since the root node can be dynamically changed, if relative ages of two non-rooted nodes is required, one of these nodes can be made the root. The layout will then be re-rendered with the new root and the temporal information with respect to the required node can be seen.

B. Circular Axes

Figure 7 shows circular arcs spanning the nodes. Each arc represents a relative distance from the root node, as labeled on the left. The farther away a concentric arc is from the center, the bigger the difference between the birthyears with respect to the root node. As mentioned previously, the hourglass layout divides the tree into ancestors on the top half and descendants on the bottom half of the tree. Hence, the birthyears increase inside out in the bottom half (old to young) and decrease in the upper half (young to old). The radial structure of the tree preserves space and allows nodes to be more spaciously placed. With the added ability to change the degree of steepness of the hourglass, the problem of node cluttering is easily addressed.

```
void findChildren(NodeItem n)
{
for (all outgoing Edges)
{
    for (all outgoing Edges)
    {
        nextEdge.setVisible();
        nextEdge.targetItem().setVisible();
        findChildren(nextEdge.targetItem());
    }
}
```

Listing 1: Ensuring visibility of direct descendants

IV. METHODS

A. Choosing the subgraph to display

To improve the layout and avoid cluttering, we chose to only display the direct ancestors and descendants of the root node. Therefore all displayable nodes can be found with a depth-first search, traversing all outgoing edges to find children and the incoming edges to find ancestors.

B. Laying out the graph

For the layout, the graph is divided into two different trees, the descendant tree and the ancestor tree, which are laid out separately using the same scheme.

The idea behind our graph layout algorithm is to combine a simple radial layout with a force directed layout. The simple radial layout is run first and it produces a layout that fulfills the basic demands of encoding age difference as distance to root node. The force directed layout is run afterwards and it performs minor adjustments in case two nodes are placed close to each other, thus avoiding clutter.

1) Simple Radial Layout: The simple Radial Layout visits every node only once, hence it is linear in runtime and very suited for interactive visualization. We placed all descendants of the root node within a cone of a certain angle. The only constraint is posed by adjusting the distance to the root to ensure the encoding of age difference.

Conveniently, the coordinates to place the nodes can be described in polar coordinates. The radius is given by the age gap to the root node. Each root node is also given a range in which it and all its descendants lie. The node is placed in the center of its range (angle β , starting angle α , see Figure 8) so the resulting angel ϕ is given by $\phi = \alpha + \frac{\beta}{2}$. The range δ for the children of each node is given by splitting the α in *n* equal slices for the *n* children of the node. The user can interactively specify the angle of the range α of the root node and hence change the steepness of the hourglass appearance of the graph.

A drawback to this method is, that it does not account for whether the laid out nodes have children themselves, since it gives all of them equal spacing.



Fig. 8: This figure shows how the nodes are placed. The blue node is place in the middle of its range β with the starting angle α . All four child nodes are assigned a new range $\delta = \frac{\alpha}{4}$ with respecting new starting angles γ (here for the red highlighted node). They are then placed in the middle of their new range δ with the right distance to the root node, and the same method is repeated for all their children.

This way, the tree can get very unbalanced, by being crowded and cluttered in one part and having no nodes to display in another part. (See Figure 9) This case is rather unlikely for a familytree, but it might occur. ²

2) Force Based Layout: The force based layout makes $\frac{4}{5}$ adjustments for cases where the first layout becomes $\frac{4}{6}$ rather unbalanced.

For a force based layout, the whole graph is modeled ⁷ as a mass-spring system. In mass-spring systems, each ⁸/₉ node is assigned a mass, and is connected by springs, ⁹ which represent edges, to other nodes. Each particle ₁₀ behaves according to the the equations of motion

$$m\ddot{\mathbf{x}} = \mathbf{F} \tag{1}^{11}$$

where *m* is the mass of the node, \ddot{x} is its acceleration and *F* are the forces that act on the node.

Since the nodes are connected by springs, a spring¹⁴ force acts on each node.

$$\mathbf{F} = k\mathbf{x} \tag{2}^{1}$$

Springs store potential energy and act with a force. The amount of force exerted by a spring depends on the spring constant k and the relative position of the spring with respect to its rest position x.



Fig. 9: a) shows a decent layout that would be produced by the simple graph layout. Fig. b) shows the case of a very unbalanced tree

```
void placeChildNodes(NodeItem n, NodeItem
      root, double angleRange, double
     startAngle)
  int out = n.getOutDegree();
       double slice = angleRange/out;
       int i=0;
       for(Iterator it=n.outNeighbors();
           it.hasNext();)
       ł
        NodeItem kid = (NodeItem) it . next();
        double angle = (2*i+1)/2.0* slice +
            startAngle;
        radius = kid.getInt(birthdate) -
            root.getInt(birthdate);
        kid.setX(root.getX() +
            radius * Math.cos(angle));
        kid.setY(root.getY()
            radius*Math.sin(angle));
        //Recursive call for children
        placeChildNodes(kid, root, slice,
            angle-slice *0.5);
        i++;
       }
17 }
```

Listing 2: Recursive placement of child nodes



Fig. 10: a) shows a the layout before applying the force directed layout. Nodes to the bottom of the picture are very close to each other. b) after applying the force directed layout the closeness of nodes is reduced

In addition to these mass spring forces, the nodes are modeled to repel each other like atoms do on a molecular level.

Force directed layout algorithms solve these equations of motion with a certain time step and updates the position of each node until the system reaches an equilibrium point, e.g., until the velocity and acceleration of each node equals zero.

For our layout, two kinds of springs are used: A spring between a parent and its child and a spring between each node and the root node. The latter keeps the node at the correct age distance. It is modeled as a "stiff" spring, a spring with a rather high spring coefficient. By giving the spring a high coefficient, it is unlikely that the spring will change its length, which corresponds to the age difference between root and child node. The edges between parent and child node are modeled as "soft springs" with a lower spring coefficient, allowing for stretching of these strings beyond their age difference. The root node is set to a fixed point and is not allowed to move during the simulation. This is achieved by setting all forces that are acting on it to zero.

This layout algorithm is run after running the simple radial layout and drives cluttered nodes away from each other. (See Figure 10) We found it convenient to use only 10 iterations of the force directed layout algorithm. This is enough to drive cluttered nodes away from each other, however it keeps the basic shape of the layout. Using more iterations can introduce crossings of edges and other unpleasant artifacts to the layout.

C. Animation between Layouts

The family tree only shows direct ancestors and descendants of the current root node. When switching the root node to another person, some subtrees need to be collapsed or others that had been collapsed need to be displayed.

1) Direct Transitions: Moving one level down in the tree (e.g., making a child node the root node) results in collapsing other children of the former root and expanding all other ancestors of the new root. (See Fig. 11) Moving one level up in the tree (e.g., making an ancestor the new root node) requires collapsing the former roots other ancestors and displaying the new root's other descendants.

To get a smooth transition when switching the focus node between two direct neighbors, the subtree that will disappear is collapsed first, then the nodes that stay visible are transitioned and finally the newly expanded subtree is made visible.

For the transitioning of nodes, a polar animation as described in "'Animated exploration of dynamic graphs with radial layout"' is used [13]. The coordinates are not linearly interpolated in the sense of a Cartesian coordinate system, but rather linearly in the sense of polar coordinates. This results in nodes moving on curves and provides smoother transitions. Prefuse already implements a class to animate those changes.

Prefuse also provides a class for animating the collapsing of subtrees.

2) Longer Distance Transitions: When transitioning between two nodes that are more distant (like a person and their great-grandmother) multiple subtrees need to be collapsed and expanded. To smoothly animate these changes, the path between those two nodes is traversed. This means that to animate the root change between a person and their great-grandmother, the change from root node to mother, from mother to grandmother and finally from grandmother to great-grandmother is animated.

V. SOFTWARE USED

There are various good software programs out there that can be used to implement our design. Protovis, JQuery, OpenGL, JavaScript and ActionScript are a few.

The requirements of our design - the radial and the force-directed layout are best supported by Prefuse. Prefuse is a set of software tools for creating rich interactive data visualizations. The original prefuse toolkit provides a visualization framework for the Java programming language. The prefuse flare toolkit provides visualization and animation tools for ActionScript and the Adobe Flash Player [14].

We created a FamilyTreeLayout that extends the ForceDirectedLayout and RadialLayout classes. The label



Fig. 11: Expanding and collapsing of subtrees when changing the root node: a) shows the subtree that needs to be collapsed, when changing the root from the red to the dark green node in b). When changing the root node further, from dark green to light green in Figure c), the parent tree of the new root node is expanded.

renderer and JPanel components introduce interactivity in the visualization. We also added a search functionality which searches over the names of the nodes in the family tree which are also used to label the nodes.

VI. RESULTS

This family tree visualization effectively encodes temporal data. The conical layout of the tree with the circular axis allows the user to easily see the age differences between a node and the root. In addition, there are many interactive features that the user can use to further the amount of information he can extract from this tree. These features are described below.

A. Interactive Features

1) Dynamic Root Node Switching: We implemented the ability to change the root node of the family tree. Clicking on a node will change the focus point of the tree and the tree is restructured to center around the new root node. The layout of the tree will change so that the new root node's ancestors are still above it and its descendents are below it. In addition, the length of the edges between two nodes, such as the new root node and a neighboring node, adjust to reflect the age difference between them. Depending on the nature of the family, any nodes that are irrelevant to the new root node disappear and previously hidden nodes that concern the new root node appear (see Figure 12).

By providing dynamic root node switching, our implementation of the family tree is more useful and interactive. The ability to change the central node around which all other calculations are done immensely adds to the amount of information that can be extracted from the tree. When users view the tree in different layouts, they can see different patterns emerge and provide more analysis. In addition, dynamic node switching reduces the clutter involved in family trees



Fig. 12: a) This image shows the family tree visualization with temporal data encoded. b) Once the root node has been changed dynamically, some nodes disappear to make the tree less cluttered.



Fig. 13: When the user hovers over a name, a tooltip with more information about that person appears, as shown. In the image above, the user can see that "'Uwe Hoffman"' was born in 1995 and his interests include swimming and jogging.

since it will hide nodes that are irrelevant to that particular root node.

2) Hover Tooltip: To provide the most amount of information with the least amount of clutter, we chose to provide a hover-over tooltip for each node. The tooltip for each node provides additional information about each person, which includes information encoded into the visualization, as well as other information the user chooses to include. For example, a tooltip may contain a person's birthdate and spouse, both of which can be easily deduced from the layout of the tree. The tooltip may also contain extra information such as a person's geographical location, hobbies or interests and medical history (see Figure 13).

The hover functionality allows the family tree to be more informational for the user. The details provided in the tooltips makes the family tree more useful for analysis, but keeps it clean and uncluttered because it does not appear unless the user explicitly chooses for it to.

3) Age Bracket Highlighting: Another feature we implemented is the ability to easily distinguish different age brackets. We provide the users with a slider that allows them to highlight nodes of a certain age bracket. The difference in age is calculated between a node and the root node. As the user moves the slider towards higher positive numbers, nodes that are older than the root node are highlighted. This means that nodes that are ancestors of the root node get highlighted as the slider is moved towards higher positive numbers. As the slider moves, more nodes are highlighted until, eventually, the entire upper half of the tree is highlighted (see Figure ??). Similarly, if the user moves the slider towards higher negative numbers, nodes that are younger than the root



Fig. 14: The user can choose to highlight nodes that are older than the root node, within a specific age range. Figure a) shows what the graph looks like when certain ancestor nodes are highlighted. Figure b) shows the slider control for this feature.

node light up. As with the ancestors, the descendent nodes will all be highlighted as the slider reaches its limit (see Figure 15).

The slider to show different age brackets showcases the temporal data trends in the family tree. It is very easy to see how far in age certain nodes are to the root node and therefore provides more useful information to the user that can be used in analysis.

4) Search: An important interactive feature we implemented is a search on each person's name. As the user searches for a name, nodes are dynamically highlighted. For each letter the user types in, if the number of results decreases, nodes that were previously highlighted are no longer lit up. This continues until the user is finished typing their search term and the appropriate nodes are highlighted, or there are no nodes that qualify, and no nodes are highlighted. In addition, the number of search results appears beside the search bar for easy access (see Figure 16).

The search feature adds a lot of functionality to the family tree visualization because it allows easier scalability. Even if there are more nodes in the tree, a user can easily check if a certain person is in the tree and can easily find the node if it is. Because what is needed can be found easily, large family trees are no longer as big a problem as before for the users.



Fig. 15: The user can also choose to highlight nodes that are younger than the root node, within a specific age range. Figure a) shows what the graph looks like when certain descendent nodes are highlighted. Figure b) shows the slider control for this feature.

19 matches	search >>	kel	X
	STARLED BAR	225.43	

Fig. 16: The search functionality is evoked through this search box. Beside the search box is a quick count of how many results match that search query.

5) Same Generation Highlighting: In addition to a slider that highlights nodes of a certain age bracket, we also implemented the ability to easily view nodes of a particular generation. This slider works similarly to the age bracket slider, but as the slider is moved, nodes of each generation are highlighted based on their distance to the root node. Starting at zero, as the slider increases, the nodes closest to the root light up in an outwards manner. As with the age bracket slider, positive numbers denote ancestors and negative numbers denote descendents. When the slider moves towards more positive numbers, the upper half of the tree is highlighted, and when the slider moves towards the more negative numbers, the bottom half of the tree is highlighted (see Figure 17).

The ability to easily see nodes of the same generation greatly adds to the family tree's visualization of temporal data. By highlighting the nodes of the same generation, it is easy to see patterns in ages among them. In encoding temporal data, an important aspect is to show what the age range within a single generation is, and this slide allows that to be easily determined.



Fig. 17: The user can highlight certain nodes of generation, as determined by its distance from the root node. Figure a) shows how this looks when an ancestor generation is highlighted. Figure b) shows the slider control for this feature.

6) *Cone Steepness:* A third slider is provided for the user to decide at which level of steepness the tree should be displayed. The steepness of the conical shape determines how spread out the nodes are. The wider the angle, the easier it is to see each individual node. The narrower the angle, the easier it is to see more nodes at once. There is a minimum and maximum limit to the cone steepness. The layout starts at the higher limit of steepness, at the most narrow angle and can be changed to be wider by the slider.

The adjustment of cone steepness allows the family tree to reduce clutter. If the user wants to focus on a certain part of the tree, he can increase the angle of the cone and view fewer nodes to decrease the noise. Similarly, if the user wants to view more nodes overall, he can decrease the cone's angle, and see more of the tree. This introduces scalability since more nodes can be better viewed, in a less cluttered way, and allows the user to gain more information from the tree.

7) Zooming and Repositioning: Additional features that add to the overall functionality and ease of use of the family tree are the zooming and repositioning features. The family tree implementation allows the user to zoom into or out of the tree as necessary using right click of the mouse. In addition, the user can interactively reposition the nodes and the tree overall as per their specifications by just holding the left click and dragging the image. The lengths of each of the edges between nodes will stay the same, as they are determined by the age differences between the nodes, but the nodes can be moved around to allow optimal placement as decided by the user.

These two features allow the family tree to be less cluttered. The user can decide whether he wants to view many small nodes, or a few very big, spaced-out nodes based on what aspect of the family tree he is analyzing. Similarly, the repositioning of the nodes allows the user to lay out the nodes in a manner he likes at a greater detail. For example, if once the user has zoomed in on a particular branch of the tree, he can choose to reposition the nodes to reduce any clutter he sees. These features help further the usefulness of the family tree visualization.

VII. DISCUSSION

This implementation of a family tree visualization is very effective in showing temporal data, as well as reducing the issue of scalability. By providing a very clean and appropriate layout, this implementation improves the information that can be extracted from the tree. The conical layout easily shows the difference between the ancestors and descendents of a central root node. In addition, the lengths of the edges encode the age differences between any node and the root node, as well as among other nodes. The circular axes easily shows the scale for the tree and provide the user an effective way at measuring the age differences. Finally, the various different interactive features that have been implemented strongly show the different ways to visualize the temporal data in the family tree.

Audiences learn much about different types of family tree visualization and how this implementation is able to properly encode temporal data. In a nutshell, this implementation of the family tree visualization is very effective in solving the problems of appropriately encoding temporal data, reducing unnecessary clutter and addressing the issue of scalability.

VIII. FUTURE WORK

There are many features included in this implementation of a family tree, but there are several ways it can be improved.

A. Dynamically add nodes and edges

Currently, the user cannot dynamically add nodes and edges to the tree. If the user wants a new node added to the tree, he must add it to the data XML file and reload the tree. In the future, a very useful feature to implement is the ability to add people dynamically to the tree. This will improve the usability of the family tree visualization greatly.

B. Add pictures for each node

In addition, a family tree could greatly improve by having pictures for each person. As it is, our implementation does not include pictures. Aside from it being useful for users to simply see the various people in the family, this feature can also be quite useful in anthropological studies.

C. Lay out nodes geographically

Another feature that will help is the ability to lay out nodes geographically. If the user provides the appropriate information for each node, a very useful feature to implement in the future is an easy way to lay out all the nodes of the tree on a map. By showing where the people live geographically, the family tree visualization can further be used in medical and anthropological studies.

IX. CONCLUSION

In conclusion, this implementation of a family tree visualization successfully encodes temporal data. In addition, this implementation effectively deals with the problems of unnecessary clutter, and scalability. The radial layout of the nodes, centered around one root node, and the circular axis help the user easily see the age differences between different nodes. Furthermore, the various interactive features improve the usability and functionality of the family tree. Overall, this implementation is very effective in solving the problems that have plagued many family trees over the years.

ACKNOWLEDGMENTS

The authors would like to ackowledge the invaluable input from Professor Maneesh Agrawala, an Associate Professor in Electrical Engineering and Computer Science at the University of California, Berkeley. Our gratitude also goes to Jeffrey Michael Heer, the owner and developer of Prefuse, the software we used to build our visualization. We would also like to thank the students of CS294 (Data Visualization) for their feedback and suggestions.

REFERENCES

- [1] S. T. Teoh, "A study on multiple views for tree visualization," 2007.
- [2] G. M. Draper and R. F. Riesenfeld, "Interactive fan charts: A space-saving technique for genealogical graph exploration," 2008.
- [3] "Paf website," http://www.progenygenealogy.com/ paf-companion-details.html. "Myheritage website," http://myheritage.com.
- [4]
- "Genopro website," http://www.genopro.com. [5]
- [6] J. Heer and D. Boyd, "Vizster: Visualizing online social networks," 2004.
- [7] G. M. Draper and R. F. Riesenfeld, "Interactive fan charts: A spacesaving technique for genealogical graph exploration," in Brigham Young University, 2008.
- [8] G. Melanon, G. M. Con, I. Herman, and I. Herman, "Circular drawings of rooted trees," 1998.
- [9] C. cheng Lin and H. chun Yen, "On balloon drawings of rooted trees," pp. 12–14, 2005. [10] T. M. J. Fruchterman, Edward, and E. M. Reingold, "Graph
- drawing by force-directed placement," 1991.
- [11] D. Tunkelang, "A numerical optimization approach to general graph drawing," Tech. Rep., 1999.
- [12] Wikipedia, "Force-based algorithms (graph drawing) wikipedia, the free encyclopedia," 2010. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Force-based_ algorithms_(graph_drawing)&oldid=357941495

- [13] K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst, "Animated exploration of dynamic graphs with radial layout," Washington, DC, USA, p. 43, 2001.
 [14] "Prefuse website," http://prefuse.org.