Visualisation of Gene Ontology and Cluster Analysis Results

Master's Thesis Report (30 credit points)

Supervisor: Prof. Dr. Andreas Kerren
Student: Vladyslav Aleksakhin

Linneuniversitetet
School of Computer Science, Physics and Mathematics

Contents

Abstract

The task of the Thesis is a new visualization method. Gene 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 over-represented ontology terms, giving insight into relevant biological processes or functional modules. Hierarchical clustering is a standard method to analyze and visualize data to find relatively homogeneous clusters of experimental data points. Both methods support the analysis of the same data set, but are usually considered independently. However, often a combined view is desired: visualizing a large data set in the context of an ontology under consideration of a clustering of the data.

The result of this work is a base for a research paper – Andreas Kerren, Ilir Jusufi, Vladyslav Aleksakhin, and Falk Schreiber. Visualization of Mappings between the Gene Ontology and Cluster Trees. Submitted to the1st IEEE Symposium on Biological Data Visualization (BioVis '11), Providence, RI, USA, 23-24 October 2011 (submitted).

Acknowledgements

I would like to express my gratitude to my supervisors Dr. Andreas Kerren and to co-superviser PhD. Ilir Jusufi. They encouraged and stimulated me in each step of this work.

Moreover, my greatest gratitude goes to my family that made my study possible, for their constant moral support even though thousands of kilometers separated us.

Finally, I would like to thank to my best friend Olga for support and patience.

1 Introduction

1.1 Motivation

Computer information analysis is the only way to work with huge amount of data produced today. In any filed of human work there are data and there are tasks of analysing it. The topic of the Thesis combined two areas of data computing: information visualisation and biological data processing.

Data for processing was provided by Plant Bioinformatics Group of Leibniz Institute of Plant Genetics and Crop Plant Research (IPK), Germany. Data consists of Gene Ontologies and hierarchical clustering, which are are both important tools in biology and medicine to study high-throughput data.

The aim of this work is to provide new visualization approach and implement it to provide useful tool for biologist in their everyday research work.

1.2 Report structure

Section 1 presents the results of the literature review, where been exploring relevant research efforts in the fields of Bioinformatic, Information Visualisation and Bio Visualization, in connection to the work. In Section 2 explains the problem and purpose of this work. Section is split in several parts which covers topics: thesis purpose and collaboration, visualisation complexity and goals, description of the input data, data format overview, overview of different graph file formats. In this section, Problem Statement and Goals, have determined the requirements, use cases, and proposed the architecture for thesis application. First part of the Section 3 describes visualisation solution for Cluster Ananlisys tree, second part explains Gene Ontology visualisation technique. On of the biggest section in the report is Section **??** which covers implementation details, architecture of the system, used libraries, project management tool, etc. Technical details of the visualisation algorithms are explained in the Sections 3.1 and 3.2. Last part of the report is Section 5 describes problems that been faced during work and future improvements.

1.3 Information Visualization

The field of computer-based information visualization draws on ideas from several intellectual traditions: computer science, psychology, semiotics, graphic design, cartography, and art. The two main threads of computer science relevant for visualization are computer graphics and human-computer interaction. The areas of cognitive and perceptual psychology offer important scientific guidance on how humans perceive visual information. A related conceptual framework from the humanities is semiotics, the study of symbols and how they convey meaning. Design, as the name suggests, is about the process of creating artifacts well- suited for their intended purpose. Cartographers have a long history of creating visual representations that are carefully chosen abstractions of the real world. Finally, artists have refined methods for conveying visual meaning in subdisciplines ranging from painting to cinematography.

Information visualization has gradually emerged over the past fifteen years as a distinct field with its own research agenda. The distillation of results from areas with other goals into specific prescriptive advice that can help us design and evaluate visualization systems is nontrivial. Although these traditions have much to offer, effective synthesis of such knowledge into a useful methodology of our own requires arduous gleaning.

The standard argument for visualization is that exploiting visual processing can help people explore or explain data. We have an active field of study because the design challenges are significant and not fully understood. Questions about visual encoding are even more central to information visualization than to scientific visualization. The subfield names grew out of an accident of history, and have some slightly unfortunate con- notations when juxtaposed: information visualization is not unscientific, and scientific visualization is not uninformative. The distinction between the two still not agreed on by all, but the definition used here is that information visualization hinges on finding a spatial mapping of data that is not inherently spatial, whereas scientific visualization uses a spatial layout that is implicit in the data.

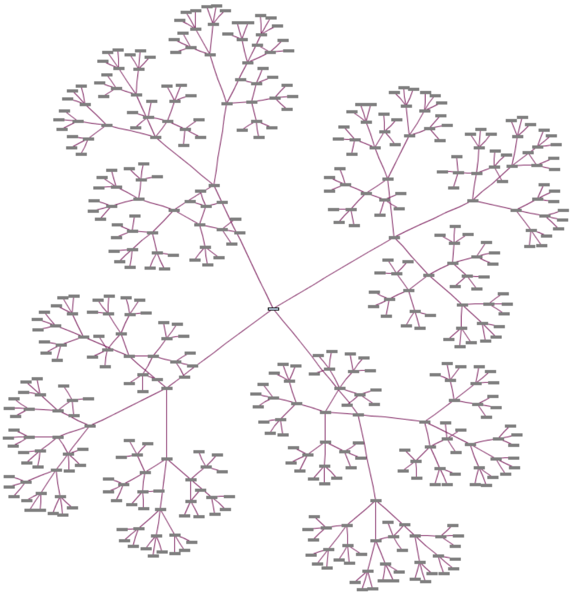


Figure 1: Sample visualisation of the tree graph

"Graph drawing or Graph layout, as a branch of graph theory, applies topology and geometry to derive two-dimensional representations of graphs. A drawing of a graph is basically a pictorial representation of an embedding of the graph in the plane (generally, with edge intersections allowed), usually aimed at a convenient visualization of certain properties of the graph in question or of the object modeled by the graph. Graph drawing is motivated by applications such as Very-large-scale (VLSI) [1] integration circuit design , social network analysis, cartography, and bioinformatics, many of which make use of information visualization." [2]

There are lots of methods and approaches to graph visualization. They are all based on various perception qualities of human. The Radial Dendrograms [3] algorithm is used for results of cluster analysis visualization.

Dendrogram [4] plot is one of the visualisation algorythms for hierarchical structures. It illustrating the outcome of decision tree-type clustering in statistics. Most commonly, dendrograms are drawn in a Cartesian layout, as an upright tree. However, this layout does not make good use of space, it is sparse towards the root and crowded towards the leaf nodes. The spacing between nodes at different levels in the hierarchy is not uniform, which is due to the shrinking number of nodes from bottom to top. For this reason, long, wide-spanning connecting lines are needed to merge nodes at higher levels. A better layout in this respect is the polar or radial layout, where leaf nodes are located on the outer ring and the root is located in the center, as a focal point. A more uniform node spacing results, leading to a better utilization of space and resulting in a better illustration of the class relationships. Recently, Barlow and Neville [5] presented an empirical user study for tree layouts (with less than 200 leaves) in which they compare some of the major schemes: organizational chart (a standard drawing of a tree), tree ring (basically a pie chart of circular segments), icicle plot (the cartesian version of the tree ring), and tree map. According to the measured performance within a group of 15 users, the three former methods yielded similar results, with the icicle plot having a slight advantage. However, given the much larger number of leaves in our case (1000 and more) and the fact that the tree ring is the most compact of the three winning configurations, a radial layout seemed to be the most favorable one for our purpose. Radial graph layouts that illustrate hierarchical relationships are very popular, and for the special application of dendrograms, we know only of one other application using a radial layout, the recent one by Kreussler and Schumann [6] that mentioned before, users often would like to focus on certain portions of the display, while compressing others, without losing context. Fisheye lenses and hyperbolic zooming have been proposed to provide these capabilities. In the context of tree rings Yang, Ward and Rundensteiner [7] have proposed a system in which users may either perform a polar zoom (i.e. expand the width of one or more adjacent rings while reducing others) or a radial zoom (i.e. expand the arc angle of some adjacent segments while reducing others). Users can perform these operations by pinning down one ring or arc segment and dragging another. A limiting factor here is that users cannot perform both operations simultaneously, which can be awkward in certain instances. To address this shortcoming, our application generalizes these concepts by allowing arbitrary warps of the dendrogram domain, i.e. we allow radial and polar zooms simultaneously.

1.4 Bioinformatic

"In the last few decades, advances in molecular biology and the equipment available for research in this field have allowed the increasingly rapid sequencing of large portions of the genomes of several species. In fact, to date, several bacterial genomes, as well as those of some simple eukaryotes (e.g., Saccharomyces cerevisiae, or baker's yeast) have been sequenced in full. The Human Genome Project, designed to sequence all 24 of the human chromosomes, is also progressing. Popular sequence databases, such as GenBank and EMBL, have been growing at exponential rates. This deluge of information has necessitated the careful storage, organization and indexing of sequence information. Information science has been applied to biology to produce the field called Bioinformatics.

The simplest tasks used in bioinformatics concern the creation and maintenance of databases of biological information. Nucleic acid sequences (and the protein sequences derived from them) comprise the majority of such databases. While the storage and or organization of millions of nucleotides is far from trivial, designing a database and developing an interface whereby researchers can both access existing information and submit new entries is only the beginning. The most pressing tasks in bioinformatics involve the analysis of sequence information" [8]

Here we can find short introduction and history of Bioinformatics: "Bioinformatics is the application of information technology to the field of molecular biology. The term bioinformatics was coined by Paulien Hogeweg in 1978 for the study of informatic processes in biotic systems. Bioinformatics now entails the creation and advancement of databases, algorithms, computational and statistical techniques, and theory to solve formal and practical problems arising from the management and analysis of biological data. Over the past few decades rapid developments in genomic and other molecular research technologies and developments in information technologies have combined to produce a tremendous amount of information related to molecular biology. It is the name given to these mathematical and computing approaches used to glean understanding of biological processes. Common activities in bioinformatics include mapping and analysing DNA and protein sequences, aligning different DNA and protein sequences to compare them and creating and viewing 3-D models of protein structures." [9] The primary goal of bioinformatics is to increase our understanding of biological processes. What sets it apart from other approaches, however, is its focus on developing and applying computationally intensive techniques (e.g., data mining, machine learning algorithms, and visualization) to achieve this goal. Major research efforts in the field include sequence alignment, gene finding, genome assembly, protein structure alignment, protein structure prediction, prediction of gene expression and protein-protein interactions, genome-wide association studies and the modelling of evolution.

1.5 Gene Ontology

As stated above, there are different knowledge databases for biologic information storage. Gene Ontology [10] project is one of the first-rate international projects. The Gene Ontology, or GO, is a major bioinformatics initiative to unify the representation of gene and gene product attributes across all species. The aims of the Gene Ontology project are threefold; firstly, to maintain and further develop its controlled vocabulary of gene and gene product attributes; secondly, to annotate genes and gene products, and assimilate and disseminate annotation data; and thirdly, to provide tools to facilitate access to all aspects of the data provided by the Gene Ontology project. The GO is part of a larger classification effort, the Open Biomedical Ontologies (OBO) [11]. The Gene Ontology project provides an ontology of defined terms representing gene product properties. The ontology covers three domains; cellular component, the parts of a cell or its extracellular environment; molecular function, the elemental activities of a gene product at the molecular level, such as binding or catalysis; and biological process, operations or sets of molecular events with a defined beginning and end, pertinent to the functioning of integrated living units: cells, tissues, organs, and organisms. Each GO term within the ontology has a term name, which may be a word or string of words; a unique alphanumeric identifier;

a definition with cited sources; and a name space indicating the domain to which it belongs. Terms may also have synonyms, which are classed as being exactly equivalent to the term name, broader, narrower, or related; references to equivalent concepts in other databases; and comments on term meaning or usage. The GO ontology is structured as a directed acyclic graph, and each term has defined relationships to one or more other terms in the same domain, and sometimes to other domains. The GO vocabulary is designed to be species-neutral, and includes terms applicable to prokaryotes and eukaryotes, single and multicellular organisms. The GO ontology is not static, and additions, corrections and alterations are suggested by, and solicited from, members of the research and annotation communities, as well as by those directly involved in the GO project. For example, an annotator may request a specific term to represent a metabolic pathway, or a section of the ontology may be revised with the help of community experts. Suggested edits are reviewed by the ontology editors, and implemented where appropriate.

1.6 Clustering

"Data clustering (or just clustering), also called cluster analysis, segmentation analysis, taxonomy analysis, or unsupervised classification, is a method of creating groups of objects, or clusters, in such a way that objects in one cluster are very similar and objects in different clusters are quite distinct. Data clustering is often confused with classification, in which objects are assigned to predefined classes. In data clustering, the classes are also to be defined." [12]

Clustering algorithms can be applied in many fields, for instance:

- Marketing: finding groups of customers with similar behaviour given a large database of customer data containing their properties and past buying records;
- Biology: classification of plants and animals given their features;
- Libraries: book ordering;
- Insurance: identifying groups of motor insurance policy holders with a high average claim cost; identifying frauds;
- City-planning: identifying groups of houses according to their house type, value and geographical location;
- Earthquake studies: clustering observed earthquake epicentres to identify dangerous zones;
- WWW: document classification; clustering web log data to discover groups of similar access patterns.

2 Problem Statement and Goals

The aim of the tool is to discover relations in separated datasets (Gene Ontology graph and cluster analysis result tree), visualize and allow interactive selection relations selection.

First step in the program work is loading data and to compute connected components. Data is stored in GML file format. More detailed information about this format and other graph file formats is in the section 2.2.

Here is program algorithm explanation using sample graphs:

1. The program visualizes Gene Ontology and cluster analysis result tree. Visualization technic is discussed in the section 3.

2. Interactively select node in the Gene Ontology graph (Figure 2).

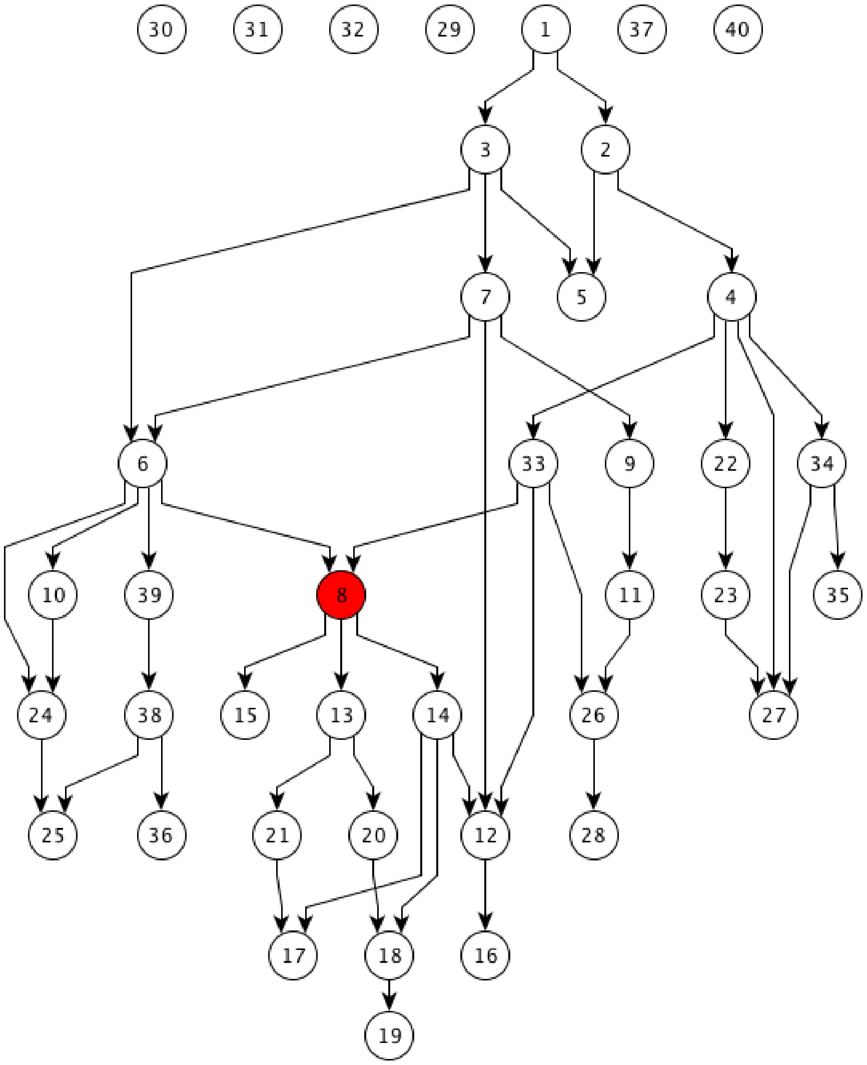


Figure 2: Selected node in the Gene Ontology

3. When node is selected in the Gene Ontology program computes all successors (Figure 3).

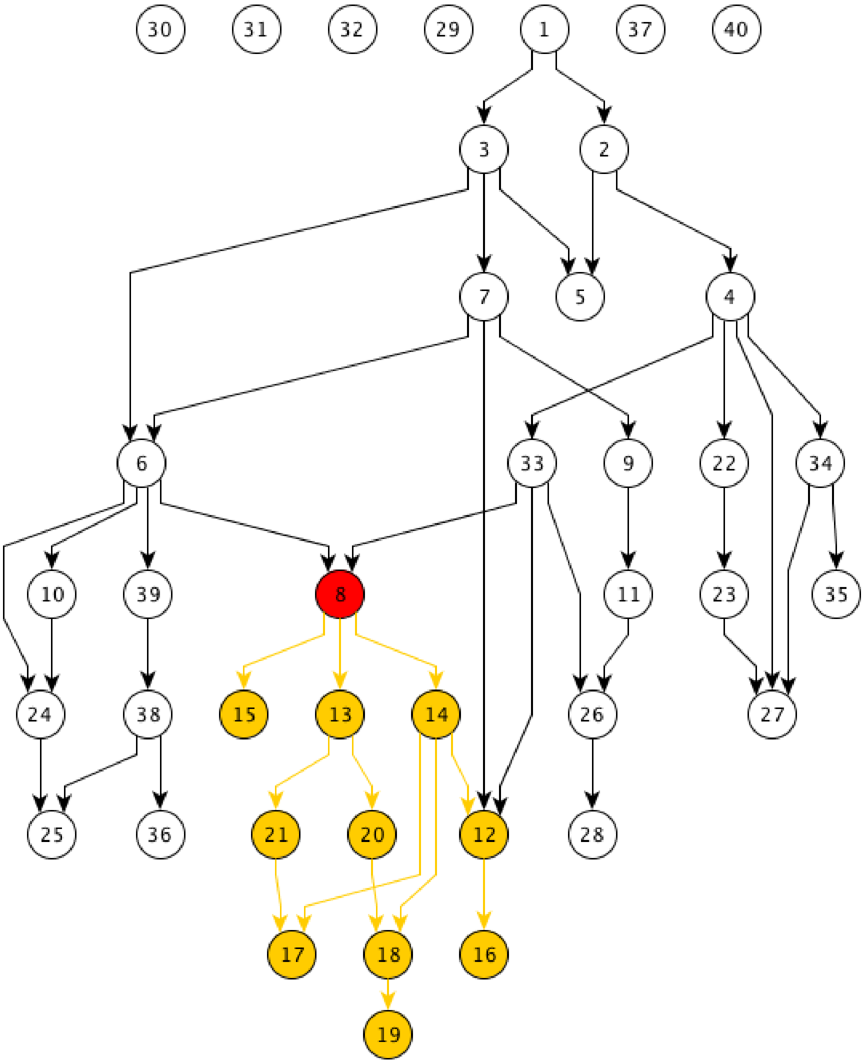


Figure 3: Corresponded successors of the selected node

4. Extract leafs from successors (Figure 4).

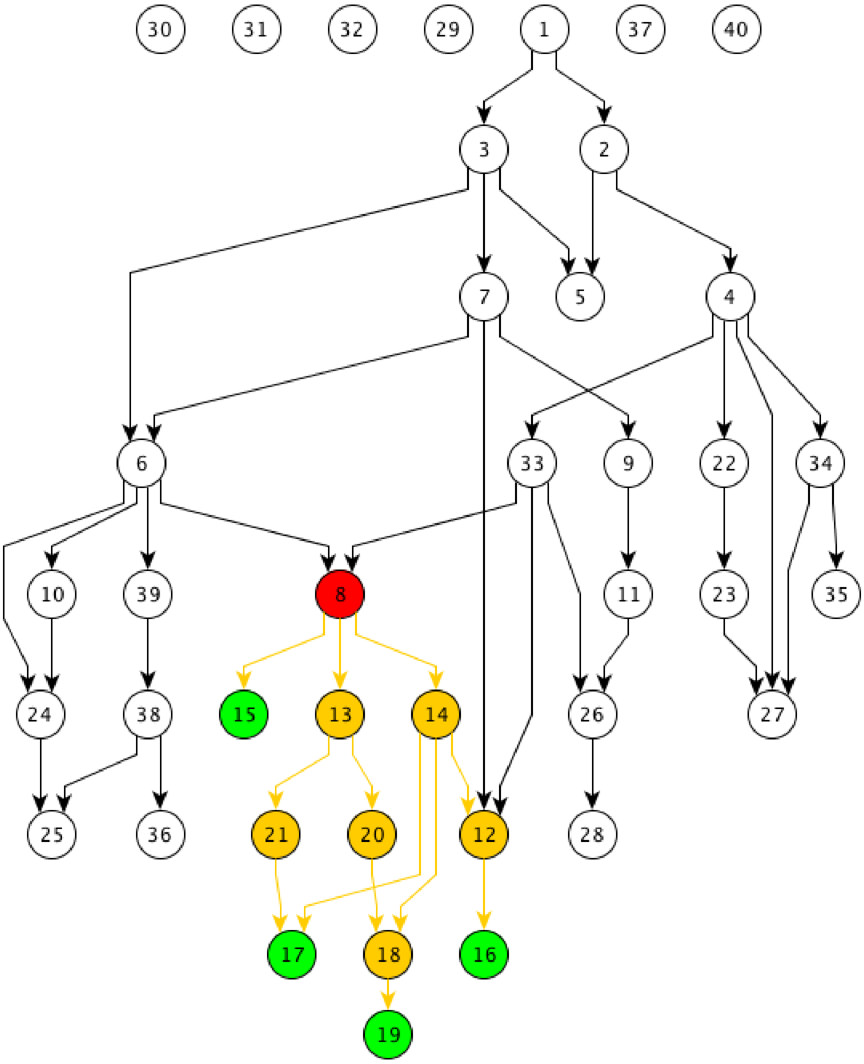


Figure 4: Extract leafs

5. Then the program searches corresponded leaves in cluster analysis result tree by label as seen on the Figure 5.

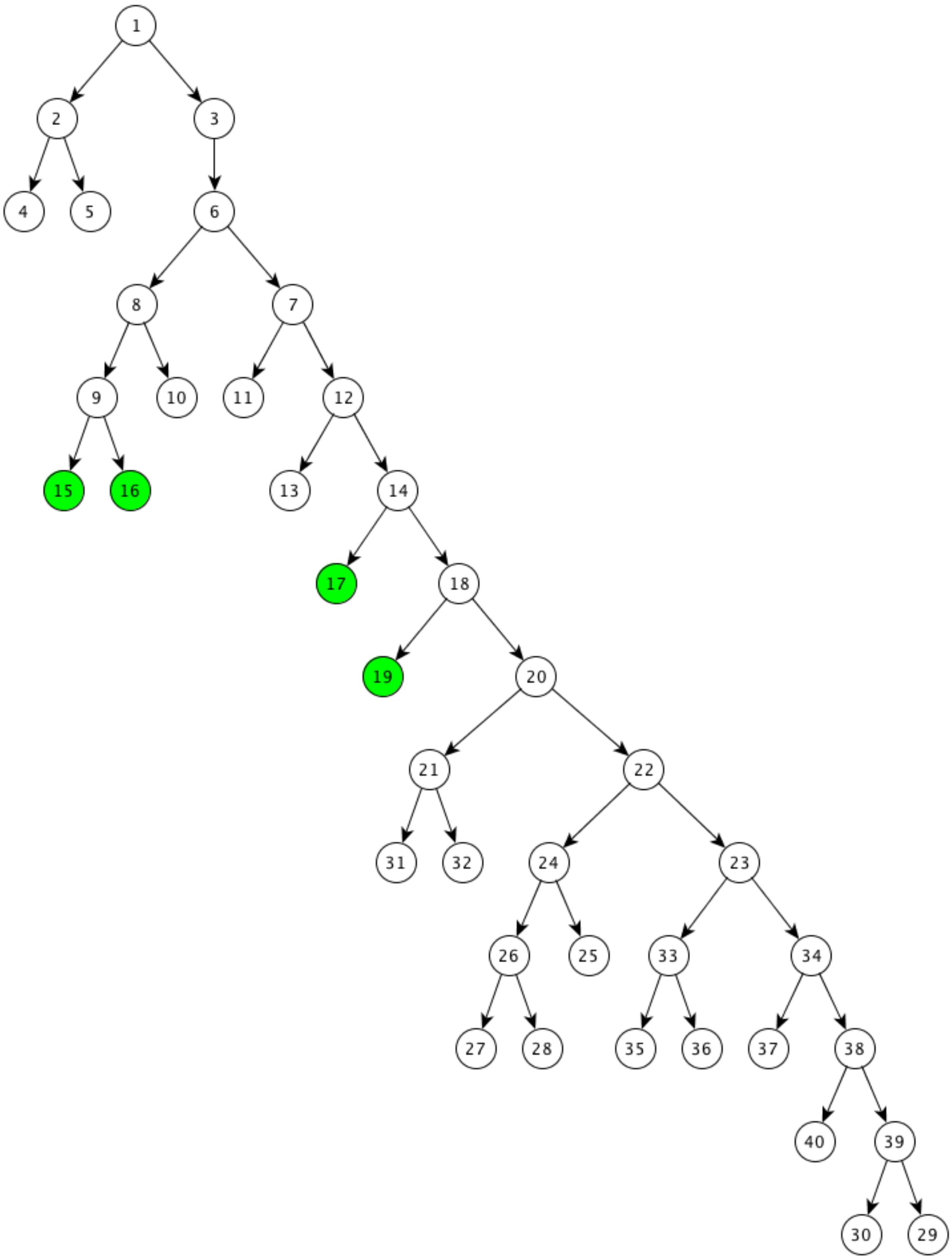


Figure 5: Corresponded leaves in the cluster tree

6. For this leaves the program founds root connected to all leaves and extract corresponding sub trees (Figure 6).

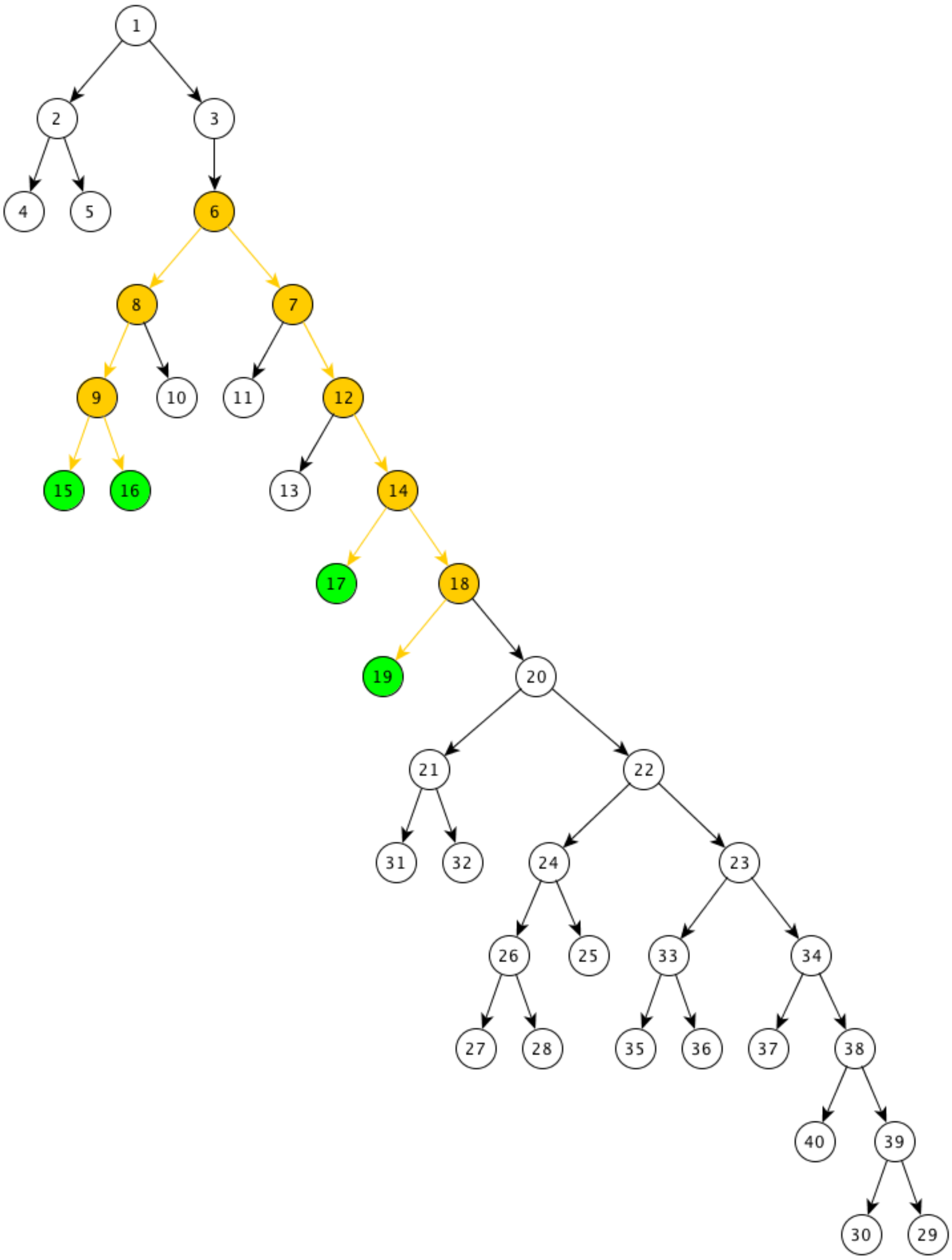


Figure 6: Build sub tree

7. Founded sub tree cached. This sub tree is highlighted in cluster analysis tree.

2.1 Thesis Collaboration

There are many datasets for analysis in biology. This work is intended for making a visualization tool for genes and gene relations, in order to help biologists with their work with genes. This project is result of collaboration between ISOVIS research group (Head: Prof. Dr. Andreas Kerren [13]) of Linneuniversitetet at Växjö, Sweden, and Plant Bioinformatics Group (Head: Prof. Dr. Falk Schreiber [14]) of Leibniz Institute of Plant Genetics and Crop Plant Research (IPK), Germany.

About the ISOVIS group: "The ISOVIS group mainly focuses on the exploration analysis and visualization of typically large information spaces, for example, in Software Engineering, Geography, or Biology. Another research topic is the use of visualizations in educational software, especially for computer science education. Hereby, we focus on so-called human-centred visualization techniques and approaches: Human-Centred Visualization combines traditional visualization techniques with the ability of the human visual-brain system and/or the haptic-motoric system to explore and analyse complex data sets comprehensively. This kind of visualization merges several aspects of different research areas, such as Information Visualization, Scientific Visualization, Human-Computer Interaction, Data Mining, Information Design, Graph Drawing, and Computer Graphics. From all sub fields in visualization, we mainly focus on Information Visualization which centres on the visualization of abstract data, e.g., hierarchical, networked, or symbolic information sources, in order to help users understand and analyse such data.

For most practical applications, researchers try to find the best visual representation of the given information. That is the core problem of each visualization but sometimes the seemingly best representation does not suffice if the human information processing and the human capability of information reception are not adequately taken into account. Additionally, these aspects depend on the data to be visualized and on the user's background. While the development of human-centred visualization tools, user abilities and requirements, visualization tasks, tool functions, and visual representations should be equally taken into account. The design of such tools is one of the large challenges of Information Visualization, Software Visualization, and of many application areas, such as the visualization of biological/biochemical or geographical information." [15]

As said on official page of Plant Bioinformatic Group: The research group focuses on modelling, analysis, simulation and visualisation of biological networks in the context of plant biological problems. Our aim is the development of methods and software tools for the analysis of complex biological networks. Therefore we integrate, process and analyse data from different areas of genome, proteome and metabolome research and present the results in a user-friendly way. The emphasis is on the linkage of experimental data about expression profiles and metabo- lite patterns with metabolic and regulatory networks. The data and complex connections are modelled using graphs. We are developing graph (network) analysis and interactive visualisation methods to discover network properties and to make the data easily accessible to the user. A subsequent step is to use the data for the simulation of metabolic and regulatory networks. [16]

2.2 Dataset Description

Gene Ontology data and cluster analysis results presented as directed graphs. They are stored in separate files in special format – GML files. GML file format covered in the next section.

GO graph is directed acyclic graph has 10042 vertices and 24155 edges. It has 1 root, 2729 nodes and 7312 leafs (terminal nodes). In the Figure 7 showed visualization of the Gene Ontology graph using yEd [20] graph editor using Hierarchical layout. It is obviously shows imperfectness of classic visualization of the graphs.

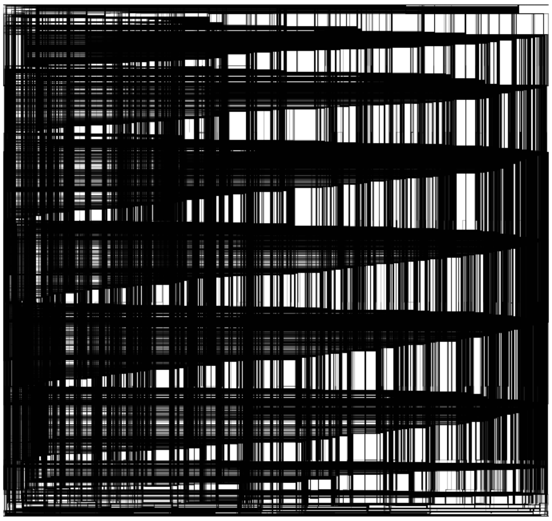


Figure 7: Gene Ontology yEd visualization

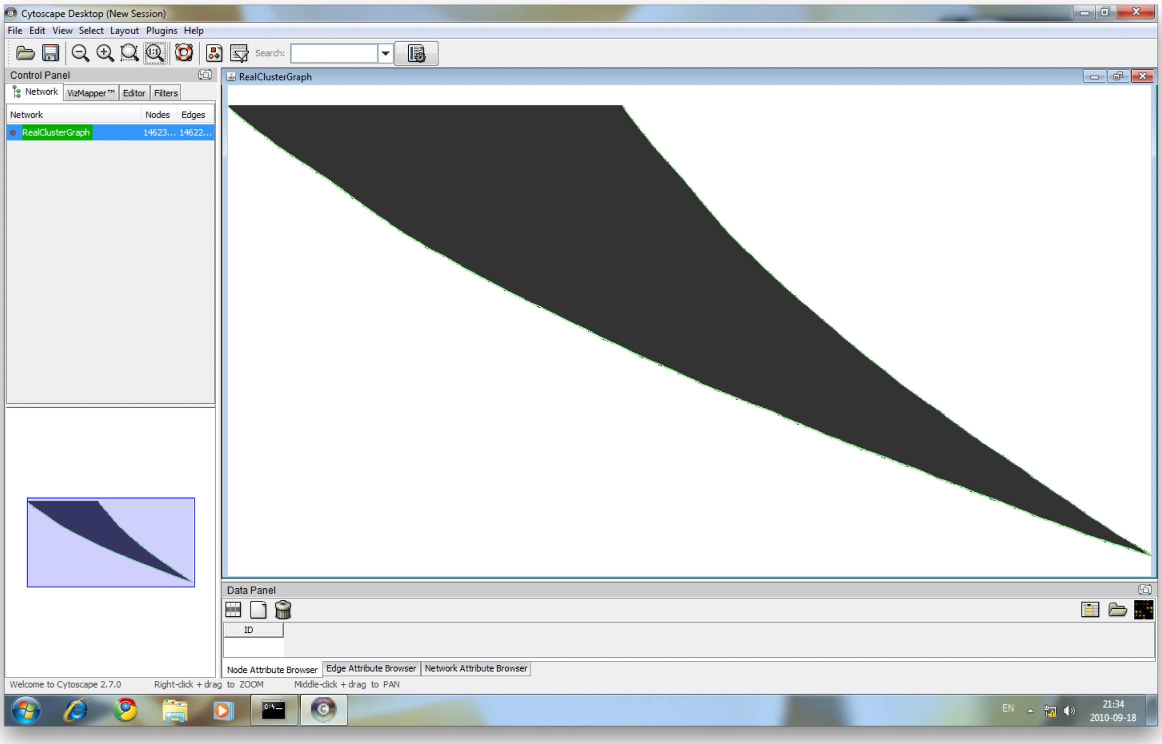
Cluster graph is directed binary tree has 14623 nodes and 14622. Cluster graph as a tree has 1 root, 7310 nodes and 7312 leafs. To get an impression of the graph on the Figure **??** is visualization of the cluster tree using Cytoscape [22] visualization tool.

Both of two graphs are independent from each other from developer point of view: they have different node ids and edge ids. But they are corresponded by graph node labels – both of this graphs have same label for terminal (leaf) graph nodes. This property is used in the sub graph extracting algorithm.

The application should work with large quantity of data over tens of thousands that is why performance is one of the main requirements. It is important to give a consideration on optimization.

2.3 GML Graph File Format

"GML, the Graph Modelling Language, is our proposal for a portable file format for graphs. GML's key features are portability, simple syntax, extensibility and flexibility. A GML file consists of a hierarchical key-value lists. Graphs can be annotated with arbitrary data structures. The idea for a common file format was born at the GD'95; this proposal is the outcome of many discussions. GML is the standard file format in the Graphlet citeGraphlet graph editor system. It has been overtaken and adapted by several other systems for drawing graphs." [18]


Cytoscape Desktop (New Session)
File Edit View Select Layout Plugins Help
Search:
Control Panel
Network
VizMapper™
Editor
Filters
RealClusterGraph
Data Panel
ID
Node Attribute Browser
Edge Attribute Browser
Network Attribute Browser
Welcome to Cytoscape 2.7.0
Right-click + drag to ZOOM
Middle-click + drag to PAN
EN
21:34
2010-09-18

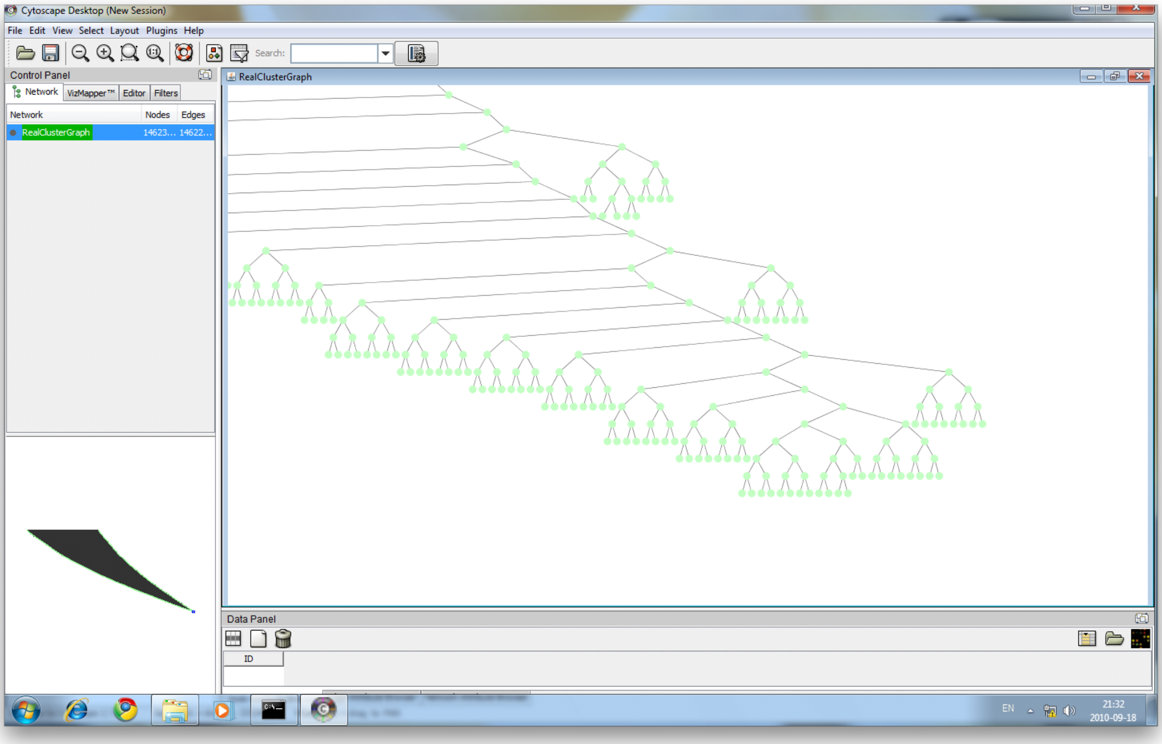
Network	Nodes	Edges
RealClusterGraph	14623...	14622...

Figure 8: Cluster analysis result tree Cytoscape visualization tool

GML format is platform independent, and easy to implement. Furthermore, it has the capability to represent arbitrary data structures, since advanced programs have the need to attach their specific data to nodes and edges. GML is flexible enough that a specific order of declarations is not needed, and that any non-essential data may be omitted. Simple graph is showed in the Listing 1

Listing 1: GML description of sample graph

```
graph [
  directed 1
  node [
    id 1
  ]
  node [
    id 2
  ]
  node [
    id 3
  ]
  edge [
    source 1
        target 2
```



Cytoscape Desktop (New Session)
File Edit View Select Layout Plugins Help
Search:
Control Panel
Network VizMapper™ Editor Filters
RealClusterGraph
Data Panel
ID
EN
21:32
2010-09-18

Network	Nodes	Edges
RealClusterGraph	14623...	14622...

Figure 9: Zoomed cluster analysis result tree

```
    ]
  edge [
    source 2
    target 3
  ]
  edge [
    source 3
    target 1
  ]
]
```

In the Figure 10 showed manual visualization of the graph using yEd [?]

More complex graph with additional properties and its visualization produced by yEd graph editor is in the Appendix A and the visualization of this graph showed on the Figure 11

Applications supporting GML [19]

- Clairlib [21], a suite of open-source Perl modules intended to simplify a number of generic tasks in natural language processing (NLP), information retrieval (IR), and network analysis (NA).
- Cytoscape [22], an open source bioinformatics software platform for visualizing molecular interaction networks, loads and save previously-constructed

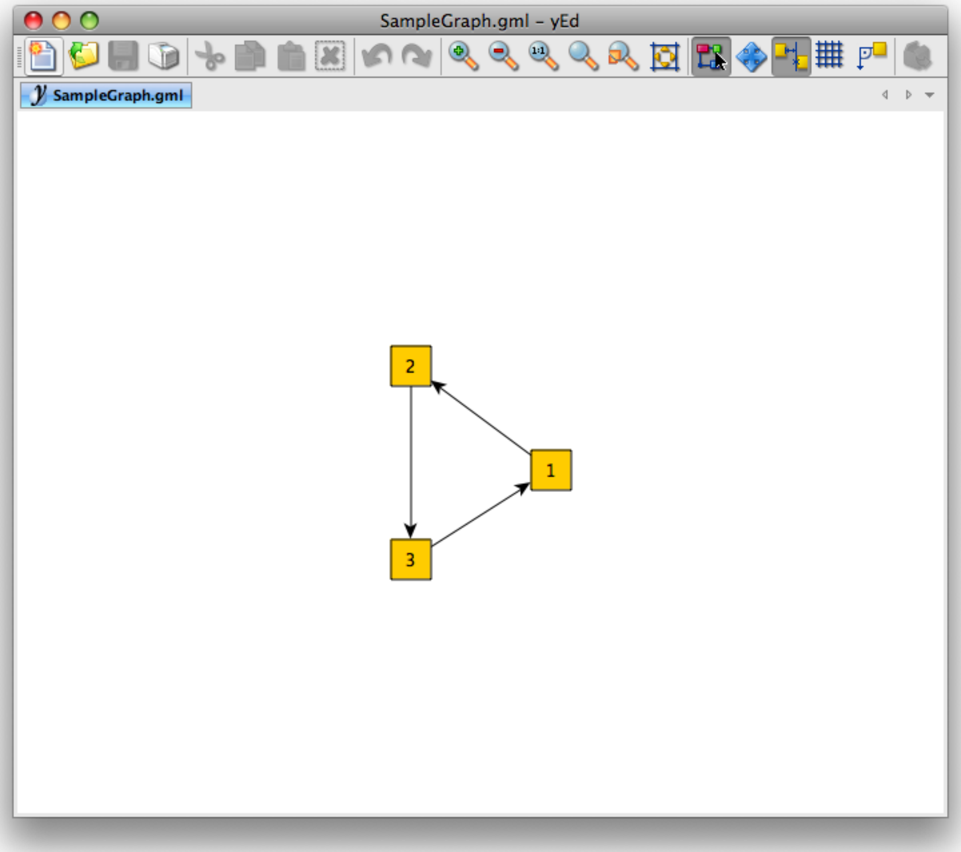


Figure 10: Manual visualization of the sample graph

interaction networks in GML.

- NetworkX [23], an open source Python library for studying complex graphs.
- ocamlgraph[24], a graph library for OCaml.
- OGDF[25], the Open Graph Drawing Framework, an open source C++ library containing implementations of various graph drawing algorithms. The library is self contained; optionally, additional packages like LP-solvers are required for some implementations.
- Tulip [26] (software) is a free software in the domain of information visualisation capable of manipulating huge graphs (with more than 1.000.000 elements).
- yEd [20], a free Java-based graph editor, supports import from and export to GML.

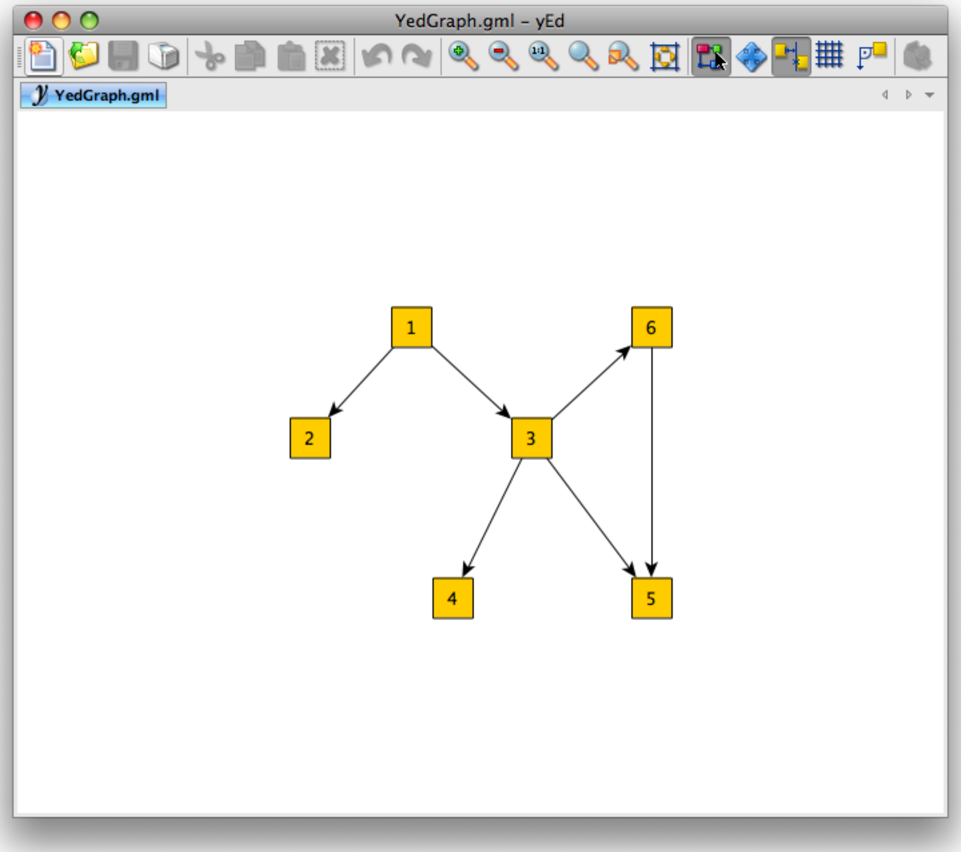


Figure 11: Manual visualization of the sample graph

2.4 Other Graph File Formats

2.4.1 GraphML

GraphML is a comprehensive and easy-to-use file format for graphs. It consists of a language core to describe the structural properties of a graph and a flexible extension mechanism to add application-specific data. [27] Its main features include support of:

- directed, undirected, and mixed graphs;
- hyper graphs;
- hierarchical graphs;
- graphical representations;
- references to external data;
- application-specific attribute data;

- light-weight parsers;

The GraphML document consists of a graphml element and a variety of sub elements: graph, node, edge. In the Figure 12 below is a simple graph. It contains 11 nodes and 12 undirected edges.

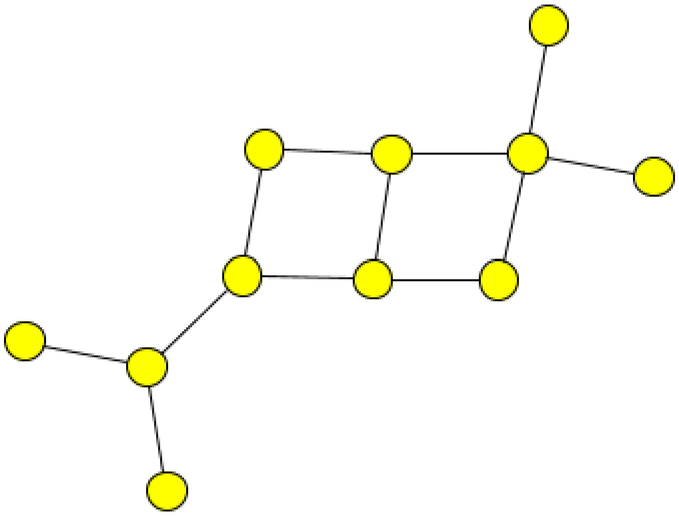


Figure 12: A simple graph

And corresponded graphml file showed in the Listing 2

Listing 2: Simple graphml file

```
<graphml>
  <graph id="G" edgedefault="undirected">
    <node id="n0"/>
    <node id="n1"/>
    <node id="n2"/>
    <node id="n3"/>
    <node id="n4"/>
    <node id="n5"/>
    <node id="n6"/>
    <node id="n7"/>
    <node id="n8"/>
    <node id="n9"/>
    <node id="n10"/>
    <edge source="n0" target="n2"/>
    <edge source="n1" target="n2"/>
    <edge source="n2" target="n3"/>
    <edge source="n3" target="n5"/>
    <edge source="n3" target="n4"/>
    <edge source="n4" target="n6"/>
    <edge source=”n6” target=”n5”/>
    <edge source=”n5” target=”n7”/>
    <edge source=”n6” target=”n8”/>
    <edge source=”n8” target=”n7”/>
    <edge source=”n8” target=”n9”/>
    <edge source=”n8” target=”n10”/>
  </graph>
</graphml>
```

2.4.2 DOT graph file format

”DOT is a plain text graph description language. It is a simple way of describing graphs that both humans and computer programs can use. DOT graphs are typically files that end with the .gv (or .dot) extension.

At its simplest, DOT can be used to describe an undirected graph. An undirected graph shows simple relations between objects, such as friendship between people. The graph keyword is used to begin a new graph, and nodes are described within curly braces. A double-hyphen (—) is used to show relations between the nodes.” [28]

Listing 3: DOT file format: undirected graph

```
graph graphname {
        a -- b -- c;
        b -- d;
}
```

”Similar to undirected graphs, DOT can describe directed graphs, such as flowcharts and dependency trees. The syntax is the same as for undirected graphs, except the digraph keyword is used to begin the graph, and an arrow (− >) is used to show relationships between nodes.” [28]

Listing 4: DOT file format: directed graph

```
graph graphname {
        a -> b -> c;
        b -> d;
}
```

Visual representation of both graphs showed in the Figure 13 below.

2.4.3 DGML

DGML is an XML-based file format for directed graphs. Here is what a simple directed graph with three nodes and two links between them looks like

Listing 5: DGML file format

```
<?xml version='1.0' encoding='utf-8'?>
<DirectedGraph xmlns=”http://schemas.microsoft.com/vs/2009/dgml”>
```

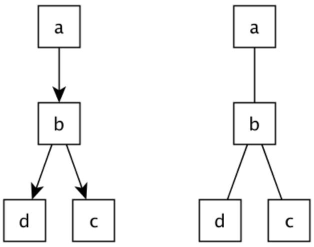


Figure 13: Directed and undirected graphs

```
  <Nodes>
    <Node Id="a" Label="a" Size="10" />
    <Node Id="b" Background="#FF008080" Label="b" />
    <Node Id="c" Label="c" Start="2010-06-10" />
  </Nodes>
  <Links>
    <Link Source="a" Target="b" />
    <Link Source="a" Target="c" />
  </Links>
  <Properties>
    <Property Id="Background" Label="Background" DataType="Brush" />
    <Property Id="Label" Label="Label" DataType="String" />
    <Property Id="Size" DataType="String" />
    <Property Id="Start" DataType="DateTime" />
  </Properties>
</DirectedGraph>
```

The complete XSD schema for DGML is available at `http://schemas.microsoft.com/vs/2009/dgml/`. DGML not only allows describing nodes and links in a graph, but also annotating those nodes and links with any user defined property and/or category.

2.4.4 GXL

"GXL (Graph eXchange Language) is a standard format for exchanging graph-based data. It is the culmination of a cooperative effort among an international group of researchers from disparate areas, including software reengineering and graph transformation. Researchers and tool builders have had a growing interest in comparing and combining approaches to their respective problems and leveraging each other's results. These collaborations provide lessons learned that are critical to advancing the maturity of the discipline. A standard exchange format for data facilitates tool interoperability and allows them to use a "best of breed" approach when building a workbench.

Interoperability is the challenge of enabling tools from different suppliers to work together. Wasserman [1] describes a taxonomy with five types of interoperability: platform, presentation, data, control, and process. Another model from Earl has three levels: control, user interface, and data. Data interoperability appears in both of them.

Data interoperability requires the data to be compatible both syntactically and semantically. In other words, tools need to agree on both the format and the meaning of this data. The graph-based data model of GXL can be used to represent both instance data and schemas.

Thus, GXL provides a standardized notation for exchanging instance data (graphs) including their structure (graph schemas). Both instance and schema graphs are encoded using the same XML (eXtensible Markup Language) DTD (Document Type Definition). While these schema graphs do not provide semantics, they serve as a basis for users to agree upon semantics. This feature is important because it helps tools and researchers communicate about the assumptions inherent in their approaches. This increased mutual understanding is a critical step in building on each other's work to increase the impact of research results.

In addition to being a generic format for representing graph structures, GXL is also suitable for object-relational data. Consequently, GXL can be used to represent data from a wider range of applications, including as data repositories and fact bases from reengineering tools." [29]

2.4.5 SVG

"SVG has been in development since 1999 by a group of companies within the W3C after the competing standards Precision Graphics Markup Language (PGML) developed from Adobe's PostScript and Vector Markup Language (VML) developed from Microsoft's RTF were submitted to W3C in 1998. SVG drew on experience from the designs of both those formats.

SVG allows three types of graphic objects:

- Vector graphics
- Raster graphics
- Text

Graphical objects, including PNG and JPEG raster images, can be grouped, styled, transformed, and composited into previously rendered objects. SVG does not directly support z-indices[5] that separate drawing order from document order for overlapping objects, unlike some other vector mark up languages like VML. Text can be in any XML name space suitable to the application, which enhances search ability and accessibility of the SVG graphics. The feature set includes nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility.

Since 2001, the SVG specification has been updated to version 1.1 (current Recommendation) and 1.2 (still a Working Draft). The SVG Mobile Recommendation introduced two simplified profiles of SVG 1.1, SVG Basic and SVG Tiny, meant for devices with reduced computational and display capabilities. SVG Tiny later became an autonomous Recommendation (current version 1.2) and the basis for SVG 1.2. In addition to these variants and profiles, the SVG Print specification (still a Working Draft) contains guidelines for printable SVG 1.2 and SVG Tiny 1.2 documents. The Canvas element in HTML 5 provides an approach to rendering dynamic graphics in HTML that's procedural rather than declarative: instead of specifying the shapes to draw in XML, the author executes drawing commands from a script. Canvas doesn't allow for static rendering, and drawn elements are not identifiable in a DOM-like way." [30]

3 Solution

3.1 Cluster Analysis Results Visualization

As was discussed before in the Section 2.2 cluster analysis result graph is binary tree.

"A binary tree is a connected acyclic graph such that the degree of each vertex is no more than 3. A rooted binary tree is such a graph that has one of its vertices of degree no more than 2 singled out as the root. With the root thus chosen, each vertex will have a uniquely defined parent, and up to two children; however, so far there is insufficient information to distinguish a left or right child. If we drop the connectedness requirement, allowing multiple connected component in the graph, we call such a structure a forest" [33] Simple binary tree showed on the Figure 14

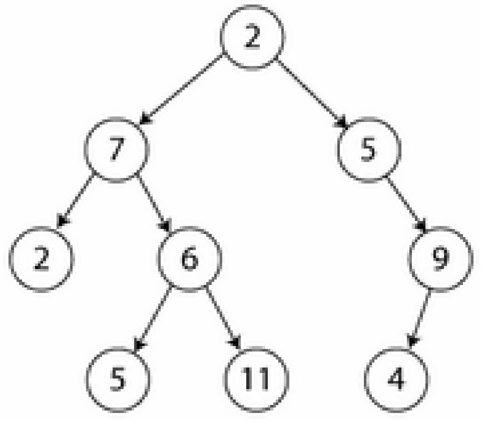
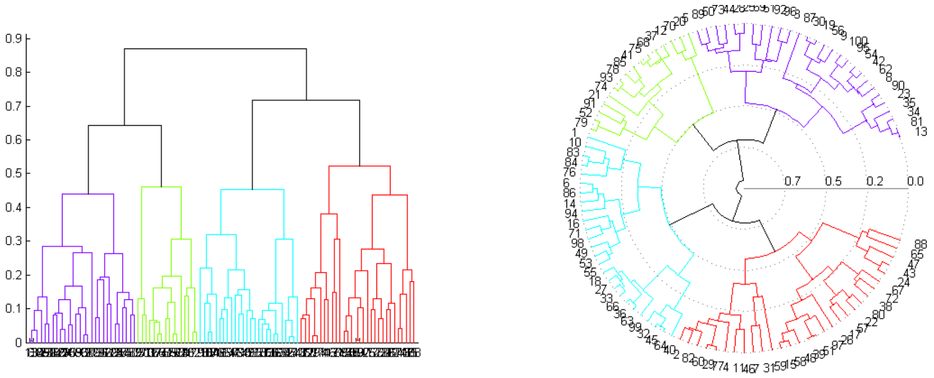


Figure 14: A simple binary tree graph

There several visualization methods for binary trees and more specific methods for cluster result. The main method for visualizing clusters is - dendrogram. Here is sample dendrogram visualization on the Figure 15a produced by MATLAB 7.2.

The Figure 15b shows polar dendrogram visualization algorithm of the same cluster tree produced by MATLAB.

One of the main ideas was to use polar dendrogram algorithm for cluster visualization. The Figure 16 shows visualization of the Cluster using native JUNG radial layout algorithm. Red are nodes and white is edges, black is background. As picture shows algorithm doesn't count nature of the cluster -



(a) Dendrogram

(b) Polar Dendrogram

Figure 15: Cluster visualisations

very deep binary tree not wide as it is common for cluster analysis results, that's why it has many edge everlappings.

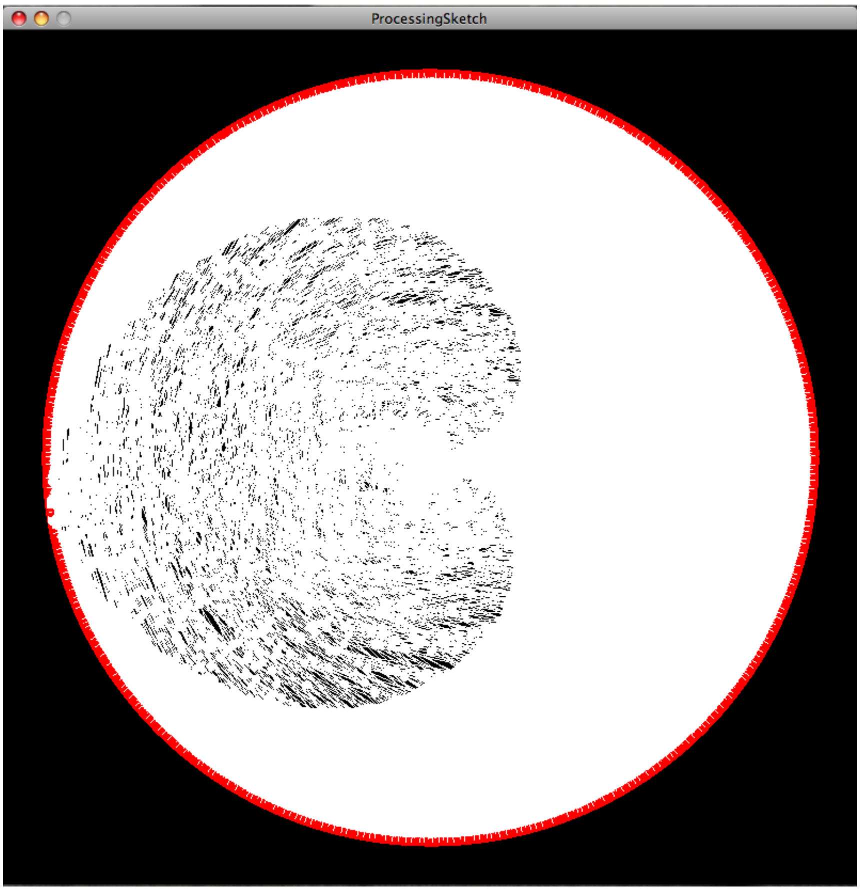


Figure 16: Cluster visualization using JUNG radial layout

Also to visualization issue it has performance issue – without any measurement was seen for an eye that program does not allow smooth interaction. Low performance issue was in nature of the visualization in the JUNG: it uses very complex hierarchical structure with many utility classes per visualized object and the cost is big memory usage. Also JUNG uses Java 2D [34] which by itself is "heavyweight" because it's part of the Java AWT – Abstract Windows Toolkit.

"The Abstract Window Toolkit (AWT) is Java's original platform-independent windowing, graphics, and user-interface widget toolkit. The AWT is now part of the Java Foundation Classes (JFC) the standard API for providing a graphical user interface (GUI) for a Java program. When Sun Microsystems first released Java in 1995, AWT widgets provided a thin level of abstraction over the underlying native user interface. For example, creating an AWT check box would cause AWT directly to call the underlying native subroutine that created a check box." [35] This technology is outdated and replaced by Swing.

"Swing is the primary Java GUI widget toolkit. It is part of Sun Microsystems' Java Foundation Classes (JFC) an API for providing a graphical user interface (GUI) for Java programs. Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit. Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform." [36]

"Since early versions of Java, a portion of the Abstract Window Toolkit (AWT) has provided platform-independent APIs for user interface components. In AWT, each component is rendered and controlled by a native peer component specific to the underlying windowing system. By contrast, Swing components are often described as lightweight because they do not require allocation of native resources in the operating system's windowing toolkit. The AWT components are referred to as heavyweight components." [36] More detail comparison can be found here. [37]

The Figure 17 shows improved JUNG radial algorithm and own visualization implementation using JOGL. JOGL will be discussed in the Section 4.2.

The Figure 18a and Figure 18b shows cluster visualization and highlighted subgraph using algorithm that was discussed in the Section 2. This pictures show the nature of the dataset. Improved version has good performance and better visualization but still has issues - there are too many elements in the scene, it is impossible to identify separate gene or trace highlighted graph genes.

As was showed before in the Figure **??** cluster graph structure is specific it is very high and unbalanced with not so deep sub-parts. It is possible to use this disadvantage as advantage and abstract sub-parts to reduce drawing area. Extract those nodes and edges that form the longest path of the cluster graph - ,,backbone". Backbone vertices are filled with yellow and showed on the Figure 20. Next step is to abstract branches as a for our Spiral Tree Layout implementation. We represent this backbone as a spiral, thus preserving space and giving us a possibility to show the complete tree in one view The sub-parts are drawn as rectangles, the rectangle size represents amount of vertices inside. Figure 19 shows how this approach works.

Then backbone formed as rectangular spiral with a root in the center and moving in clockwise direction. Figure 20 shows complete visualisation result. This will have to reuse space as much as possible and still gives overview of

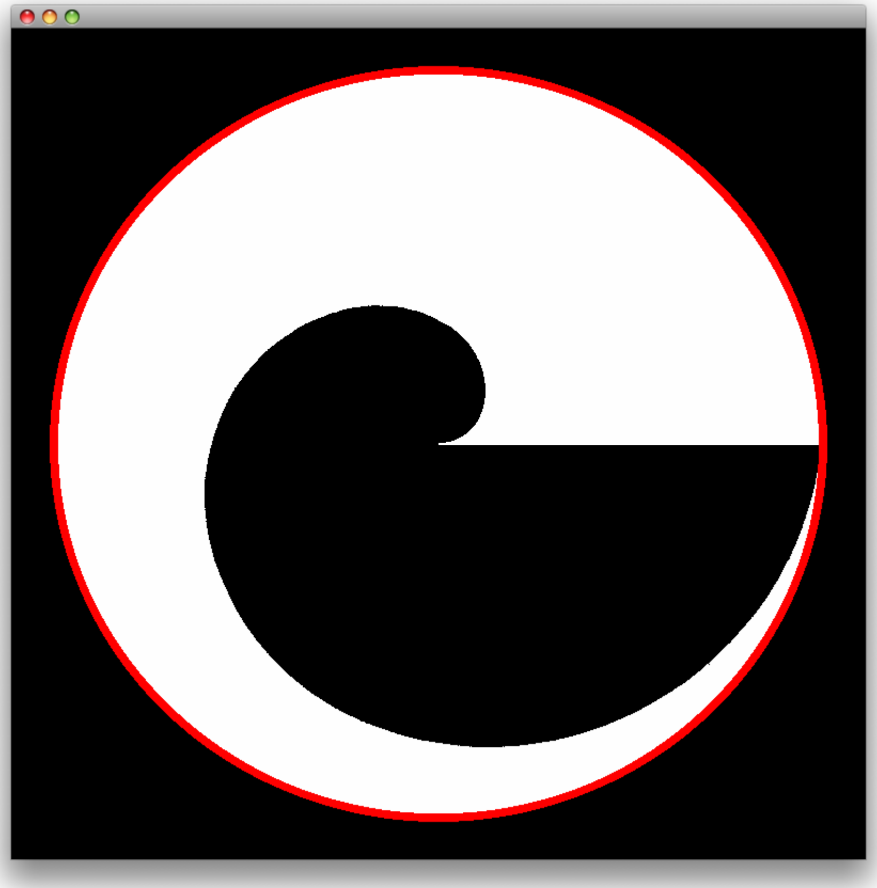


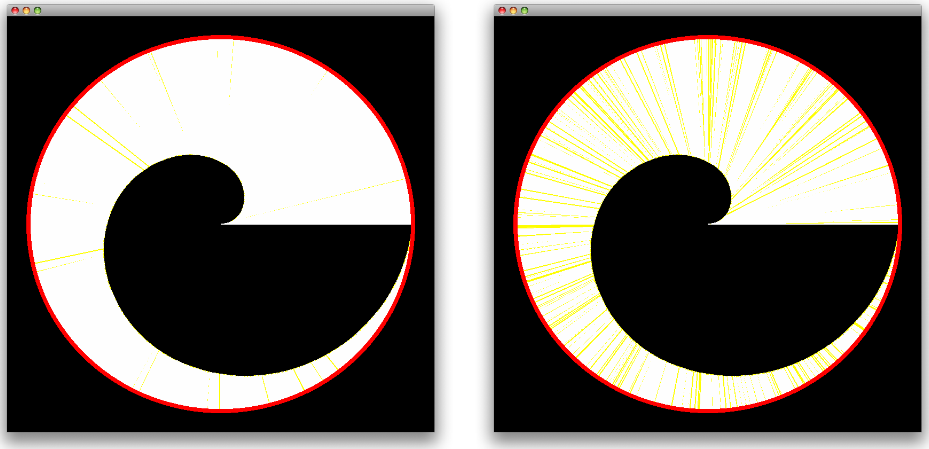
Figure 17: Cluster visualization using JOGL and improved JUNG radial layout

location of the highlighted vertices in cluster hierarchy – how far from a root.

It is possible to explore sub-part (rectangles) of the Cluster graph using lens. User can interactively choose any sub-part and lens with inned content will appear. There are two different lens layouts: polar 21a and HV-tree 21b. Both implementations are own made and are not depend of any external library.

3.2 Gene Ontology Visualization

There are two visualisation implementations for Gene Ontology. The first layering approach is called Levels Layout and places the leaves (red pixels) and nodes (white pixels) into their corresponding layer depending on their graph-theoretic distance from the 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 22 shows an example of this layout strategy in the GO view on the left hand side, whereas Figure 23 displays the situation if the user zooms in



(a) Cluster graph and highlighted subgraph

(b) Cluster graph and highlighted subgraph

Figure 18: Cluster graph visualisation using improved JUNG polar dendrogram layout

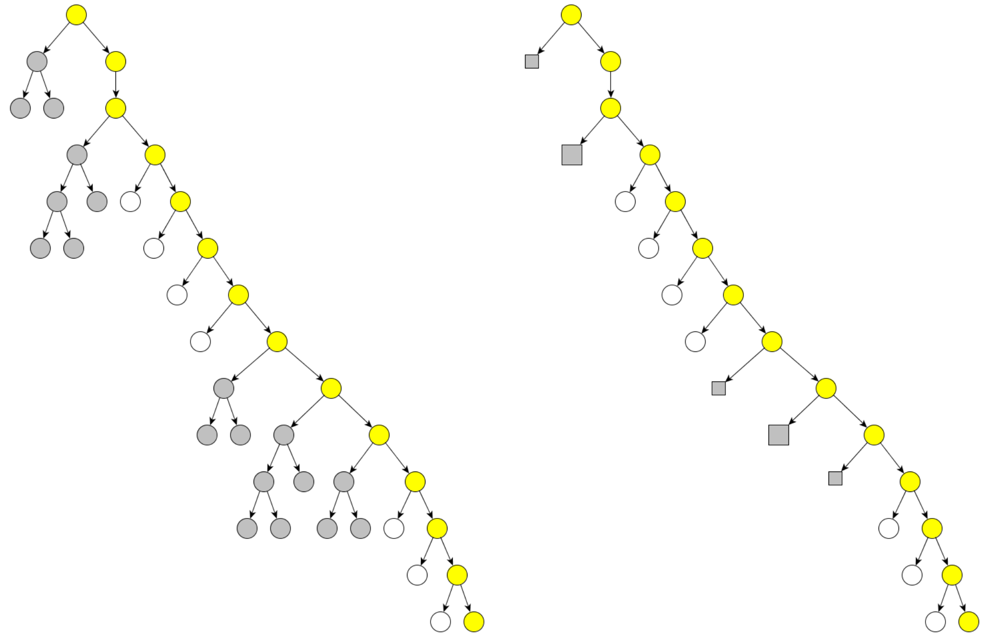


Figure 19: Cluster graph sub-parts abstraction

the view. 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. In other words, the covered area depends on the specific layer density.

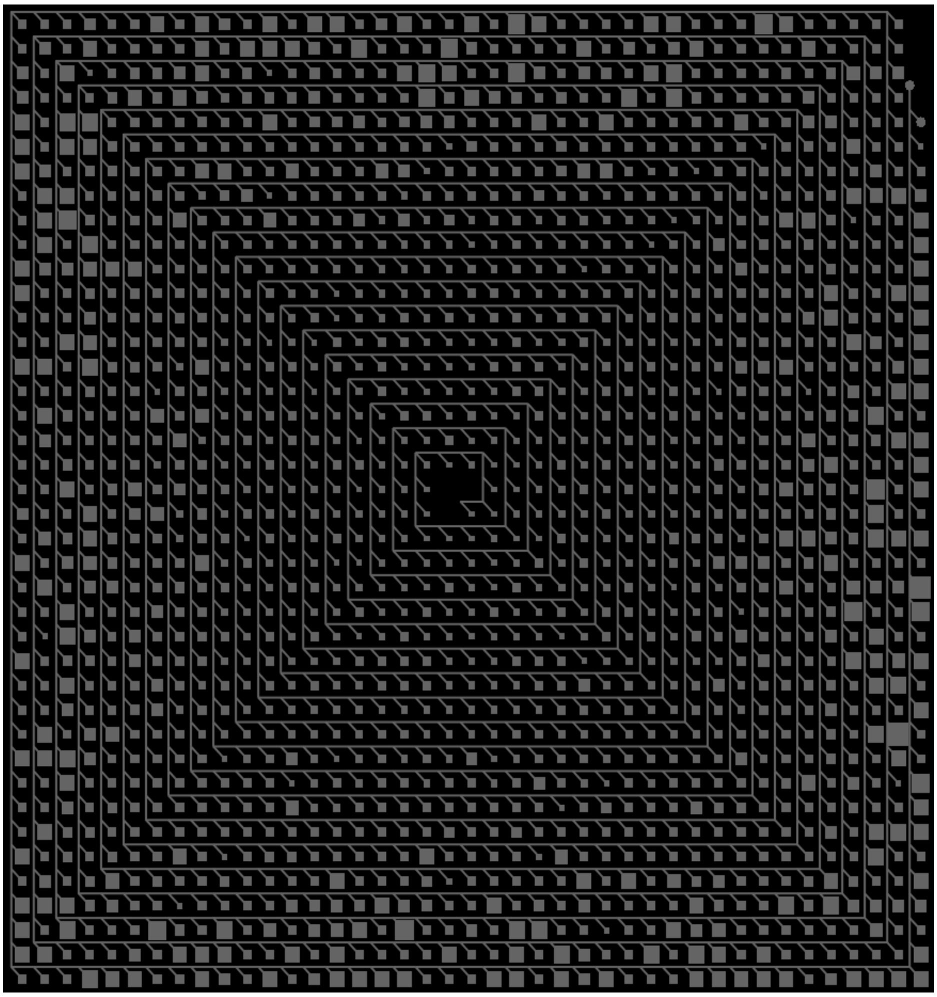
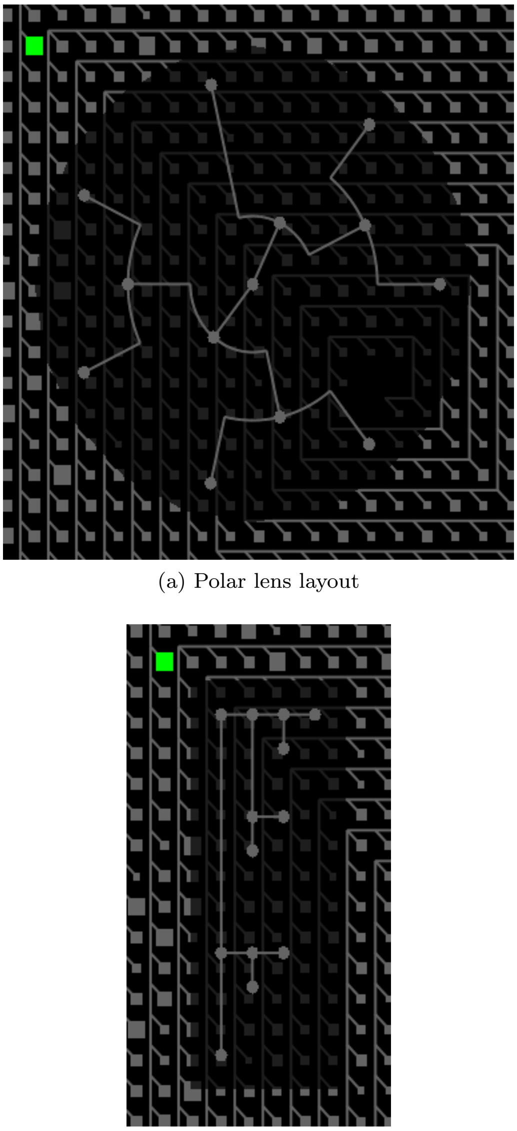


Figure 20: Rectangular spiral Cluster graph visualisation

There are unconnected components in the Gene Ontology graph. Unconnected nodes are placed in the top layer number – zero. There is an option to show-hide unconnected components from the main menu. The spatial arrangement of the node pixels within a layer, except the placing of leaves and nodes in specific regions, is random.

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 together with unconnected nodes at the bottom of the GO view, i. e., in the layer with the highest number. Unconnected nodes can be filtered out if necessary. 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. More screenshots can be found in the Appendix C.



(a) Polar lens layout

(b) HV-tree lens layout

Figure 21: Lens layouts

4 Implementation

Main language and platform for developing thesis is Java. "Java is a programming language and computing platform first released by Sun Microsystems in 1995. It is the underlying technology that powers state-of-the-art programs including utilities, games, and business applications. Java runs on more than 850 million personal computers worldwide, and on billions of devices worldwide, including mobile and TV devices." [32] Also, there are a lot of different libraries in Java. Every "common" part of the system could be replaceable. In the future sections would be detailed overview libraries for graphs, graph visualizations

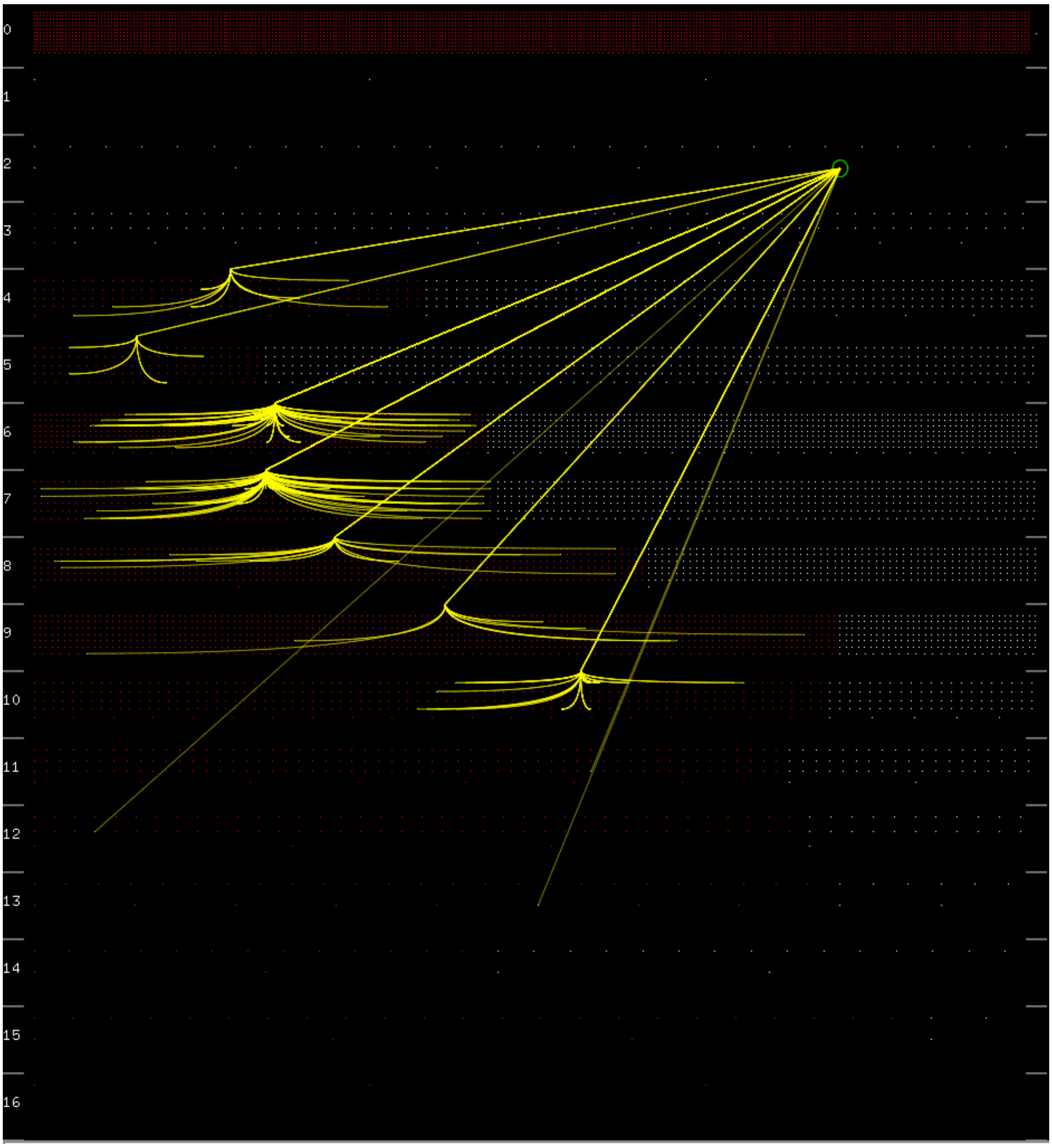


Figure 22: Levels Layout visualisation with sub-graph edges

and graphic libraries.

Java is flexible platform which has big amount of different libraries. It helps not to write twice things already made but flexibility cause project structure complexity and library management. On the early stage there are no problems to control project throw sophisticated IDE (integrated development environment) but as project complexity grows more powerful building tool need appears. Maven is used during thesis work.

Maven [38] is free, open-source and de facto project management standard on

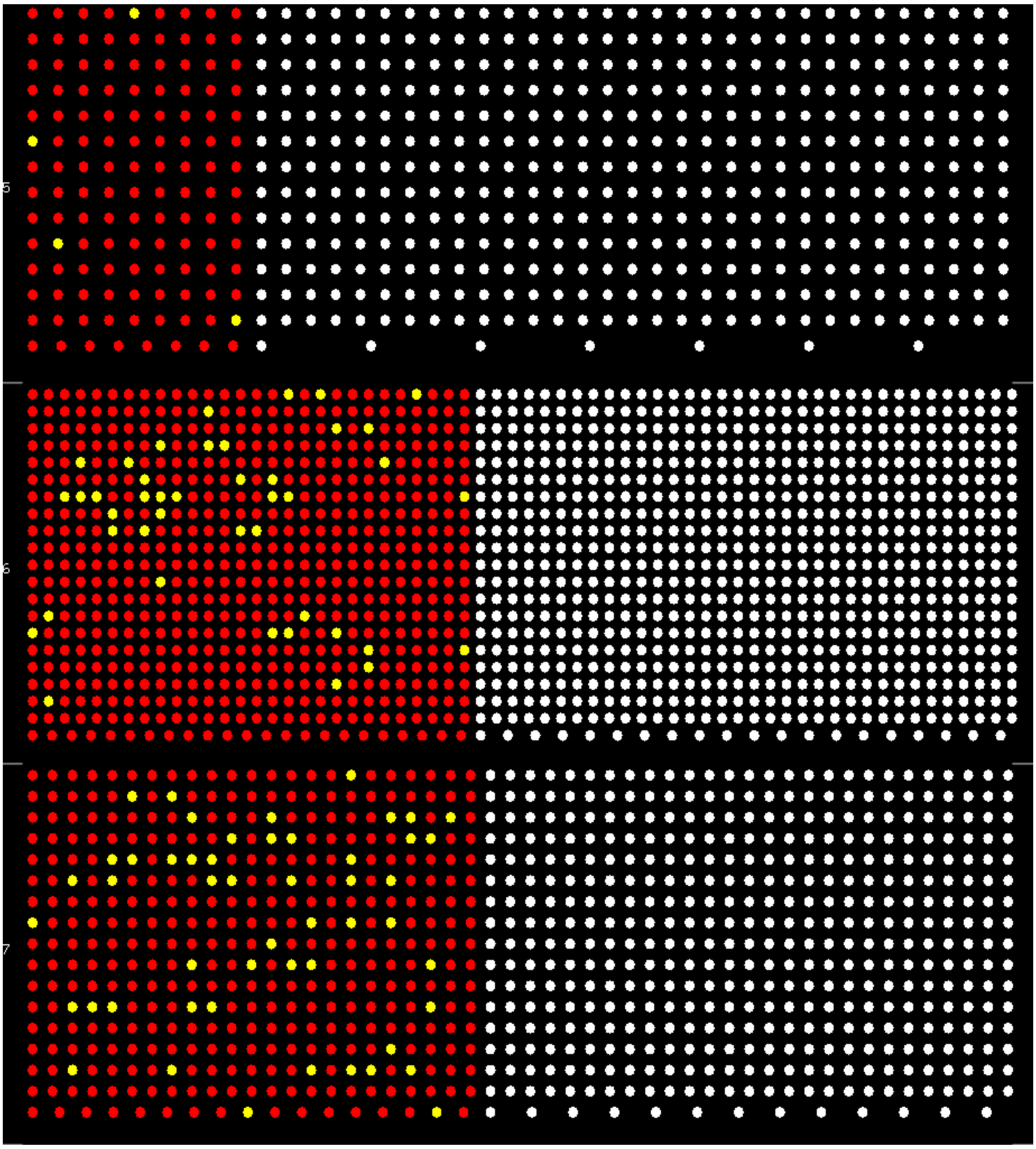


Figure 23: Zoomed view

the Java platform, is part of Apache software project developed and supported by ASF (Apache Software Foundation) [39]. "Maven provides a comprehensive approach to managing software projects. From compilation, to distribution, to documentation, to team collaboration, Maven provides the necessary abstractions that encourage reuse and take much of the work out of project builds." [40]

"Maven is a project management framework, but this doesn't tell you much about Maven. It's the most obvious three-word definition of Maven the authors could come up with, but the term project management framework is a meaning-less abstraction that doesn't do justice to the richness and complexity of Maven. Too often technologists rely on abstract phrases to capture complex topics in three or four words, and with repetition phrases such as project management and enterprise software start to lose concrete meaning.

When someone wants to know what Maven is, they will usually ask What exactly is Maven?, and they expect a short, sound-bite answer. Well it is a build tool or a scripting framework Maven is more than three boring, uninspiring words. It is a combination of ideas, standards, and software, and it is impossible to distill the definition of Maven to simply digested sound-bites. Revolutionary ideas are often difficult to convey with words. If you are interested in a fuller, richer definition of Maven read this introduction; it will prime you for the concepts that are to follow.

If Maven isn't a project management framework". what is it? Here's one attempt at a description: Maven is a set of standards, a repository format, and a piece of software used to manage and describe projects. It defines a standard life cycle for building, testing, and deploying project artifacts. It provides a framework that enables easy reuse of common build logic for all projects following Maven's standards. The Maven project at the Apache Software Foundation is an open source community which produces software tools that understand a common declarative Project Object Model (POM). This book focuses on the core tool produced by the Maven project, Maven 2, a framework that greatly simplifies the process of managing a software project.

You may have been expecting a more straightforward answer. Perhaps you picked up this book because someone told you that Maven is a build tool. Don't worry, Maven can be the build tool you are looking for, and many developers who have approached Maven as another build tool have come away with a finely tuned build system. While you are free to use Maven as just another build tool, to view it in such limited terms is akin to saying that a web browser is nothing more than a tool that reads hypertext.

Maven and the technologies related to the Maven project are beginning to have a transformative effect on the Java community. In addition to solving straightforward, first-order problems such as simplifying builds, documentation, distribution, and the deployment process, Maven also brings with it some compelling second-order benefits.

As more and more projects and products adopt Maven as a foundation for project management, it becomes easier to build relationships between projects and to build systems that navigate and report on these relationships. Maven's standard formats enable a sort of "Semantic Web" for programming projects. Maven's standards and central repository have defined a new naming system for projects. Using Maven has made it easier to add external dependencies and publish your own components.

So to answer the original question: Maven is many things to many people. It is a set of standards and an approach as much as it is a piece of software. It is a way of approaching a set of software as a collection of interdependent components which can be described in a common format. It is the next step in the evolution of how individuals and organizations collaborate to create software systems. Once you get up to speed on the fundamentals of Maven, you will wonder how you ever developed without it." [41]

Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal there are several areas of concern that Maven attempts to deal with:

- making the build process easy
- providing a uniform build system
- providing quality project information
- providing guidelines for best practices development
- allowing transparent migration to new features

On the Figure 24 showed structure of thesis project folder. Project uses stardard Maven project folder format.

Here is overview of key parts: java source code stored in the "src/main/java" folder, tests source code stored in the "src/test/java". Necessary resources such as log4j configuration file stored in the "src/main/resources", in the same folder thesis properties file is stored. During "package" phase all resources are copied by Maven.

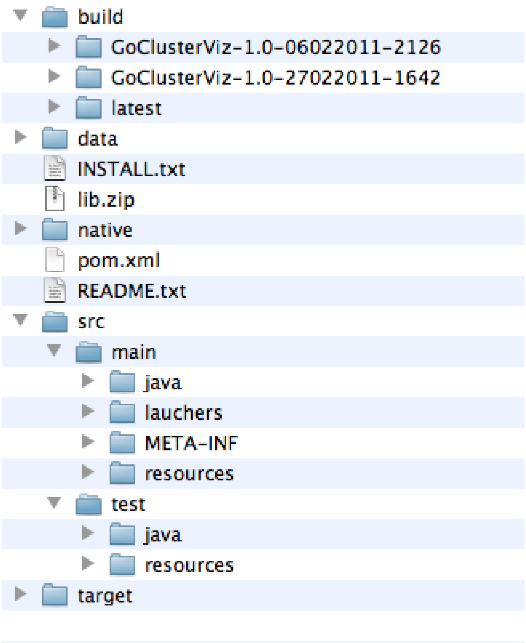
"Native" directory stores JOGL native libraries to different platform such as Linux, Windows and Mac OS X. In the "lib.zip" stored jars for JUNG library which should be manually placed in the local Maven repository because they do not exist in central Maven repository. All other dependencies are exist in Maven repositories and handled by default dependency process.

Complete builds are stored in the "build" folder. Latest build stored in the "build/latest" folder, other builds stored in the folder that has such name pattern:`GoClusterViz-%version%-%build date%-%build time%`. Each build folder contains corresponded native build for each of three platforms Linux, Windows and Mac OS X.

Cluster graph and Gene Ontology graphs stored in the "data" folder.

Second necessary part of the normal development process is revision control. "Revision control, also known as version control or source control (and an aspect of software configuration management or SCM), is the management of changes to documents, programs, and other information stored as computer files. It is most commonly used in software development, where a team of people may change the same files. Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or simply "revision". For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged." [42]

During thesis development was used Git version control system. "Git is a distributed revision control system with an emphasis on speed. Git was initially designed and developed by Linus Torvalds for Linux kernel development. Every Git working directory is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server. Git's current software maintenance is overseen by Junio Hamano. Git is free software distributed under the terms of the GNU General Public License version 2. after he claimed that Andrew Tridgell had reverse-engineered the BitKeeper protocols.) Torvalds wanted a distributed system that he could use like BitKeeper, but none of the available free systems met his needs, particularly his performance needs." [43] Git allows to store and work local on



```
build
  GoClusterViz-1.0-06022011-2126
  GoClusterViz-1.0-27022011-1642
  latest
data
INSTALL.txt
lib.zip
native
pom.xml
README.txt
src
  main
    java
    lauchers
    META-INF
    resources
  test
    java
    resources
target
```

Figure 24: Thesis folder structure

machine but for more safety reasons all source code stored on GitHub. GitHub is a web-based hosting service for software development projects that use the Git revision control system. GitHub provides free hosting open-source project. Thesis project home page on GitHub is `http://github.com/vadyalex/thesis` and source code url is `git://github.com/vadyalex/thesis.git`

4.1 Java Graph Libraries Overview

There are overview of a few libraries for working with graphs in Java

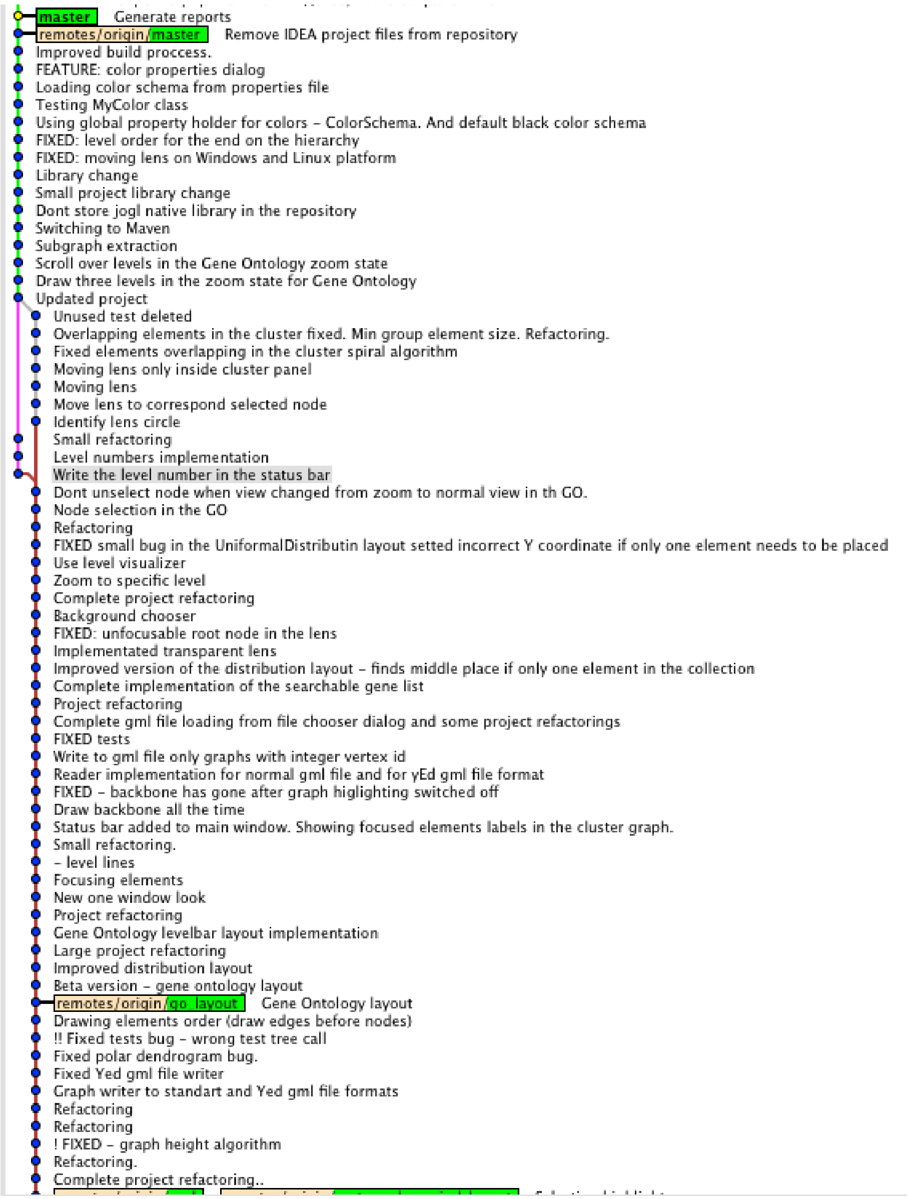
1. Java Graph Editing Framework (GEF) [45]

   The aim of project consists in generation of library for graph editing, which can be used for construction of high-end (high-quality) custom applications for working with graphs. GEF facilities (opportunities):

   - simple and clear design, which allows a developer to expand library's functionality
   - Node-Port-Edge model of graph's presentation, which permits to perform overwhelming majority of tasks occurring in working with graphs applications
   - future XML-based format support (SVG)

2. ILOG JViews [46]

   ILOG JViews gives (grants) components, aimed for using in custom applications, and also in common with Ajax and Eclipse platform.



```
master  Generate reports
remotes/origin/master  Remove IDEA project files from repository
Improved build proccess.
FEATURE: color properties dialog
Loading color schema from properties file
Testing MyColor class
Using global property holder for colors - ColorSchema. And default black color schema
FIXED: level order for the end on the hierarchy
FIXED: moving lens on Windows and Linux platform
Library change
Small project library change
Dont store jogl native library in the repository
Switching to Maven
Subgraph extraction
Scroll over levels in the Gene Ontology zoom state
Draw three levels in the zoom state for Gene Ontology
Updated project
  Unused test deleted
  Overlapping elements in the cluster fixed. Min group element size. Refactoring.
  Fixed elements overlapping in the cluster spiral algorithm
  Moving lens only inside cluster panel
  Moving lens
  Move lens to correspond selected node
  Identify lens circle
  Small refactoring
  Level numbers implementation
  Write the level number in the status bar
  Dont unselect node when view changed from zoom to normal view in th GO.
  Node selection in the GO
  Refactoring
  FIXED small bug in the UniformalDistributin layout setted incorrect Y coordinate if only one element needs to be placed
  Use level visualizer
  Zoom to specific level
  Complete project refactoring
  Background chooser
  FIXED: unfocusable root node in the lens
  Implementated transparent lens
  Improved version of the distribution layout - finds middle place if only one element in the collection
  Complete implementation of the searchable gene list
  Project refactoring
  Complete gml file loading from file chooser dialog and some project refactorings
  FIXED tests
  Write to gml file only graphs with integer vertex id
  Reader implementation for normal gml file and for yEd gml file format
  FIXED - backbone has gone after graph higlighting switched off
  Draw backbone all the time
  Status bar added to main window. Showing focused elements labels in the cluster graph.
  Small refactoring.
  - level lines
  Focusing elements
  New one window look
  Project refactoring
  Gene Ontology levelbar layout implementation
  Large project refactoring
  Improved distribution layout
  Beta version - gene ontology layout
  remotes/origin/go_layout  Gene Ontology layout
  Drawing elements order (draw edges before nodes)
  !! Fixed tests bug - wrong test tree call
  Fixed polar dendrogram bug.
  Fixed Yed gml file writer
  Graph writer to standart and Yed gml file formats
  Refactoring
  Refactoring
  ! FIXED - graph height algorithm
  Refactoring.
  Complete project refactoring..
```

Figure 25: Commit graph of the local repository made by gitk tool

3. JGraphT [47]

   JGraphT is open source library, which provides with mathematical tool of graphs theory. JGraphT supports different kinds of graphs, including: oriented and unoriented graphs, graphs with weighted/non weighted/nominate (named) or anything else arc format, appointed by user, non upgradeable graphs - supported access to internal graphs in "Read Only" mode. Listenable graphs: allows outer listener to trace events appearance; sub graphs: graphs which are a view about other graphs. Being a powerful feature, JGraphT has been developed as easy and type-safe (with Java code generators use) feature for working with graphs. For example, any object can be node of a graph. You can build graphics on basis of: line, URL, XML documents and so forth, you can even build graphs of graphs.

4. Java Universal Network / Graph Framework (JUNG) [48]

   ”JUNG the Java Universal Network/Graph Framework–is a software library that provides a common and extendible language for the modeling, analysis, and visualization of data that can be represented as a graph or network. It is written in Java, which allows JUNG-based applications to make use of the extensive built-in capabilities of the Java API, as well as those of other existing third-party Java libraries.

   The JUNG architecture is designed to support a variety of representations of entities and their relations, such as directed and undirected graphs, multi-modal graphs, graphs with parallel edges, and hypergraphs. It provides a mechanism for annotating graphs, entities, and relations with metadata. This facilitates the creation of analytic tools for complex data sets that can examine the relations between entities as well as the metadata attached to each entity and relation. The current distribution of JUNG includes implementations of a number of algorithms from graph theory, data mining, and social network analysis, such as routines for clustering, decomposition, optimization, random graph generation, statistical analysis, and calculation of network distances, flows, and importance measures (centrality, PageRank, HITS, etc.).

   JUNG also provides a visualization framework that makes it easy to construct tools for the interactive exploration of network data. Users can use one of the layout algorithms provided, or use the framework to create their own custom layouts. In addition, filtering mechanisms are provided which allow users to focus their attention, or their algorithms, on specific portions of the graph.

   As an open-source library, JUNG provides a common framework for graph/network analysis and visualization. We hope that JUNG will make it easier for those who work with relational data to make use of one anothers' development efforts, and thus avoid continually re-inventing the wheel.” [49]

   JUNG library is widely used in differ amount of projects. Here is a list of projects using JUNG:

   - ExtC: an Eclipse plug-in that is useful for locating large, noncohesive classes and for recommending how to split them into smaller, more cohesive classes. (Keith Cassell) [50]
   - Djinn: a tool for visualizing java artifacts dependencies in a project: jars, directories, packages, classes. (Fabien Benoit) [51]
   - Angur: An XML visualization/WYSWYG Editor (Amir Mohammad shahi) [52]
   - RDF Gravity: a tool for visualizing RDF/OWL graphs/ontologies. (Sunil Goyal, Rupert Westenthaler) [53]
   - GUESS from HP Labs is a database-driven network analysis tool that provides flexible visualizations, scripting capabilities with Python/Jython,

and interfaces with JUNG to let users take advantage of its algorithm library. (Eytan Adar, David Feinberg) [54]

- ADAPTNet is an applet that visualises the families of short oligo microarray probesets associated through common gene transcripts. (Michal Okoniewski, Tim Yates) [55]
- Augur [56] is a visualization tool designed to support the distributed software development process. (Jon Froehlich) [57]
- Ariadne is an Eclipse plug-in (under development) that links technical and social dependencies. (Cleidson de Souza et al.) [58]
- Netsight is a proof-of-concept tool for the visual exploratory data analysis of large-scale network and relational data sets. (Yan-Biao Boey, Joshua O'Madadhain, Scott White, Padhraic Smyth) [59]
- InfoVis CyberInfrastructure provides an unified architecture in which diverse data analysis, modeling and visualization algorithms can be plugged in and run. [60]
- PWComp is a graph comparative metabolic pathway tool. (Joshua Adelman, Josh England, Alex Chen) [61]
- Google Cartography, featured in Google Hacks, uses the Google Search API to build a visual representation of the interconnectivity of streets in an area. (Richard Jones) [62]
- GINY is a project with similar aims to that of JUNG, which contains some code derived from JUNG. (Rowan Christmas) [63]
- GraphExplore is a JAVA application that renders networks of objects in a graphical form, which uses modified forms of the JUNG layout algorithm implementations. (Quanli Wang) [64]
- TOTEM (TOolbox for Traffic Engineering Methods) provides a framework where researchers can integrate their traffic engineering algorithms. These algorithms can therefore be applied on models of real networks. The TOTEM toolbox also gives network operators the opportunity to experiment the currently developed traffic engineering algorithms on their own network. Today, the TOTEM toolbox already federates a large set of traffic engineering algorithms published in the scientific literature. This project uses JUNG for the graphical representation of the network topology. (S. Balon, O. Delcourt, J. Lepropre and F. Skivee) [65]
- D2K ("Data to Knowledge") is a visual programming environment for building complicated data mining applications; T2K is a library of D2K modules that implements sophisticated algorithms for text analysis. Each of these uses JUNG for network visualization. [66]
- graphBuilder is an application that allows users to build network representations of relational databases and data files. It has been designed as a tool for exploring online scientific data repositories. (Ben Raymond) [67]
- Semiophore is an application for exploring large graphs where there are many variables on both nodes and links (including time-based/event variables). It works with a relationnal database. It provides several

visualization approaches. One of them is based on JUNG. It provides several analysis routines, featuring SNA measures among them. User can interact with the network : dynamic multi-variables filtering, dynamic aggregation, network editing and production of quicktime videos from longitudinal analysis are possible. Semiophore can handle text/XML documents with NLP information extraction and text summarization routines [English and French support only] in order to automatically build network maps of actors/information. [68]

- Xholon uses JUNG to represent and visualize networks such as biochemical pathways and models (screenshots). (Ken Webb) [69]
- Flink is a website presenting the social networks and research activity of Semantic Web researchers based on a number of sources (web pages, publication databases, email archives, FOAF data). Flink uses JUNG for network representation and visualization as well as for computing network measures. Flink has won 1st prize at the Semantic Web Challenge [71] of 2004. (Peter Mika) [70]
- T-Prox(approve sites) is a proxy, designed to be used for usability analyzes of websites. It uses JUNG to visualize the users path through the site. (Sven Lilienthal) [72]
- Simple C-K Editor is a visualisation tool built on the C-K Design Theory. Its main purpose is to provide an easy tool to create, manipulate, edit and print C-K diagrams. [73]
- PCOPGene is web-based application to analise microarray data with large sample-series. The user can identify several kinds of non-linear expression relationships inside the gene network, study the expression dependence fluctuations in detail, and crossing the results with external biomedical data-servers. [74]

There are a lot more graph visualization frameworks for Java: Piccolo [78], The Visualization Toolkit (VTK) [79], The InfoVis Toolkit [80], Improvise [81]. All of them can be used as for storing and visualizing graphs and networks.

In the scope of current thesis work was used JUNG graph library to store graph structures.

4.2 OpenGL visualization standard

"OpenGL is a software interface to graphics hardware. This interface consists of about 120 distinct commands, which you use to specify the objects and operations needed to produce interactive three-dimensional applications.

OpenGL is designed to work efficiently even if the computer that displays the graphics you create isn't the computer that runs your graphics program. This might be the case if you work in a networked computer environment where many computers are connected to one another by wires capable of carrying digital data. In this situation, the computer on which your program runs and issues OpenGL drawing commands is called the client, and the computer that receives those commands and performs the drawing is called the server. The format for transmitting OpenGL commands (called the protocol) from the client to the server is always the same, so OpenGL programs can work across a network even if the client and server are different kinds of computers. If an OpenGL program isn't running across a network, then there's only one computer, and it is both the client and the server.

OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, you must work through whatever windowing system controls the particular hardware you're using. Similarly, OpenGL doesn't provide high-level commands for describing models of three-dimensional objects. Such commands might allow you to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules. With OpenGL, you must build up your desired model from a small set of geometric primitive - points, lines, and polygons." [75]

There are many wrappers over OpenGL for developing on different programming languages. One of the favourite are JOGL and LWJGL.

"The Lightweight Java Game Library (LWJGL) is a solution aimed directly at professional and amateur Java programmers alike to enable commercial quality games to be written in Java. LWJGL provides developers access to high performance crossplatform libraries such as OpenGL (Open Graphics Library) and OpenAL (Open Audio Library) allowing for state of the art 3D games and 3D sound. Additionally LWJGL provides access to controllers such as Gamepads, Steering wheel and Joysticks. All in a simple and straight forward API.

LWJGL is not meant to make writing games particularly easy; it is primarily an enabling technology which allows developers to get at resources that are simply otherwise unavailable or poorly implemented on the existing Java platform. We anticipate that the LWJGL will, through evolution and extension, become the foundation for more complete game libraries and "game engines" as they have popularly become known, and hide some of the new evils we have had to expose in the APIs.

LWJGL is available under a BSD license, which means it's open source and freely available at no charge." [76]

In the thesis used JOGL wrapper library over OpenGL. "Java OpenGL (JOGL) is a wrapper library that allows OpenGL to be used in the Java programming language. It was originally developed by Kenneth Bradley Russell and Christopher John Kline, and was further developed by the Sun Microsystems Game Technology Group. As of 2010, it is an independent open source project under a BSD license. It is the reference implementation for Java Bindings for OpenGL (JSR-231).

The base OpenGL C API and associated operation system API, are accessed in JOGL via Java Native Interface (JNI) calls. As such, the underlying system must support OpenGL for JOGL to work. JOGL differs from some other Java OpenGL wrapper libraries in that it merely exposes the procedural OpenGL API via methods on a few classes, rather than trying to map OpenGL functionality onto the object-oriented programming paradigm. Indeed, most of the JOGL code is autogenerated from the OpenGL C header files via a conversion tool named GlueGen, which was programmed specifically to facilitate the creation of JOGL.

This design decision has both its advantages and disadvantages. The procedural and state machine nature of OpenGL is inconsistent with the typical method of programming under Java, which is bothersome to many programmers. However, the straightforward mapping of the OpenGL C API to Java methods makes conversion of existing C applications and example code much simpler. The thin layer of abstraction provided by JOGL makes runtime execution quite efficient, but accordingly is more difficult to code compared to higher-level abstraction libraries like Java3D. Because most of the code is autogenerated, changes to OpenGL can be rapidly added to JOGL." [77]

4.3 Program architecture

An overview on the tool's architecture is given in Figure 26. The implementation is divided into several modules specialized for various tasks. The IO module implements data loading from gml files. The data is stored in extended gml file format, which contains additional properties for nodes, such as the node label. The Graph Core module extends the JUNG graph model [48] in order to fit it to our requirements. The implementation of Swing GUI and OpenGL user interactions is realized by the User Interaction module. The Graph Visualization module and its submodules contain all code for the whole visualization process, including our own layout implementation, primitive drawing abstraction, and program state machine. One of the most important modules of is of course Subgraph Extraction that contains the implementation of the subtree calculation algorithm.

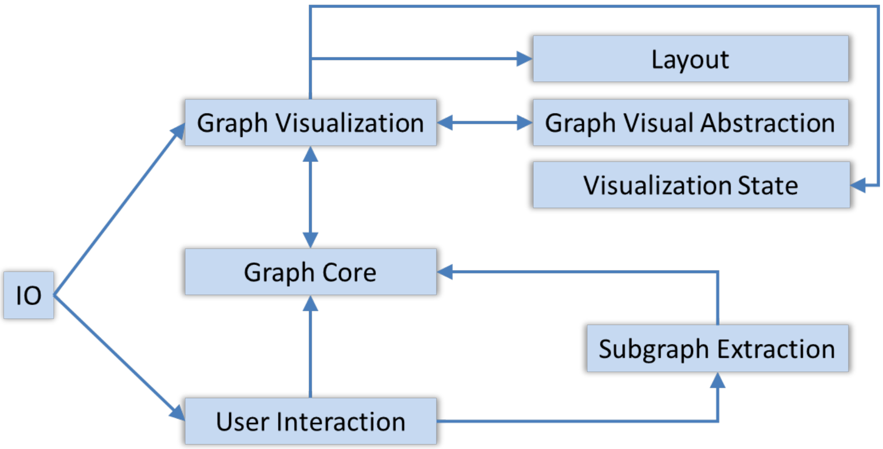


Figure 26: Module architecture of GoClusterViz tool

One of the main module in the program is Subgraph Extraction module. Extraction algorithm was explained in the Section 2 and in the Appendix B is pseudocode, implementation is in the class se.lnu.thesis.algorithm.Extractor.

Complete UML class diagram of the GoClusterViz can be found in the Appendix D. Here is a brief overview of the package content.

Main class and program entry point is class – GoClusterViz.

Package se.lnu.thesis.gui contains all interface abstractions and Swing implementations for main window, menus and event handling.

se.lnu.thesis.element is hierarchy for graph visualisation. It is implementation of the Composite pattern: "In software engineering, the composite pattern is a partitioning design pattern. The composite pattern describes that a group of objects are to be treated in the same way as a single instance of an object. The intent of a composite is to ”compose” objects into tree structures to represent part-whole hierarchies. Implementing the composite pattern lets clients treat individual objects and compositions uniformly.” [82]

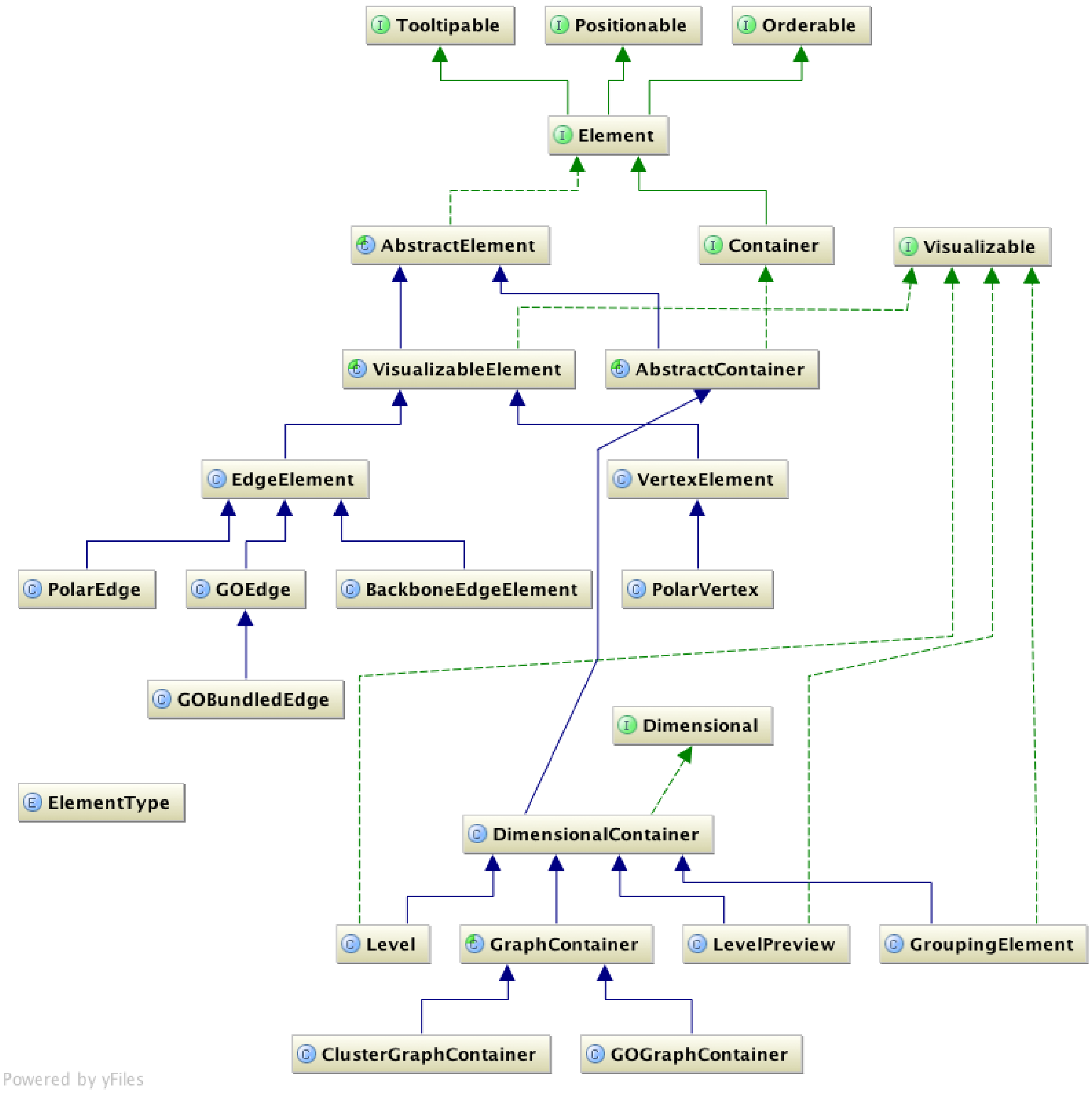


Figure 27: Graph elements visualisation hierarchy abstraction

Package se.lnu.thesis.paint contains all drawing implementations. Graph elements do not implement visualization shape, for that purpose there is ElementVisualizer abstraction. Diagram on the Figure 28 shows complete UML diagram of the package se.lnu.thesis.paint.visualizer, there are several different shape visualizer. To reduce memory usage all visualizers are stored in the ElementVisualizerFactory – Flyweight design pattern.

”Flyweight is a software design pattern. A flyweight is an object that minimizes memory use by sharing as much data as possible with other similar objects; it is a way to use objects in large numbers when a simple repeated representation would use an unacceptable amount of memory. The term is named after the boxing weight class. Often some parts of the object state can be shared and it's common to put them in external data structures and pass them to the flyweight objects temporarily when they are used.

A classic example usage of the flyweight pattern is the data structures for graphical representation of characters in a word processor. It might be desirable to have, for each character in a document, a glyph object containing its font outline, font metrics, and other formatting data, but this would amount to hundreds or thousands of bytes for each character. Instead, for every character there might be a reference to a flyweight glyph object shared by every instance of the same character in the document; only the position of each character (in the document and/or the page) would need to be stored internally.

In other contexts the idea of sharing identical data structures is called hash consing.” [83]

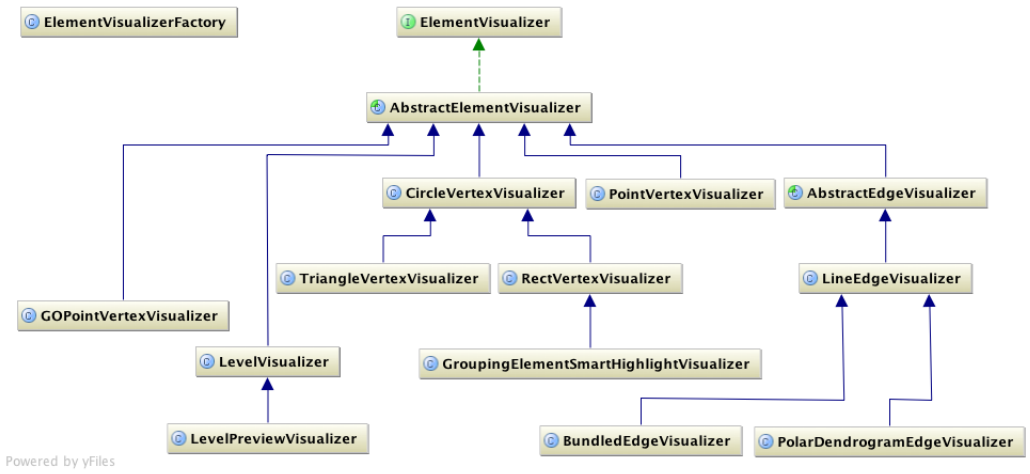


Figure 28: Elements visualizers

User interaction state machine is implementation of the State software development pattern located in the package se.lnu.thesis.paint.state. ”A monolithic objects behavior is a function of its state, and it must change its behavior at run-time depending on that state. Or, an application is characterized by large and numerous case statements that vector flow of control based on the state of the application. The State pattern does not specify where the state transitions will be defined. The choices are two: the context object, or each individual State derived class. The advantage of the latter option is ease of adding new State derived classes. The disadvantage is each State derived class has knowledge of (coupling to) its siblings, which introduces dependencies between subclasses. A table-driven approach to designing finite state machines does a good job of specifying state transitions, but it is difficult to add actions to accompany the state transitions. The pattern-based approach uses code (instead of data structures) to specify state transitions, but it does a good job of accomodating state transition actions.” [84]

Layout implementations are in the se.lnu.thesis.layout package 29.

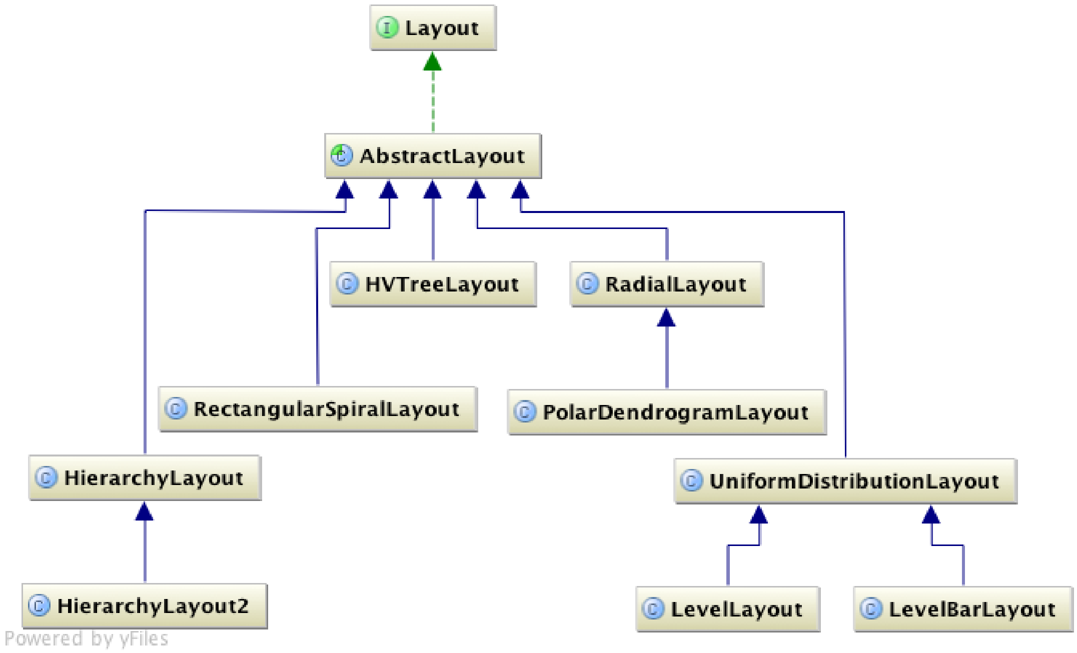


Figure 29: Layouts class diagram

5 Conclusion and future work

We presented a new method for the combined visualization of an ontology (represented as DAG) and an hierarchical clustering (represented as tree) of one data set. The proposed method interactively visualizes all the data without scrolling, thereby presenting an complete overview. It also allows for interactive selection and navigation to explore the data. We have showed that the program is able to tackle the problem in our research focus, i.e., the visualization and visual mapping between two huge and conceptually different data sets. However, there are some improvements that should be performed in the future.

The current state of the prototype does not provide a way to visualize a direct mapping between a terminal GO DAG node and a cluster tree leave. A simple way to overcome this problem for one specific node is to highlight the corresponding nodes in the GO view and/or Cluster Tree view on mouse-over action. This could be easily implemented as a part of our future work.

As explained previously, the zoomed-in view for the GO shows three levels at the same time while displaying the subgraph by highlighting the nodes only. The edges are omitted due to clutter problems that can occur since edges from a higher level might go through the zoomed-in view to nodes in the lower layers.

Since we are zoomed in, this does no make sense to show, because we have no insight from which layer those edges are coming from, nor to which layer they are going to. However, an improvement is possible by showing only edges between the three layers shown in the zoomed-in GO view. At the same time, the edge bundling algorithm could also be improved.

We are also working on a heavily improved version of our spiral tree metaphor to cope with more balanced binary trees. One possible solution is to create something we call nested spiral trees. The idea is to draw smaller spirals instead of aggregating larger subtrees that pass over a certain threshold of nodes into box glyphs. This approach will introduce more unused spaces, making the approach less space-filling.

Glossary

GO – Gene Ontology
XML – eXtensible Markup Language
GML – Graph Modelling Language
SVG – Scalable Vector Graphics
IDE - Integrated Development Environment
JUNG – Java Universal Network / Graph Framework
JOGL – Java OpenGL
JNI – Java Native API
LWJGL – Lightweight Java Game Library

References

[1] VLSI is available on the web http://en.wikipedia.org/wiki/Very-large-scale_integration. Last access in September 2010.

[2] Graph drawing general information is available on the web http://en.wikipedia.org/wiki/Graph_drawing. Last access in September 2010.

[3] Radial Dendrogram is available on the web cs.sunysb.edu/~vislab/papers. Last access in September 2010.

[4] Introduction into Dengrograms is available on the web en.wikipedia.org/wiki/Dendrogram. Last access in September 2010.

[5] T. Barlow and P. Neville, A comparison of 2D visualization of hierarchies, Information Visualization pp 131-138, 2001.

[6] M. Kreussler and H. Schumann, A flexible approach for visual data mining, IEEE Trans. VisualGraphics, vol. 8, no. 1, pp. 39-51, 2002.

[7] J. Yang, M. Ward, and E. Rundensteiner, InterRing: An interactive tool for visually navigating and manipulatiung hierarchical structures, IEEE 2002 Symposium on Information Visualization, pp. 77-92, 2002.

[8] Biotech is available on the web http://biotech.icmb.utexas.edu/pages/bioinfo.html. Last access in September 2010.

[9] Information about Bioinformatics is available on the web http://www.absoluteastronomy.com/topics/Bioinformatics. Last access in September 2010.

[10] Gene Ontology is available on the web http://www.geneontology.org/. Last access in September 2010.

[11] The Open Biomedical Ontologies is available on the web http://www.obofoundry.org/. Last access in September 2010.

[12] G. Gan, C. Ma and J. Wu, Data Clustering. Theory, Algorithms, and Applications, Society for Industrial and Applied Mathematics pp 3, 2007.

[13] Prof. Dr. Andreas Kerren home page is available on the web http://w3.msi.vxu.se/users/akemsi/. Last access in September 2010.

[14] Prof. Dr. Falk Schreiber home page is available on the web http://bic-gh.ipk-gatersleben.de/~schreibe/. Last access in September 2010.

[15] ISOVIS research group home page is available on the web http://cs.lnu.se/isovis/. Last access in September 2010.

[16] Plant Bioinformatic research Group home page is available on the web http://nwg.bic-gh.de/. Last access in September 2010.

[17] Graphlet is available on the web http://www.fmi.uni-passau.de/Graphlet/. Last access in September 2010.

[18] GML is available on the web http://www.infosun.fim.uni-passau.de/Graphlet/GML/gml-tr.html. Last access in September 2010.

[19] GML supported programs is available on the web http://en.wikipedia.org/wiki/Graph_Modelling_Language. Last access in September 2010.

[20] yEd is available on the web http://www.yworks.com/en/products_yed_about.html. Last access in September 2010.

[21] Clairlib is available on the web http://www.clairlib.org/. Last access in September 2010.

[22] Cytoscape is available on the web http://en.wikipedia.org/wiki/Cytoscape. Last access in September 2010.

[23] NetworkX is available on the web http://en.wikipedia.org/wiki/NetworkX. Last access in September 2010.

[24] ocamlgraph is available on the web http://ocamlgraph.lri.fr/. Last access in September 2010.

[25] OGDF is available on the web http://www.ogdf.net/. Last access in September 2010.

[26] Tulip is available on the web http://tulip.labri.fr/TulipDrupal/. Last access in September 2010.

[27] GraphML specification is available on the web http://graphml.graphdrawing.org/. Last access in September 2010.

[28] DOT graph language is available on the web http://en.wikipedia.org/wiki/DOT_language. Last access in September 2010.

[29] GXL introduction is available on the web http://www.gupro.de/GXL/Introduction/section1.html. Last access in September 2010.

[30] SVG information is available on the web http://en.wikipedia.org/wiki/Scalable_Vector_Graphics. Last access in September 2010.

[31] Graph Drawing Conference '95 is available on the web www.informatik.uni-trier.de/~ley/db/conf/gd/gd95.html. Last access in September 2010.

[32] Java FAQ is available on the web http://www.java.com/en/download/faq/whatis_java.xml. Last access in September 2010.

[33] Binary Tree definition is available on the web http://www.wordiq.com/definition/Binary_tree. Last access in September 2010.

[34] Java 2D reference is available on the web http://java.sun.com/products/java-media/2D/reference/index.html. Last access in September 2010.

[35] Java AWT overview is available on the web http://en.wikipedia.org/wiki/Abstract_Window_Toolkit. Last access in September 2010.

[36] Java SWING overview is available on the web http://en.wikipedia.org/wiki/Swing_(Java). Last access in September 2010.

[37] AWT vs SWING comparison is available on the web http://edn.embarcadero.com/article/26970. Last access in September 2010.

[38] Maven project home page is available on the web http://maven.apache.org/. Last access in September 2010.

[39] Apache Foundation home page is available on the web http://www.apache.org/. Last access in September 2010.

[40] Vincent Massol, Jason van Zyl, Better builds with Maven, Mergere Library Press, pp. 16, 2006.

[41] Vincent Massol, Jason van Zyl, Better builds with Maven, Mergere Library Press, pp. 18, 2006.

[42] Revision Control definition is available on the web http://en.wikipedia.org/wiki/Revision_control. Last access in September 2010.

[43] Git overview is available on the web http://en.wikipedia.org/wiki/Git_(software). Last access in September 2010.

[44] is available on the web . Last access in September 2010.

[45] Java Graph Editing Framework is available on the web http://gef.tigris.org/. Last access in September 2010.

[46] ILOG Jview is available on the web http://www.ilog.com/products/jviews/. Last access in September 2010.

[47] JGraphT) is available on the web http://jgrapht.sourceforge.net/. Last access in September 2010.

[48] Java Universal Network / Graph Framework (JUNG) is available on the web http://jung.sourceforge.net/. Last access in September 2010.

[49] JUNG graph library overview is available on the web http://jung.sourceforge.net/index.html. Last access in September 2010.

[50] ExtC is available on the web http://code.google.com/p/ext-c/. Last access in September 2010.

[51] Djinn is available on the web http://www.jnovation.net/djinn. Last access in September 2010.

[52] Angur is available on the web http://angur.sourceforge.net/. Last access in September 2010.

[53] RDF Gravity is available on the web http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html. Last access in September 2010.

[54] GUESS is available on the web http://www.hpl.hp.com/research/idl/projects/graphs/index.html. Last access in September 2010.

[55] ADAPTNet is available on the web http://bioinformatics.picr.man.ac.uk/adaptnet. Last access in September 2010.

[56] Augur is available on the web http://www.cs.washington.edu/homes/jfroehli/publications/GROUP2005_SeekingTheSource.pdf. Last access in September 2010.

[57] Jon Froehlich web page is available on the web http://www.cs.washington.edu/homes/jfroehli/. Last access in September 2010.

[58] Ariadne is available on the web http://projects.ischool.washington.edu/mcdonald/cscw04/papers/desousa-cscw04.pdf. Last access in September 2010.

[59] Netsight is available on the web http://jung.sourceforge.net/netsight. Last access in September 2010.

[60] InfoVis CuberInfrastructure is available on the web http://iv.slis.indiana.edu/sw/. Last access in September 2010.

[61] PWComp is available on the web http://www.ocf.berkeley.edu/\~Ejadelman/pwcomp/. Last access in September 2010.

[62] Google Cartography is available on the web http://richard.jones.name/google-hacks/google-cartography/google-cartography.html. Last access in September 2010.

[63] GINY is available on the web http://csbi.sourceforge.net/. Last access in September 2010.

[64] GraphExplorer is available on the web http://graphexplore.cgt.duke.edu/. Last access in September 2010.

[65] TOTEM is available on the web http://totem.run.montefiore.ulg.ac.be/. Last access in September 2010.

[66] D2K is available on the web http://alg.ncsa.uiuc.edu/do/downloads. Last access in September 2010.

[67] graphBuilder is available on the web http://aadc-maps.aad.gov.au/analysis/gb.cfm. Last access in September 2010.

[68] Semiophore is available on the web http://www.semiophore.net/. Last access in September 2010.

[69] Xholon is available on the web http://sourceforge.net/projects/xholon/. Last access in September 2010.

[70] Flink is available on the web http://flink.semanticweb.org/. Last access in September 2010.

[71] Semantic Web Challenge is available on the web http://challenge.semanticweb.org/. Last access in September 2010.

[72] T-Prox is available on the web http://www.t-prox.net/. Last access in September 2010.

[73] Simple C-K Editor is available on the web http://code.google.com/p/ckeditor. Last access in September 2010.

[74] PCOPGene is available on the web http://revolutionresearch.uab.es/. Last access in September 2010.

[75] Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis, OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R), Version 2.1 (6th Edition), Addison-Wesley Professional, 2007.

[76] LWJGL is available on the web http://www.lwjgl.org/. Last access in September 2010.

[77] JOGL is available on the web http://jogamp.org/. Last access in September 2010.

[78] Piccolo is available on the web http://www.cs.umd.edu/hcil/jazz/. Last access in September 2010.

[79] Visualisation Toolkit is available on the web http://www.vtk.org/. Last access in September 2010.

[80] InfoVis Toolkit) is available on the web http://ivtk.sourceforge.net/. Last access in September 2010.

[81] Improvise project homepage is available on the web http://www.cs.ou.edu/~weaver/improvise/index.html. Last access in September 2010.

[82] Gamma, Erich; Richard Helm, Ralph Johnson, John M. Vlissides (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. pp. 395.

[83] Flyweight design pattern explanation is available on the web http://en.wikipedia.org/wiki/Flyweight_pattern. Last access in September 2010.

[84] State design pattern explanation is available on the web http://sourcemaking.com/design_patterns/state. Last access in September 2010.

Appendix A: Listing of the complex gml file

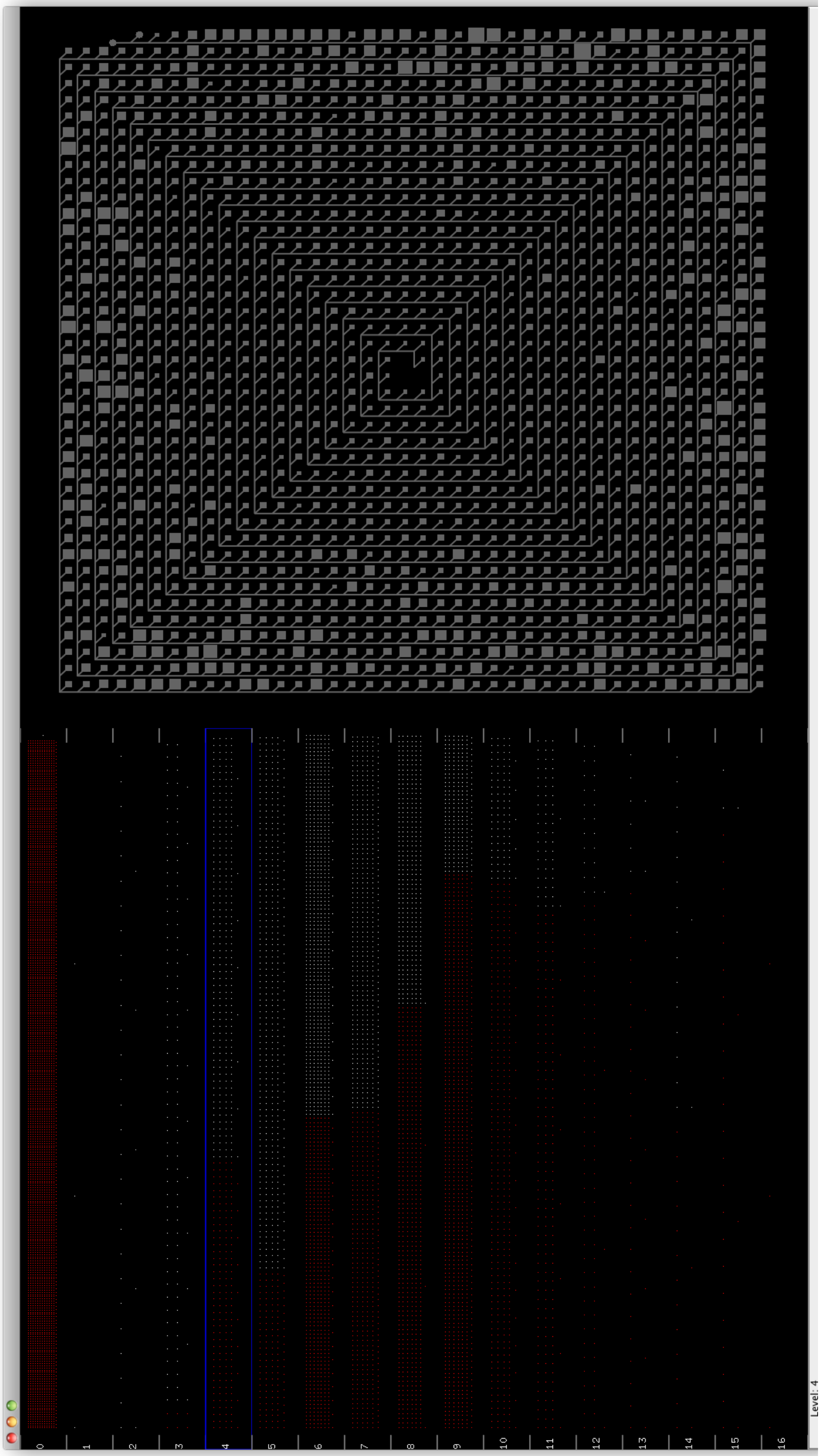
```
Creator ”yFiles”
Version ”2.7”
graph
[
  hierarchic  1
  label ””
  directed  1
  node
  [
    id  0
    label ”1”
    graphics
    [
      x 360.0
      y 67.0
      w 30.0
      h 30.0
      type  ”rectangle”
      fill  ”#FFCC00”
      outline ”#000000”
    ]
    LabelGraphics
    [
      text  ”1”
      fontSize  13
      fontName  ”Dialog”
      anchor  ”c”
    ]
  ]
  node
  [
    id  1
    label ”2”
    graphics
    [
      x 284.0
      y 150.0
      w 30.0
      h 30.0
      type  ”rectangle”
      fill  ”#FFCC00”
      outline ”#000000”
    ]
    LabelGraphics
    [
      text  ”2”
      fontSize  13
      fontName  ”Dialog”
      anchor  ”c”
    ]
  ]
  node
  [
    id  2
    label ”3”
    graphics
    [
      x 450.0
      y 150.0
      w 30.0
      h 30.0
      type  "rectangle"
      fill  "#FFCC00"
      outline "#000000"
    ]
    LabelGraphics
    [
      text  "3"
      fontSize  13
      fontName  "Dialog"
      anchor  "c"
    ]
  ]
  node
  [
    id  3
    label "4"
    graphics
    [
      x 391.0
      y 270.0
      w 30.0
      h 30.0
      type  "rectangle"
      fill  "#FFCC00"
      outline "#000000"
    ]
    LabelGraphics
    [
      text  "4"
      fontSize  13
      fontName  "Dialog"
      anchor  "c"
    ]
  ]
  node
  [
    id  4
    label "5"
    graphics
    [
      x 540.0
      y 270.0
      w 30.0
      h 30.0
      type  "rectangle"
      fill  "#FFCC00"
      outline "#000000"
    ]
    LabelGraphics
    [
      text  "5"
      fontSize  13
      fontName  "Dialog"
      anchor  "c"
    ]
  ]
  node
  [
    id  5
    label "6"
    graphics
  [
    x 540.0
    y 67.0
    w 30.0
    h 30.0
    type ”rectangle”
    fill ”#FFCC00”
    outline ”#000000”
  ]
  LabelGraphics
  [
    text ”6”
    fontSize 13
    fontName ”Dialog”
    anchor ”c”
  ]
]
edge
[
  source 0
  target 1
  graphics
  [
    fill ”#000000”
    targetArrow ”standard”
  ]
]
edge
[
  source 0
  target 2
  graphics
  [
    fill ”#000000”
    targetArrow ”standard”
  ]
]
edge
[
  source 2
  target 5
  graphics
  [
    fill ”#000000”
    targetArrow ”standard”
  ]
]
edge
[
  source 2
  target 3
  graphics
  [
    fill ”#000000”
    targetArrow ”standard”
  ]
]
edge
[
  source 2
  target 4
  graphics
```

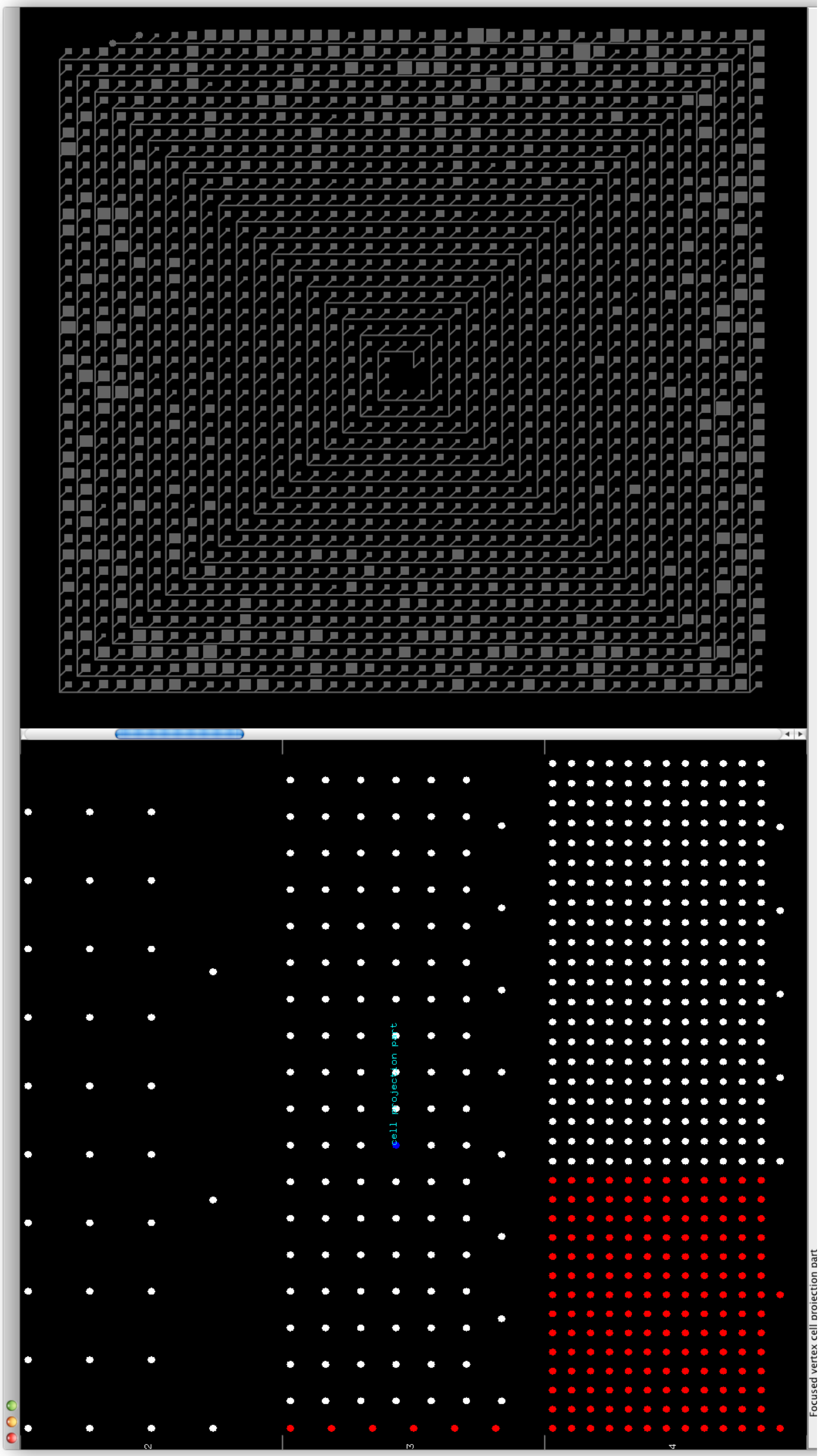
```
      [
        fill   "#000000"
        targetArrow "standard"
      ]
    ]
    edge
    [
      source   5
      target   4
      graphics
      [
        fill   "#000000"
        targetArrow "standard"
      ]
    ]
]
```

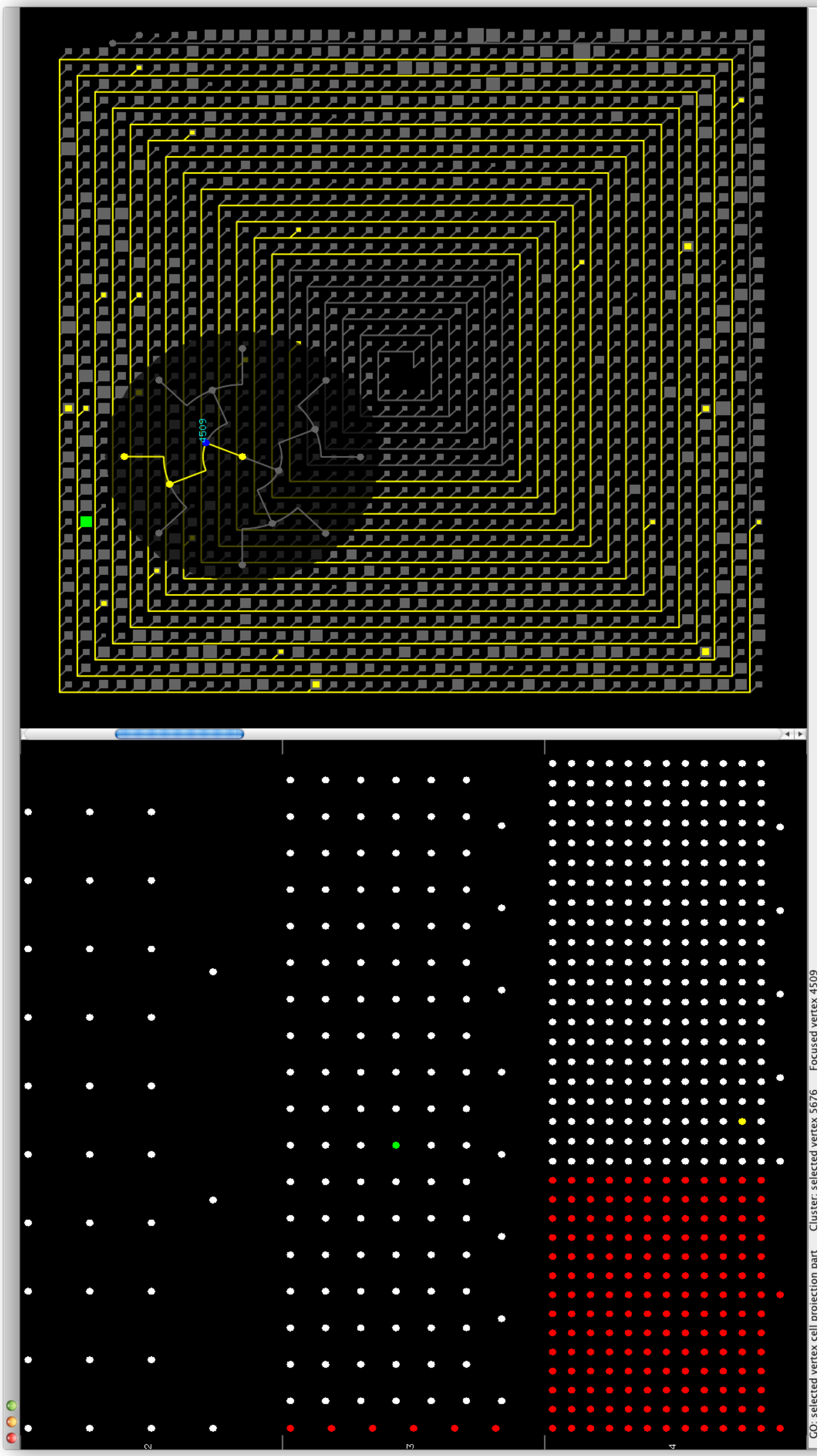
Appendix B: Subgraph extraction algorithm

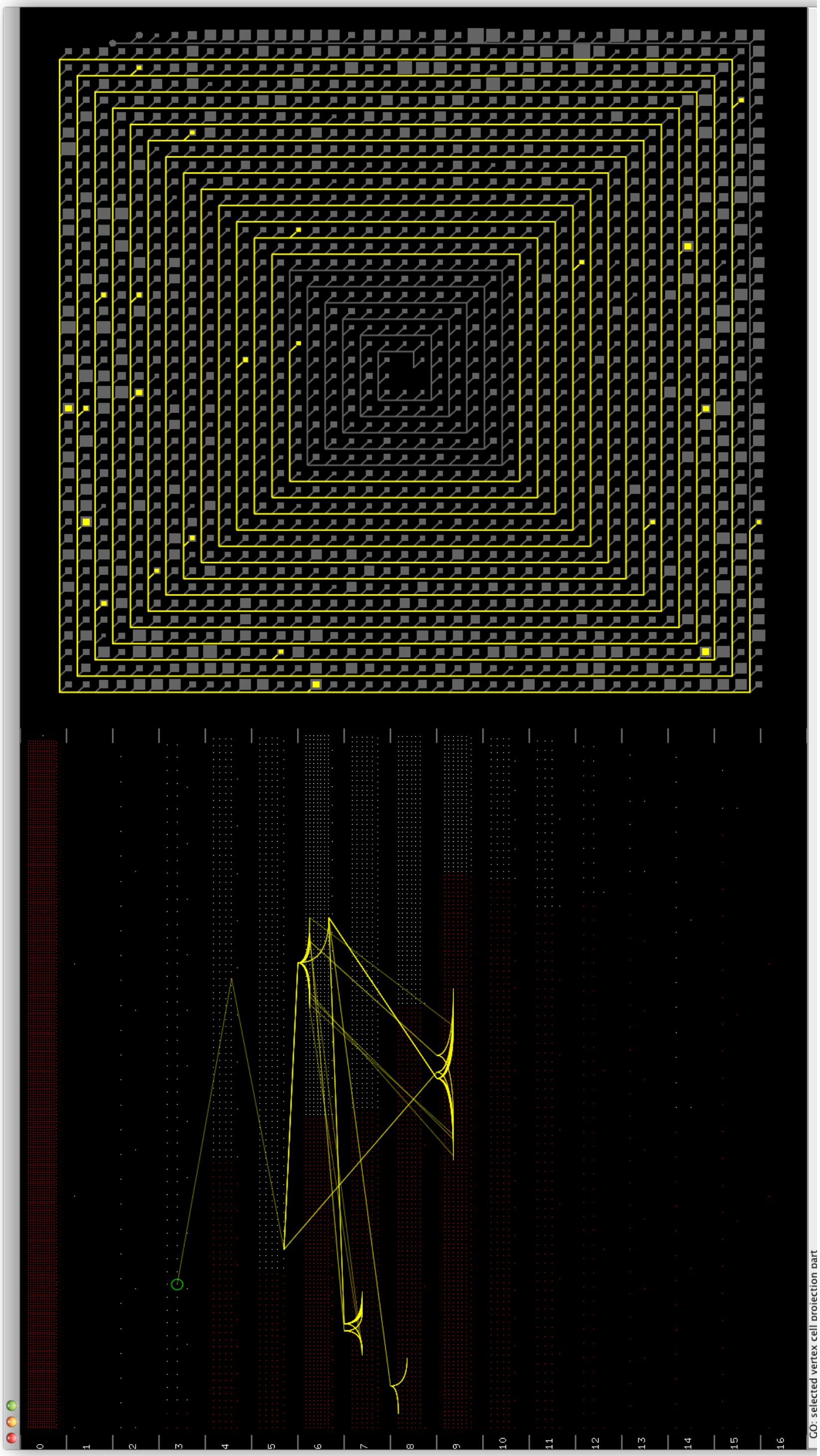
```
// selected vertex in the Gene Ontology by user
GO_vertex;

if GO_vertex is in cache {
  // already computed, load from cache
  load from cache GO_subgraph and Cluster_subgraph;
  return;
}

// extract subgraph using non-recursive DFS
// starting from vertex GO_vertex as root
GO_subgraph = extractSubgraph(GO_graph, GO_vertex);
save GO_subgraph to cache with key GO_vertex

// create empty subgraph
Cluster_subtree = new Graph;

for each vertex in GO_subgraph {
  if vertex is leaf {
    // leaf labels are the same for both graphs,
    // but vertex objects are different
    l_label = GO_graph.getLabel(vertex);
    leaf = Cluster_graph.getVertexByLabel(l_label);
    // get all connected vertices
    // from current leaf up to the Cluster root
    connectedVertices = invertDFS(Cluster_graph, leaf);

    // add connected vertices to cluster subtree
    // create edges
    for (int i=0; i<=connectedVertices.size()-1; i++) {
      node1, node2 = null;
      node1 = connectedNodes.get(i);
      Cluster_subtree.addVertex(node1);

      if (i+1<=connectedNodes.size()-1) {
      node2 = connectedNodes.get(i+1);

      Cluster_subtree.addVertex(node2);
      Cluster_subtree.addEdge(node2,node1);
    }
  }

  // find lowest common root for subgraph
  // and remove vertices from the root to common root
  removeRootChain(Cluster_graph, Cluster_subtree);
}

save Cluster_subtree to cache with key GO_vertex
```

Appendix C: Program screenshots

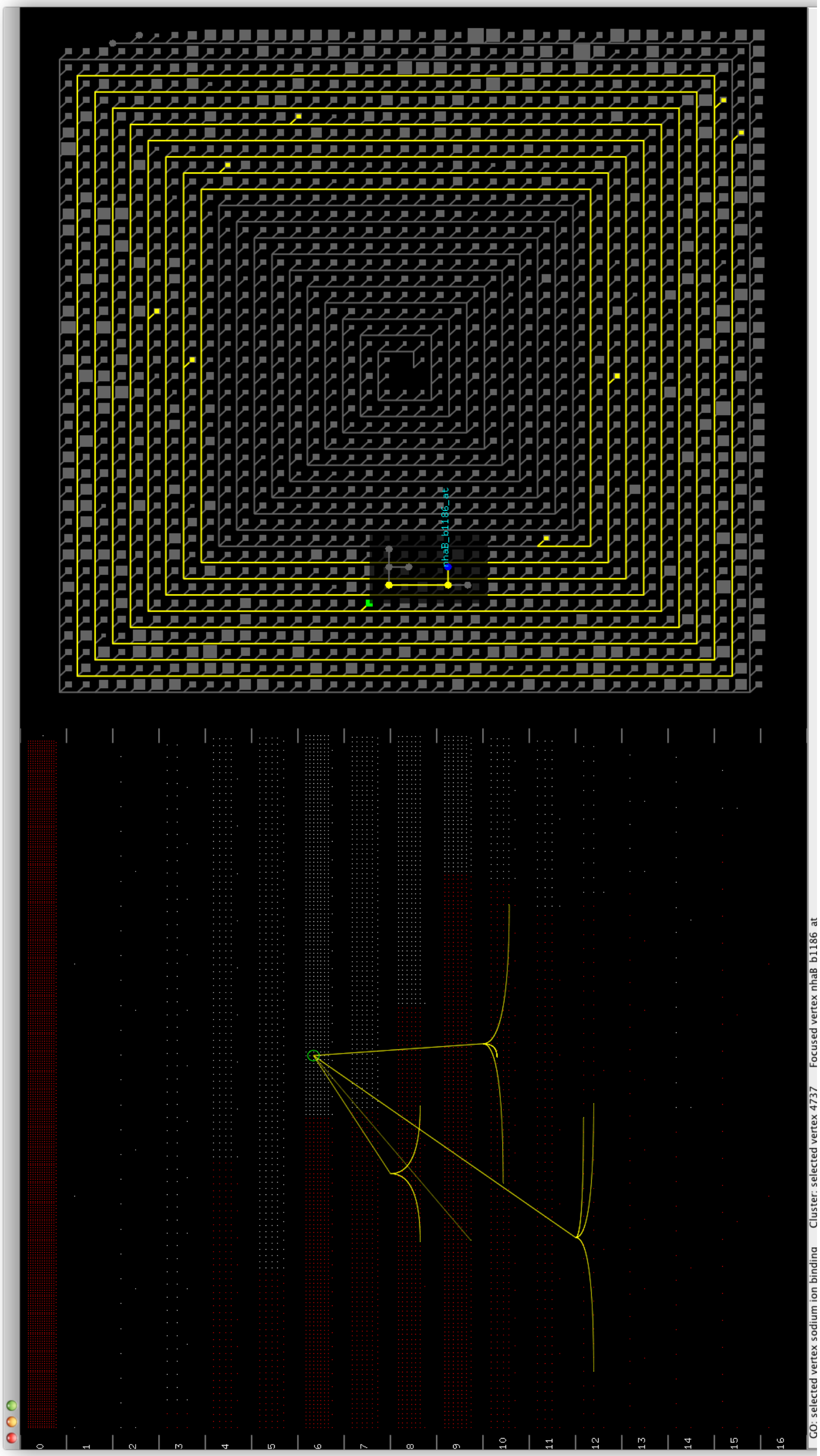



cell projection part
2
3
4
Focused vertex cell projection part

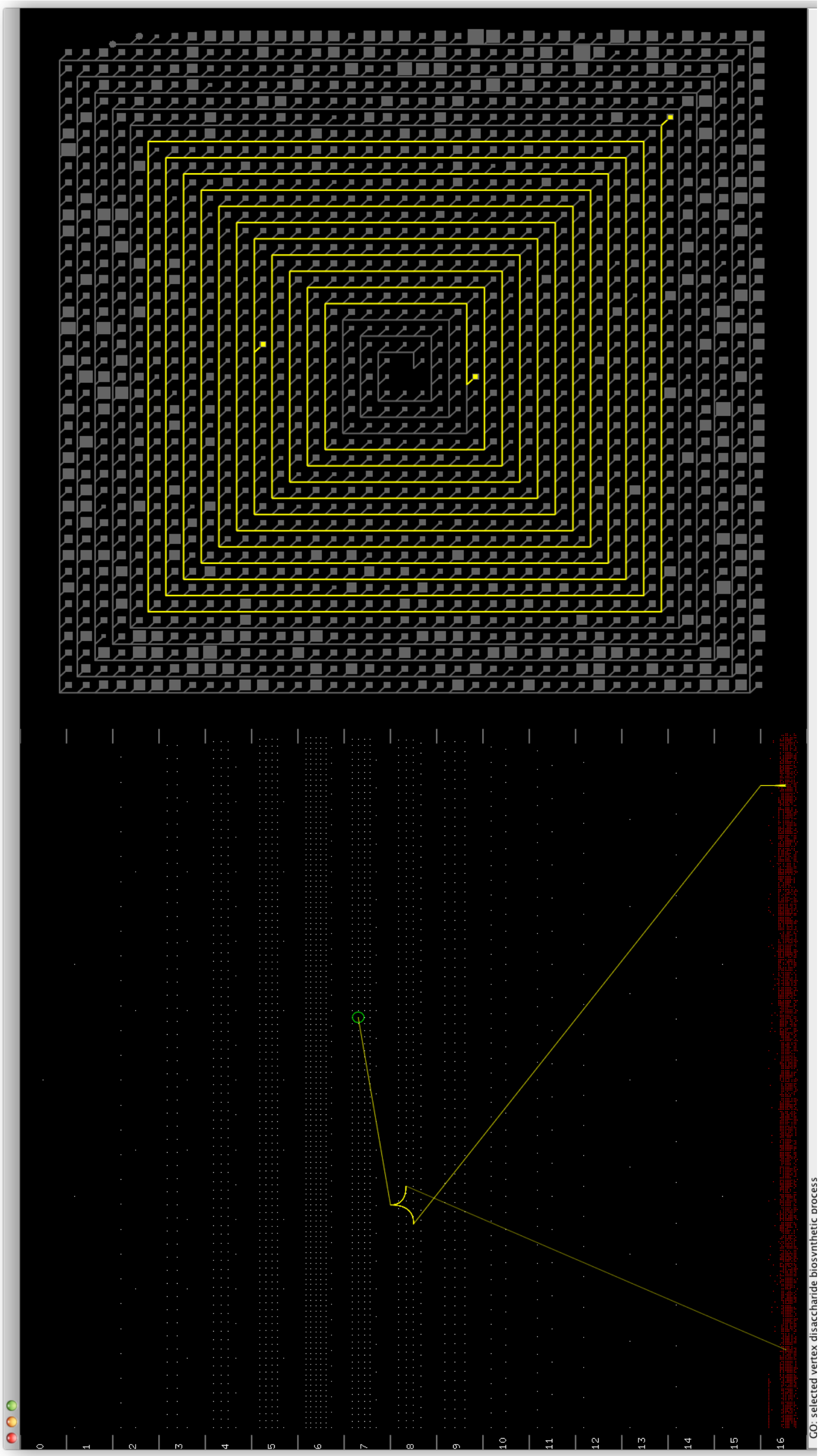

4509
2
3
4
GO: selected vertex cell projection part
Cluster: selected vertex 5676
Focused vertex 4509



0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
GO: selected vertex cell projection part

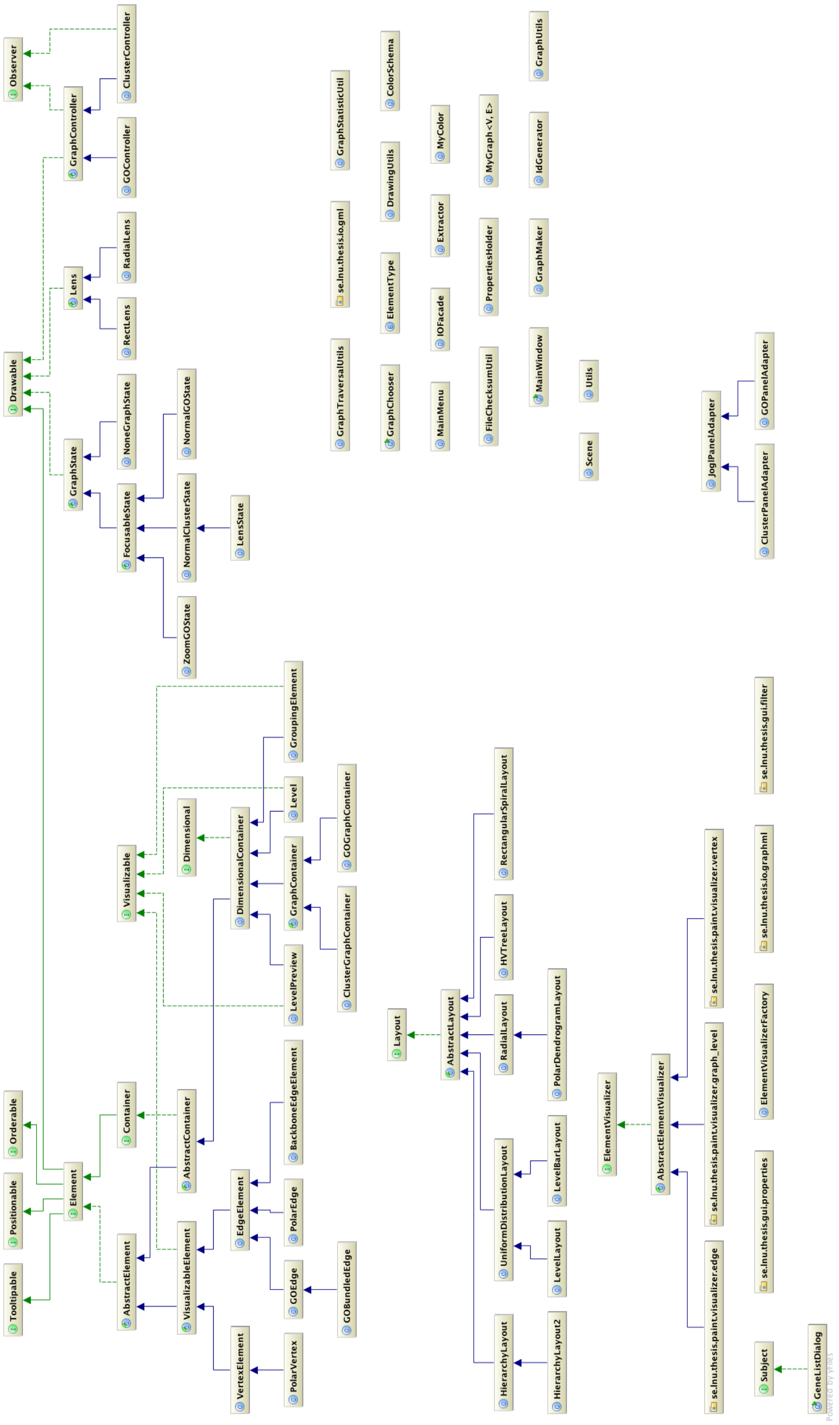


GO: selected vertex sodium ion binding   Cluster: selected vertex 4737   Focused vertex nhaB_b1186_at



GO: selected vertex disaccharide biosynthetic process

Appendix D: Complete UML class diagram


Observer
ClusterController
GraphController
GOController
Drawable
Lens
RadialLens
RectLens
GraphState
NoneGraphState
FocusableState
NormalGOState
NormalClusterState
LensState
ZoomGOState
GraphStatisticUtil
se.lnu.thesis.io.gml
GraphTraversalUtils
ColorSchema
DrawingUtils
ElementType
GraphChooser
MyColor
Extractor
IOFacade
MainMenu
MyGraph<V, E>
PropertiesHolder
FileChecksumUtil
GraphUtils
IdGenerator
GraphMaker
MainWindow
Utils
Scene
JoglPanelAdapter
GOPanelAdapter
ClusterPanelAdapter
Visualizable
GroupingElement
Dimensional
DimensionalContainer
Level
GraphContainer
GOGraphContainer
LevelPreview
ClusterGraphContainer
Layout
AbstractLayout
RectangularSpiralLayout
HVTreeLayout
RadialLayout
PolarDendrogramLayout
se.lnu.thesis.gui.filter
se.lnu.thesis.paint.visualizer.vertex
se.lnu.thesis.io.graphml
Orderable
Positionable
Tooltipable
Element
Container
AbstractContainer
AbstractElement
VisualizableElement
EdgeElement
BackboneEdgeElement
PolarEdge
GOEdge
GOBundledEdge
VertexElement
PolarVertex
ElementVisualizer
AbstractElementVisualizer
UniformDistributionLayout
LevelBarLayout
LevelLayout
HierarchyLayout
HierarchyLayout2
se.lnu.thesis.paint.visualizer.graph_level
se.lnu.thesis.paint.visualizer.edge
ElementVisualizerFactory
se.lnu.thesis.gui.properties
Subject
GeneListDialog
Powered by yFiles