

UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Computer Science Curriculum

Jordan Valdma

**gPathways : a visualization tool to identify global
connections among biological pathways**

Master's Thesis (30 ECTS)

Supervisor: Balaji Rajashekar, PhD

Tartu 2014

gPathways : a visualization tool to identify global connections among biological pathways

Abstract:

Molecular biology experiments involving 'omics' studies always yield in several gene lists that are important for their study. These gene lists have to be further explored by one or many computational methods to gather additional information about the genes.

One such method is to query a biological pathway database. These databases have important information about proteins which are involved in a process where a series of reactions occur between the molecule that are guided by proteins leading to a product. Such reactions are happening in every cell constantly.

The current available tools can query a gene list on a pathway database and will result in a table showing genes associated to different pathways. The tools can also provide network visualization on gene interactions from pathway analysis.

Our aim was to develop a novel interactive web-application tool for querying and visualising all pathways and their interconnectedness in a global context. We summarized all genes in a pathway as node and the edges represented the relationships between the pathways. The application also shows the result of querying genes, provide a clear overview of pathway data, while maintaining ability to query and zoom into details. Graphical rich functionalities, mentioned above, were implemented in gPathways which can be accessed from the link <http://biit.cs.ut.ee/gpathways/>.

Keywords:

Biological pathways, KEGG, genes, visualization, network

gPathways: bioloogiliste radade vaheliste ühenduste visualiseerimise ja identifitseerimise tööriist

Lühikokkuvõte:

Molekulaarbioloogia katsed 'omics' vallas, sisaldavad alati katse seisukohast olulisi geenijärjendit. Neid geenijärjendeid uuritakse edasi ühe või mitme arvutusliku meetodiga, et saada lisa informatsiooni geenide kohta.

Üks selliseid meetodeid on päringud bioloogiliste radade andmebaasis. Vastavates andmebaasides sisaldub oluline info proteiinide kohta, mis osalevad protsessis, kus toimub jada reaktsioone molekulide vahel, mis viivad produktini. Sellised reaktsioonid toimuvad igas rakus konstantselt.

Hetke tööriistad, mis pärivad geenijärjendeid radade andmebaasist, näitavad tabeleid geenide seostest radadega. Tööriistad pakuvad ka võrgustiku visualiseerimist geenide baasil, radade analüüsist.

Meie eesmärk oli arendada uudne interaktiivne veebirakendus radade omavaheliste ühenduste visualiseerimiseks ja bioloogiliste radade pärimiseks, globaalses kontekstis. Me summeerime kõik geenid rajas graafi punktina ja servad väljendavad radade vahelisi ühendusi. Rakendus näitab geeni päringute tulemusi ja annab selge ülevaate radade andmestikust, jättes võimaluse pärida ja vaadata detaile. Ülalmainitud graafiliselt rikkalik funktsionaalsus implementeeriti gPathways rakenduses, millele saab ligi lingilt <http://biit.cs.ut.ee/gpathways/>.

Võtmesõnad:

Bioloogilised rajad, KEGG, geenid, visualiseerimine, võrk

Table of Contents

1	Introduction.....	6
1.1	Motivation and background.....	6
1.2	Contributions of this work.....	8
1.3	Structure of the thesis.....	10
2	Development - Methods and Tools.....	11
2.1	Overview.....	11
2.2	Technologies.....	11
2.3	Data handling.....	13
	Data structures.....	14
	Web-application.....	15
2.4	How development was done.....	16
2.5	Environments.....	17
	Development.....	17
	Live.....	17
3	Features.....	18
3.1	Import.....	18
3.2	Used tools.....	19
3.3	Analytics.....	19
3.4	Graph.....	19
3.5	Querying genes.....	20
3.6	Querying pathways.....	24
3.7	Graph Options.....	25
3.8	Pathway selection.....	25
3.9	Zoom in to pathway.....	26
3.10	Static URL.....	26
3.11	Layout.....	26
3.12	Output.....	27
3.13	Feedback.....	28
4	Results.....	29
4.1	Example 1.....	29
4.2	Example 2.....	31
5	Conclusion.....	32
6	Future works.....	34

7	References.....	35
	Appendix.....	36
	I. License.....	36
	II. JSON input Schema	37
	III. Example query data	41

1 Introduction

1.1 Motivation and background

A biological pathway is a process within a cell - a series of actions among molecules, catalyzed by proteins which lead to a certain product or a change in the cell. Such a pathway can trigger the assembly of new products as molecules.

One way to understand biological pathways is to bring out an analogy with the concepts of object oriented programming¹. When we think about an organism as a software program, which could cope with some tasks, then classes/objects within this software could represent biological pathways. A function/method in this class could be represented as a process within a biological pathway. Those process interchange among each other the results, just like functions in a software.

As there are tasks for software that require cooperation of multiple classes there are tasks for an organisms that require multiple pathways functioning together as a whole to process substance or produce a products. These can be called pathway networks or metabolic networks. In a process of converting sugar to energy several reactions are involved in several pathways. This happens in the aerobic respiration pathway where the following pathways are involved Glycolysis → Pyruvate Decarboxylation → Citric acid cycle → Oxidative phosphorylation (Electron Transport Chain + ATP synthase)².

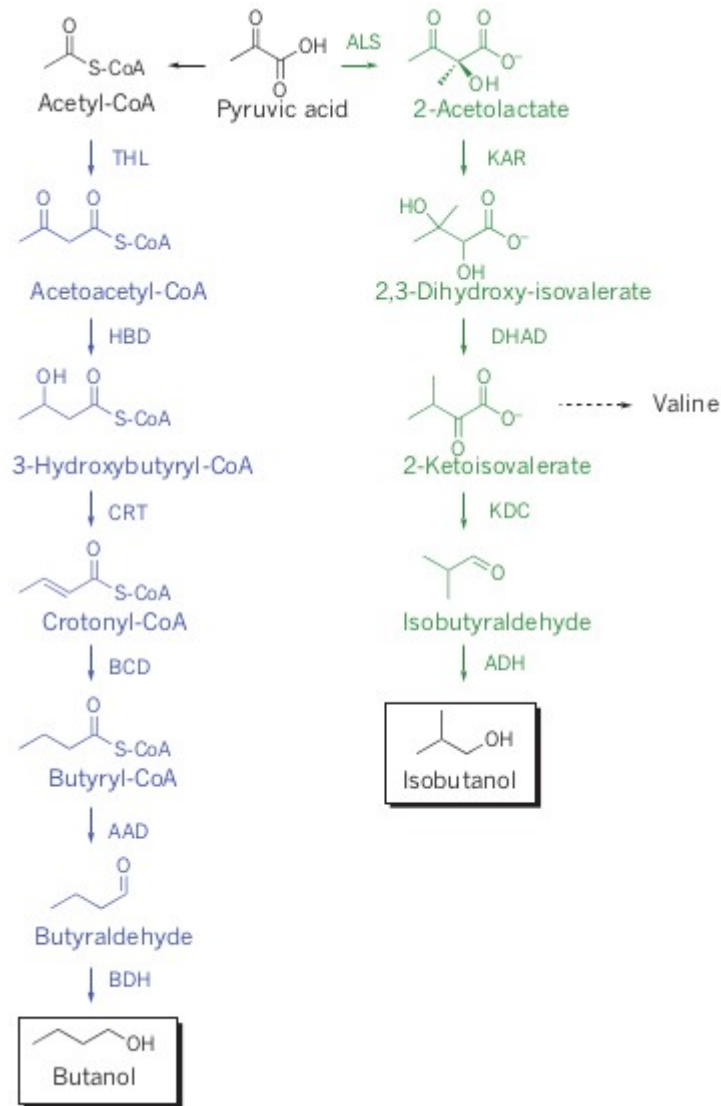
The need to understand the importance of pathways lies in one such application where metabolic pathways play a crucial role in the production of alcohol-based biofuels which are similar to petrol based fuels. We need an alternative fuel as our demands for fuel is ever growing. An alternative way is to engineer and modify the biology of the microorganisms to maximize the production of biofuels. Several pathways have to be optimized to produce biofuels efficiently for commercial success. The following metabolic pathways are engineered for the production of alcohol-based biofuels. In blue, Clostridium's butanol pathway converts acetyl-CoA into butanol. AAD, butyrladehyde dehydrogenase; BCD, butyryl-CoA dehydrogenase; BDH, butanol dehydrogenase; CRT, crotonase; HBD, 3-hydroxybutyryl-CoA dehydrogenase; THL, thiolase. In green, the 2-keto acid pathway

¹<http://www.codeproject.com/Articles/22769/Introduction-to-Object-Oriented-Programming-Concep>

²http://en.wikipedia.org/wiki/Metabolic_network

produces isobutanol from pyruvic acid. ADH, alcohol dehydrogenase; ALS, acetolactate synthase; DHAD, dihydroxy-acid dehydratase; KAR, ketol-acid reductoisomerase; KDC, keto-acid decarboxylase. [1]

Figure 1. Reactions involved in production of biofuels [1]



There are several databases storing the gene pathway information, KEGG³ (Kyoto Encyclopedia of Genes and Genomes) is one of the popular database which comprises of manually defined functional units dealing with genomes, enzymatic pathways, and biological chemicals. The pathway database records networks of molecular interactions in the cells, and variants of them specific to particular organisms. We used KEGG as our first data source and

³<http://www.genome.jp/kegg/>

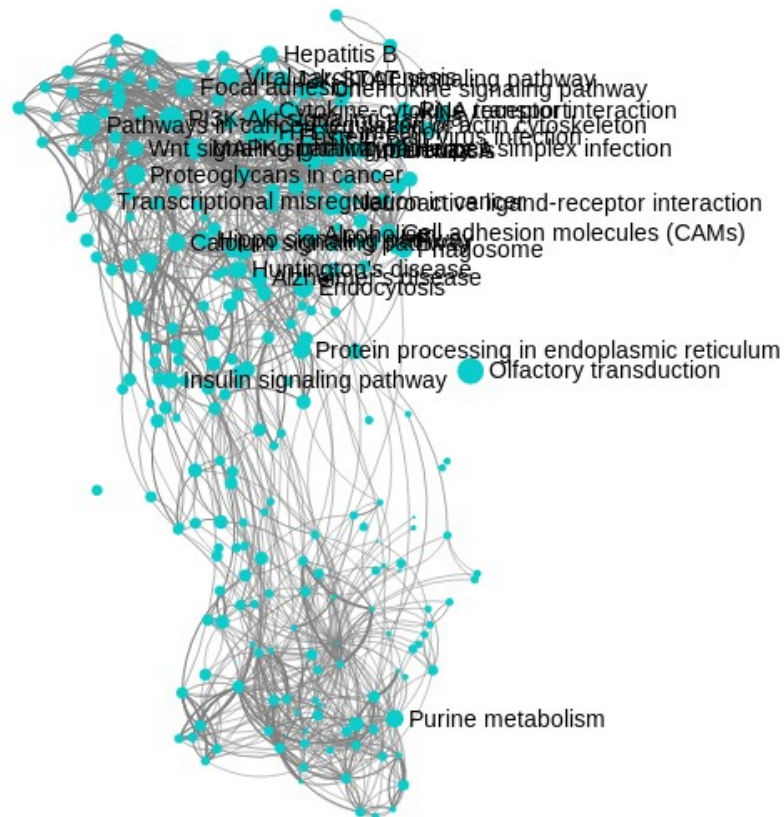
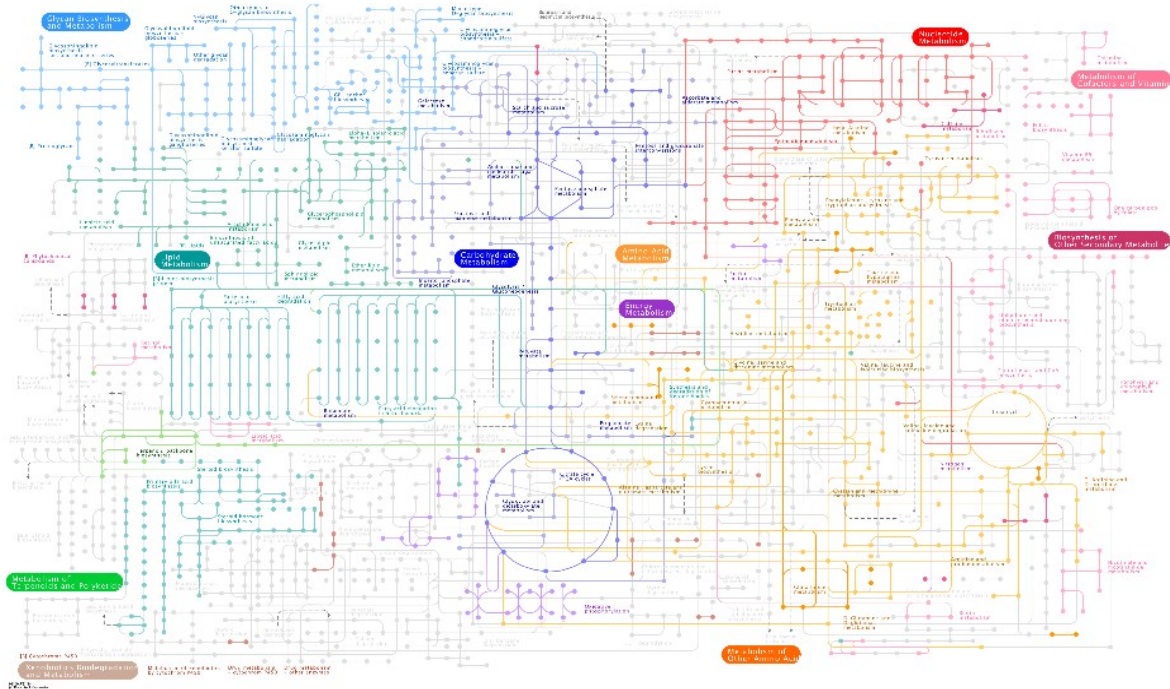
in using several web-application tools implemented a graphical and interactive pathway visualization tool, which is currently available as gPathways.

1.2 Contributions of this work

This thesis experiments with visualization of biological data. It endeavors to provide an intuitive insight into a complex network. An image is a way of presenting information where it can be easily grasped. As a result a graph representation and a couple of informative lists were chosen.

Another aspect is pathway information query and analysis tool - the network of biological pathways is generated based on the gene and pathway relationships obtained from KEGG as a first source of information. This tool provides a feature rich visual overview on the pathway repositories in a networked graph showing the global relationships within an organism. The idea is to reduce a complex network of reactions into graspable, queryable visual interface.

Figure 2. visualization of global relationships of pathways within an organism to a network.



1.3 Structure of the thesis

In section 2 technical details, methods and tools are presented. section 3 gives an overview of features. Section 4 presents some practical use-cases with example queries to assess quality of this tool. Conclusion is given in section 5 and future works are discussed in section 6.

2 Development - Methods and Tools

In this section used data, technologies and methods are described.

2.1 Overview

On the server-side, current application is based on Grails framework, which is an open source web application framework that uses Groovy language (which is based on the Java platform). It is intended to be a high-productivity framework by following the "coding by convention" paradigm. It actually uses existing Java technologies such as Hibernate⁴ and Spring under a single interface.⁵

On front-end JavaScript with extensive use of HTML5 and canvas, with SigmaJS⁶ for graph are used.

2.2 Technologies

Following is a list of used technologies, in gPathways, with usage written at the end, after each item.

- Java Virtual Machine (JVM)⁷ - is a cross-platform Java bytecode interpreter, runs Java applications. Used as a base technology.
- Groovy⁸ - a modern and dynamic language that executes on JVM. Used as server-side language
- Grails⁹ - Open Source, full stack, web application framework for the JVM, which is built upon Spring MVC and uses Groovy with convention over configuration in mind. Used for web-application.

⁴<http://hibernate.org/>

⁵<https://grails.org/>

⁶<http://sigmajs.org/>

⁷http://en.wikipedia.org/wiki/Java_virtual_machine

⁸<http://groovy.codehaus.org/>

⁹<https://grails.org/>

- H2¹⁰ - Java SQL database. Used for development environment.
- PostgreSQL¹¹ - Extensible, recognized, open source database server. Used for live environment.
- Twitter Bootstrap¹² - Front-end design framework. Used for web-application layout.
- SigmaJS¹³ - JavaScript library for drawing graphs. Used for web-application front-end.
- jQuery¹⁴ - JavaScript library for DOM traversal and manipulation. Used for web-application front-end.
- Lodash¹⁵ - JavaScript performance and utility library. Used for web-application front-end.
- Git¹⁶ - Free, open source distributed version control system. Bitbucket¹⁷ provider, used for source code management.
- Google Analytics¹⁸ - Website analytics tracker. Used for web-application visit and query tracking.
- Mail server - gMail¹⁹ used for sending messages of feedback.
- Apache Tomcat²⁰ - An open source implementation of Java Servlet²¹ and JavaServer Pages²² technologies, which allows running Java web-applications. Used for running web-application.

10<http://www.h2database.com>

11<http://www.postgresql.org/>

12<http://getbootstrap.com/>

13<http://sigmajavascript.com/>

14<http://jquery.com/>

15<http://lodash.com/>

16<http://git-scm.com/>

17<http://bitbucket.org/>

18<http://www.google.com/analytics/>

19<https://www.gmail.com>

20<http://tomcat.apache.org>

21http://en.wikipedia.org/wiki/Java_Servlet

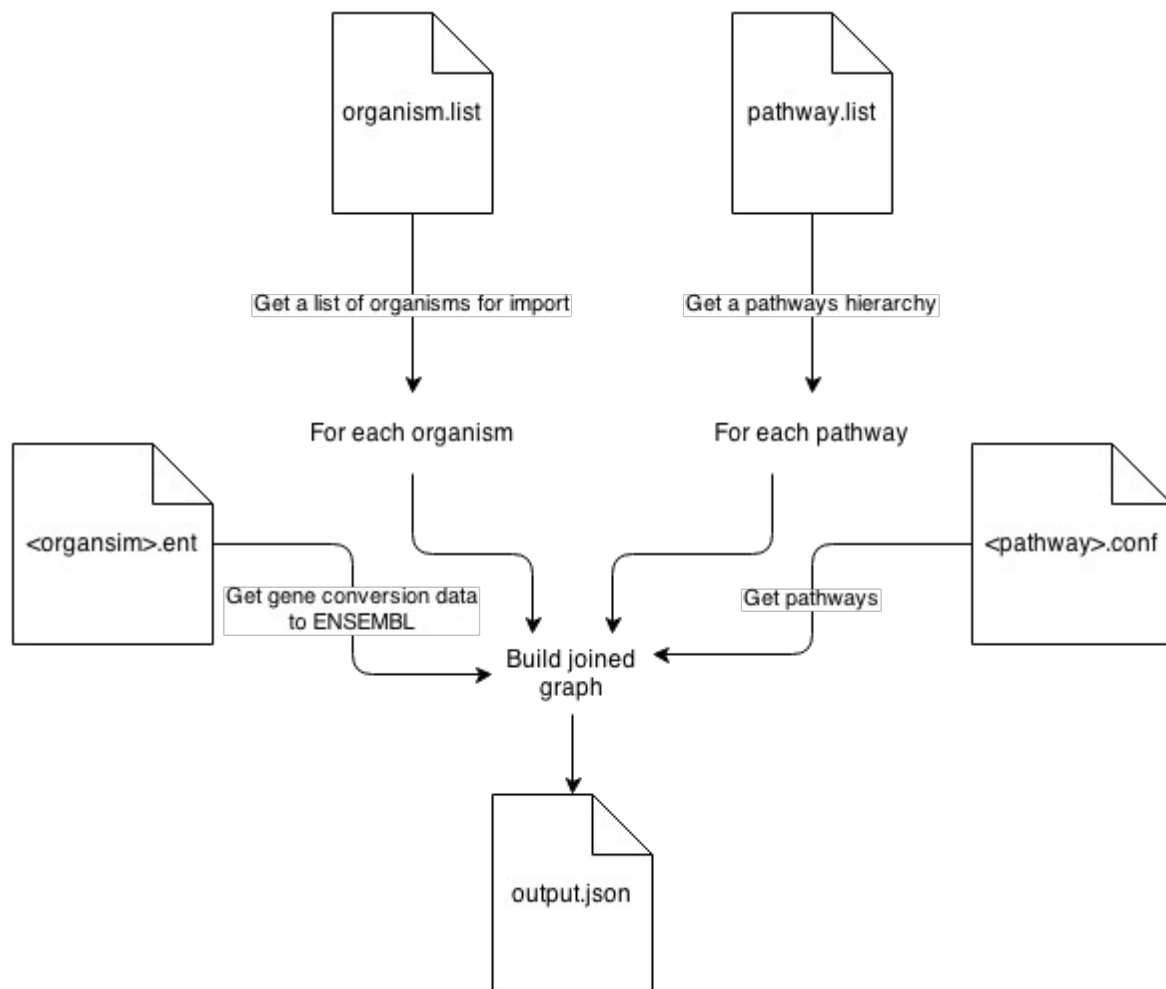
22http://en.wikipedia.org/wiki/JavaServer_Pages

- Apache HTTP Server²³ - A secure, efficient and extensible open source server. Used as live HTTP server.

2.3 Data handling

The KEGG database has been in development by Kanehisa Laboratories since 1995. It is a reference base for integration and interpretation of large-scale molecular data sets generated by genome sequencing and other high-throughput experimental technologies. Currently it is used as a resource for understanding high-level functions and utilities of the biological system, such as the cell, the organism and the ecosystem, from molecular-level information, especially large-scale molecular datasets generated by genome sequencing and other high-throughput experimental technologies [3]. Below, figure 3, explains how we use KEGG data.

Figure 3. Working with KEGG data



²³<http://httpd.apache.org/>

Files from KEGG were grouped into folders by organism. Each pathway has a separate file with KEGG convention, which had to be read and parsed in order to extract firstly the genes in the pathway and secondly pathways interconnectedness.

Besides pathway files, organism.list and pathway.list, which are internal additional files for indexes were used. Last one also represented the hierarchy of pathways within an organism – this allowed us to group pathways into classes and subclasses ie. Cellular Processes and Cell growth and Death. This allowed to create a JSON output file, which we actually input to the application, which in turn validates the data and uses it for features. Organism.list and pathway.list are combined with pathway files <pathway>.conf and ensembl id-s file <organism>.ent and joined for output.json - a graph representation. This is done using Groovy as a language.

Generation outputs a JSON file and a log into console as follows:

Reading ensembl ID-s

Rendering: Homo sapiens

Reading ensembl ID-s

Rendering: Mus musculus

Reading ensembl ID-s

Rendering: Arabidopsis thaliana

...

Data structures

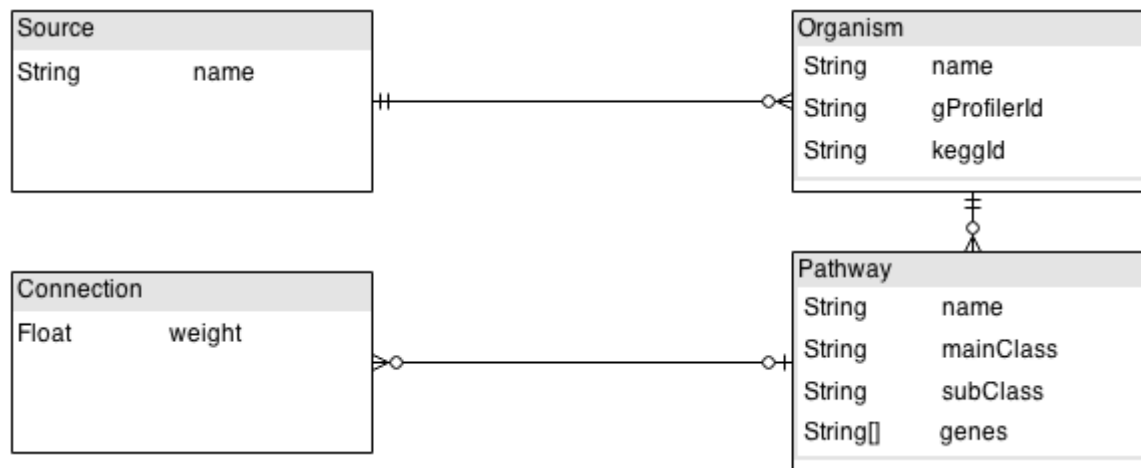
Imported data has a JSON²⁴ interface, which is brought out in appendix II. The meaning of this interface is to help adding further data sources. Just like Kegg is added for example Wikipathways²⁵ can be added as a data source. As application keeps its internal representation of every added source, the data must be converted to JSON format as defined by schema.

²⁴<http://json.org/>

²⁵<http://wikipathways.org/index.php/WikiPathways>

Figure 4 displays internal representation/model, which is represented in database as well.

Figure 4. Internal data model



Web-application

Web-application uses Grails framework. It was chosen over Spring MVC²⁶ for its quick prototyping features and convention over configuration idea. In front-end SigmaJS was used, even before 1.0 version²⁷ and after that changes to API²⁸ required a rewrite on client side.

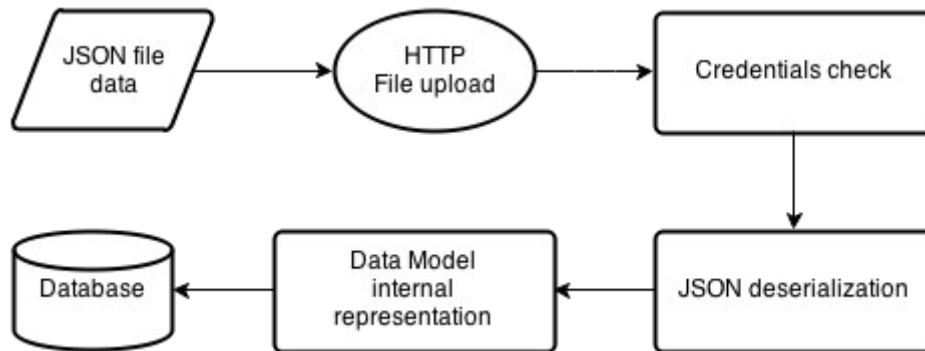
There is an overview, in section 2.3, of how to put KEGG data together for the application. Here is a data flow diagram on figure 5 of what happens on inputting the generated file to application.

²⁶<http://spring.io/>

²⁷<http://semver.org/>

²⁸http://en.wikipedia.org/wiki/Application_programming_interface

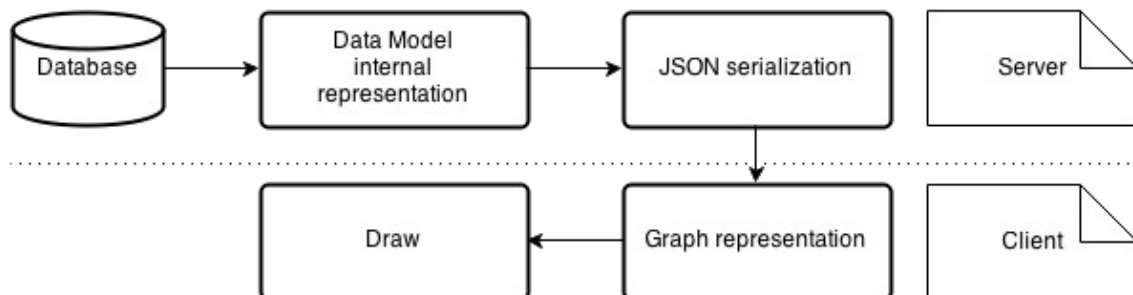
Figure 5. Data import



In section 3.1 import is fully described as a feature.

When web-application is requested from server and application has been loaded into a web-browser, it asks for graph data. Data flow looks like on figure 6 below.

Figure 6. Graph data flow to client



2.4 How development was done

On the organizational and planning side, regular meetings were held about status, requested features and any questions. After that there was usually a deadline for work, after what another meeting was held. For an overview of features, user stories, Google Docs²⁹ was used. For more detailed, technical tasks and bugs, Bitbucket's issue list was used.

During development initial feedback from interested sources, was valued and features, layout were put together accordingly.

²⁹<http://docs.google.com>

2.5 Environments

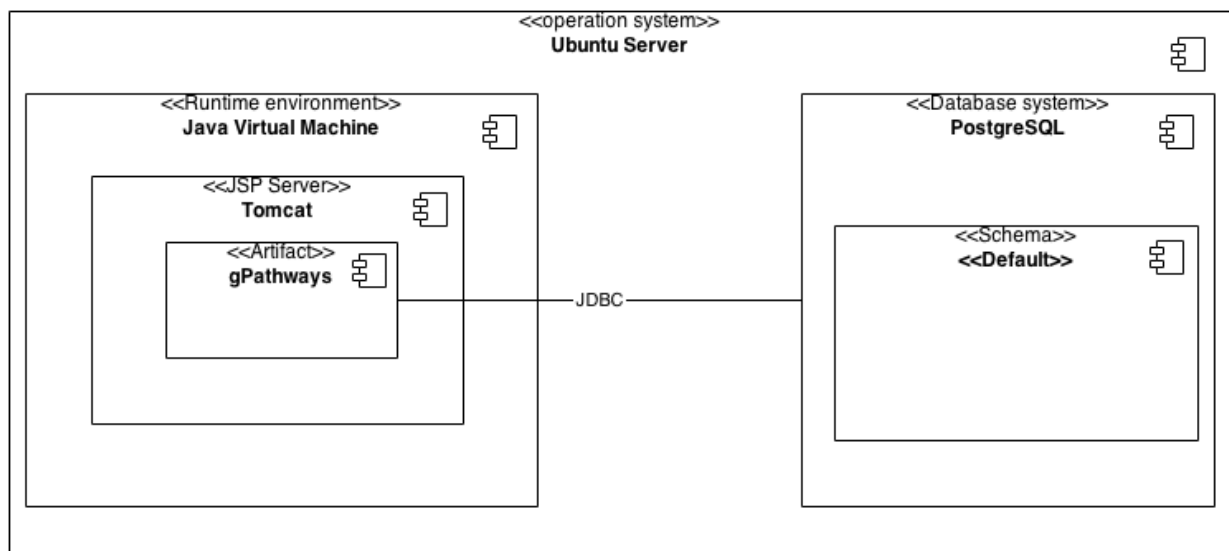
Development

Development environment was run with Grails built-in Tomcat server, initialized with a command `grails run-app`. H2 database, in-memory version as application data source provider was used.

Live

Live environment resides in a virtual machine, which has an instance of Tomcat and Apache HTTP, which serves HTTP requests. PostgreSQL is used as a database engine.

Figure 7. Live environment



3 Features

In this section a list of features, there were implemented in the application, is presented. There is some background information given on implementation alternatives and example usage scenarios whenever possible.

Every feature, whenever possible, is followed by an example and an image.

3.1 Import

Importing is for application, a basic functionality that allows administrators to modify the data contained in database. Importing is done through JSON file, that should correspond interface mentioned in 2.3. In that section also the data flow implemented for incoming source is also elaborated.

JSON file needs to be uploaded to the web, with an administration password, which is take from configuration.

Figure 8. Administration tool



The image shows a web interface for the 'gPathways administration tool'. At the top, the title 'gPathways administration tool' is displayed. Below the title, there is a section labeled 'Pathways JSON file'. This section contains a 'Choose File' button followed by the text 'output.json'. Below this is a text input field containing a series of dots, representing a password. At the bottom of the form is a 'Submit' button.

On success application returns the list of imported organisms joined with imported pathways as exemplified below

Source: KEGG

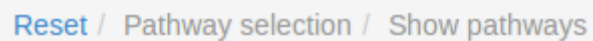
-Organism: Homo sapiens

--Pathway: *Metabolic pathways*
--Pathway: *Glycolysis / Gluconeogenesis*
--Pathway: *Citrate cycle (TCA cycle)*
...

3.2 Used tools

Used tools list resides in the top of user interface and serves a purpose of helping to orient within the many features of the application. Whenever an action is carried out that modifies the state of global pathway graph, it is added the bar. This feature was implemented in the latter part of development process and was a direct result of a user request after being overwhelmed and confused as what graph options were currently having an effect.

Figure 9. List of used tools



Reset / Pathway selection / Show pathways

The first element in the used tools list is always “Reset”, which is a link. Upon clicking the reset button the user interface will be reset and all options in effect will be reset as well.

3.3 Analytics

As there is a need to analyse usage of this application Google Analytics³⁰ is included. It is usage tracking tool with variety of functionality and output. In gPathways applications every visit and a query is collected as usage and logged.

3.4 Graph

Graph is single most important feature in the application – it visualized pathways and their interconnectedness. Graph can be zoomed and panned. Node/pathway size on the graph represents a total number of genes within the pathway. Edges between nodes are ordered such

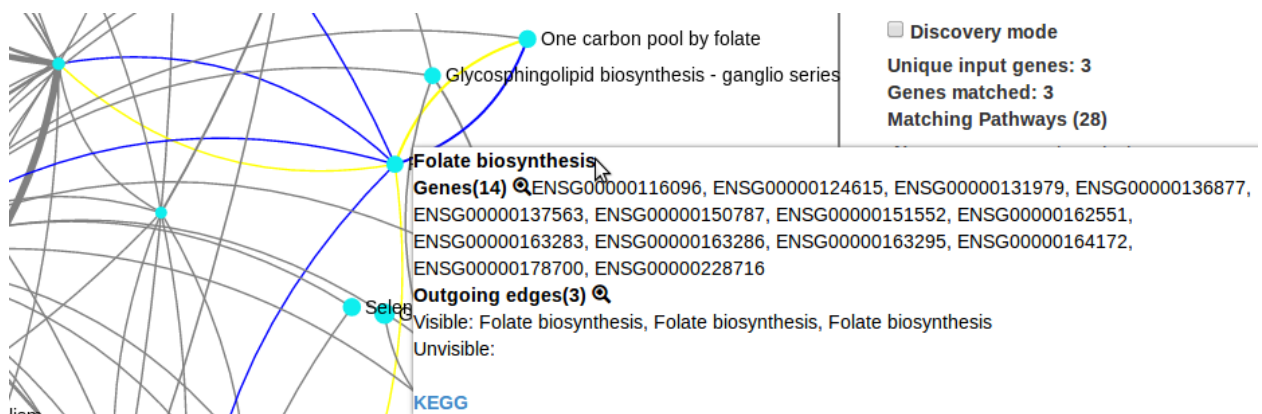
³⁰<http://www.google.com/analytics/>

that: clockwise curves display outgoing and counter-clockwise, incoming edges. Edge thickness is calculated by the number of shared product between individual pathways and is presented in a logarithmic scale.

When hovering a node edges are colored: outgoing edges are colored yellow and incoming edges blue. This feature was developed to enable user browse graph and connections easily. On hover pathway information window is displayed which contains following information:

- Pathway name
- Genes and count in pathway - on clicking, reveals the names for genes for copying and highlights those that are matching a query
- Outgoing edges and count - on clic, reveals the names of pathways with labeled by visibility
- Link to KEGG - a link to corresponding KEGG pathway for detailed introspection

Figure 10. A selected pathway showing futher details



3.5 Querying genes

The second thing in the layout that draws attention, after the graph, is a label “Querying genes” and a text field, where one can input gene/list of genes, gene time series or gene times series with values. For each of the three options mentioned, there is an example query, which can be accessed by clicking on the links (example 1, example 2, example 3) just in the help text, below the input.

Next overview of each query type is presented:

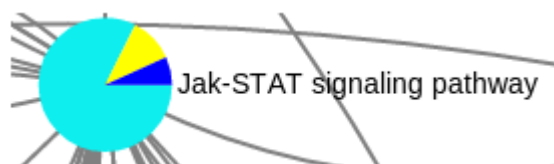
1. gene / list of genes – allows user to query one gene or a list of genes separated by commas, examples: “ENSG00000125740” or “ENSG00000125740, ENSG00000036565”. Each pathway on the graph that has a match on the query will be colored as a pie: one part showing the proportion of unmatched genes in the pathway and other portion matched genes.

Figure 11. Simple query result



2. Gene time series – this query type follows the convention with commas, but an additional character “|” (pipe) is introduced, which allows user to separate genes / gene lists into time series. For each time series a sector of the matched pathway pie will be created. An example query would be: “ENSG00000125740, ENSG00000036565, ENSG00000005339 | ENSG00000026103, ENSG00000102882”.

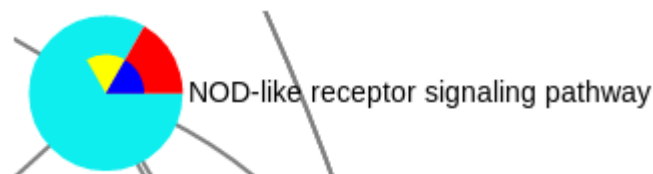
Figure 12. Time series query result



3. Gene time series with values – this type of query goes on with commas and pipes, but introduces a different input layout, numeric values and a cutoff value(s) separated by comma, marked inside curly braces “{}”. For each time series one gene and numeric values, separated by commas, is expected. As before each time series will generate an additional segment on the pie, but cutoff value(s) is used to create layers in the pie. With this query type unmatched values are proportional to matches. An example “ENSG00000125740: 1,2,3 | ENSG00000102882: 1,2,3 {2}”. Here the cutoff value

will split each time series values into two groups (currently : 1,2 and 3) each time series gene is then matched to pathways to find the proportion of how many values for each cutoff is matched. One can also enter multiple cutoffs ie. “ENSG00000125740: 1,2,3 | ENSG00000102882: 1,2,3 {1,2}”. Then pie and values will segmented into 3 layers. For this type of input equal count of time series values for each time series is expected.

Figure 13. Time series with values query result



A feature worth noticing is that when a query is made, all nodes default color will become lighter – just to have more contrast with colors indicating matching.

There is a legend in the bottom-left of the graph, which highlights colors with corresponding labels.

Figure 14. Graph legend



By default, colors are selected so that there would a contrast among them that diminishes as there are more colors. A feature of customizing colors and labels is also allowed by introducing brackets “[]”. Where in brackets first value denotes a label and a second value, separated by a comma, denotes a color. An example with second query type: “[List1, blue] ENSG00000125740, ENSG00000036565, ENSG00000005339 | [List2, red] ENSG00000026103, ENSG00000102882”, an example with third query type: “[Time0, Red | Time1, Green | Time2, Blue] ENSG00000125740: 1,2,3 | ENSG00000102882: 1,2,3 {2}”

After a query is made, on the right sidebar query results will be displayed. First row has a label “Unique input genes” and a number which shows the count of unique input genes and when hovered will show genes as a tool-tip.

Second row is labeled “Genes matched”, followed by a number which shows count of genes matched from input and when hovered genes are displayed in a tool-tip. Third row is labeled “Matching pathways”, followed by a number which shows the count of matching pathways. Then below, a list of matching pathways will be visible, in the table as “Name, Count of genes in pathway, Count of matches, Count of unique matches”. Hovering over values applies here also. When clicked on a pathway, graph will be zoomed to corresponding pathway.

Figure 15. Query result list

Unique input genes: 2
Genes matched: 2
Matching Pathways (68)

Name	*	*	*
Alcoholism	157	6	2
Osteoclast differentiation	120	6	2
Long-term depression	55	3	1
Viral carcinogenesis	190	3	1

There is an additional option “Convert genes” to the query as a check box below the query input– when checked, each gene will be standardized to system value, for example user may enter a gene “fosb” which is converted to “ENSG00000125740” as ENSEMBL³¹ format. This is done by using HTTP requests to gProfiler³², module gConvert³³.

³¹<http://www.ensembl.org/index.html>

³²<http://biit.cs.ut.ee/gprofiler>

³³<http://biit.cs.ut.ee/gprofiler/gconvert.cgi>

Figure 16. Gene conversion feature

Convert gene names ?

Figure 17. Query tool

Querying genes

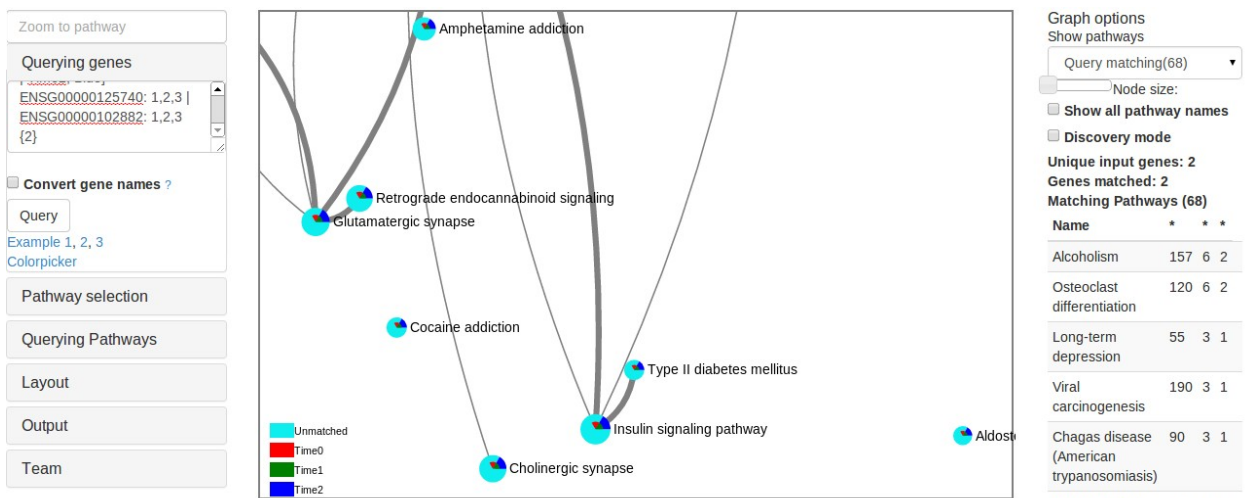
Query genes

Convert gene names ?

Query

Example 1, 2, 3
Colorpicker

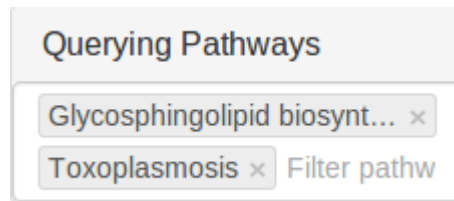
Figure 18. Query with graph and a result list



3.6 Querying pathways

Pathways can be queried by name, thus filtering out unimportant pathways.

Figure 19. Querying pathways

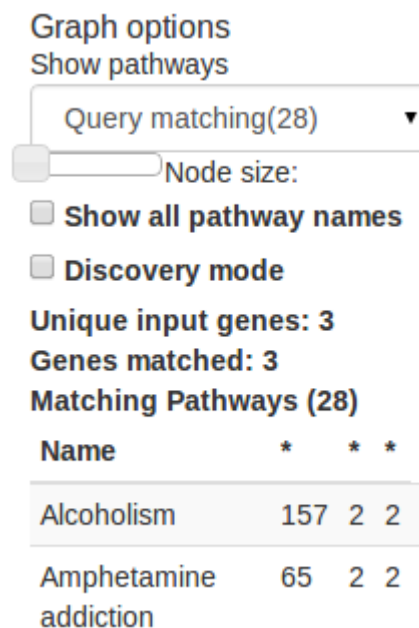


3.7 Graph Options

Graph option section at the right sidebar has following features:

1. Show pathways – selection, which will display corresponding pathways and count
2. Node size – allows to choose between different node sizes for visibility
3. Show all pathway names – use default or show all names
4. Discovery mode – When clicked on a node, node neighbors will be toggled (shown or hidden), to traverse the network
5. List of currently queried pathways

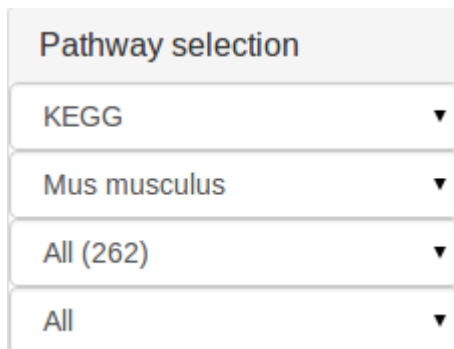
Figure 20. Graph options



3.8 Pathway selection

As mentioned in the Methods part of this document, pathways are allocated into a hierarchy that originates from KEGG. This hierarchy is visible and selectable in the “Pathway selection” sidebar on the left.

Figure 21. Pathway selection



A dropdown menu titled "Pathway selection" with four options: "KEGG", "Mus musculus", "All (262)", and "All". Each option has a small downward-pointing triangle on its right side.

3.9 Zoom in to pathway

On the top left on the left sidebar there is an auto-complete field for zooming into a pathway. Upon pressing key enter, graph will zoom to corresponding pathway on the auto-complete field.

Figure 22. Zoom to pathway



A rectangular input field with the placeholder text "Zoom to pathway" inside.

3.10 Static URL

On the top left there is a "Static URL" button, when clicked a static URL containing pathway selection and query will be displayed in the browser location.

Figure 23. Static URL



A blue rectangular button with the text "Static URL" in white.

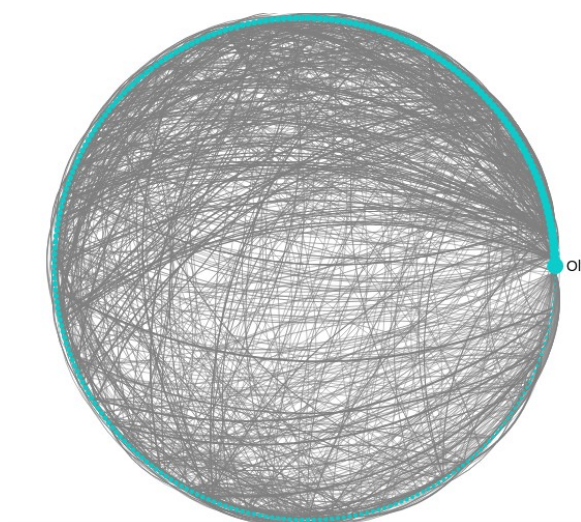
3.11 Layout

Layout option allows choosing between two layout possibilities:

1. Fixed layout, which will allocate pathways into 360 degree circle, ordered ascending clockwise. An example image is presented in the section, after the text.

2. Clustered, which will cluster pathways using Force Atlas 2³⁴ algorithm. It is a continuous algorithm, that allows you to manipulate the graph while it is rendering (a classic force-vector, like Fruchterman Rheingold, and unlike OpenOrd) and has a linear-linear model (attraction and repulsion proportional to distance between nodes). [2] It is modified to work on two ways - cluster based on edges and cluster based on query matches. Clustering by query matches can make interested pathways more visible, separated from the others and keep visualizing all of the pathways at the same time. An example can be seen in section 3.4.

Figure 24. Graph fixed layout



3.12 Output

Three types of output are available:

1. JSON – textual representation in JSON format, this is added to be used by scripts and other applications, that use JSON as an input.
2. Image – PNG image is displayed, which might be downloaded and presented in documents when necessary. Image captures current state of a graph, everything that is visible on it will be put to image as seen.
3. CSV³⁵ – a format that can be used in table calculators such as Calc or Excel. This feature is particularly good for post analysis with common tools.

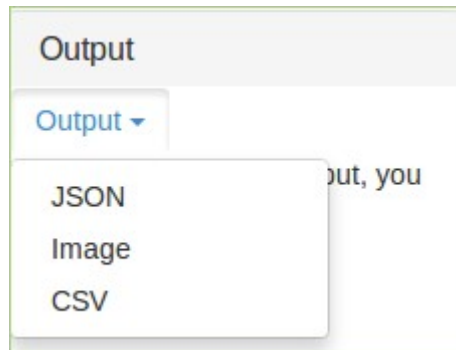
With JSON and CSV outputs, query results, with all available information, is written out as

³⁴<https://gephi.org/2011/forceatlas2-the-new-version-of-our-home-brew-layout/>

³⁵http://en.wikipedia.org/wiki/Comma-separated_values

well.

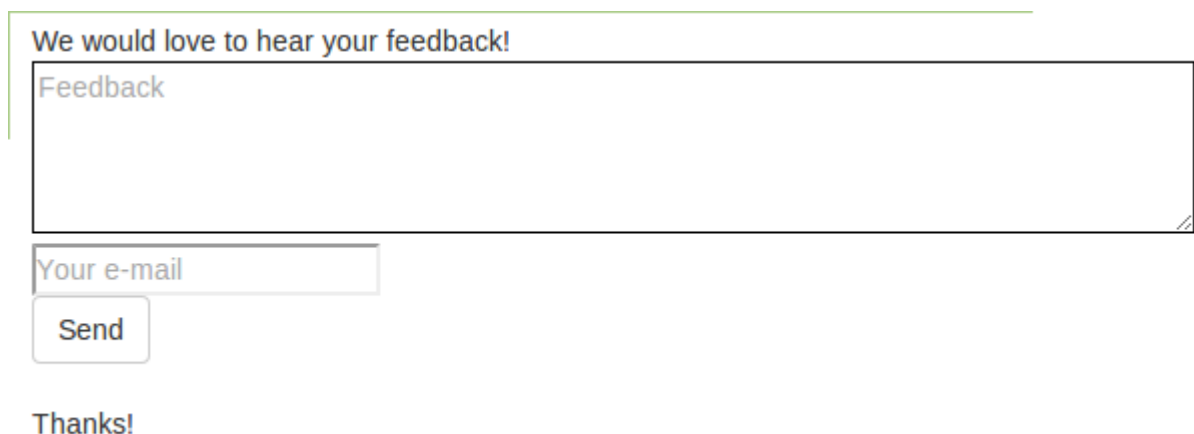
Figure 25. Output



3.13 Feedback

This is a feature for collecting feedback. As the aim of this application is to cover areas of interest for biologist and bioinformaticians, feedback is valued. On top left corner there is a “Feedback!” link, which open a form and feedback is collected and sent by e-mail to administrators.

Figure 26. Feedback

A screenshot of a feedback form. The form has a title 'We would love to hear your feedback!' followed by a large text area for feedback. Below the text area is an input field for 'Your e-mail' and a 'Send' button. The form is enclosed in a light green border. Below the form, the text 'Thanks!' is displayed.

4 Results

In gPathways we have pathway information from seven model organisms. In *Homo sapiens* there are 266 KEGG pathways, 6410 genes are represented in these pathways, 1385 edges between the pathways, 240 connected pathways. All of the pathways are organized into a hierarchy by class and subclass of pathways obtained from KEGG. We have performed two studies and gPathways results are shown below.

4.1 Example 1

We selected a microarray expression dataset E-GEOD-5281 from Alzheimer's disease . This dataset has 161 brain tissue samples from both sexes belonging to ages 63 to 102 years. We then performed an differential expression test (Limma) on the diseased and normal groups using DiffExp tool [5]. Default statistical parameters were used to obtain the up and down regulated genes.

The differential expression analysis results and their gene list can be accessed from the link http://biit.cs.ut.ee/diffexp/?project=agedbrain&platform=A-AFFY-44&dataset=E-GEOD-5281&organism=hsapiens&attribute=DiseaseState&group1=Alzheimer%27s%20Disease&group2=normal&stat_test=limma&n_max_results=&invalidate_cache=0&tp_pval=0.05&tp_foldchange=1&tp_adjmethod=fdr&

Differential analysis on the above resulted in 601 genes to be up regulated and 756 genes to be down regulated. These two gene lists are given as an input to gPathways to identify the pathways, their inter connections and their relevance to the disease.

Table 1. Example 1 query data summary

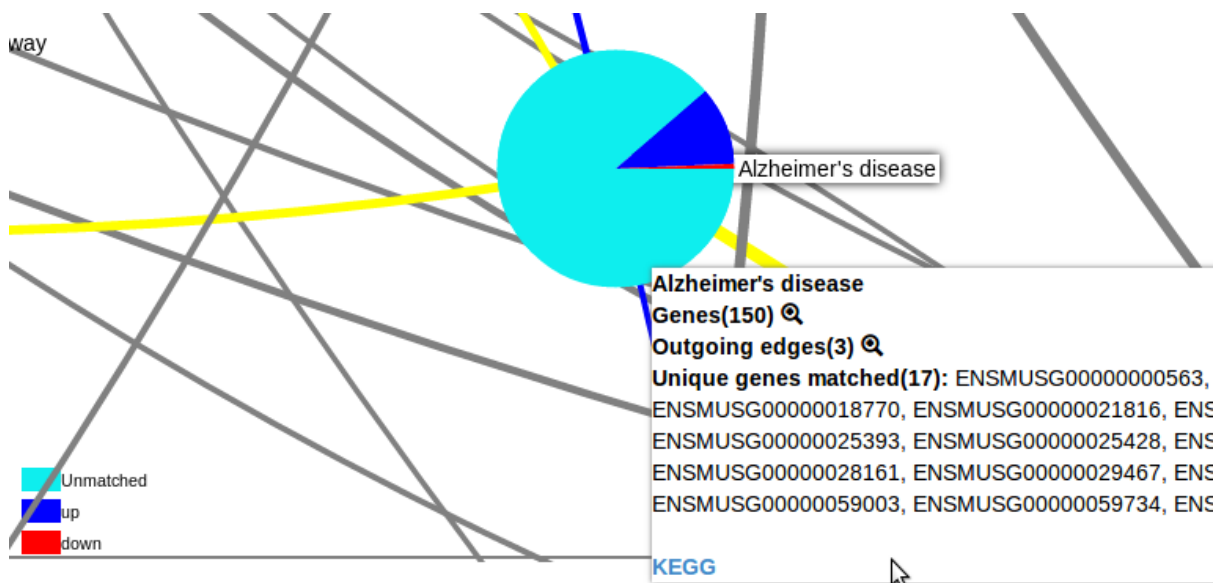
G1 vs G2	Number
Up	601

Down	756
All	1357

Given data is formatted and a query is made, see Appendix III Example query data.

Current tool recognises 717 unique input genes, which are previously converted to ENSEMBL format. gPathways identified 17 genes to Alzheimer's disease pathway, figure 27.

Figure 27. Example 1, gene lists from Alzheimer's study matching to Alzheimer's disease pathway



As seen from the figure large portion of up-regulated genes and a small list of down-regulated genes matched the Alzheimer's pathway, which is what one would expect from the query data based on the experiment selection.

By looking at the result list and graph, user can identify interesting pathways and traverse whole of the pathway network at the same time.

4.2 Example 2

We selected a time-series experiment from expression studies on skin cancer in mouse (unpublished data). The gene lists for the time series were obtained from gene expression data on an experiment involved in understanding mechanisms in cancer initiation.

As seen in figure 28, there were 1111 unique input genes, and the tool identifies „Pathways in cancer“, with 72 matching genes, as a first result.

Figure 28. Example 2, Mouse pathways in skin cancer

Unique input genes: 1111

Genes matched: 365

Matching Pathways (220)

Name	*	*	*
Pathways in cancer	295	72	24
HTLV-I infection	259	66	22
Endocytosis	206	66	22
Huntington's disease	156	57	19
PI3K-Akt signaling pathway	317	57	19
Parkinson's disease	101	51	17

5 Conclusion

We explored with the idea of summarizing metabolic pathways and visualizing them in a concise way. One of the main requirements was to visualize all pathways of an organism in a one image. The tool should be interactive, graphically enriched and allow user modification and export the analyzed the result.

To achieve these ideas, we selected KEGG as our data resource, which had information on genes and pathways. Raw data was converted into a structured format for a graph representation. The data was presented using web-application tools like HTML5 and JavaScript. gPathways is a resulting application which, allows user to browse a pathway of interest, all pathways in an organism, visually interpret the relationships between pathways, query by one or multiple gene sets, provision to search in different genes ID formats, possibility to visualize genes with expression or significant values from other bioinformatics analysis and visualize time series by genes data on pathway.

The resulting output of matching genes to pathways can be visualized as global pathway interaction maps, the pathways as nodes can be clustered, non-matching pathways can be filtered out, co-regulated and interacting pathways can be visualized. The table output shows input genes matching to KEGG pathways. The results can be saved as JSON, image, CSV format or static links. gPathways does not model the complete biological network of any organism.

A challenging tasks, connected to data analysis, was integrating data source from KEGG. It had manually curated data, where experimenting with different approaches, to extract meaningful information, was needed.

As query results within application can be output with data in multiple ways - image, JSON, CSV, it is possible and encouraged to integrate it into research pipelines. An example of Alzheimer, provided with query data gives a meaningful answer and demonstrated an accuracy of current tool. Although a need for an implementation of p-value for results is evident to better bring out relevant ones.

A goal is to enhance visual imagery - query results clustering and filtering pathways in pathway networks are implemented. The tool can handle large gene lists, but optimizing it for performance and adding more data mining features are in the list. Valued feedback from

biologists will help to prioritize important features to integrate the tool into different analysis pipelines.

6 Future works

As it can be seen gPathways can be used for variety of tasks, with visualizing interconnectedness of pathways being the main feature.

We have observed the limitations of pathway information in KEGG. Adding pathway information from other sources like Reactome will be beneficial as it has additional information about edges and regulation which is not present in KEGG. From the feedback that we gathered, this is one of the most important things. After that, next data sources, for example Wikipathways, will follow the importance set by the users.

Ideas for additional features include making queries seamless - users could copy-paste tab-separated query data in example from excel. Users would be able to insert huge queries by file upload. Users could compare organisms - ie. analyze similar networks of pathways and genes, find homologous genes. User could compare obtained results from different data sources. User could add or remove pathways from the output. Users could request new organisms that will be automatically added by update scripts. Add IntergenomeDB³⁶ for gene conversion, just like gConvert in gProfiler. Integrate Gene Ontology³⁷ in similar lines with KEGG hierarchy. Integrate Protein Atlas³⁸ for proteins. In summary, any hierarchical data in format of pathways and genes can be added to gPathways.

36<http://integromedb.org/>

37<http://www.geneontology.org/>

38<http://www.proteinatlas.org>

7 References

- [1] Pamela P. Peralta-Yahya, Fuzhong Zhang, Stephen B. del Cardayre & Jay D. Keasling
“Microbial engineering for the production of advanced biofuels” *Nature*, vol 488, pp.
320–328, 2012
- [2] ForceAtlas2, the new version of our home-brew Layout. (2011, June) Gephy [online]
<https://gephi.org/2011/forceatlas2-the-new-version-of-our-home-brew-layout/>
- [3] Kegg. [Online]
<http://www.genome.jp/kegg/>
- [4] Biological pathway (2013. November) Wikipedia [online]
http://en.wikipedia.org/wiki/Biological_pathway
- [5] <http://biit.cs.ut.ee/diffexp/>

Appendix

I. License

Non-exclusive licence to reproduce thesis and make thesis public

I, Jordan Valdma (date of birth: 02.12.1988),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

- 1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
- 1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

gPathways: a visualization tool to identify global connections among biological pathways,

supervised by Balaji Rajashekar,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **03.06.2014**

II. JSON input Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "gPathways",
  "description": "gPathways import schema",
  "type": "object",
  "properties": {
    "sources": {
      "description": "Example: Kegg, Reactome etc.",
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "name": {
            "type": "string",
            "description": "Data source name, ie. 'KEGG'"
          },
          "organisms": {
            "type": "array",
            "items": {
              "type": "object",
              "properties": {
                "keggOrganismId": {
                  "type": "string",
                  "description": "kegg id to link"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

```
"name":{
"type": "string",
"description": "organism name ie. 'Homo sapiens'"
},
"gProfilerId":{
"type": "string",
"description": "gProfiler id to gene conversions"
},
"pathways":{
"type": "array",
"items": {
"type": "object",
"properties":{
"name":{
"type": "string",
"description": "pathway name ie. 'Pentose phosphate pathway'"
},
"keggId":{
"type": "string",
"description": "for linking ie. '00030'"
},
"mainClass":{
"type": "string",
"description": "classification ie. 'Cellular processes'"
},
"subClass":{
```

```

"type": "string",
"description": "classification ie. 'Circulatory systems'"
},
"genes":{
"type": "array",
"items": {
"type": "string"
}
},
"connectedTo":{
"type": "object",
"properties": {
"name":{
"type": "string",
"description": "connecting pathway name ie. 'Pentose phosphate pathway'"
},
"weight":{
"type": "number",
"description": "edge thickness ie. '0.5'"
}
}
},
"required": ["name", "keggId", "mainClass", "subClass", "genes", "connectedTo"]
}
}

```

```
},  
"required": ["keggOrganismId", "name", "gProfilerId", "pathways"]  
}  
}  
},  
"required": ["name", "organisms"]  
}  
}  
},  
"required": ["sources"]  
}
```


VPS35, RP11-93O14.2, SLC16A14, STS, PARP2, PPIAP29, PPIAP22, PPIAP11, PPIA, TMEM178B, KLC1, KLC1, OPA1, OPA1, TOMM20, GLS, MAP4, ACP1, OCIAD1, MAP7D2, PPIAP29, MAP2K1, CCT4, NRXN3, HSP90AB1, N/A, ATP5G3, PSMA1, PSMA1, MALSU1, SLC39A10, DHX15, PPP3CB, FAIM2, N/A, EML6, PARM1, SLC17A7, KIAA0368, PPIAP29, PPIAP22, PCLO, SLC25A3, RIT2, STMN2, LMBRD2, CASD1, FKBP1B, PPIAP22, PPIAP11, PPIA, RP11-460H18.1, ANKS1B, PGM2L1, PTGR1, SUB1, SERPINI1, ATP5F1, N/A, EXOC8, SYT1, SCG2, ELOVL4, CLASP2, FAHD1, NRXN3, TSPAN7, TMEM200A, BTBD10, SCG5, NAP1L5, ATRNL1, SLITRK4, N/A, RGS4, MTMR4, SMYD2, ACTG1, ACTG1, ACTG1, PNMAL1, MEF2C, MEF2C, DYNLL1, WDR7, TRAPPC4, STYK1, SYNE1, DYNC1L1, ARL6, KIF3A, SOD1, ZFPM2, TCEAL7, ATP5A1, NRSN1, PPP2CA, SUB1, DNAJC6, ZNF365, N/A, TSPYL1, N/A, MMADHC, GRIA1, RASGRF2, RP11-58A12.2, BX255923.3, AL078621.3, RP11-561O23.5, RP11-143M1.3, DCLK1, PSMD12, PSMB1, ST6GALNAC5, OLFM3, DCAF6, GABRB3, CEP41, PSMD10, ARHGAP32, CACNA2D3, CDS1, TMEM70, SNAP91, MEF2C, MEF2C, EID1, AC012379.1, RP13-514E23.1, EPHA5, MFSD6, SSBP1, ANKMY2, ERC2, GNG2, N/A, NFU1, KCTD4, YWHAB, N/A, SUCLA2, NAPB, GSTO1, RPRD1A, GRIN2A, N/A, TMEM14A, TRIM36, SYNJ1, GABRA1, SPHKAP, NEFL, MED21, PEG3, RCHY1, PDE1A, TMEM178A, CHURC1, ARPP21, ARHGEF7, GDA, SCN3B, OAZ1, VSNL1, UGP2, UNC80, TAC1, TMX4, SRD5A1, CALN1, GAD1, PRKCB, CDC42, SLITRK4, RAB27B, EPHA4, EIF5A2, RP11-676M6.1, DNAJB6, PVALB, PRDX5, KCNQ5, GLRB, LDB2, PPP3CA, PNMA2, CALM1, CALM1, CALM1, FGF13, NEFM, KCNA1, PCDH8, RGS4, GABRB2, BHLHE22, SYT1, RPS4Y1

|

[down, red]

NEAT1, RP11-74E24.2, ZC3H11A, ZC3H11B, ANP32B, EP400, COL27A1, BBX, PRR11, TFEB, NAV1, WNK1, HIPK2, TNPO1, CTD-3051D23.4, EPC1, POLR1B, JUND, SLC12A7, FAM161B, HIP1R, ADAM33, SLC35E1, CCDC152, RP11-194N12.2, FMNL2, ACACB, MKNK2, NAV2, SNRNP48, RP11-118B22.3, DDR1, THRA, MAP4K4, NOTCH2NL, C9ORF64, SRRM2, GOLIM4, MAFF, KLF15, MSI2, PDE4C, MT1M, RP3-341D10.4, ZNF160, MAPKBP1, PXDC1, LIFR, RHOQ, ANKRD13D, MED13L, ZFR, ZBED6, RP11-403P17.4, Y_RNA, FBXO32, CDK13, NFKBIA, NFAT5, SMC3, ZFP36L1, TBL1X, BCL6, DNAJC1, ITPKB, TCF3, RP6-109B7.3, ARHGEF40, EZR, ITPRIPL2, DTNA, FAM107B, AK4, NDUFS8, FLCN, FAM123A, SKI, SCAF11, ZFH3, VTI1A,

LATS2, RNPC3, KLC1, TNS1, AC009469.1, SOX5, PMP2, SEMA3F, TBL1XR1, CFLAR, RNU7-45P, PALLD, NADKD1, ZC3H7B, RAB18, JPX, WWTR1, RFX4, RGPD6, RGPD5, RGPD3, QKI, PTAR1, PTMAP2, PTMA, PTMAP5, CTB-89H12.4, NFIA, FRMD4A, SLC7A2, NBPF14, NBPF11, WI2-3658N16.1, NBPF24, NBPF10, NBPF20, NBPF12, NBPF9, FOXO1, ZCCHC24, TP53INP1, RAB13, RP11-603J24.7, RORA, AC004951.6, NOTCH2, RP11-147I3.1, PREX1, IQCA1, STAG2, FAM181B, SAT1, KIAA1731, C4B, C4A, CPEB4, MALAT1, DGKG, ERBB2IP, PPP2R1B, ANAPC16, AC009963.3, PLGLA, PLG, MRPS5, VCAN, NFASC, NPAS3, BOD1L1, PDE4DIP, MSX1, CXCR4, DCHS1, FAM120A, POU3F2, U6, PARP11, ITGB5, CDC42EP4, TOB2, BMPR1B, XAF1, ZFP36L2, RASSF4, ABLIM1, ANKRD36, ANKRD36C, ANKRD36B, MTUS1, 8-Sep, ID4, AKAP10, CHST11, PNISR, OTUD7B, MXI1, TGFB2, LRP4, RBM25, COL5A3, PLGLB2, PLGLB1, KCNJ10, RP11-433M22.2, CREBBP, MAP7, SOX2, MAML2, KIF1B, KANK1, RAB11FIP3, LEF1, VEZF1P1, VEZF1, PLSCR4, MT2A, AHNAK, SASH1, FAM65C, AC013461.1, PLEKHA5, SPG7, CWC27, NFIC, RP11-316E14.6, YLPM1, CHD9, PBXIP1, DDX46, C5ORF24, ID3, GFAP, KANK2, DDIT4, FXR1, ARHGAP21, ZIC1, CEBPD, AFF1, AJAP1, PPFIBP1, SFSWAP, USP36, IL6ST, UBE2D3, SAMD4A, AQP4, MKNK1, EFEMP2, PCSK5, NACC2, YAP1, SRGN, MEGF10, BCL2, PSAT1, KTN1, BDP1, KIF5B, PTK2, PKN2, SEPT7P2, TNFRSF10B, CBFA2T2, CSDA, KLF9, MYO10, TULP4, RP11-732M18.3, GPAM, HLA-E, RHOBTB3, ADD3, ANLN, ZNRF3, PON2, ZIC2, PREX2, HSPA1A, ARRB1, EMX2, IL17RB, IRF2BP2, PDZD2, SNHG14, BAG3, FAT1, PLOD2, SEPP1, MECOM, BGN, ZBTB20, TGFBR3, PKP4, RP11-271C24.3, ESF1, ATRX, PTN, ACSS3, RP11-389C8.2, SOX9, CTC-228N24.3, HES1, USP47, ANKDD1A, MIR568, RP11-553L6.5, ARHGEF26, FOXC1, TOB1, SPEN, AIF1L, SLC39A12, OSMR, SPAG9, SNORD112, NIPBL, RBMS3, ASCL1, UBE2W, CCDC88A, SLC1A2, KDM4B, BCAS1, SMAD6, KLF4, RIN2, KCNJ16, SPTBN1, TTBK2, FGFR2, TNS3, ADAMTS2, LPAR1, NASP, EMP1, CIT, SCARA3, EGFR, NR2F2, MOBP, SCAMP1, HSPA2, DDIT4L, GJA1, AC005013.1, TRIL, HIGD1B, TJP2, CXCR7, MT-ND6, J01415.21, TAF1D, IGFBP5, SDC4, SLC5A3, SLC35E2, IGFBP7, RP11-738E22.2, DDX17, MIAT, PRDM16, CUX1, C11ORF96, SPP1, GABRG1, PDK4, ATP1A2, CUX2, KIF13A, EPAS1, ID2, FN1, CHI3L1, RGS1, XIST