DigitalTree: A Tool for Displaying Biological Data in Tree Structure

Robin Kramer¹, Victor Olman², Ying Xu², and Dong Xu¹

¹Computer Science Department and Christopher S. Bond Life Sciences Center,
University of Missouri-Columbia, Columbia MO 65211-2060, USA

kramer@mizzou.edu, xudong@missouri.edu

²Department of Biochemistry and Molecular Biology and Institute of Bioinformatics,
University of Georgia-Athens, Athens, GA 30602-7229, USA

olman@csbl.bmb.uga.edu, xyn@bmb.uga.edu

Abstract. We developed a computational tool, DigitalTree, for displaying a two-dimensional weighted tree of biological data. Existing methods, Radial Layout, Hyperbolic Layout, Phylloclastic Placement, and Force-Directing were integrated into DigitalTree. DigitalTree preserves the distances between nodes in the tree and the trees topology. The tool has been applied in analyzing microarray data and phylogenetic trees. DigitalTree is available for download at <http://digbio.missouri.edu/~digtree>.

1 Introduction

Weighted trees, representing biological data, have been widely used. Phylogenetic trees can represent biological species and genes in a tree structure. Biological data can be organized in a Minimum Spanning Tree (MST) for clustering [9]. Most of these biological data are complex and have high dimensions. Visualizing the tree structure for the biological data on a two-dimensional (2D) plane is useful in terms of understanding the relationships among the vertices in the tree. Various techniques have been developed for displaying trees or graphs, including Force-Directed[4], Radial Layout[7], Hyperbolic Layout[3], Phylloclastic Layout[1], and Information Visualization [5]. Although some distortion in the weight relationship among vertices is expected due to the projection from high dimension to 2D, existing methods typically do not preserve the ratios among edges of the displayed tree. In addition, these tools generally lack utilities for handling standardized biological data, such as the widely used phylogenetic tree format.

We developed an algorithm for laying out general weighted, both rooted and unrooted, trees on a 2D plane. This algorithm was implemented as a tool for displaying specific biological data on a personal computer. A weighted tree contains a set of weighted edges such that no vertex can be its own ancestor. A layout is a set of abstract coordinates that can be displayed, with only simple transformations (rotation, translation, scaling, and shearing) being performed. Computer visualization preprocesses for the eye and as such adheres to aesthetic

principles. Conflicting aesthetic principles like spreading every vertex away from each other vertex and preventing edge crossings are balanced by DigitalTree, while maintaining the integrity and quantity of information being displayed.

2 Algorithm and Methods

In this section, we will discuss various techniques that we used for tree display, including choosing a root, calculating angles, placing vertices, optimizing layouts, adjusting angles, and choosing orderings of the children. The layout includes two parts: (1) placing an initial 2D layout based on certain angles of children around their parent vertex, while ratios among the edge weights are preserved; (2) optimizing the placement through an objective function using the steepest descent optimization.

2.1 Choosing a Root

From a tree $T = \{V, E\}$, with the set V of vertices with parents $P(v)$ or children $C(v)$ and a set E of edges with weights $w(e)$ where $e \in E, v \in V$, DigitalTree first chooses a root that balances the tree. For this purpose, we define a path γ between two vertices i and j as $\gamma_{i,j}$ containing edges $e \in \gamma_{i,j}$. A path's length is the sum of the weights of edges on the path. The root quality of the i -th vertex is calculated as follows.

$$\Delta_i = \sum_{j \neq i} \left\{ \sum_{e \in \gamma_{i,j}} w(e) \right\}$$

Root is a vertex with index, such that

$$i_{root} = \arg \left\{ \min_i \Delta_i \right\}.$$

2.2 Calculating Angles

Once the root is determined, all descendants are placed on a 2D plane at angles relative to the first child placed, with the lengths of edges proportional to corresponding edge weights. Then the first grandchild of each child is placed relative to the boundary between the child and its closest sibling, and the rest of the children for a vertex are placed relative to the previously placed sibling, and so on and so forth. Although the placement algorithm for the edges is different, the angles themselves are chosen in a similar manner to the algorithm in Nicheworks [7].

For a sub-branch of T rooted at a vertex v , $L(v)$ is the number of vertices that are leaves. Note that $L(v)$ for a parent vertex is equal to the sum of the $L(v)$ values of its children. Therefore, $L(v)$ of the root has as many leaves as the entire tree, and a leaf has $L(v) = 1$, i.e.,

$$L(v) = \begin{cases} 1 & \text{if } v \text{ a leaf,} \\ \sum_{u \in C(v)} L(u) & \text{otherwise.} \end{cases}$$

Fig. 1. A drawing of the edges and boundaries for a tree layout. The solid straight lines represent the edges of the tree. The dashed lines extending from the vertex represent the boundaries on the portion of the drawing surface which is allotted for drawing an edge and its descendant edges. The β s represent the angles between the two dashed lines.

As shown in Figure 1, we define the boundaries between the siblings and the parents, relative to the first boundary of the first child of the root. The children of a vertex are ordered $v = 0, 1 \dots v_n$. The angle β_v at each child vertex is proportional to its parent angle $\beta_{P(v)}$ and its leaf count $L(v)$, and inversely proportional to the parent leaf count $L(P(v))$. In the case of the root vertex the angle is 2π , i.e.,

$$\beta_v = \begin{cases} 2\pi & \text{if } v \text{ is the root,} \\ \beta_{P(v)} \left(\frac{L(v)}{L(P(v))} \right) & \text{otherwise.} \end{cases} \quad (1)$$

We align the first child of the root relative to the horizontal axis of the 2D plane. Then the angle of a boundary relative to the horizontal axis is

$$\theta_v = \begin{cases} 0 & \text{for the first child of the root,} \\ \theta_{P(v)} & \text{for the first child of a parent,} \\ \theta_{(v-1)} + \beta_{(v-1)} & \text{for every consecutive child with the same parent.} \end{cases} \quad (2)$$

Then the initial layout of the angles $\alpha_v^{(0)}$ between an edge and the horizontal axis can be calculated as

$$\alpha_v^{(0)} = \frac{\beta_v}{2} + \theta_v.$$

As a result, for each vertex v and its vertex $(x_v, y_v) = V(\alpha_v)$, we place the coordinates of the root vertex at $(0, 0)$, and the coordinates for every other vertex (x_v, y_v) according to α_v , with preserved edge weight ω_v , i.e.,

$$V(\alpha_v) = \begin{cases} (\cos(\alpha_v)\omega_v + x_{P(v)}, \sin(\alpha_v)\omega_v + y_{P(v)}) & v \neq \text{root}, \\ (0, 0) & v = \text{root}. \end{cases}$$

We can prove that an initial placement of the vertices has no crossing between edges.

proof: For a child vertex v and the corresponding angles α_v and θ_v , an edge can be represented by

$$\{t\cos(\alpha_v) + x_{P(v)}, t\sin(\alpha_v) + y_{P(v)}\}, 0 < t < \omega_v, \quad (3)$$

$$\theta_v < \alpha_v < \theta_{v+1}. \quad (4)$$

The boundaries of the edge can be represented by

$$\{t\cos(\theta_v) + x_{P(v)}, t\sin(\theta_v) + y_{P(v)}\}, 0 \leq t, \quad (5)$$

$$\{t\cos(\theta_{v+1}) + x_{P(v)}, t\sin(\theta_{v+1}) + y_{P(v)}\}, 0 \leq t. \quad (6)$$

By Equations (3-6) the above edge will not intersect the boundaries of its siblings or its parent given Equation (2):

$$\theta_{P(v)} \leq \theta_v, \quad (7)$$

$$\theta_{v+1} \leq \theta_{P(v)+1}. \quad (8)$$

with the boundaries of the parent of this edge represented as

$$\{t\cos(\theta_{P(v)}) + x_{P(P(v))}, t\sin(\theta_{P(v)}) + y_{P(P(v))}\}, 0 \leq t, \quad (9)$$

$$\{t\cos(\theta_{P(v)+1}) + x_{P(P(v))}, t\sin(\theta_{P(v)+1}) + y_{P(P(v))}\}, 0 \leq t. \quad (10)$$

Therefore, given that a vertex $(x_{P(v)}, y_{P(v)})$ is chosen within its own bounding area according to Equations (9,10), the edges of its children cannot intersect their own bounding areas.

2.3 Optimizing the Layout

Based on force-directed methods for creating MST layouts [4], the initial set of angles are refined. The coordinates of vertices are then evaluated with an objective function $f(x_v, y_v)$ given by:

$$f(x_v, y_v) = \sum_{i=1}^n \frac{1}{1 + e^{-(x_i - x_v)^2 - (y_i - y_v)^2}}.$$

where n is the total number of vertices on the tree. The set of angles α and coordinates $(x_v, y_v) = V(\alpha_v)$ can be found through minimizing $f(x_v, y_v)$, i.e.,

$$\alpha = \arg \left\{ \min_{\alpha} \sum_i f(V(\alpha_i)) \right\}.$$

A step size of δ equal to one half of the smallest β between any of the boundaries was chosen for calculating the next set of angles $\alpha' = \alpha + \delta(-1)^n$. Each vertex was updated as its gradient was calculated (rather than after all the gradients had been calculated). This way each edge responds to the updated position of the previous edge, and will not accidentally try to occupy the same location. Also, as one branch is updated all descendants are updated by that same amount. Hence, the gradient algorithm would be less likely to make edge crossings. The next α was calculated according to

$$\alpha_v^{(l+1)} = \delta \{f(x'_v, y'_v) - f(x_v, y_v)\} + \alpha_v^{(l)},$$

where l is the optimization iteration step.

To take advantage of the space introduced by displacing the child vertex from the parent, and to facilitate the gradient optimization, the angle β_i is enlarged proportional to the distance from the parent ω_i , i.e.,

$$\beta'_v = \beta_v + \frac{c\omega_v}{\log(n)},$$

where β_v is from Equation (1), n is the number of vertices in the tree, and c is a constant, which was empirically found $c \approx 0.1$ for good results. The adjustment improves initial appearance and convergence time for both large and small trees.

2.4 Ordering the Children: Phyllotaxis

Phyllotaxis is the placement of new leaves, branches, petals, etc. based on ordering of the children for improving the quality of the tree layout. The idea of Phyllotaxis came from plants. Although plants with limited surface areas are uncertain about the final shape of their branches, plants know that earlier branches are likely to be larger with more sub-branches than later branches. The Golden Ratio $\phi \approx 1.618$ is used to derive the Golden Angle $\chi = 2\pi \frac{1}{1+\phi}$ for placing the new petal of a flower, i.e., $\rho_{(i+1)}$ relative to the last ρ_i satisfies the phyllotactic angles ($\rho_i = \rho_{(i+1)} + \chi$). Analogously, the final shape of each DigitalTree branch is uncertain, and placing the longer branches away from each other, takes better advantage of space. In particular, at a vertex in tree with N children and N phyllotactic angles, we have

$$\{c_k : k = (1 \dots N), \beta_k > \beta_{(k+1)}\},$$

$$\{\rho_i : \rho_i \in (0, 2\pi), i = (1 \dots N), \rho_i < \rho_{(i+1)}\}.$$

The phyllotactic arcs are $\{\hat{\rho}_i = \frac{1}{2}(\rho_{(i-1)} - \rho_{(i+1)}), i_0 = i_N\}$. With the ordering $\{\hat{\rho}_k : \hat{\rho}_k > \hat{\rho}_{(k+1)}\}$, the order of the children c_k is mapped to the phyllotactic order c_i . This ensures that the longest branches are placed where the most space in the phyllotaxis would be.

2.5 Complexity Analysis

Picking the root vertex is $O(n^2)$, and is a small portion of the time involved in calculating the layout. Breadth first traversals can be used to calculate both $\{\theta, \beta, V(\alpha^{(0)})\}$ and $V(\alpha^{(l)})$. These are linear in the number of vertices in the data set $O(n)$. At each gradient step and with m gradient steps, the objective function is evaluated n times. The objective function for general unweighted force-directed methods is $O(n)$ [6], because there is only one vertex moving at a time. Hence, with the average number of descendants from a vertex approximated by $O(\log(n))$, the objective function is on average $O(n \log(n))$ as only the vertex and its descendants need to measure its distance from each other. This makes DigitalTree's total computational complexity $O(mn^2 \log(n))$.

3 Implementation and Features

The algorithm was implemented in the C programming language for speed, while the graphical user interface (GUI) was implemented in Java which called the C code through native interfaces. The algorithm makes use of breadth first and depth first searches in a number of places, these are all coded in while loops for efficiency, rather than with recursive function calls. DigitalTree can handle more than 10,000 vertices including optimization and display in 10 minutes on a typical

Fig. 2. A screen shot of DigitalTree interface

desktop PC with 1 GB memory. There is a DigitalTree file format output from the C package, which the GUI reads and writes. This format stores the parent-child relationships, weights, vertex numbers, vertex names, and coordinates and colors of the vertices.

DigitalTree is a standalone package and it is also incorporated into EXCAVATOR [8], a MST-based clustering package. To produce even better displays, there are several editing features, which allow the user to rotate, zoom, re-center the tree (see Figure 2). DigitalTree also allows the user to move branches of the tree. DigitalTree can parse a number of input formats, in particular the most common phylogenetic tree file format, Newick. The Newick file parser was written using `spirit++`. The outputs of common phylogenetic tree construction tools Phylip, ClustalW and PAUP* can be used. In Phylip there is no need to change formats. In ClustalW, the result can be saved into the Phylip format. In PAUP*, one can use the command (with the file extension `.dnd`):

```
savetrees file=BestMLtree.phy brlens format=phylip;
```

4 Discussion

In summary, we developed a package DigitalTree for displaying a weighted tree on a 2D plane with preserving the ratios among edges. It incorporates and improves various existing algorithms. DigitalTree can handle large-scale biological data and support various existing formats (e.g., the Newick and EXCAVATOR file formats). It facilitates pattern discovery and visualization with interactive features, such as editable vertices, a scale key, and vertex search.

DigitalTree provides a nice layout for weighted tree structures. The layout is especially useful when the underlying measure of distance behaves according to the properties of norms, such as Euclidean distance or Manhattan distance, because inferences about vertices which are connected through other vertices can be made. DigitalTree is also useful for displaying trees with correlation coefficients and phylogenetic distances, which do not satisfy the properties of norms. With correlation coefficients it is useful to recognize which vertices are highly correlated with many other vertices. With phylogenetic trees it is especially useful to see what the evolutionary distance might be.

DigitalTree fills a unique niche which has not been covered by previous layout algorithms. Force directed methods with random layouts, performing well on cyclic graph structures, converge with edge crossings on large tree structures. While edge crossings are admissible or necessary in large cyclic graph structures, they are not in tree structures. Although Radial Layouts are linear in the number of nodes for tree structures and have no edge crossings, they do not take full advantage of the screen space. Hyperbolic Radial Layouts provide a user with a method to zoom in on parts of the graph at the expense of distorting the relative sizes of the tree. DigitalTree provides a method which preserves the ratios among the lengths of edges, while minimizing edge crossings and optimizing screen usage.

Acknowledgments

This project is supported by a National Science Foundation grant NSF/ITRIS-0407204. RK was also supported by a short-term fellowship sponsored by National Library of Medicine, National Institute of Health during the summer of 2005. VO and YX's work is also supported in part by the U.S. Department of Energys Genomes to Life program under project Carbon Sequestration in *Synechococcus* sp: From Molecular Machines to Hierarchical Modeling, National Science Foundation Grants NSF/DBI-0354771, and a "distinguished scholar" grant from Georgia Cancer Coalition.

References

1. Carpendale, S., Agarawala, A.: PhylloTrees: Harnessing Nature's Phyllotactic Patterns for Tree Layout. InfoVis. (2004).
2. Chen, C., Morris, S.: Visualizing Evolving Networks: Minimum Spanning Trees versus Pathfinder Networks. InfoVis. (2003) 9.
3. Lamping, J., Rao, R., Pirolli: A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. Conference on Human Factors in Computing Systems. (1995) 401-408.
4. Noel, S., Chu, C.H., Raghavan, V.V.: Visualization of document co-citation counts. In Proceedings of the Intl. Conf. on Data Mining. (2002) 691-696.
5. Plaisant, C.: The challenge of information visualization evaluation. In Proceedings of the Working Conference on Advanced Visual interfaces. (2004) 109-116.
6. Pulo, K.J.: Recursive space decompositions in force-directed graph drawing algorithms. In Australian Symposium on Information Visualisation. 9 (2001) 95-102.
7. Wills, G.J.: NicheWorks-Interactive Visualization of Very Large Graphs. Journal of Computational and Graphical Statistics. 8 No. 2 (1999) 190-212.
8. Xu, D., Olman, V., Wang, L., Xu, Y.: EXCAVATOR: a computer program for gene expression data analysis. Nucleic Acid Research. 31 (2003) 5582-5589.
9. Xu, Y., Olman, V., Xu, D.: Clustering Gene Expression Data Using a Graph-Theoretic Approach: An Application of Minimum Spanning Trees. Bioinformatics. 18 (2002) 526-535.