BIOINFORMATICS APPLICATIONS NOTE
Vol. 25 no. 4 2009, pages 557–558
doi:10.1093/bioinformatics/btn656

Databases and ontologies

A novel method for large tree visualization

Jeff Heard[1,*], William Kaufmann[2] and Xiaojun Guan[1,*]

[1]Renaissance Computing Institute, 100 Europa Drive, Chapel Hill, NC 27519 and [2]Department of Pathology and Laboratory of Medicine, University of North Carolina, Chapel Hill, NC 27599, USA

Received on September 24, 2008; revised on November 24, 2008; accepted on December 18, 2008

Advance Access publication January 7, 2009

Associate Editor: Jonathan Wren

ABSTRACT

Summary: Many genomic and proteomic analyses generate as a result a tree of genes or proteins. These trees are often large (containing tens of thousands of nodes and edges), and need a visualization tool to fully display all the information contained in the tree. Clustering analysis can be performed on these trees to obtain clusters of proteins, and we need an efficient way to visualize the clustering results. We present a novel tree visualization tool to help with such analyses.

Availability: http://www2.renci.org/~jeff/software/bin/win32/ProteinVis-2.1.6-win32.zip

Contact: jeff@renci.org; xguan@renci.org

1 INTRODUCTION

When processing large amounts of genomic or proteomic data, many analyses produce results as a tree of genes or proteins, for example, to classify genes based on results of microarray analysis, or to classify proteins based on sequence similarity. The resulting trees are large with tens of thousands of nodes. Very often, researchers need to perform analysis on these trees, such as identifying gene clusters based on Gene Ontology (GO, http://www.geneontology.org/), or identifying groups of genes that are up- or down-regulated under certain conditions.

There are tree viewers, such as the Java TreeView (Saldanha, 2004), Hierarchical Cluster Explorer (Seo *et al.*, 2006), PhyloWidget (Jordan and Piel, 2008) and Dendroscope (Huson *et al.*, 2007). But these programs either cannot handle trees with tens of thousands of nodes very well, or do not have the built in capabilities to identify and visualize clusters of nodes based on certain biological properties (such as GO annotations). When clustering analysis is performed, we need a way to represent the resulting clusters in the tree, and to be able to interact with these clusters to check the accuracy of the clusters. Here, we present a new tool for visualizing a large tree and the results of clustering analysis on the tree.

2 METHODS

Java TreeView and others provide linear views of hierarchical trees. While this is appropriate for small trees, large trees show keenly the drawbacks of such an approach. In the binary tree, the total number of internal nodes cannot exceed the number of leaves (Cormen, 2001). Linearly arranged trees devote the same amount of layout space per level to nodes in the tree further from the leaves as they do for nodes near the leaves.

The protein trees in our analysis contain upwards of 18 000 leaf nodes, and therefore we want to spread things out for better viewing while maximizing the amount of the whole tree that we can see on the screen at once. Thus, we have developed a radial view (Sheth, 2003) of the binary protein tree. This radial view takes advantage of the decreasing number of nodes in the interior at higher levels and compresses them in favor of spreading the nodes at the leaves out further. The difference between our tree and Sheth's tree is that our layout remains static while Sheth's tree rearranges itself every time a user clicks on a node. Since the ordering of our nodes is important, we prefer to see the user-highlighted node in the context of the entire tree as is. There are more space optimal ways to represent a tree than this (Nguyen and Huang, 2007), but the advantage of the radial view for our tree is that leaf nodes are adjacent to each other, and the nature of our clustering means that adjacent leaf nodes close together on the rim should have similar function.

The algorithm for laying out the tree starts with the leaves. Leaves are first sorted by an in-order walk of the tree, so that nodes that share a parent will be next to each other, recursively up the tree. Then angles along the rim of a circle are calculated for each leaf node based on this ordering. Then the immediate parents are situated at angles halfway in-between their children. Internal nodes are situated at radii proportional to the number of steps to the root versus the number of steps to the furthest leaf. Once this geometry is created, it is converted to Cartesian coordinates and passed through OpenGL for rendering. Coloring is based on a numerical attribute of each node, usually similarity. For example, if each protein has a vector of features (expression levels), the similarity between a pair of proteins can be measured as Euclidean distance of the two vectors.

The input to the application is a binary tree, generated by any program, in the format of:

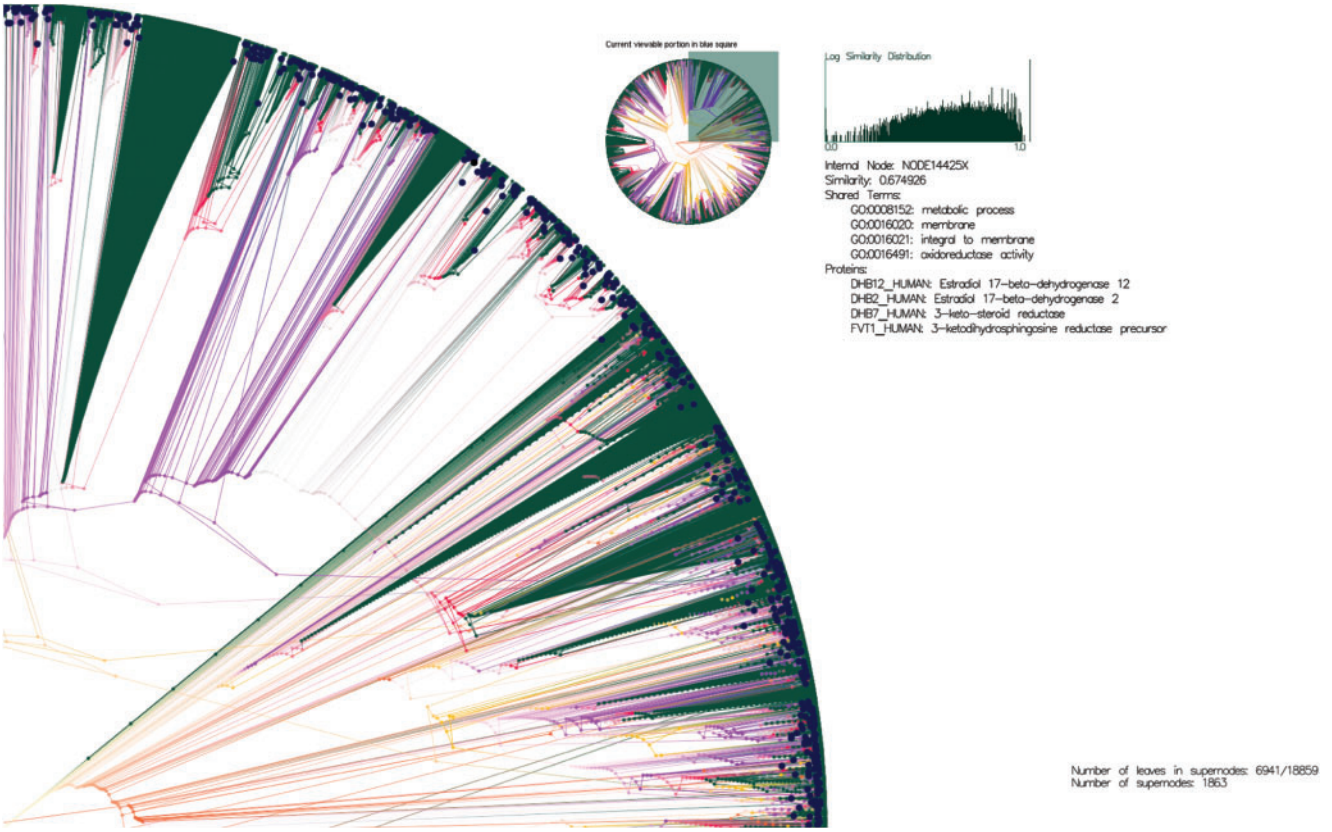
$$Node,\ left-child,\ right-child,\ similarity$$

along with any ancillary metadata about the node. The tree can be constructed with hierarchical clustering. For example, our example data were generated using the Cluster 3.0 program (http://www.geo.vu.nl/~huik/cluster.htm). Leaf nodes represent proteins or genes. At construction time, each internal node is associated with its children, and all descendant leaf-nodes are inspected for shared GO-terms. The intersection of all sets of GO terms in the reachable leaves are then associated with the internal nodes.

Users can add annotations for leaf nodes. Additionally, there are various clustering approaches that can be used to group and collapse subtrees at will. For an example, any internal node whose leaf nodes share at least k GO terms can represent a cluster, where k is adjustable by the users. We call these internal nodes 'super-nodes' and they are highlighted with an enlarged circle and with a special color.

Users can click internal nodes (including super-nodes) to examine the GO terms that they share. Clicking on an internal node also lists the leaf nodes (proteins or genes) that are reachable by descending the tree from that node. Users can collapse and expand a sub-tree by right-clicking a node, thus expanding the rest of the rim to fill the space, allowing one to effectively zoom in on an interesting part of the tree. Users can change the k parameter and be able to check the accuracy of the resulting clusters.

*To whom correspondence should be addressed.



Users can search for individual nodes by name, and the node will snap to the 45° position (centered, in other words) and be highlighted.

The picture above is a snapshot of a tree generated by hierarchical clustering of human proteins based on how they share GO terms. We have a set of 18 000 human proteins that are in our interactome database (idea.renci.org) and we have downloaded all GO terms that these proteins are associated with. Each protein has a vector of GO terms. The hierarchical clustering is performed on these vectors. Then we identify all 'super-nodes' whose leaf children share at least k GO terms and highlight them in blue and in enlarged circles. When clicking a node, all the leaf nodes under this node and GO terms that they share are displayed on the right.

The colors reflect the similarity between nodes. Nodes towards the bottom left (root of the tree) are higher up in the tree, further from the leaves. Leaf nodes are spread around the rim. Enlarged blue nodes are the super-nodes. Proteins are listed to the right. A glyph to the upper right of the main viewing portion shows where in the whole tree the user currently is. One quarter of the full tree can be viewed at any one time. At the bottom right are statistics about the number of total nodes and number of collapsed ones. At the top right is a histogram of showing a breakdown of internal nodes and similarity scores.

The visualization uses the full gamut of mouse functions, and can be controlled, except for node searching entirely by the mouse. The left button selects nodes and displays their details. The scroll wheel rotates the tree interactively. The right button brings up a menu for changing k and searching for nodes. The middle button expands and contracts subtrees. We have tested this visualization with success on both specialized visualization hardware (a $9' \times 14'$ high-resolution visualization wall), and typical desktop and laptop hardware.

Funding: National Institute of Environmental Health Sciences (#5P01ES014635-01, P30-ES10126, in part).

Conflict of Interest: none declared.

REFERENCES

Cluster 3.0 - http://www.geo.vu.nl/~huik/cluster.htm.

Cormen,T. *et al*. (2001) Introduction to Algorithms. 2nd edn. McGraw-Hill, Boston, pp. 253.

Gene Ontology - http://www.geneontology.org.

Huson,D.H. *et al*. (2007) Dendroscope: an interactive viewer for large phylogenetic trees. *BMC Bioinformatics*, **8**, 460–465.

Jordan,G. and Piel,W. (2008) PhyloWidget: webbased visualizations for the tree of life. *Bioinformatics*, **24**, 1641–1642.

Nguyen,Q.V. and Huang,M.L. (2007) A space efficient clustered visualization of large graphs. In *Fourth International Conference on Image and Graphics (ICIG 2007) 2007*, Sichuan, China, pp. 920–927.

Saldanha,A. (2004) Java Treeview—extensible visualization of microarray data. *Bioinformatics*, **17**, 3246–3248.

Seo,J. *et al*. (2006) A interactive power analysis tool for microarray hypothesis testing and generation. *Bioinformatics*, **7**, 808–814.

Sheth,N. *et al*. (2003) Treemap, Radial Tree and 3D Tree Visualizations. In *IEEE Information Visualization Conference*, Seattle, Washington, pp. 28–129.