Biogranat: An Extensible Framework for the Visualization and Analysis of Gene Regulatory Networks

Volker Ahlers, Frauke Sprengel, and student team

Fachhochschule Hannover (Germany)
University of Applied Sciences and Arts
Faculty IV, Dept. of Computer Science

Thomas Schlitt and Benjamin Lehne

King's College London (UK)
School of Medicine, Dept. of Medical and Molecular Genetics

FHH Student Team

Master project in winter semester 2007/08

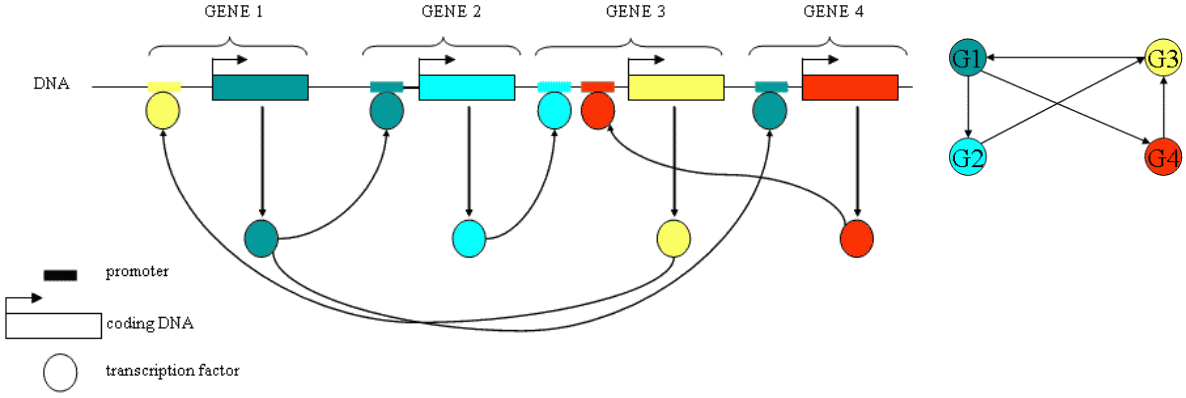
Ingo Bente, Hendrik Kammeyer, Andreas Mendig, Mike Steinmetz, Jörg Vieweg, Björn Zimmer

Bachelor theses in 2007 and 2008

Bastian Hellmann, Stefan Kirstein, Michael Klaas, Alexander Schulz

Gene Regulatory Networks

Gene regulatory networks (GRNs) are used in bioinformatics to model interaction between genes and transcription factors (proteins).

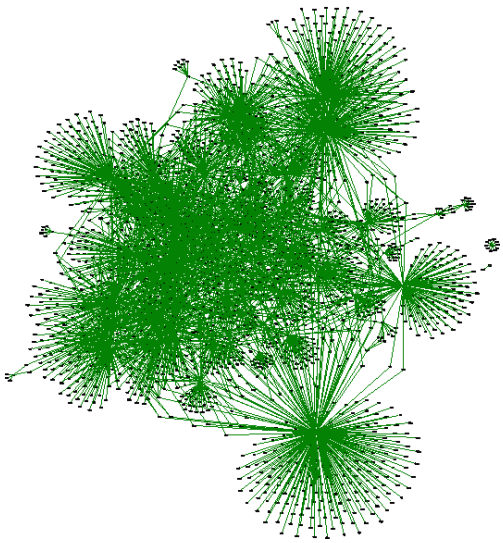


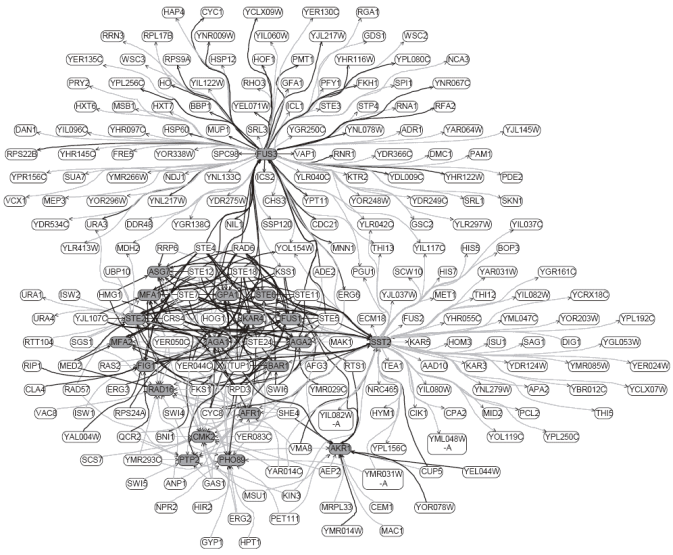
Weighted directed graph: Weights represent strength of gene activation or repression.

Fachhochschule Hannover
University of Applied Sciences and Arts
Fakultät IV – Wirtschaft und Informatik

Gene Regulatory Networks

GRNs tend to be large, but have a specific structure: hub nodes with large number of outgoing edges.

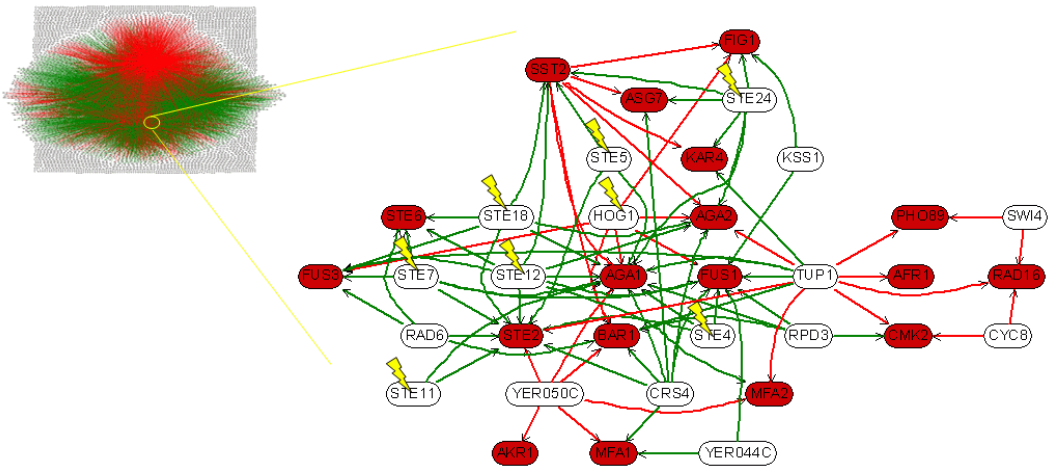




Analysis of Gene Regulatory Networks

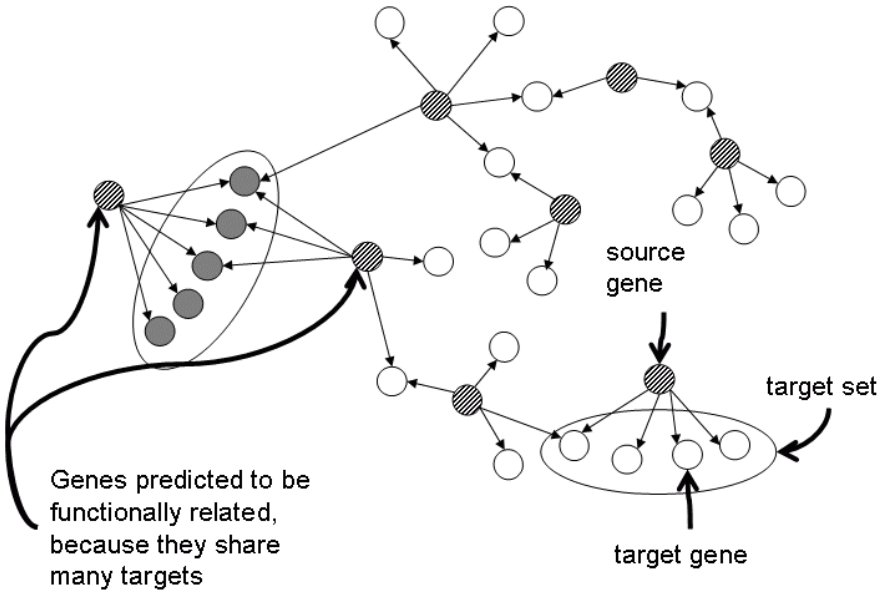
Main aim of GRN analysis:
Finding “important” genes and understanding their function.

Subnetworks are of interest, e. g., red core set and next neighbors.



Analysis of Gene Regulatory Networks

Example: Finding functionally related genes.



Analysis of Gene Regulatory Networks

Other things of interest for GRN analysis:

- Connectivity: strongly or weekly connected components.
- Clustering of nodes and edges.
- Statistics, e. g., distribution of node degrees (number of incoming and outgoing edges).
- Mutant networks: change of graph properties after removing "important" nodes or edges.
- Randomization: change of graph properties after randomly changing nodes or edges.

Graph Drawing

Aesthetic criteria of graph drawing may contradict each other:

- Crossing minimization.
- Bend minimization.
- Area minimization.
- Angle maximization.
- Length minimization.
- Symmetry preservation.
- Clustering.

Graph Drawing

Classic approach: force-directed layout.

- Define energy function $U(\boldsymbol{r})$ of node positions $\{\boldsymbol{r}\}$, e. g., based on a spring model with optimal node distance d_0.
- Start with a random layout (or better, a sophisticated guess).
- Compute forces $\boldsymbol{F}(\boldsymbol{r}) = -\nabla U(\boldsymbol{r})$ acting on nodes.
- Minimize energy using simulated annealing.

Fruchterman-Reingold model (with nodes/vertices V, edges E),

$$U = \sum_{\{u,v\}\in E} \frac{1}{3\,d_0} \left\|\boldsymbol{r}(u) - \boldsymbol{r}(v)\right\|^3 - \sum_{\{u,v\}\in V^2} d_0^2 \ln \left\|\boldsymbol{r}(u) - \boldsymbol{r}(v)\right\| .$$

- Attractive forces: $\left\|\boldsymbol{F}_{\text{att}}\right\| \propto \frac{\|\boldsymbol{r}(u)-\boldsymbol{r}(v)\|^2}{d_0}$.
- Repulsive forces: $\left\|\boldsymbol{F}_{\text{rep}}\right\| \propto \frac{d_0^2}{\|\boldsymbol{r}(u)-\boldsymbol{r}(v)\|}$.

Requirements

Requirements for a GRN analysis tool:

- Network visualization, using graph drawing methods, generating “nice” and usable graph layouts.
- Network analysis, using fundamental graph algorithms (e. g., connected components, shortest paths).
- Extensibility by new algorithms implemented as plugins, preferably in Java.
- Possibility to use existing plugins within new plugins.
- Platform-independent open-source software.
- Command logging/scripting functionality, batch execution.

Existing Solutions

Existing software tools and libraries (selection):

- Tools: Cytoscape, Pajek, BioLayout Express 3D.
- Matlab toolboxes: Bioinformatics, SimBiology.
- Graph libraries: LEDA, Graphviz/Grappa, JUNG, yFiles.

Miscellaneous drawbacks:

- Exclusive focus on graph drawing, missing extensibility.
- Proprietary software.
- Non-open-source and/or platform-specific software.
- “Strange” programming languages like Python or C++.

System Architecture: OSGi

OSGi (formerly *Open Services Gateway initiative*) provides the core plugin architecture of Biogranat.

- Open standard, developed since 1999 by *OSGi Alliance*.
- Implementations: Apache Felix, Eclipse Equinox, many others.
- Dynamic, component-based service platform for Java.
- Service-oriented architecture on local machine.
- OSGi framework manages lifecycle of bundles (components).
  - Bundles can be installed, updated, and uninstalled at runtime.
  - Bundle dependencies are resolved.

System Architecture and Data Model

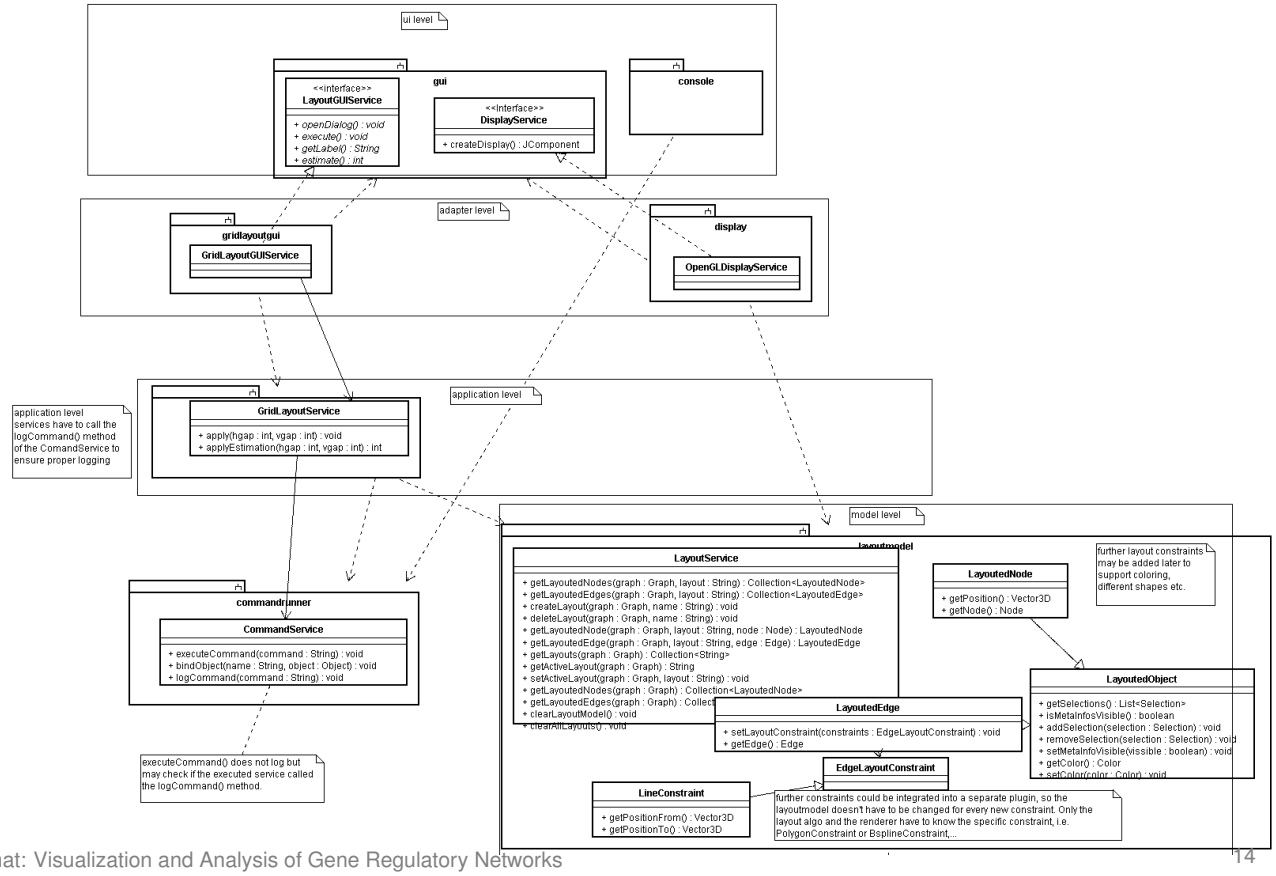
System architecture

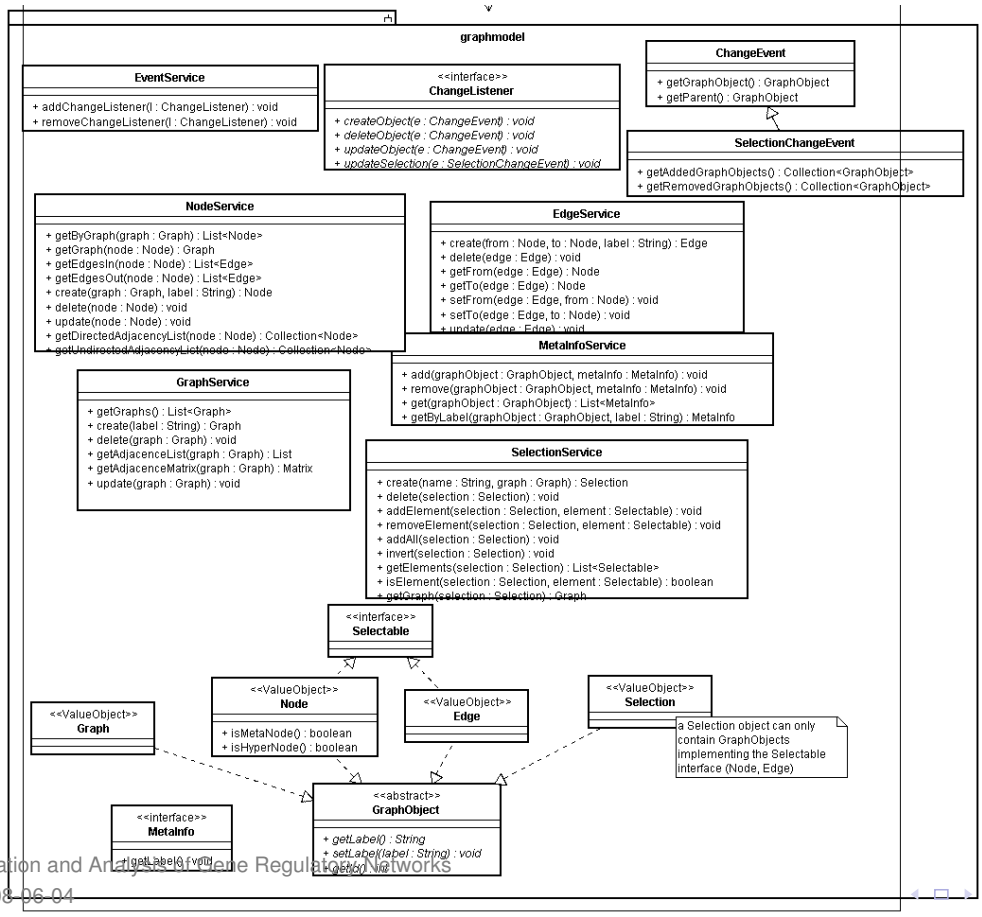
- Every system component is implemented as an OSGi bundle, including GUI, OpenGL renderer, and graph model.
- Every bundle provides one or more services, e. g., graph, node, and edge services, connected component service.
- User interface and implementation of a bundle are separated.

Data model

- Graph model: nodes, edges, selections, meta information (e. g., analysis results).
- Graph layout model: node and edge positions, colors, etc.

Fachhochschule Hannover
University of Applied Sciences and Arts
Fakultät IV – Wirtschaft und Informatik





Example: Accessing Services

```
// get bundle context from bundle activator
BundleContext context = Activator.getContext();

// create dynamic graph service tracker
ServiceTracker activeGraphServiceTracker =
    new ServiceTracker(context,
        ActiveGraphService.class.getName(), null);
activeGraphServiceTracker.open();

// access graph service
ActiveGraphService activeGraphService =
    (ActiveGraphService)
        activeGraphServiceTracker.getService();
```

Example: Obtaining Graph Information

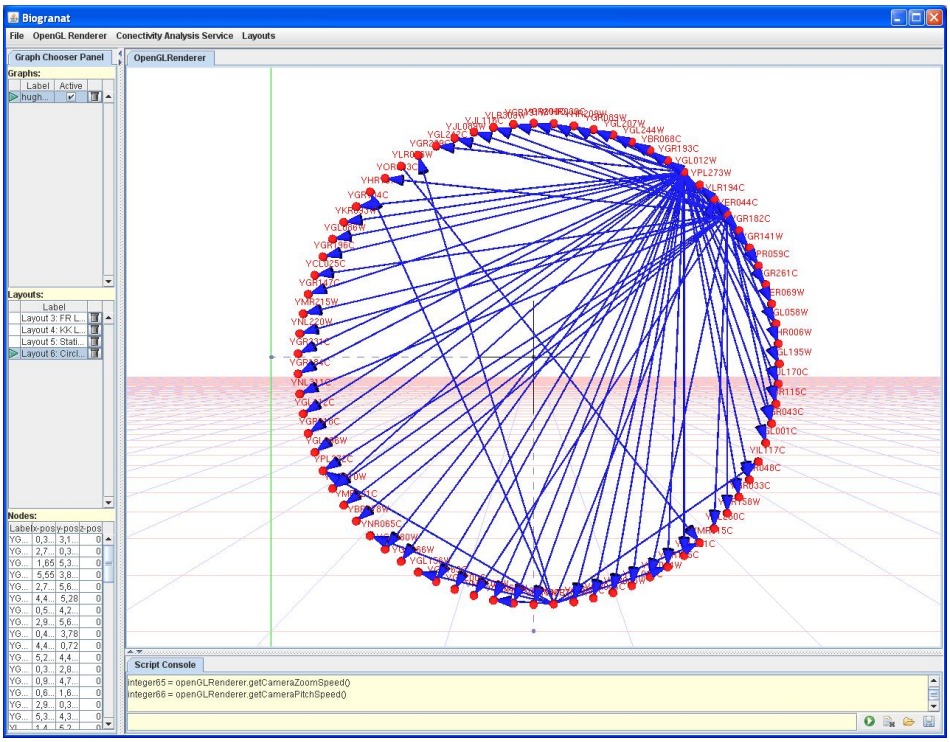
```
ArrayList<Graph> graphs = (ArrayList<Graph>)
    activeGraphService.getAll();
String message = "Graphs: " + graphs.size() + "\n";
for (Graph graph : graphs) {
  ArrayList<Node> nodes = (ArrayList<Node>)
      nodeService.getByGraph(graph);
  ArrayList<Edge> edges = (ArrayList<Edge>)
      edgeService.getByGraph(graph);
  message += "- Graph " + graph.getId() + " -\n"
      + "Nodes: " + nodes.size() + "\n"
      + "Edges: " + edges.size() + "\n";
}
JOptionPane.showMessageDialog(null, message);
```


Features

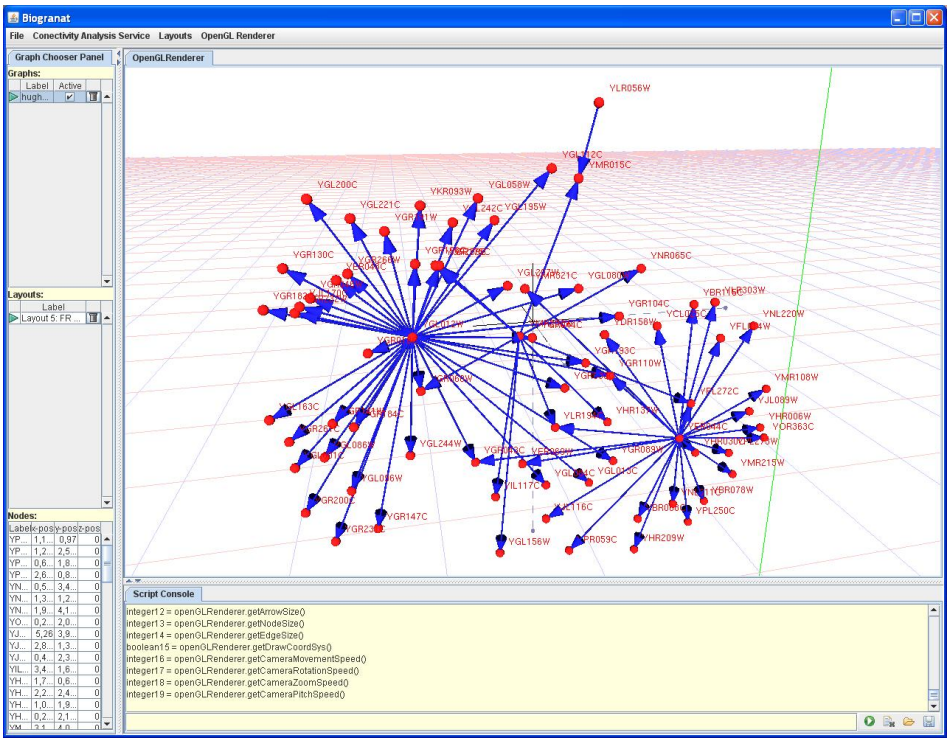
Overview of user features of Biogranat:

- Different graph layouts using JUNG library (*Java Universal Network/Graph Framework*), including circle, Fruchterman-Reingold (FR), and Kamada-Kawai (KK) layouts.
- Command logging using Java Scripting Framework (Java 6 `javax.script`).
- Configurable 3D OpenGL renderer JOGL (*Java OpenGL Bindings*).
- Import and export of graphs in ASCII and XML-based GraphML formats.
- Fundamental graph algorithms, e. g., connected components, reverse topological sort.

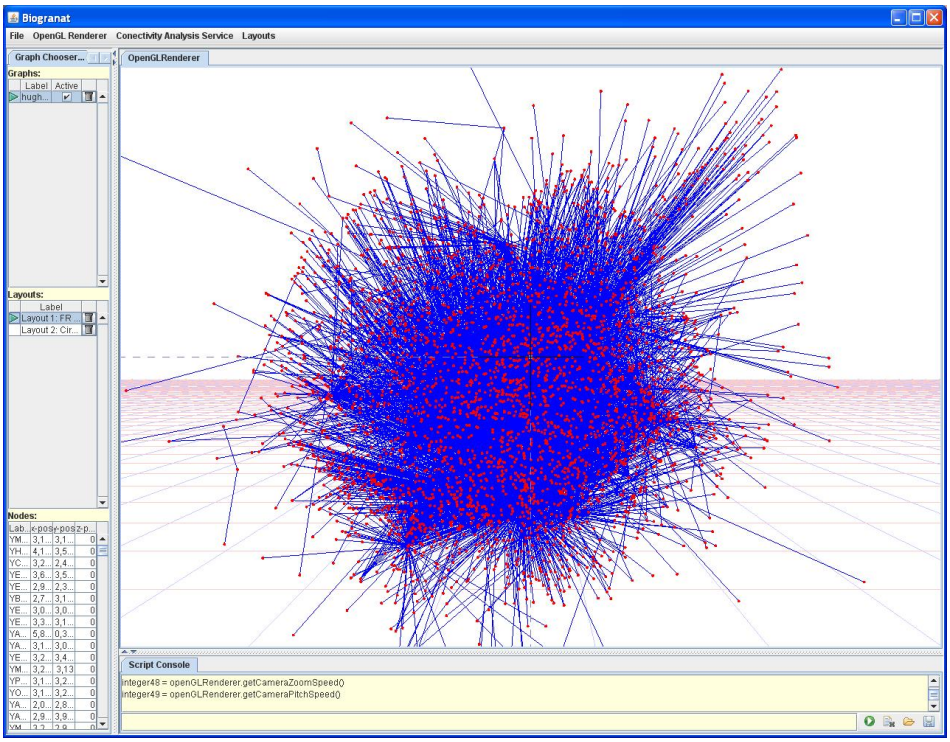
Demonstration: Circle Layout



Demonstration: Fruchterman-Reingold (FR) Layout



Demonstration: Large Graph (3197 nodes)



Project Organization

Bilateral meetings take place at Hannover and London.

- DAAD/British Council travel grant for 2007–2009 (PPP/ARC, *Programm des projektbezogenen Personenaustauschs/Academic Research Collaboration*).

Most work has been done within student projects.

- Bachelor thesis in summer semester 2007.
- Master project in winter semester 2007/08, OSGi architecture.

Drawbacks of student projects:

- Limited documentation and long-term support.
- Missing continuity.

Ongoing Work

Master and PhD theses at London

- Analysis algorithms for bioinformatics research, e. g., studying effects of network randomization.

Bachelor and Master theses at Hannover

- Vector graphic (SVG) export for 2D projections of 3D layouts.
- Specialized graph and tree layouts for gene regulatory networks.
- Wrappers for further graph libraries, e. g., *Graphviz*.
- Computing 3D graph layouts on the graphics processing unit (GPGPU approach).

Conclusion

Key features of Biogranat graph analysis tool:

- Extensible plugin architecture based on Java 6 and OSGi.
- 2D and 3D graph layouts using JUNG library and OpenGL renderer.
- Command logging using Java Scripting Framework.

Next tasks:

- Stabilizing the graph model kernel.
- Implementing algorithms for analyzing gene regulatory networks.
- Adding wrappers for further graph libraries.
- Adding interfaces to bioinformatics tools.

Literature I

- Battista, G. di, P. Eades, R. Tamassia, and I. G. Tollis: *Graph Drawing*.
  Prentice-Hall, Upper Saddle River, NJ, 1999.
- Jünger, M. and P. Mutzel (eds.): *Graph Drawing Software*.
  Springer, Berlin, 2004.
- Kaufmann, M. and D. Wagner (eds.): *Drawing Graphs*.
  Springer, Berlin, 2001.
- Klaas, M.: *Graph Drawing: Entwicklung einer Software zur Visualisierung großer Graphen für biomedizinische Anwendungen*.
  B. Sc. thesis, Fachhochschule Hannover, 2007.

Literature II

- Rung, J., T. Schlitt, A. Brazma, K. Freivalds, and J. Vilo: *Building and analysing genome-wide gene disruption networks*.
  Bioinformatics, 18:S202–S210, 2002.
- Schlitt, T.: *Towards Reverse Engineering of Gene Regulatory Networks*.
  Ph. D. thesis, University of Cambridge, 2003.
- Schlitt, T. and A. Brazma: *Modelling in molecular biology: describing transcription regulatory networks at different scales*.
  Phil. Trans. R. Soc. B, 361:483–494, 2006.
- *JUNG graph library*.
  `http://jung.sourceforge.net/`.
- *OSGi service platform*.
  `http://www.osgi.org/`.