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

Acknowledgements

1 Introduction

1.1 Motivation

1.2 Purpose

1.3 Report structure

Section 1 presents the results of the literature review, where been exploring relevant research efforts in the fields of Bioinformatic, Information 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. In Section **??**,

we describe possible scenarios to identify the features of the envisioned system. Chapter 6 consists of the initial ideas and interactions we had for our GUI. Chapter 7 is about the technical approach of our application. In that chapter, we describe the visualization techniques, the tag clouds, the clustering algorithm that we have chosen, and the interaction techniques that we have used for having a robust and flexible geovisualization system for visualizing notes and their content. In Chapter 8, we give some explanation about technologies that we have used, and explained the classes that we implemented. In the next chapter (Case Study), we used the first imagined scenario based on what we presented in Chapter 4 and tested it with the application we have implemented. Finally, in Chapter 10 we give our conclusions and we present future work that can help our system to deliver better results.

1.4 Information Visualization

1.5 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" [1]

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." [2] 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.

1.6 Gene Ontology

As stated above, there are different knowledge databases for biologic information storage. Gene Ontology [3] 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) [4]. 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.

1.7 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." [5]

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 1).
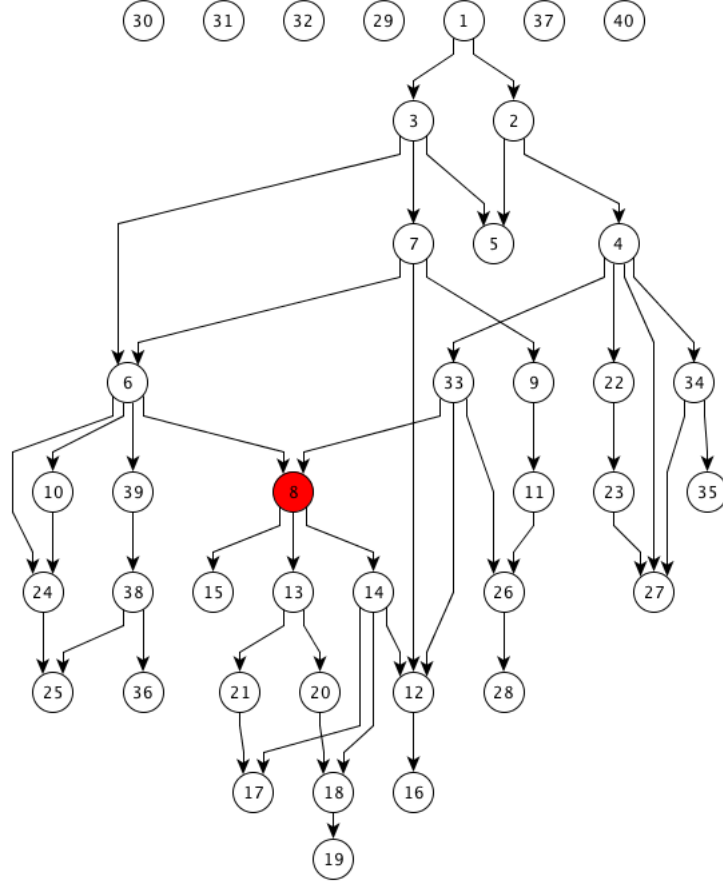

Figure 1: Selected node in the Gene Ontology

3. When node is selected in the Gene Ontology program computes all successors (Figure 2).

4. Extract leafs from successors (Figure 3).

5. Then the program searches corresponded leaves in cluster analysis result tree by label as seen on the Figure 4.

6. For this leaves the program founds root connected to all leaves and extract corresponding sub trees (Figure 5).

7. Founded sub tree cached. This sub tree is highlighted in cluster analysis tree.
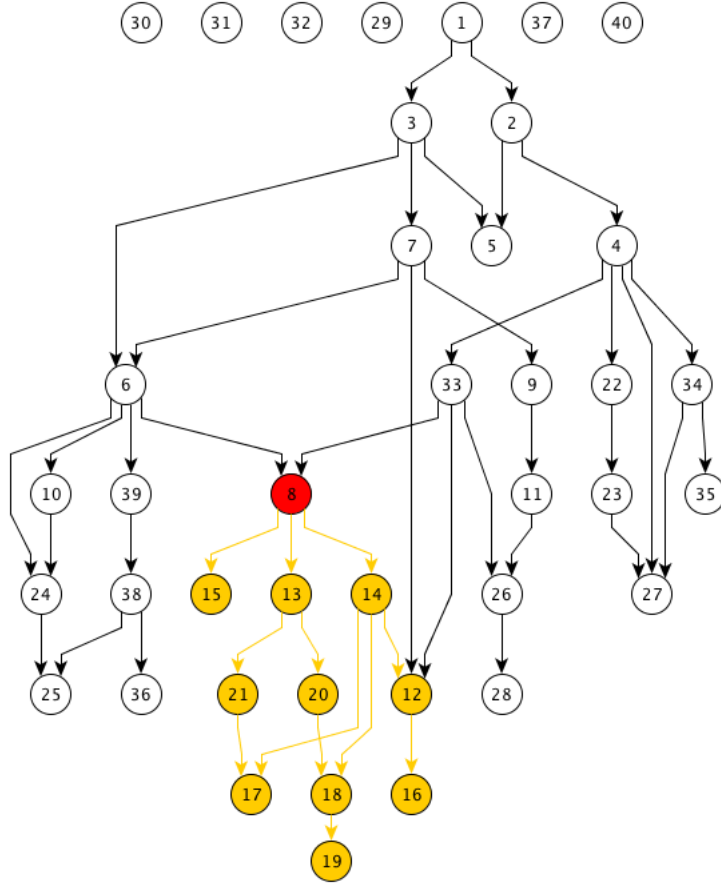
Figure 2: Corresponded successors of the selected node

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 [6]) of Linneuniversitetet at Växjö, Sweden, and Plant Bioinformatics Group (Head: Prof. Dr. Falk Schreiber [7]) 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
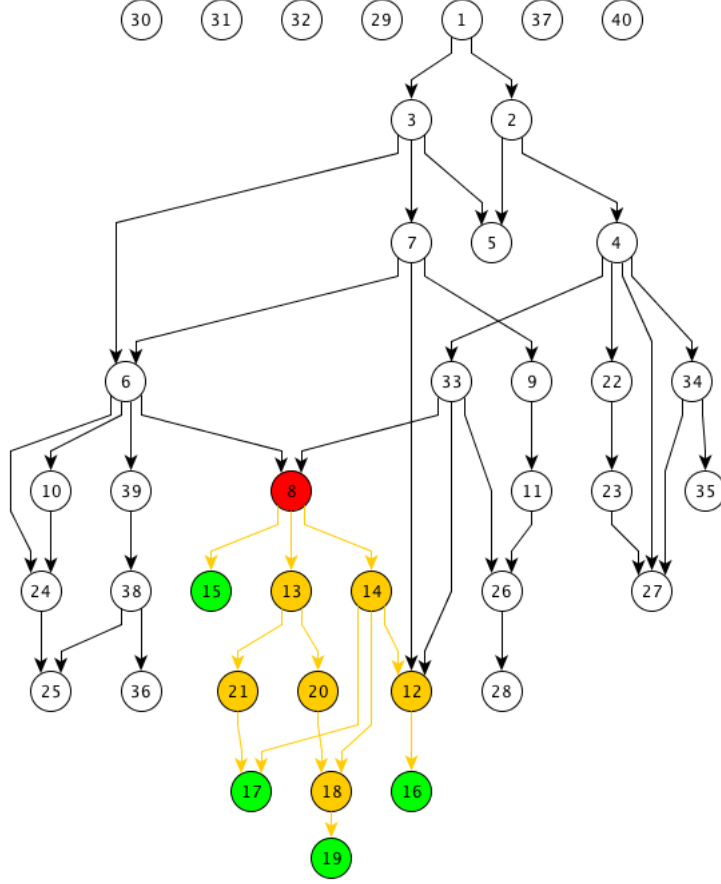
Figure 3: Extract leafs

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
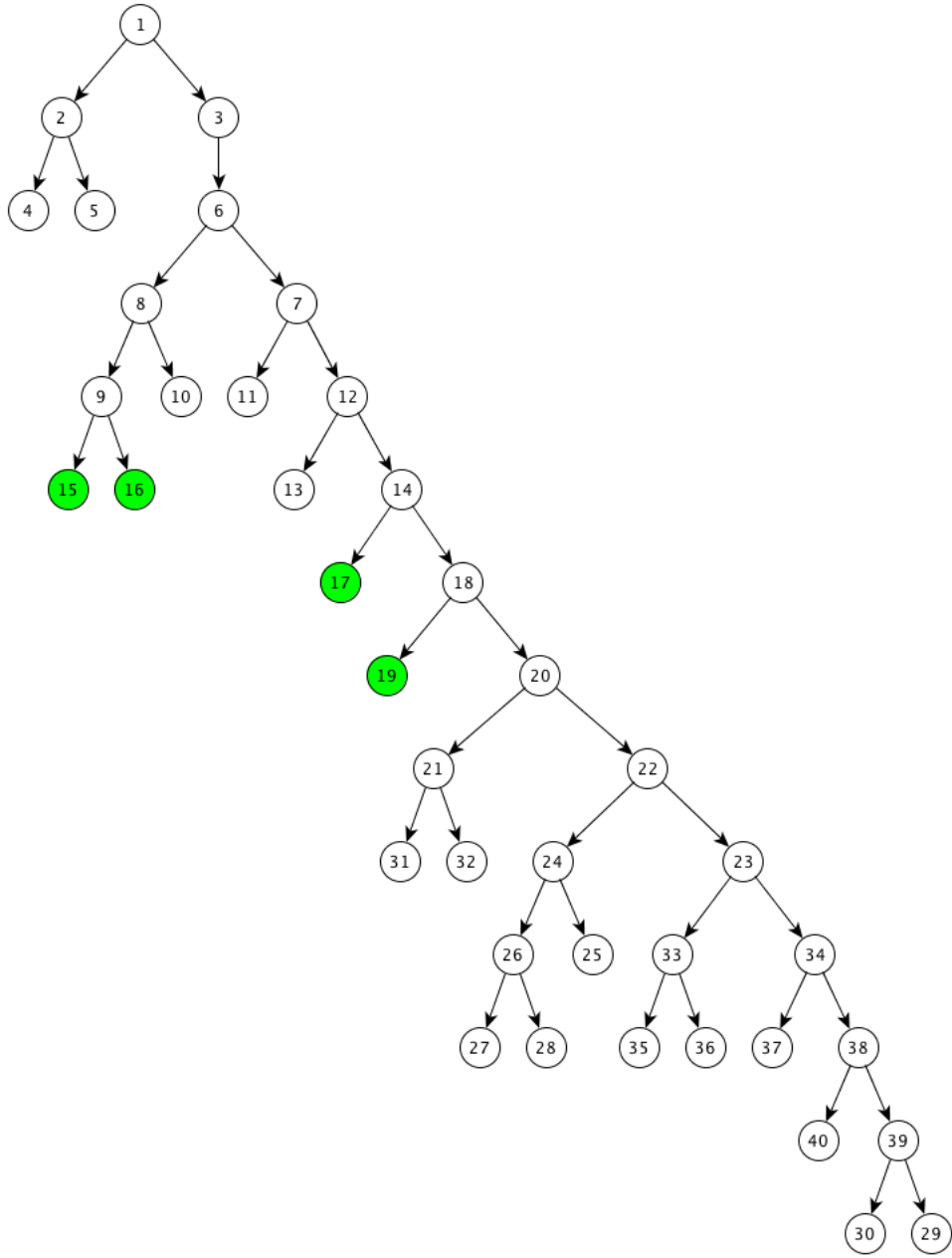
Figure 4: Corresponded leaves in the cluster tree

the visualization of biological/biochemical or geographical information." [8]

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.

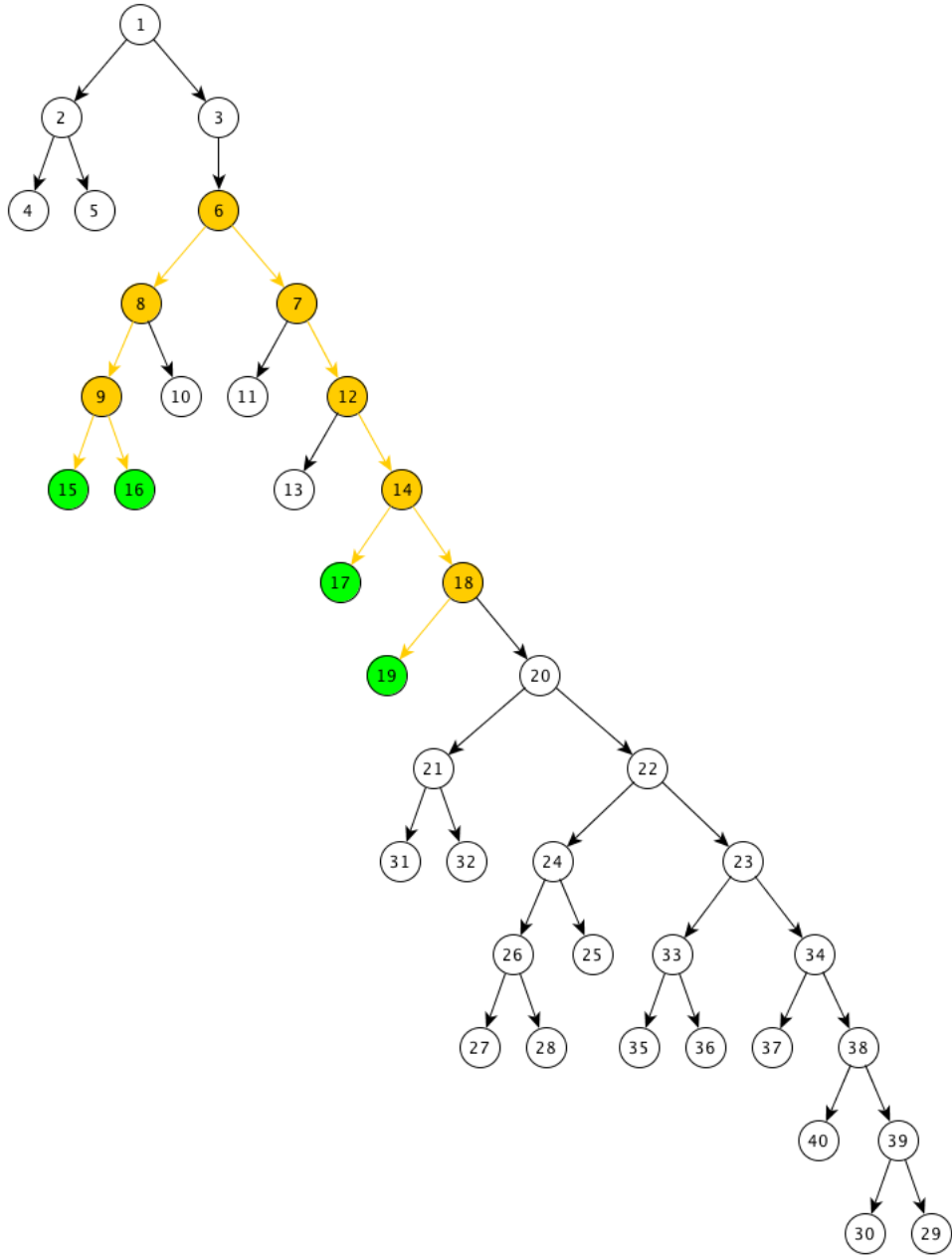Figure 5: Build sub tree

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 develop-

ing 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. [9]

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 2.2 showed visualization of the Gene Ontology graph using yEd [**?**] graph editor using Hierarchical layout. It is obviously shows imperfectness of classic visualization of the graphs.
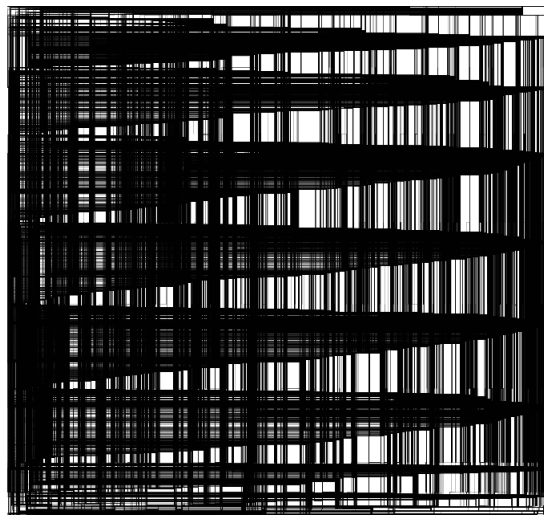
Figure 6: Gene Ontology yEd visualization

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 2.2 is visualization of the cluster tree using Cytoscape [15] 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.

Figure 7: Cluster analysis result tree Cytoscape visualization tool

2.3 GML Graph File Format

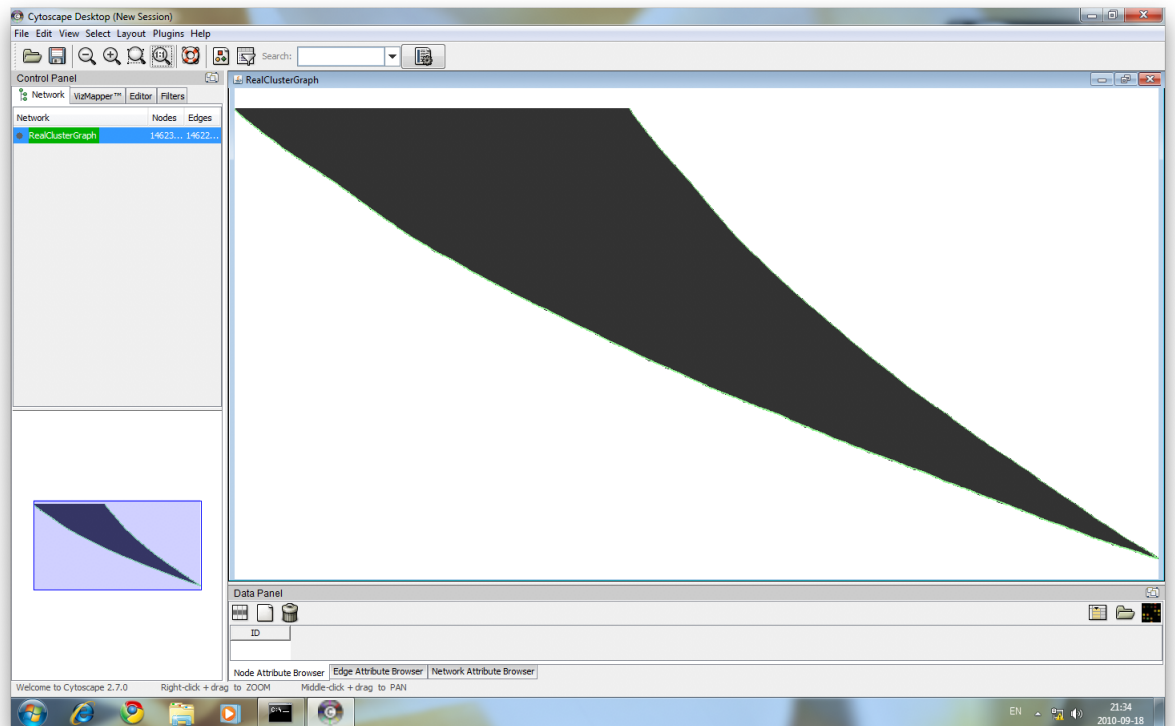"GML, the Graph Modelling Language, is our proposal for a portable file format for graphs. GML's key features are portability, simple syntax, extensibility and flexibility. A GML file consists of a hierarchical key-value lists. Graphs can be annotated with arbitrary data structures. The idea for a common 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." [11]

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
```
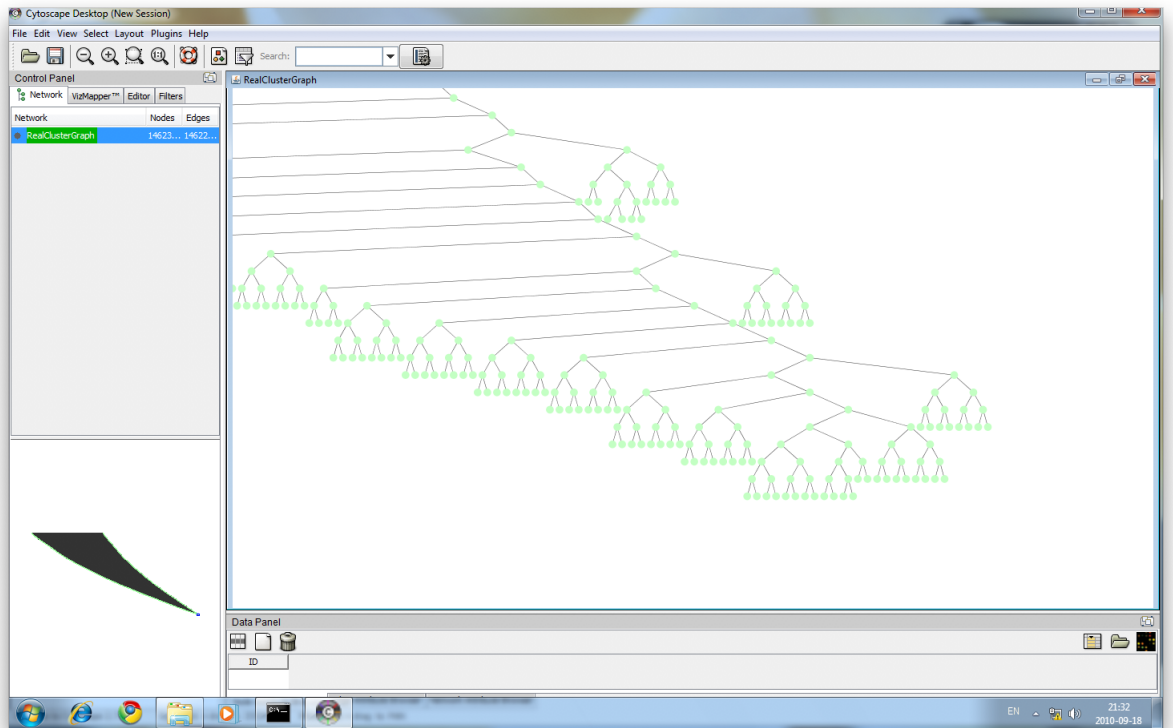
Figure 8: Zoomed cluster analysis result tree

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

In the Figure 9 showed manual visualization of the graph using yEd [13]
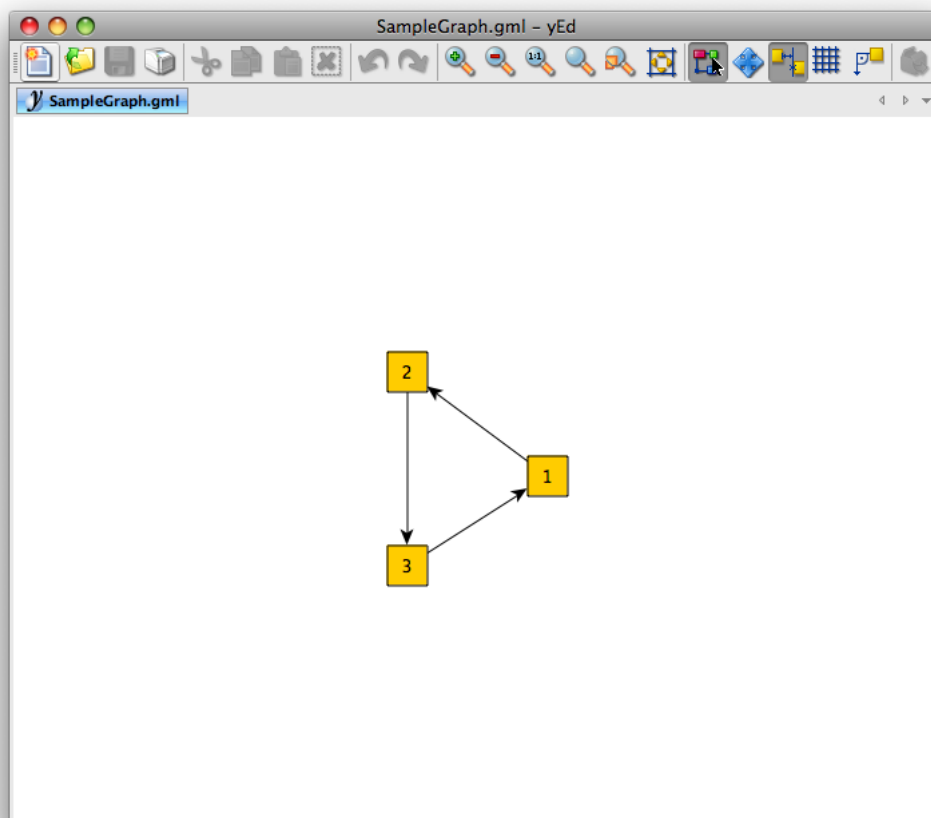
Figure 9: Manual visualization of the sample graph

More complex graph with additional properties and its visualization produced by yEd graph editor is in the 5.2 and the visualization of this graph showed on the Figure 10

Applications supporting GML [12]

- Clairlib [14], 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 [15], an open source bioinformatics software platform for visualizing molecular interaction networks, loads and save previously-constructed interaction networks in GML.

- NetworkX [16], an open source Python library for studying complex graphs.

- ocamlgraph[17], a graph library for OCaml.
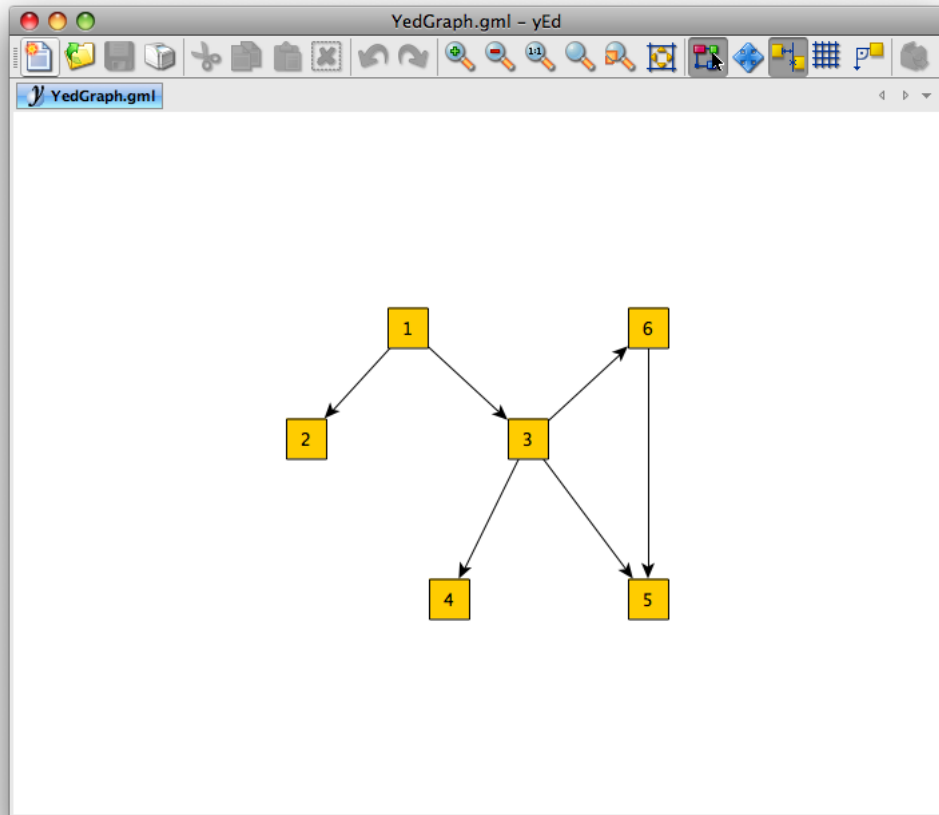
Figure 10: Manual visualization of the sample graph

- OGDF[18], 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 [19] (software) is a free software in the domain of information visualisation capable of manipulating huge graphs (with more than 1.000.000 elements).

- yEd [13], a free Java-based graph editor, supports import from and export to GML.

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. [20] 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 11 below is a simple graph. It contains 11 nodes and 12 undirected edges.

Figure 11: 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>
```
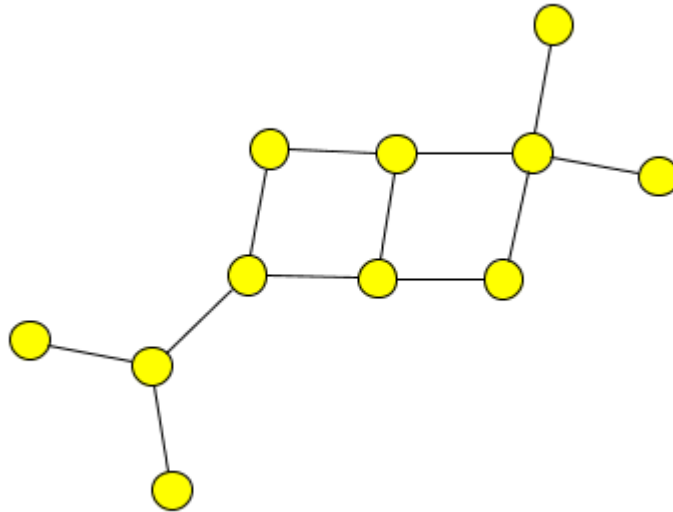
2.4.2 DOT graph file format

"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." [**?**]

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." [**?**]

Listing 4: DOT file format: directed graph
```
graph graphname {
    a −> b −> c ;
    b −> d ;
}
```

Visual representation of both graphs showed in the Figure 12 below.
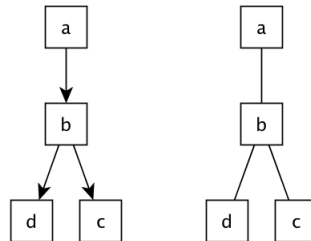
Figure 12: Directed and undirected graphs

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
<?xml version='1.0' encoding='utf-8'?>
<DirectedGraph xmlns="http://schemas.microsoft.com/vs/2009/dgml">
  <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." [21]

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." [22]

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" [25] Simple binary tree showed on the Figure 13
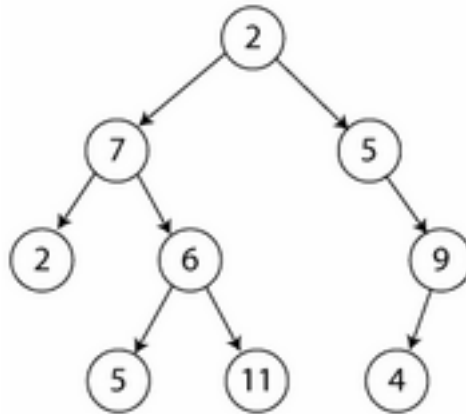
Figure 13: A simple binary tree graph

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 14 produced by MATLAB 7.2.
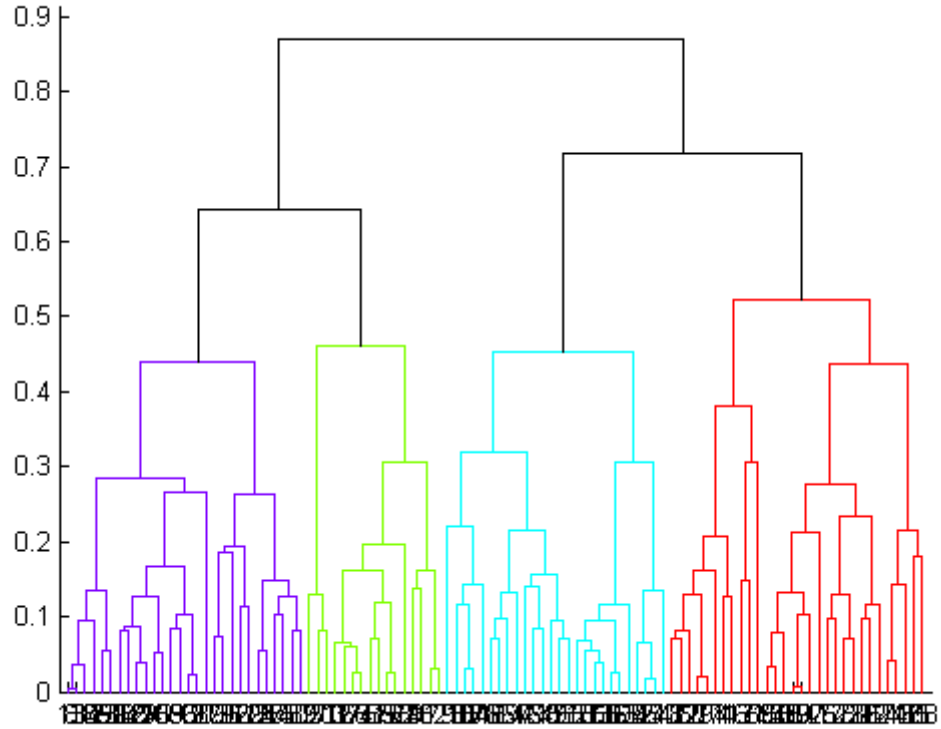
Figure 14: Dendrogram

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

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 [26] which by itself is "heavyweight" because it's part of the Java AWT – Abstract Windows Toolkit.

21
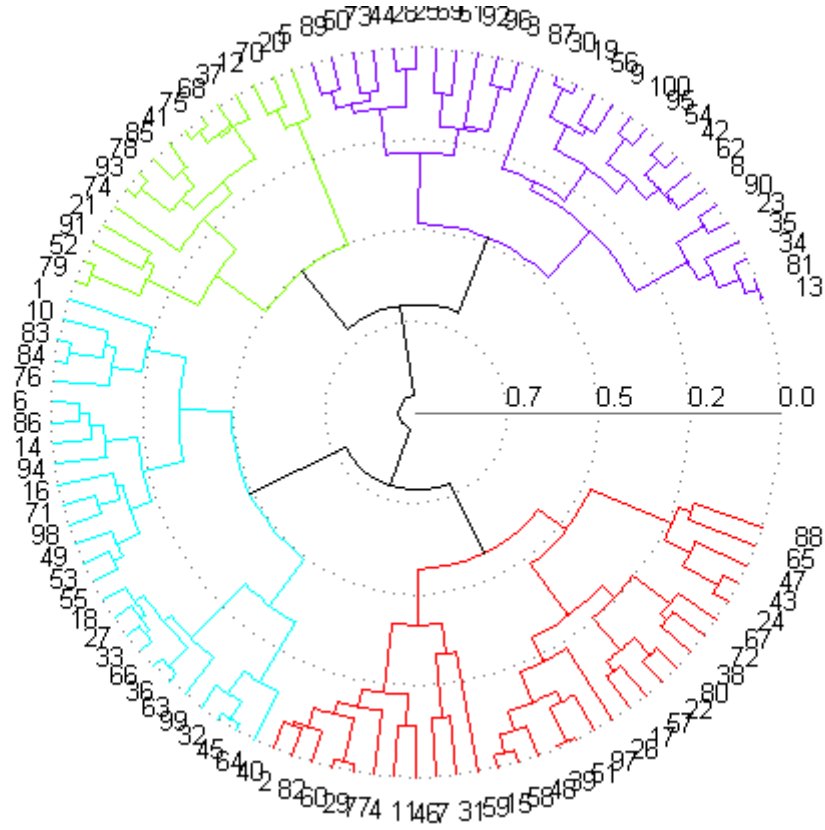
Figure 15: Polar Dendrogram

"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." [27] 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." [28]

"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

22
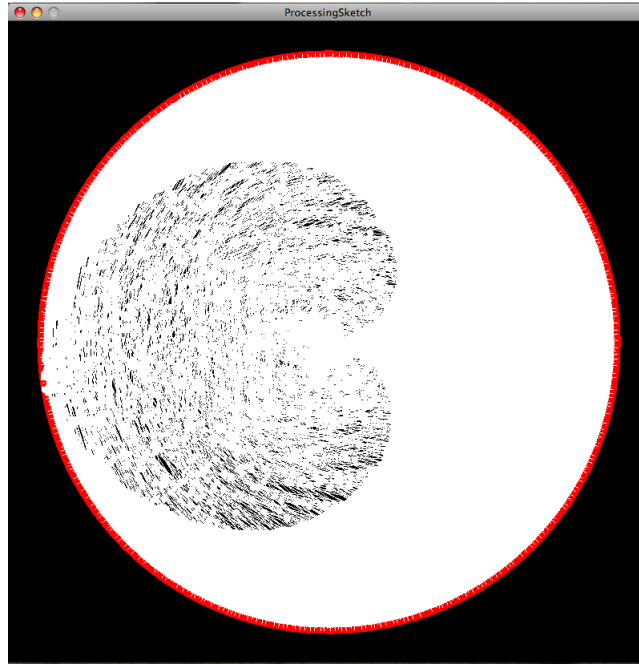
Figure 16: Cluster visualization using JUNG radial layout

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." [28] More detail comparison can be found here. [29]

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 18 and Figure 19 shows cluster visualization and highlighted subgraph using algorithm that was discussed in the Section **??**. This pictures show the nature of the dataset. Improved version has good performance and better visualization but still has issues. Too many elements are in the scene, it is impossible to identify separate gene and trace highlighted graph genes.

3.2 Gene Ontology Visualization

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." [24] 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 and graphic libraries.

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

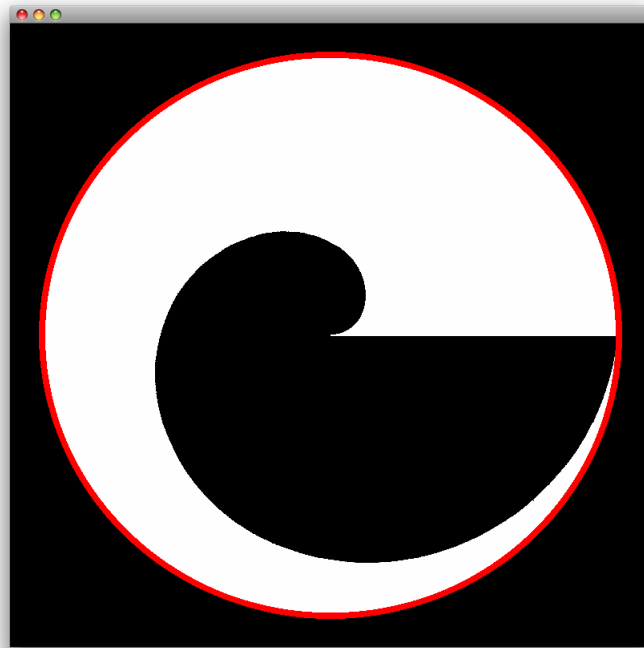Java is flexible platform which has big amount of different libraries. It helps not to write twice things already made but flexibility cause project structure complexity and library management. On the early stage there are no problems to control project throw sophisticated IDE (integrated development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

Maven [30] is free, open-source and de facto project management standard on the Java platform, is part of Apache software project developed and supported by ASF (Apache Software Foundation) [31]. "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." [33]

"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 meaningless 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.

24
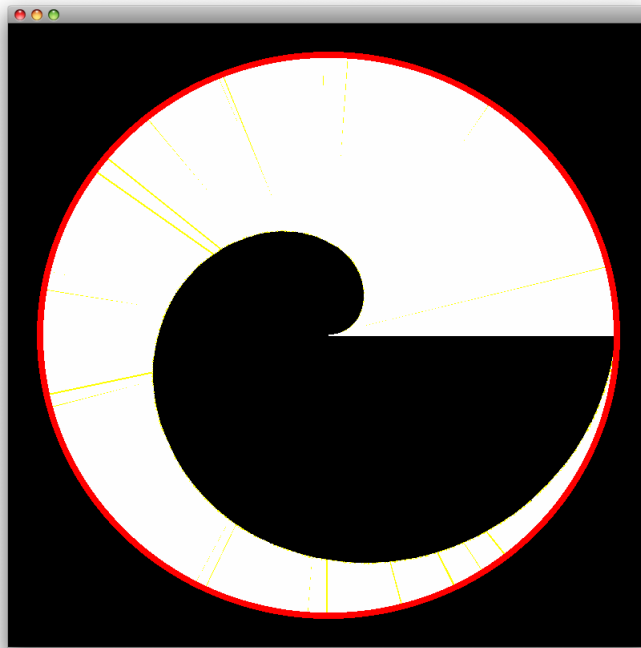
Figure 18: Cluster graph and highlighted subgraph

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 com-
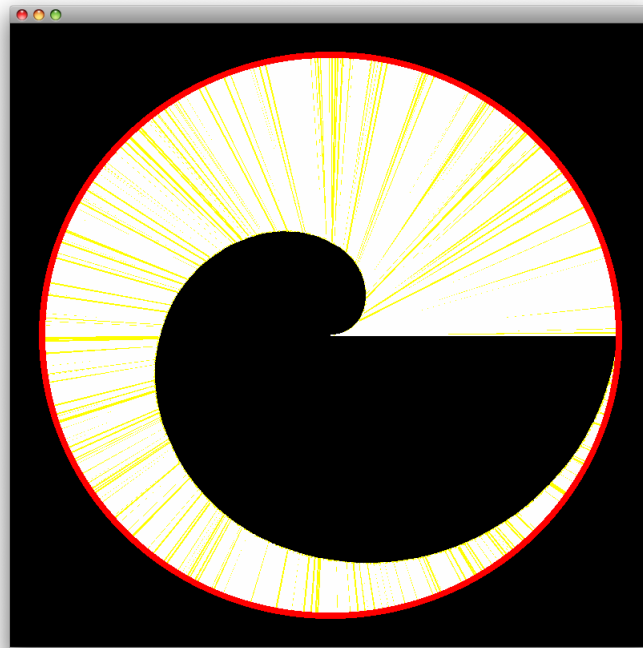
Figure 19: Cluster graph and highlighted subgraph

pelling 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." [?]

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

26

- allowing transparent migration to new features

On the Figure 20 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.
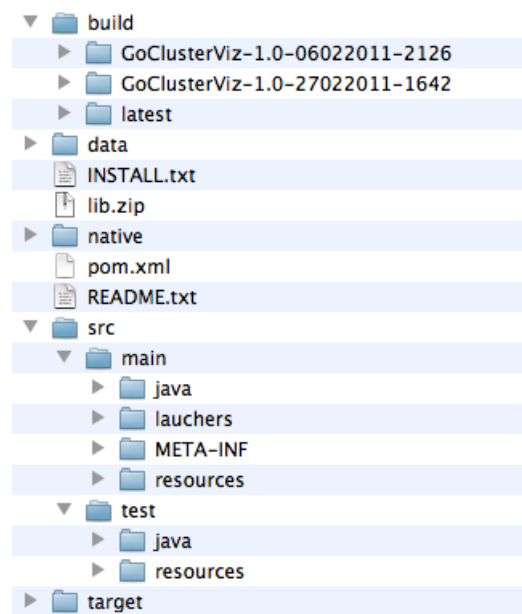
Figure 20: Thesis folder structure

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." [34]

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." [35] Git allows to store and work local on 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) [37]

   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 [38]

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

3. JGraphT [39]

   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;

Figure 21: Commit graph of the local repository made by gitk tool

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) [40]

   "JUNG the Java Universal Network/Graph Framework–is a software library that provides a common and extendible language for the modeling,

29

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." [41]

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) [42]

- Djinn: a tool for visualizing java artifacts dependencies in a project: jars, directories, packages, classes. (Fabien Benoit) [43]

- Angur: An XML visualization/WYSWYG Editor (Amir Mohammad shahi) [44]

- RDF Gravity: a tool for visualizing RDF/OWL graphs/ontologies. (Sunil Goyal, Rupert Westenthaler) [45]

- 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) [46]

- ADAPTNet is an applet that visualises the families of short oligo microarray probesets associated through common gene transcripts. (Michal Okoniewski, Tim Yates) [47]

- Augur [48] is a visualization tool designed to support the distributed software development process. (Jon Froehlich) [49]

- Ariadne is an Eclipse plug-in (under development) that links technical and social dependencies. (Cleidson de Souza et al.) [50]

- 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) [51]

- InfoVis CyberInfrastructure provides an unified architecture in which diverse data analysis, modeling and visualization algorithms can be plugged in and run. [52]

- PWComp is a graph comparative metabolic pathway tool. (Joshua Adelman, Josh England, Alex Chen) [53]

- 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) [54]

- GINY is a project with similar aims to that of JUNG, which contains some code derived from JUNG. (Rowan Christmas) [55]

- 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) [56]

- 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) [57]

- 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. [58]

- 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) [59]

- 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. [60]

- Xholon uses JUNG to represent and visualize networks such as biochemical pathways and models (screenshots). (Ken Webb) [61]

- 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 [63] of 2004. (Peter Mika) [62]

- 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) [64]

- 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. [65]

- 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. [66]

There are a lot more graph visualization frameworks for Java: Piccolo [70], The Visualization Toolkit (VTK) [71], The InfoVis Toolkit [72], Improvise [73]. 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 standart

"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." [67]

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." [68]

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." [69]

4.3 IO layer Implementation

4.4 Gene Ontology Visualization Implementation

4.5 Cluster Visualization Implementation

5 Conclusions

5.1 Future work

5.2 Problems during thesis work

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[23] 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.

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

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

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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[67] 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.

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

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

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

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

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

[73] Improvise project homepage is available on the web `http://www.cs.ou.edu/~weaver/improvise/index.html`. 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"
    ]
  ]
]
```