CluMa-GO: Bring Gene Ontologies and Hierarchical Clusterings Together

Andreas Kerren*
Linnaeus University, School of Computer Science, Physics and Mathematics (DFM)
Växjö, Sweden

Ilir Jusufi†
Linnaeus University, School of Computer Science, Physics and Mathematics (DFM)
Växjö, Sweden

Vladyslav Aleksakhin‡
Linnaeus University, School of Computer Science, Physics and Mathematics (DFM)
Växjö, Sweden

Falk Schreiber§
IPK Gatersleben and Martin-Luther University Halle-Wittenberg
Germany

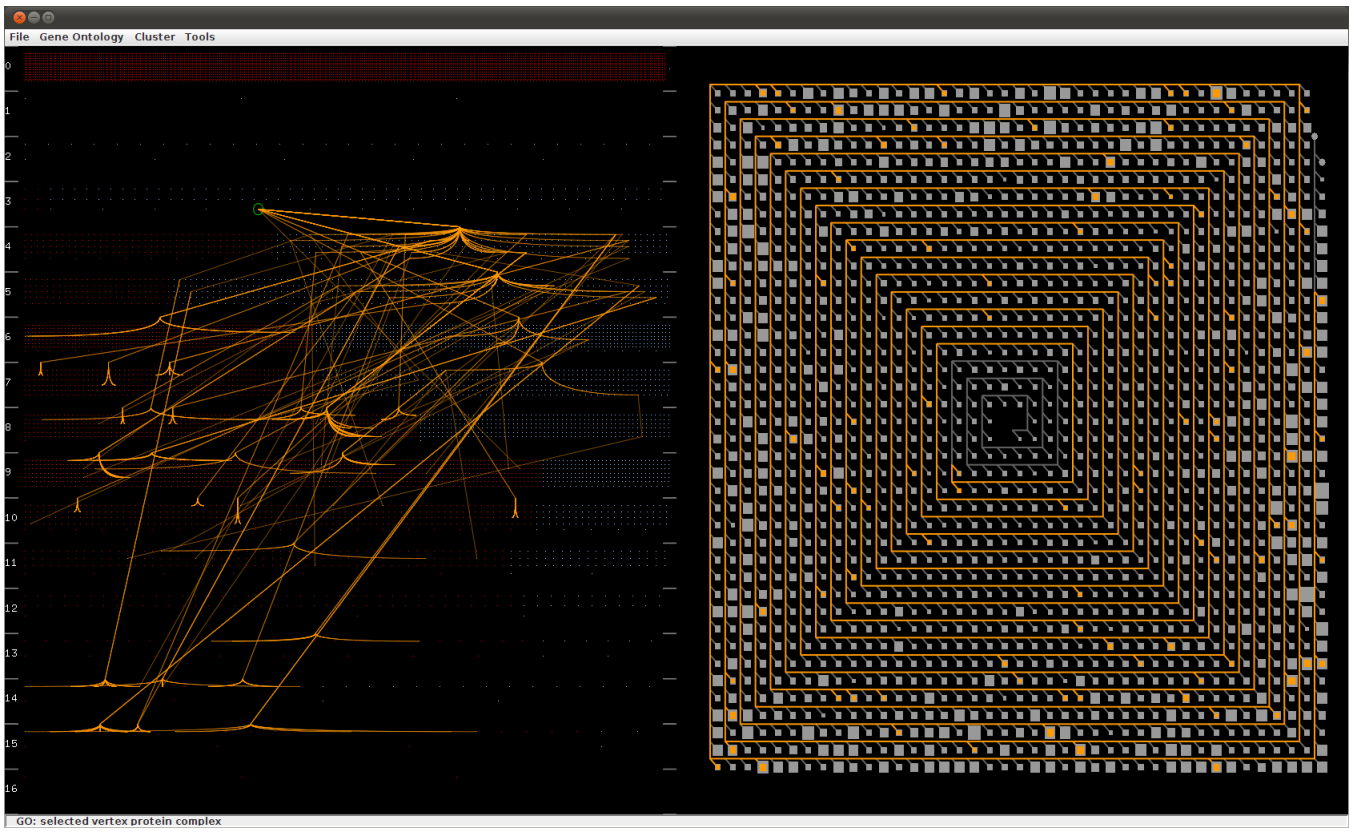

Figure 1: GUI of CluMa-GO. On the left hand side, the used Gene Ontology is represented in the GO view (Levels Layout). The green circle represents the currently selected GO term whereas the visible edges represent its relationship to subterms. On the right hand side, the clustering view is located.

1 Introduction

Ontologies and hierarchical clustering are both important tools in biology and medicine to study high-throughput data such as transcriptomics and metabolomics data. Enrichment of ontology terms in the data is used to identify statistically overrepresented ontology terms, giving insight into relevant biological processes or functional modules. In this paper, we focus on Gene Ontology (GO) [4] which is an on-line database that provides a set of structured vocabularies (ontologies) for the annotation of genes, gene products and sequences. Hierarchical clustering is a standard method to analyze and visualize data to find relatively homogeneous clusters of experimental data points [3]. It is a statistical method for finding relatively homogeneous clusters and is based on two steps: computing a distance matrix containing the pairwise distances between biological objects and a hierarchical clustering algorithm. Both, ontologies as well as hierarchical clustering, are widely used to support the analysis of molecular-biological data obtained by high throughput technologies and lead to huge data sets of DAG- [1] and tree-like structures. To help analyzing this data, often two views are desired: visualizing a large data set (such as the expression levels of the genes in an organism) in the context of an ontology (such as the GO) and in the context of a clustering of the data (such as an hierarchical clustering).

Due to the complexity and huge amount of data to be visualized, we visualize the GO DAG and the clustering in two separated and coordinated views [6]. We use a transcriptomics data set representing different expression levels of genes in E. coli. The initial data set has been reduced to 7,312 genes, which are significantly up- or down-regulated. The hierarchical clustering has been computed based on the distance of the expression levels of genes at different time points. The resulting binary tree of the cluster analysis, called *Cluster Tree*, has 14,623 nodes (7,311 non-terminal nodes and 7,312 leaves) and 14,622 edges. Furthermore, the GO is a DAG consisting of more than 34,000 inner nodes and a large amount of leaf nodes depending on the organism under considera-

*e-mail: andreas.kerren@lnu.se
†e-mail: ilir.jusufi@lnu.se
‡e-mail: vladyslav.aleksakhin@gmail.com
§e-mail: schreibe@ipk-gatersleben.de

tion. We only consider the nodes representing the 7,312 genes and those nodes, which are on paths between the GO root node and leaf nodes (genes). Thus, the final GO data set consists of 10,042 nodes and 24,155 edges. The graph has one root, 2,729 non-terminal nodes and 7,312 other nodes, which are not all leaves of the GO. There is also a considerable amount of unconnected nodes as not all genes are assigned to GO terms and therefore do not form part of the DAG.

Our main challenge was to relate two huge data sets of different nature to each other: a DAG and a binary tree. Both graphs are technically independent from each other as they have different node and edge IDs. However, the graphs share identical labels for terminal nodes (genes), which can be used to compute a mapping: for each interactively selected node in the GO visualization, a corresponding subtree in the Cluster Tree is determined. We use brushing techniques to show the mapping between both and implemented specific representations for the GO DAG and clustering that address the aforementioned challenges. A complete overview of the user interface of our prototype implementation, called CluMa-GO, is shown in Figure 1.

2 Visualization Approach

We took inspiration from pixel-based approaches, which usually cope with large data sets [5]: GO nodes are represented by colored pixels (*node pixels*), whereas edges are hidden to avoid clutter. Edges are shown optionally in case the user selects a particular GO term (non-terminal node) for further exploration. We also implemented a simple edge bundling algorithm to reduce clutter. Red pixels represent leaf and light-blue pixels non-terminal nodes. DAGs can be hierarchically layered and have a "flow direction" as there are no cycles. This allows us to place the nodes into several layers, which provide some insight into the topology of the GO graph as shown on the left hand side of Figure 1. The user can zoom in at the specific layer and scroll up or down between three layers simultaneously if needed.

We have implemented two different layering approaches. The first one (*Levels Layout*) places the leaves (red pixels) and non-terminal nodes (light-blue pixels) into their corresponding layer depending on their graph-theoretic distance [2] from the source node (root). Moreover, leaf nodes are distributed in the left part of their assigned layer; all other nodes are arranged on the right. This feature gives us further insight into the topology of a specific layer by gaining information about the distribution of leaf nodes and non-terminal nodes on a particular layer. Figure 1 shows an example of this layout strategy in the GO view on the left hand side. Although the resulting visualization looks to mimic bar charts, the number of leaves cannot be precisely compared between different layers, as the area the red node pixels (leaves) cover is not proportional to the total number of leaves in each layer. But, it is proportional to the sum of nodes in that particular layer. The spatial arrangement of the node pixels within a layer, except the placing of leaves and non-terminal nodes in specific regions, is random.

Our second layering approach (*Bottom Layout*) is similar to the first one in terms of placing the nodes into corresponding layers based on the distance from the source node and random distribution of the node pixels within each layer. However, all leaves are placed into one single layer at the bottom of the GO view, i. e., in the layer with the highest number. This approach gives insight into the distribution of nodes among different layers without the distraction of the leaves, thus enriching the perception of the graph topology.

The challenge with clustering visualizations is that the application of conventional tree drawing algorithms would produce rather high tree drawings or wide ones. We have noticed that the trees in our data sets are particularly high and unbalanced with not so deep branches (subtrees) and decided to take this disadvantage of typically space-consuming drawings and turn it into an advantage

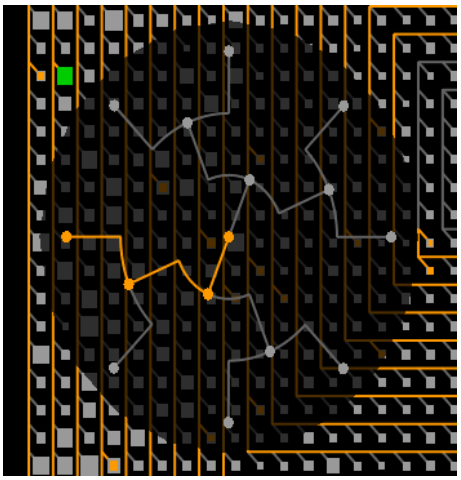

Figure 2: Subtree (branch) view. The more detailed view of the selected branch (*green box glyph*) is visualized as a dendrogram. Circles represent single nodes, whereas a box in the background represents a specific subtree (branch) of the Cluster Tree.

when dealing with large trees of such nature. We decided to use those nodes and edges that form the longest path that connects all branches as a "backbone" for our *Spiral Tree Layout*. We represent this backbone as a spiral, thus preserving space and giving us a possibility to show the complete tree in one view. We implemented this space-filling tree visualization approach, which deals particularly with large unbalanced binary trees. The direction of the flow in the spiral is counter-clockwise from the center towards out as shown in Figure 1 on the right hand side.

The subtrees connected to the backbone are aggregated into box glyphs as the data set is too large. Thus, we allow a specific amount of abstraction in our visualization approach: each small box glyph attached to the spiral in Figure 1 corresponds to one subtree branching out from the backbone with an angle of 135° from the vertical. The size of a box glyph represents the number of nodes of the corresponding subtree. This approach helps to identify interesting patterns of distributions of subtree branches in the clustering. To support a deeper analysis, the user can explore the details of each subtree visualized in the spiral. This is done by clicking on a box glyph. CluMa-GO displays then the half-transparent tree visualization widget with two possible dendrogram layouts: a radial method (see Figure 2) and a so-called HV-drawing method as well.

References

[1] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

[2] J. A. Bondy and U. S. R. Murty. *Graph Theory*, volume 244 of *Graduate Texts in Mathematics*. Springer, 3rd corrected printing edition, 2008.

[3] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[4] The Gene Ontology, last accessed: 2011-08-22. `http://www.geneontology.org`.

[5] D. A. Keim, M. Ankerst, and H.-P. Kriegel. Recursive pattern: A technique for visualizing very large amounts of data. In *Proceedings of the 6th conference on Visualization '95*, VIS '95, pages 279–286. IEEE Computer Society, 1995.

[6] J. C. Roberts. Exploratory visualization with multiple linked views. In A. MacEachren, M.-J. Kraak, and J. Dykes, editors, *Exploring Geovisualization*. Elseviers, December 2004.