Institute for Computer Graphics and Vision
Graz University of Technology
Graz

Metabolic Pathway Visualization Using Gene-Expression Data

Master's Thesis

Marc Streit, Bakk.rer.soc.oec.
mstreit@icg.tugraz.at

May 2007

Supervision:
Univ. Prof. DI Dr. techn. Dieter Schmalstieg
Ing. DI Michael Kalkusch

Abstract

Visualizing pathways is a substantial field of the computer assisted analysis of proteins, genes, and cellular processes. Due to the high complexity of the metabolic processes in living cells and the massive amount of data, novel tools have to be developed to address these issues. This thesis presents a pathway visualization environment, which allows an integration of metabolic pathways from the KEGG database, a publicly available high quality pathway resource. The proposed tool empowers biologists and health professionals to perform visual data mining in cellular functional networks called metabolic pathways. To handle the high degree of complexity a combination of 2D and 3D techniques is proposed. The planar pathways are arranged in a 3D scene which results in a 2.5D visualization. This novel approach allows the integration and investigation of multiple pathways in the proposed system at the same time. During this interactive visual data mining process, the user is accompanied by an extensive support of meta-information about the elements and relations of the pathways. The introduced tool focuses on the interactive visualization of pathways to create and validate new hypothesis.
The proposed software implementation is embedded in the *GeneView* visualization framework. Using this powerful software suite enables the user to synchronize different views and their in-between data.

Keywords: Pathway, Metabolic, Biochemical, Visualization, OpenGL, Graph, JOGL

Zusammenfassung

Die Visualisierung von Stoffwechselwegen ist ein wichtiges Gebiet in der computerunterstützten Anaylse von Proteinen, Genen und zellulären Prozessen. Aufgrund der hohen Komplexität der Stoffwechselprozesse in lebenden Zellen und des immensen Datenvolumens sind neue Werkzeuge erforderlich, welche in der Lage sind, diese Probleme zu bewältigen. In dieser Arbeit wird eine Visualisierungumgebung präsentiert, welche das Einbinden von bestehenden Stoffwechselwegen von der KEGG Datenbank, einer öffentlich verfügbaren und qualitativ hochwertigen Ressource von Signalwegen, erlaubt. Die im Rahmen dieser Arbeit entwickelte Software befähigt Biologen und Mediziner eine visuelle Auswertung der zellulären Funktionsprozesse durchzuführen. Um den hohen Grad an Komplexität handhaben zu können, wird eine Kombination aus 2D und 3D Methoden vorgeschlagen. Die planaren Stoffwechselwege können in der 3D Szene beliebig angeordnet werden. Dieser neue Ansatz ermöglicht eine gleichzeitige Integration und Betrachtung mehrerer Stoffwechselwege im System. Eine umfangreiche Bereitstellung von Meta-Informationen unterstützt den Benutzer während der gesamten Datenauswertung. Das Hauptaugenmerk des vorgeschlagenen Softwaresystems liegt in der interaktiven Visualisierung von Stoffwechselwegen sowie in der Erstellung und in weiterer Folge der Validierung von Hypothesen.
Die präsentierte Applikation ist eingebettet in die *GeneView* Visualierungsplattform. Dies ermöglicht eine Synchronisierung der verschiedenen Ansichten und deren Daten.

Schlagworte: Stoffwechsel, Visualisierung, OpenGL, Graph, JOGL

I hereby certify that the work presented in this master's thesis is my own and that work performed by others is appropriately cited.

Ich versichere hiermit, diese Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient zu haben.

Danksagung

An erster Stelle möchte ich mich bei meinen lieben Eltern und meiner Familie für ihre tolle Unterstützung während des gesamten Studiums und vor allem in der heißen Phase des Diplomarbeitschreibens bedanken, sowie dafür, dass sie mir diesen Bildungsweg ermöglicht haben.

Weiters möchte ich mich ganz herzlich bei meinen Betreuern bedanken. Dieter Schmalstieg bin ich sehr dankbar, dass er es mir ermöglicht hat, am Institut zu arbeiten und das wissenschaftliche Arbeiten im Alltag mitzuerleben. Ohne seine Unterstützung wäre diese Arbeit nicht möglich gewesen. Michael Kalkusch, der die Arbeit maßgeblich mitgeprägt hat, danke ich besonders für die fruchtbare Zusammenarbeit in den letzten Monaten sowie die unzähligen Verbesserungsvorschläge und Diskussionen beinahe rund um die Uhr.

Mein großer Dank gilt auch allen Mitarbeiterinnen und Mitarbeitern am Institut, die immer ein offenes Ohr für meine Anliegen hatten und in deren Mitte ich mich stets wohl gefühlt habe.

Danke auch an meine Freunde und Kollegen, mit denen ich gemeinsam einen großen Teil des Studiums verbringen durfte - allen voran Alex, Chris, Tobias, Martin und Richard. Ohne euch hätte das Großprojekt Studium sicher nicht so gut geklappt. Es war schön mit so kompetenten und liebenswürdigen Menschen zusammenzuarbeiten - in dem Wissen, dass man sich in jeder Situation aufeinander verlassen kann.

Bei Bernhard, Alex und Barbara bedanke ich mich dafür, dass sie sich die Zeit genommen haben, meine Arbeit gewissenhaft durchzulesen und konstruktives Feedback einzubringen.

Ganz besonders möchte ich meiner lieben Freundin Barbara danken, die in dieser stressigen Zeit immer für mich da war.

Contents

Chapter 1

Introduction

Cellular processes can be represented as a highly complex network with enormous measure. Figure 1.1 shows a snapshot of the metabolic network [Michal1999]. These kind of maps are created and layouted manually in painful legwork. The fact that these huge posters of the global metabolic network are still pinned on the walls of numerous laboratories illustrates why alternative ways of visualization need to be explored [Jourdan2003]. Even these huge models can only show the processes and connections on a high level of abstraction, hence the network can be broken down into a much more detailed view.

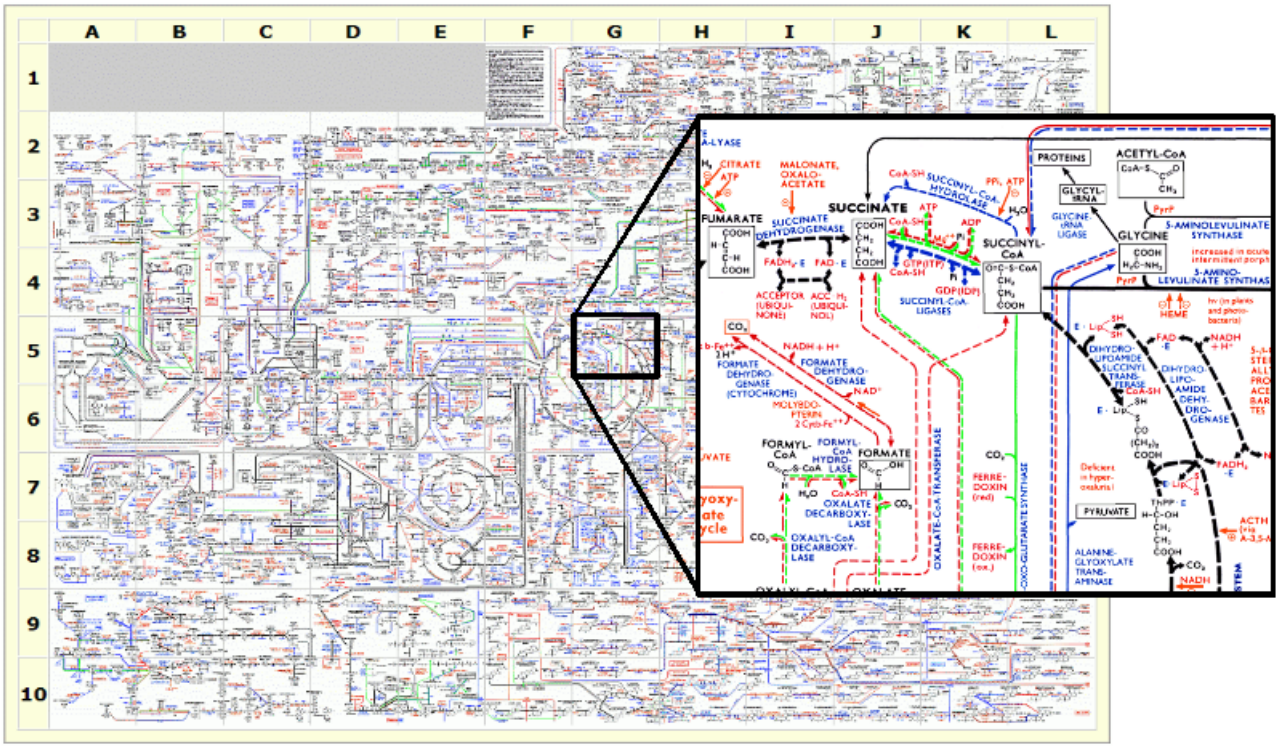


Figure 1.1: *Roche Applied Science wall chart on metabolic pathways [Michal1999]*

(Roche Applied Science - available from `http://www.expasy.org/cgi-bin/show_thumbnails.pl`*).*

New methods for handling these metabolic networks would therefore be more than welcome. Due to the dynamic developments in the field of bioinformatics the problem is getting even worse. [Eisenberg2006] defines bioinformatics as the scientific field of management and analysis of biological information. This definition shows the data centered

nature of bioinformatics. Daily new sub-graphs, cellular function, and coherences are explored. Consequently the growth of information is exponential. For example the information in GenBank[1], the most popular genetic sequence database, increased from about 1 billion base pairs in 1.8 million DNA sequences in 1997 to over 80 billion base pairs in 18 million DNA sequences in 2006 (see figure 1.2). The same growth pattern can be observed for all biological entities. This phenomenon partly originates in the discovery of modern high-throughput methods that emerged in the last decade (such as micro array analysis - see section 2.5 on page 32). Figure 1.2 shows that the trend will continue in the future.

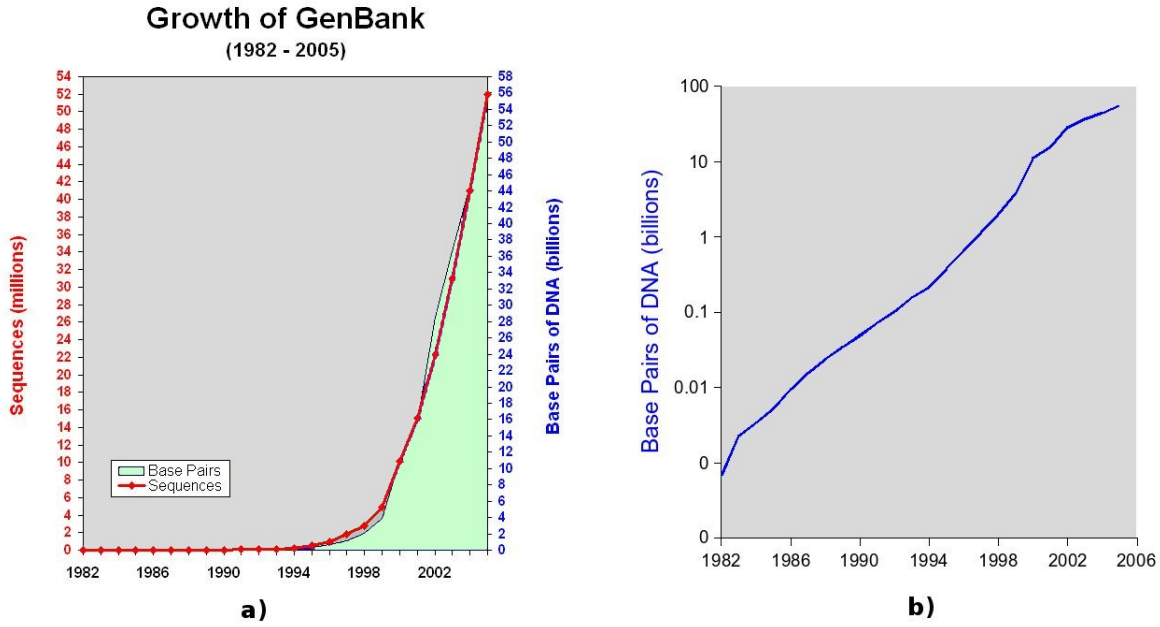


Figure 1.2: *GenBank growth rate.*
Chart (a) shows the increase of DNA sequences (red) and bases (blue) in the period from 1992 to 2005. Chart (b) depicts the same information using a logarithmic scale for the base pairs.
(Figure (a) is available on `http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html`*.)*

What motivates work in that field is the idea of creating a visualization framework that supports the understanding of the results of experiments conducted by biologists and medical doctors. Modern *Information Visualization* techniques serve as the toolbox to achieve this ambitious goal.

[1] `http://www.ncbi.nlm.nih.gov/Genbank/`

Problem statement and contribution

New hypotheses on cellular functions and processes emerge during the research of biologists and health professionals. The thesis aims at the creation of a pathway visualization tool that enables researchers to interactively explore complex metabolic pathways representing biomedical processes. The system should empower them to refine and finally validate their hypotheses. This process is illustrated in figure 1.3.

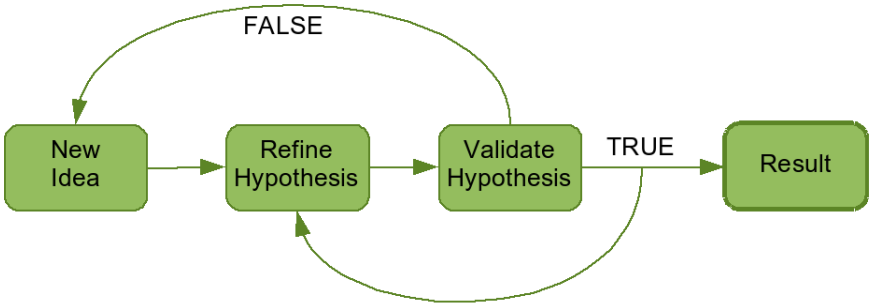


Figure 1.3: *Process of a hypothesis evaluation.*

Furthermore the comprehension of genetic data into metabolic networks opens possibilities in the determination of gene functions. Though human DNA has been examined and broken down to sequences of genes, the meaning of most genes and their influence on each other is still unknown. This knowledge would give a deeper insight into functions of cells and lots of genetic caused diseases.

In collaboration with our medical partners we prepared a list of requirements to understand the users' needs. Some of them are already implemented in available software systems on the market. Others are integrated in our software framework for the first time. Features ranked at the top of the list created with our medical experts mostly deal with user interaction and novel ways of data representation. This outcome reflects also the fact that currently available software packages have a deficit with respect to interactive data exploration.

The main requirements for the design and implementation of our pathway visualization framework are:

- Pathway loading from existing databases
- Navigation inside the pathway network
- Multiple pathway visualization
- Cross reference and dependency visualization
- Neighborhood visualization in the pathways
- Integration of additional meta-information

The proposed pathway visualization system does not include features regarding the creation, editing and layouting of the pathways. Our main focus lies on the visualization of the pathways and their interrelations. Public databases provide the input data including the layouting. Therefore our tool has to support an easy and fast integration of external

pathways. The system must consider the hierarchical structure of the metabolic network which is determined by the cellular functions. Moving up- and downwards in the hierarchy is needed. Not only the vertical navigation in the hierarchy is essential - also the switching between pathways must be supported. With respect to the size of complex pathways zooming capabilities are needed. Here it is vital to assure the users' orientation in the network at any time.

These requirements refer to the interaction of a single pathway. Even more important is the simultaneous display of multiple pathways inside the framework. In contrast to the single pathway inspection, the multiple pathway approach allows a better examination of the overall network. This feature is the basis for the visualization of dependencies between the pathways. Elements in one pathway occur multiple times in the cellular network. Hence, an identification of these identical nodes in related pathways is crucial. From information visualization point of view pathways are graphs. The display of topological neighborhoods in metabolic pathways can provide a better insight into the involvement of elements in the processes of a sub-graph. The neighborhood algorithm must be applicable in arbitrary depth.
Due to the amount of contextual information on the pathways and its elements a proper integration in the system is needed. However, an overloading of the display must be avoided.

Currently available tools do not meet these requirements to a satisfactory degree. The contribution of this thesis is a visual data mining and interaction system for the visualization of metabolic pathways that covers these requirements.

Chapter 2

Related work

The first part of this chapter provides an overview of the related work in the field of *Information Visualization*. First general methods, which are important for visualization of medical data in this area of application, are described. This rather high level discussion is followed by a more specialized view on pathway and gene-expression visualization. The combination of these two emerging research areas leads to the application of gene-expression data onto biochemical pathways. Finally the current state in the domain of pathway visualization is summarized and potential improvements are identified.

2.1 What are medical pathways?

Pathways can be divided into:

- **Metabolic pathways**
  Metabolic pathways are models that describe chemical reaction cascades in cells.
- **Regulatory pathways**
  Regulatory pathways focus on signal-transduction and cell cycles. An example for a cell cycle would be the apoptosis which is the programmed cell death (PCD).

Only metabolic pathways are in the scope of this paper because metabolism can be modeled in reference pathways. These general pathways are valid for several organisms [Kanehisa2000]. Building on a specific reference pathway, organism specific pathways can be generated. In contrast, regulatory pathways vary in detail for specific organism and are much harder to consolidate in common reference pathways [Kanehisa2000].
In [Bourqui2006] a metabolic network is defined as interconnected metabolic pathways.

Metabolic pathways aim at modeling cellular functions in graphs. Basic building blocks of pathways are chemical compounds (so called metabolites) and enzymes. Chemical compounds act as substrates and products in chemical reactions. Enzymes catalyze these reactions by using substrates as input. The output of the chemical reaction are one or more products which are in turn compounds. This process is illustrated in figure 2.1.
This model of the chemical reaction in pathways is a simplification of the real process. Figure 2.2 shows the chemical process that occurs in pathways in more detail. Basically the reaction is regulated by the energy flow. Adenosine triphosphate (ATP) is a possible energy source inside cells that is needed by the chemical reaction to run. During the chemical reaction ATP is converted to adenosine diphosphate plus phosphate (ADP+P). The chemical reaction is regulated by the energy level and co-factors.

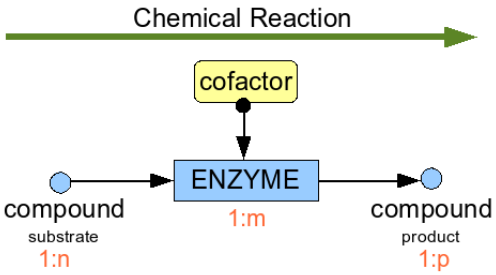


Figure 2.1: *A chemical reaction inside the cellular system takes one or more compounds (i.e. substrates) as reactants and generates one or more products. This process is catalyzed by at least one enzyme. Co-factors are external influences that affect the reaction.*

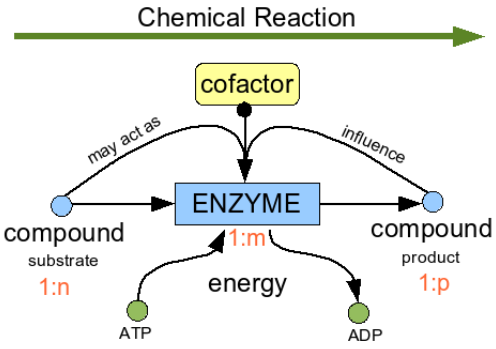


Figure 2.2: *External regulation of the chemical reaction inside metabolic pathways.*

Additionally substrates and products could act as co-factors. Imagine a special chemical reaction that only starts if a certain amount of substrate is present as input. It is like switching the reaction on. This could result in an acute reaction that quickly regulates the enzyme up which produces a certain amount of compound as output. This compound subsequently influences the reaction. For example the process stops at a certain level of the product. Therefore forward regulation and back coupling have a high impact on these reactions. So far these complex processes have not yet been entirely integrated in an interactive visualization system. It would be interesting to have a system that enables the user to change the parameters (e.g. energy level) at runtime and immediately get feedback on the influences on the network. Thus a simulation of the network would give more insight into the cellular processes.

2.2 Biomedical databases

In the last two centuries biomedical databases emerged to organize the massive amount of information. Database projects are often conducted by different objectives. For instance one might aim for completeness while others strive for quality and correctness. Due to the wide range of databases and competing projects it is hard to make a choice. Therefore the

decision for the right database will highly depend on the specific task.

Some important criteria that contribute to the success of biomedical databases are:

- Availability
- Update rate / update process
- ID management (Unique Key)
- Links to other databases
- Software to access database (API, scripts, etc.)
- Documentation
- Full data dump

In contrast to commercial databases, publicly available resources in the field of bioinformatics are not stored in relational database management systems. A prevalent approach is the deposit of information in standardized text files. A popular example is the *Abstract Syntax Notation One (ASN.1)* file format. One reason for this development are the special requirements for databases in this field. An important standard feature is the comparison of DNA sequences. This can be performed quickly in a text file based storage system.

Selected databases that are relevant in conjunction with this work are discussed below. According to the focus of the databases, we group them in pathway and gene databases respectively.

2.2.1 Pathway databases

Popular databases are:

- Kyoto Encyclopedia of Genes and Genomes (KEGG)[1]
- BioCarta[2]
- PANTHER[3]
- MetaCyc[4]
- ExPASy[5]

Only two of the listed resources are discussed in more detail: KEGG, because it is our choice for the input data that is visualized by our system, and BioCarta to show an alternative pathway resource that strongly differs in the representation of the pathways.

[1] `http://www.genome.jp/kegg/`
[2] `http://www.biocarta.com`
[3] `http://www.pantherdb.org/pathway`
[4] `http://metacyc.org/`
[5] `http://expasy.org/cgi-bin/search-biochem-index`

2.2.1.1 Kyoto Encyclopedia of Genes and Genomes

The *Kyoto Encyclopedia of Genes and Genomes (KEGG)*[6] [Kanehisa2000; Kanehisa2006] is a biomedical resource that started its online service in 1995 and belongs to the Japanese GenomeNet. The KEGG database suite consists of four major components that are basically self-contained [Kanehisa2006]:

- **PATHWAY**
  The database gives the community access to a fast increasing number of pathways. A lot of literature research has been conducted to create these pathways. The graphs are layouted by hand and provided in XML format.
- **LIGAND**
  The LIGAND database holds chemical information like enzymes, compounds and reactions. These entities are the building blocks of the pathways.
- **GENE**
  The GENE component includes catalogues of genes from various species.
- **BRITE**
  BRITE was added in 2006 to convey functional hierarchies and higher-order functions by classifying proteins, compounds, drugs, diseases, and so on.

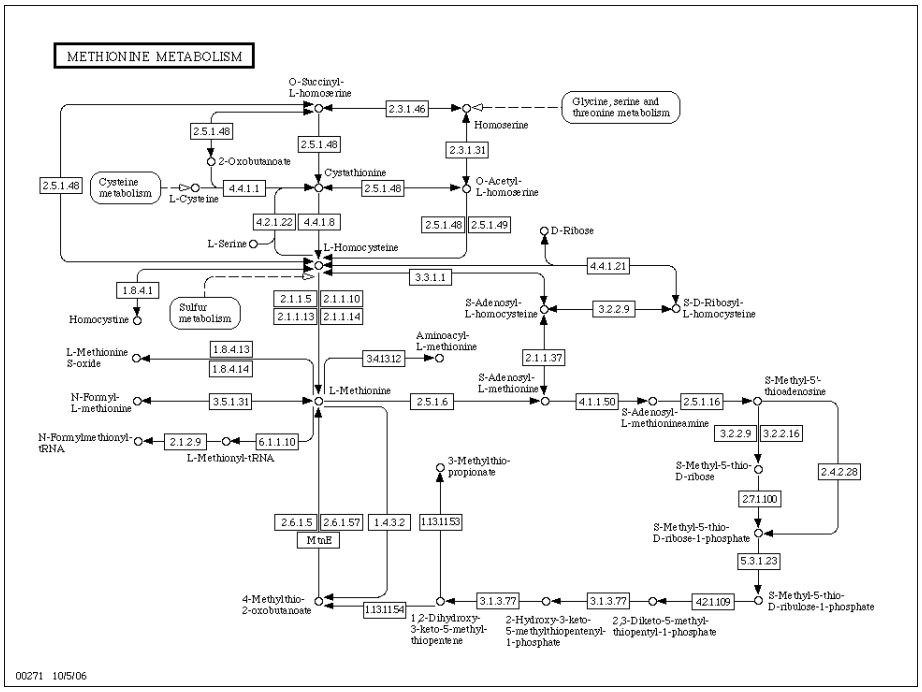
The information of each component is fully linked to the other databases. For example the database entry of an enzyme contains information in which pathways the specific protein is involved and which genes encode this enzyme.

KEGG PATHWAY database

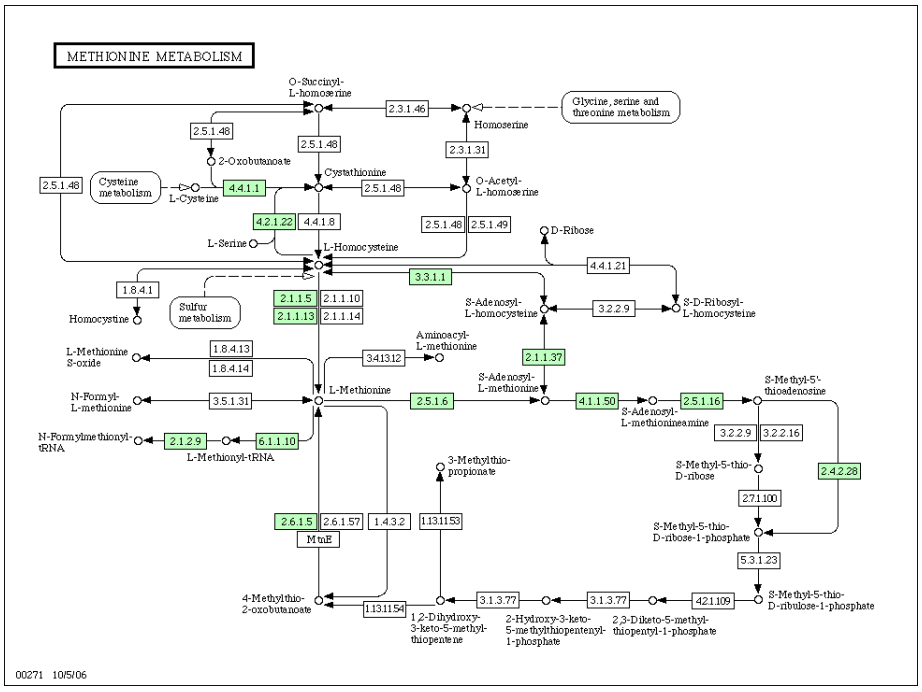
The resource provides over 140 metabolic pathways for various organisms. Pathways are represented by graphs which are layouted by hand and stored as static images. In the graph enzymes are visualized by rectangular nodes and compounds are represented by small circles. Round rectangular nodes depict nested pathways or links to others. This points to the fact that a pathway is only an artificial subset of a huge complex network.

Throughout this paper the *Methionine Metabolism* pathway serves as prime example. This special amino acid pathway is of average size and includes all types of nodes and edges as well as cyclic reaction cascades.
The database holds so called reference pathways that are valid for various organisms. Therefore these reference pathways can be specialized per organism. Figure 2.3 shows the reference pathway and the specialized homo sapiens pathway for *Methionine Metabolism*. The graphs share the same composition and connection of nodes but vary in the color coding of enzymes. Hence in the specialized pathways enzymes are replaced by genes that are involved in the production of these enzymes. In this example human genes that encode enzymes in the pathway are highlighted in green. KEGG is able to generate these specialized colored maps for numerous organisms.

[6] `http://www.genome.jp/kegg/`



(a) *KEGG Methionine Metabolism reference pathway.*
(taken from `http://www.genome.jp/kegg/pathway/map/map00271.html` *)*



(b) *KEGG Methionine Metabolism pathway for homo sapiens.*
(taken from `http://www.genome.jp/dbget-bin/get_pathway?org_name=hsa&mapno=00271` *).*
Green highlighted enzymes depict proteins that are encoded by the human genome.

Figure 2.3: KEGG Methionine Metabolism pathway.

2.2.1.2 BioCarta

The *BioCarta*[7] platform comprises a high number of pathways. The maps are created and contributed by an active community. Templates that are designed to make the drawing of the pathways easier are provided on their website. Additional descriptions are also delivered with the hand-drawn images. The platform supports a reviewing and revision process of the contributed drawings.

The *BioCarta* pathways are drawn in a non-photorealistic rendering (NPR) style. The appearance of the drawings also reminds at a cartoon-like drawing style which is very appealing to the scientists [Saraiya2005]. The scientific community highly appreciates the embedding of biological context knowledge in this easy perceptable way. Therefore some *BioCarta* pathways show the cellular context in the cell where the processes happen. The sample pathway in figure 2.4 depicts demonstrative locations in the cell like the cell plasma, the nucleus, and the lumen. Also obvious by just looking at the pathway image is the fact that the liver and the intestinal cell are involved in the process. This kind of contextual information is mostly missing in the visual representation of KEGG pathways.

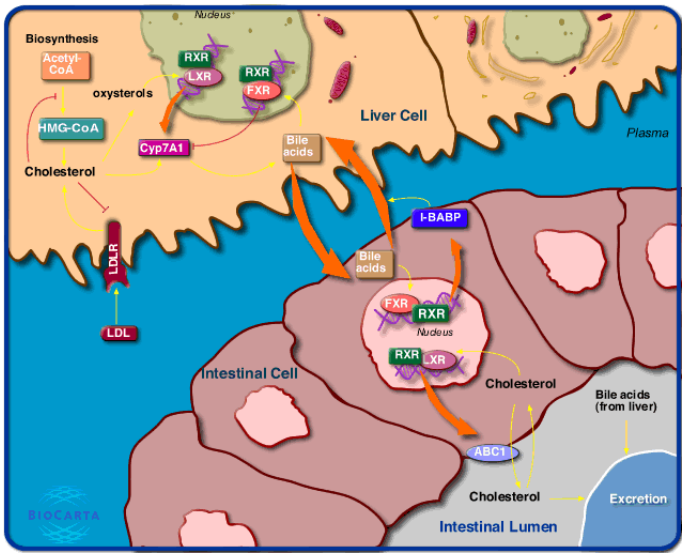


Figure 2.4: *BioCarta sample pathway for FXR and LXR Regulation of Cholesterol Metabolism* (*taken from* `http://www.biocarta.com/pathfiles/m_fxrPathway.asp`)

[7] `http://www.biocarta.com`

2.2.2 Gene databases

2.2.2.1 Entrez

Entrez[8] is a web-based search and data retrieval service. The system is part of the National Center for Biotechnology Information (NCBI) which is integrated in the United States National Library of Medicine (NLM). NLM belongs to the National Institutes of Health (NIH) the focal point for medical research in the United States. The *Entrez* system is connected to all major gene and protein databases. The databases can either be searched simultaneously via a text-based web interface or by using Entrez tools[9]. These are scripts and APIs for various programming languages based on the Simple Object Access Protocol (SOAP).

2.2.2.2 International Nucleotide Sequence Database Collaboration

The *International Nucleotide Sequence Database Collaboration (INSDC)*[10] is a collaborative network of DNA sequence databases that consists of three major members:

- GenBank[11] (USA)
- European Molecular Biology Laboratory (EMBL)[12] (Europe)
- DNA Data Bank of Japan (DDBJ)[13] (Japan)

The data are synchronized between the partner databases on a daily basis. The INSDC network is the biggest resource for DNA sequences and related information. Futhermore it is widely accepted in the community as the standard reference.

2.2.2.3 Gene Ontology

The *Gene Ontology (GO)*[14] project tries to describe and categorize genes and proteins in ontologies. They focus on assigning genes according to:

- molecular functions,
- biological processes and
- cellular components.

The GO project has great potential to boost the analysis of metabolic networks in the next decade.

[8] `http://www.ncbi.nlm.nih.gov/entrez`
[9] `http://www.ncbi.nlm.nih.gov/entrez/query/static/advancedentrez.html`
[10] `http://www.insdc.org`
[11] `http://www.ncbi.nlm.nih.gov/Genbank`
[12] `http://www.ebi.ac.uk/embl`
[13] `http://www.ddbj.nig.ac.jp`
[14] `http://www.geneontology.org`

2.2.3 Biomedical identifiers

During the last decade biochemical database projects have become numerous. Some projects have proven their value and are established inside the community. Each database project introduced its own identification numbers for genes, proteins, nucleotides, and so on. This fact makes it hard to write flexible software systems that can handle a wide range of identifiers.

2.2.3.1 Biological entities

This section discusses annotation systems that were designed for entities within the context of pathways.

Enzyme ID

The *Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (NC-IUBMB)* agreed on the Enzyme Commission number (EC number)[15] [Bairoch2000]. Enzymes are divided into classes and subclasses which are separated by a "." (e.g.: 3.1.1.1). The enzyme nomenclature database[16] is provided online by the Swiss-Prot platform. The release from the 3rd of April 2007 contains 4026 active enzyme entries. Each contains:

- EC-number
- Official name
- Alternative names
- Reactions catalyzed
- Co-factors (if any)
- Cross-links to other databases

Gene ID

The nomenclature for enzymes became widely accepted. However, in the case of genes, unique identifiers are not available. Several identification systems are used by different groups and projects. The main problem is the uniqueness of the gene identifiers.
Some annotation systems often assign new identification numbers when sequences get updated or corrected. The accession number annotation system does not suffer from this problem. Therefore it seems to be the ideal choice for unique identification in software applications.

2.2.3.2 Annotation mapping (ID conversion)

An inherent problem when working with biomedical resources is the conversion between the different identification systems. Some of these annotation systems are ambiguous.

[15] http://www.chem.qmul.ac.uk/iubmb/enzyme/
[16] http://www.expasy.org/enzyme/

For the design of an application in this area it is a far-reaching decision how to address this problem. Over the time there were many attempts to get this issue under control. But still, a reliable mapping of identification numbers by including various databases is a challenging task.

The easiest path is the usage of an existing external annotation mapping system. [Alibes2007] proposed a server-side web application[17] that is capable of performing such mapping tasks. It takes a list of identifiers in a web form and returns the result either in HTML, as a text file or as a spreadsheet.

Although these kind of web services support a wide range of annotations and generate feasible results, they are not suitable for the integration in a standalone framework. A better solution is the implementation of a dedicated mapping database. However the design of such a system is time-consuming and demands a high degree of expertise. Nonetheless, local availability of annotation mapping is essential.

2.3 Information visualization methods

Information Visualization (InfoVis) needs to be strictly differentiated from *Scientific Visualization (SciVis)*. InfoVis deals with abstract data like for example movies in a movie database. SciVis operates on real-world data with spatial character. *Flow Visualization* and *Volume Visualization* can be assigned to the class of *Scientific Visualization*. Another important visualization research field that needs to be mentioned is the *Medical Visualization (MedVis)*. Examples for medical data are CT, MRI and ultrasound.

Applied to the context of this thesis, medical pathways and gene-expression analysis rather belong to *Information Visualization*.

[Shneiderman1996] published the often cited ***Visual Information Seeking Mantra***:

> *Overview first, zoom and filter, then details on demand*

The mantra is proposed as a design guidance for the creation of novel information visualization systems. [Craft2005] discussed why the mantra became a widely accepted design principle, although it has not been substantiated by scientific studies yet. However, nobody really puts Shneiderman's guiding paradigm into question. Parts of the mantra can be found in every modern visualization system. A user centered visualization pipeline is published in [Card1999].

Action cycle

Some of the well-established visualization and interaction methods seem to be obvious and intuitive approaches - sometimes even trivial [Kosara2003]. However, the proper implementation and integration of these methods is hard to accomplish. When the system is laid out for interaction and exploration of complex data the problem becomes worse.

[17] `http://idconverter.bioinfo.cnio.es`

[Spence2007] introduces the action cycle that is illustrated in figure 2.5 which models the dynamical data exploration process.

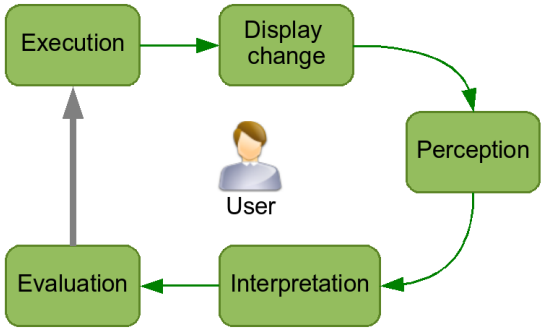


Figure 2.5: *The action cycle [Spence2007] depicts the process of user interaction with a dynamic system.*

The cycle starts with the execution of the data followed by a display change. At this point the user becomes involved in the process by visually perceiving the display's content. After a successful information perception the user needs to interpret and evaluate the results.

2.3.1 Geometric zoom and pan

The actions zoom and pan are important standard features in many visualization applications. [Furnas1995] tried to visualize these features in the so called space-scale diagrams. Figure 2.6 shows the original 3D version from their paper.

The diagram shows a pyramid turned upside down containing all possible magnification levels of a 2D image. Therefore the vertical axis represents scale while the horizontal axis depicts the spatial extension. The magnification is 0 at the tip of the pyramid (maximum zoomed out) and can go to infinity (i.e. zoom in). In literature this action is often described as geometric zooming (in opposition to semantic zoom - see section 2.3.2 on page 23). Also a viewing window is introduced in this diagram which illustrates the portion of the image that the user sees. Shifting the frame around is equal to a pan action. Figure 2.7 shows the trajectories for pan and zoom actions.

Furnas and Bederson also present a 2D version of the space-scale diagram where the spatial dimension is broken down into one dimension. With the reduction to a 2D diagram the information is still preserved but the drawing of a 2D space-scale diagram is much simpler. [Wijk2003] proposed a method that aims for a smooth and optimzed way of zooming and panning.

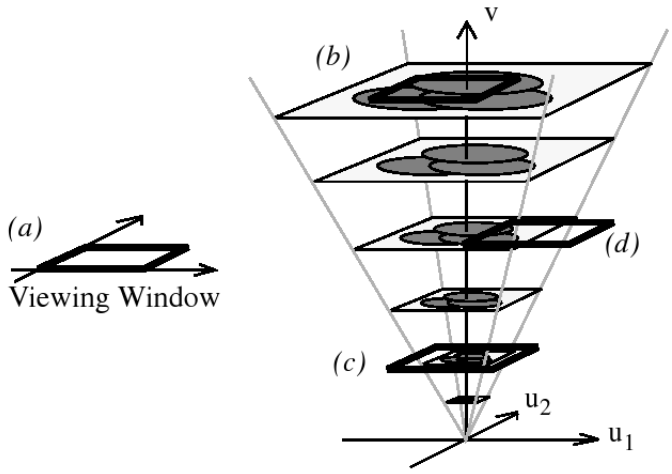


Figure 2.6: *Zooming and panning in a 3D space-scale diagram (see figure 2 in [Furnas1995]). (a) represents the viewing frame that is moved around on the image; (b) shows the viewing frame in a zoomed in image; (c) shows the whole 2D image (i.e. zoomed out). (d) represents a pan action to the right side of the image.*

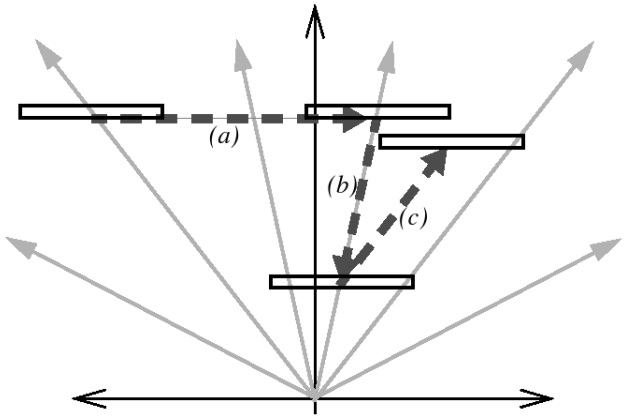


Figure 2.7: *Zoom and pan trajectories in a space-scale diagram (see figure 6 in [Furnas1995]). (a) shows a pan action performed on an image. The trajectory for a zoom-out action is shown in (b). Finally panning and zooming is performed on the image which is depicted with (c).*

2.3.2 Semantic zoom

The semantic zoom visualization technique provides information while using another representation [Spence2007]. Semantic and geometric zoom are often deployed in combination. We applied the idea of semantic zoom to pathways in figure 2.8. In this example chemical compounds in cellular networks are originally represented by a simple circle. When the user zooms into a predefined magnification, the circular symbols are replaced by the chemical structure of the compound.

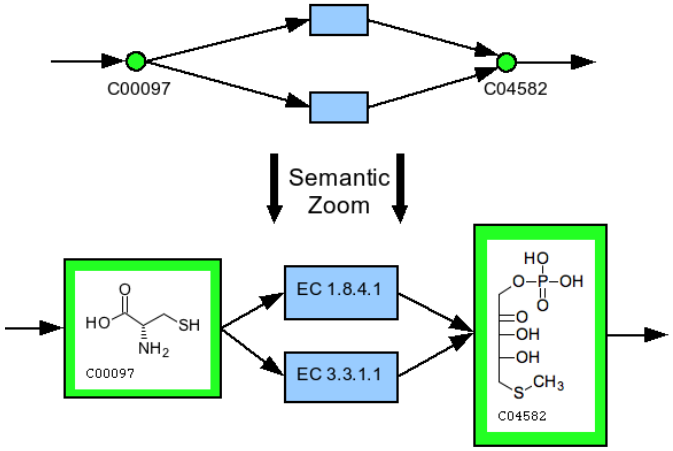


Figure 2.8: *Semantic zoom example. A zoom action reveals the chemical structure of the compounds in a graph.*

2.3.3 Multiple views

[Jacobson1994] proposed the LinkWinds system that already integrated the multiple view technique. A screenshot of the system in figure 2.9 shows how several views are positioned on the screen.

For a real understanding and exploration of a data entity it can be of vital importance to depict the data in different windows or views [Roberts2000]. The approach of multiple views is also a powerful paradigm to avoid misinterpretation by showing data in alternative representations. A lot of previous work has been carried out on this topic. [Baldonado2000] published eight guidelines to support the creation process of a multiple view system which aim at dispelling problems early in the design phase of a software project.

[Kosara2003] stated that by using currently available graphical user interface libraries the integration of multiple views in an application is no problem at all. However, only with the integration of user interaction capabilities multiple view environments can produce a real added value.

A good implementation of multiple views always keeps the canvas areas synchronized. A user that interacts with a view changes rotation, orientation, and translation - the so called viewing parameters. In various situations the tight coupling of these parameters between

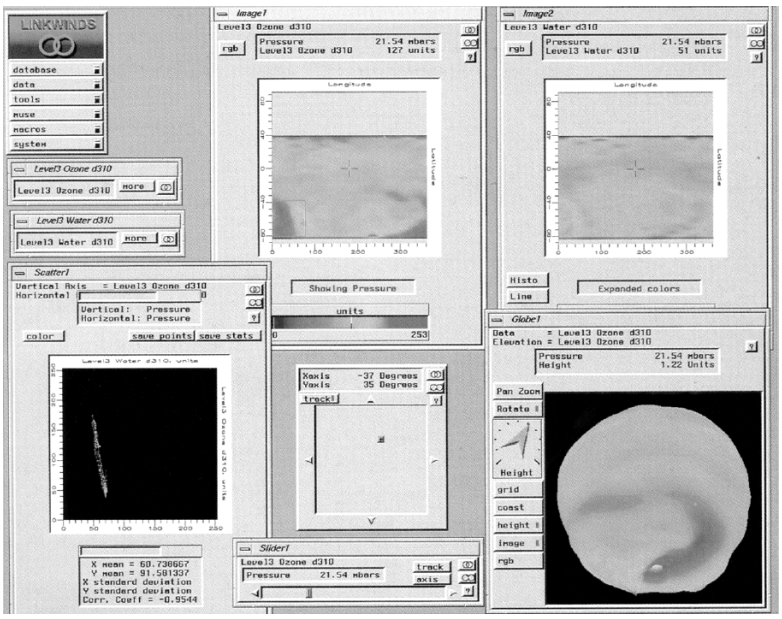


Figure 2.9: *Screenshot of the LinkWinds system (see figure 4 in [Jacobson1994]). Views are arranged side by side and partly overlapped.*

views can be useful. An example would be the comparison of similar data for revealing minor differences.

In 2000 North and Shneiderman published the Snap-Together Visualization [North2000]. Using their system a user can dynamically connect views without programming. They provide an API as well as a user interface that sets up on their programming interface. Remarkable is the fact that the Snap-Together Visualization supports the combination of views by using techniques as *Linking & Brushing* (see section 2.3.7 on page 27), details on demand (see section 2.3.4 on page 25), synchronized scrolling, and the *Focus + Context* overview method (see section 2.3.5 on page 26) in a very flexible and easy useable way. This approach is taken as a role model in the *GeneView* framework for the design of the update and synchronization mechanism (see section 3.2.4 on page 50).

Special attention has to be turned on the consistency of the visual representation [Baldonado2000]. For example a color that is chosen for depicting certain data values in one view should consequently be used in other views for highlighting. Same applies for properties as shape, size, etc.

Model-view separation

According to the Model-View Controller (MVC) design pattern (see [Krasner1988] [Gamma1995; Heer2006]) the input data needs to be separated from the viewing data. Figure 2.10 illustrates how the same data can be visualized in different visual representations. Especially when designing a visualization framework, the strict consideration of this principle is essential. The importance of the MVC pattern becomes apparent when information is displayed in several views at the same time.

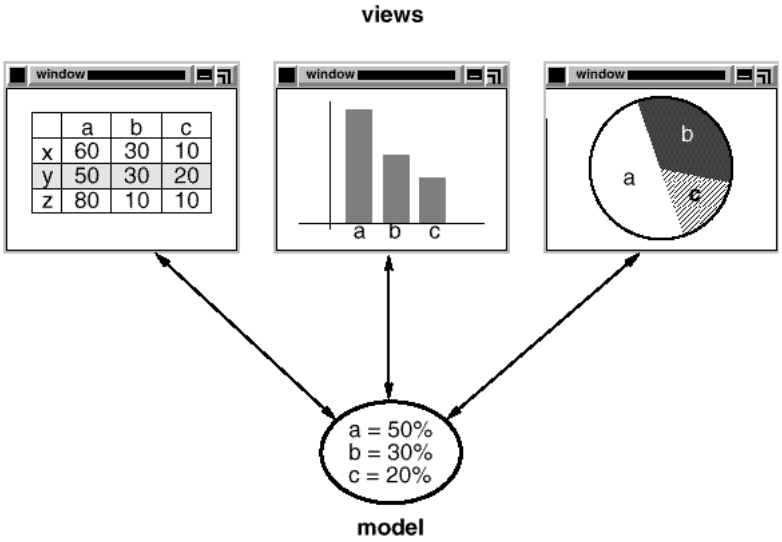


Figure 2.10: *Relation between model (i.e. data) and view [Gamma1995].*

By following this abstract design pattern, views can react on data updates without knowing the details of the controller and other views. When data is changed in one view only the model needs to be updated. All related views can perform a refresh action for presenting the current data state. Without the clear distinction between model and visual representation the view changed by user interaction would have to hold references to all other views to notify them. This decoupling of views is also covered by the Observer design pattern [Gamma1995].

2.3.4 Details on demand

Due to the limited screen space it is impracticable or even impossible to show all information in-depth at the same time. Details on demand is the process of providing extra information for an element or a group of elements in addition to the current view without changing the shown representation [Craft2005]. A typical task that is solved by this method is the identification of a specific data item. One possible trigger (among others) for the details on demand feature can be a mouse-over action that opens a pop-up window [Shneiderman1996].

In 1995 an early implementation of details on demand was integrated in the *Information Visualization & Exploration Environment (IVEE)*. The screenshot in figure 2.11 shows data points on a map for which detailed information can be requested on demand.

The application of the details on demand feature in a multiple-view environment gives the possibility to nicely embed the extra information in the graphical user interface.

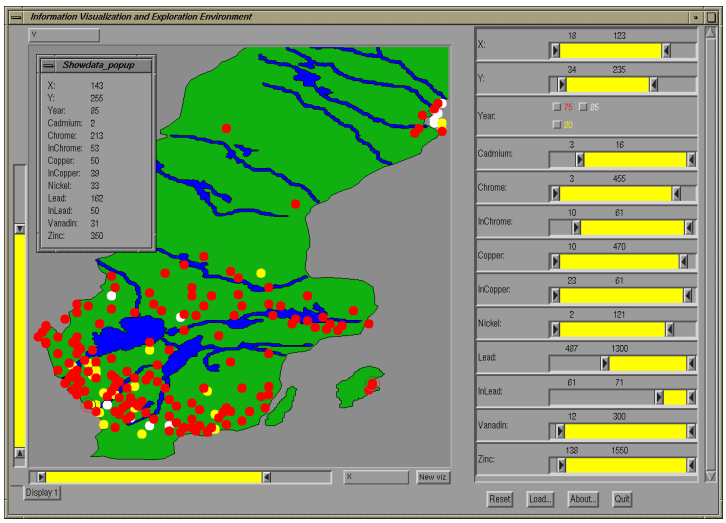


Figure 2.11: *Screenshot of the details on demand support in the IVEE framework [Ahlberg1995]. The colored objects on the map of Sweden indicate measurement of heavy metals. The pop-up window on the upper right corner is fed by the detailed information that is dynamically requested from the database.*

2.3.5 Focus + Context

Imagine a person looks up a street on a detailed city map of New York City. The regular way to accomplish this task is by searching the name of the street in the map's index. What you usually get is some kind of raster instruction (i.e. column B row 1). This information cuts down the searching area to one raster segment where it is easy to find the desired street. However, in which district is the street? What are the main streets leading to this district? The circumstance that the map is arbitrary folded reinforces the problem even more. For answering these questions an overview map is often enclosed to the city map. The problem is to correlate the information in the overview map with the detailed one [Spence2007]. In information visualization this problem is addressed by the *Focus+Context (F + C)* visualization technique. The user wants to have orientation considering the whole data set while being focused on a sub portion.

In the state-of-the-art report [Kosara2003] classified the *Focus + Context* problems into four sub-methods:

- **Overview method**
  The introductory example with the overview city map is a typical application of this *Focus + Context* method.
- **Filtering**
  The filtering methods presents a part of the information altered or refined. An example would be the magic lens technique where the user moves an arbitrary object over the screen and the information inside the borders of the object change.
- **Distortion-oriented**
  The idea behind image distortion is to increase the space for the data in focus. The

contextual information is distorted. One popular and early adopted method was the bifocal lens [Spence2007]. Other popular implementation of geometric distortion are the fish-eye views [Furnas1986] and the perspective wall [Mackinlay1991].

- **In-place technique**
  In contrast to the filtering method the application (instead of the user) decides which information should be altered or enriched.

2.3.6 2D vs. 3D

Whether to use 2D or 3D visualization is still a controversial topic. The clear advantage of planar visualization techniques (i.e. 2D) lies in their simplicity. Even untrained users are able to visually perceive information presented in a 2D view in a fast manner. However, the 2D version reaches its limit when relations among graphs and/or other views are in the user's interest. 3D representations overcome the space limitation to some extent but encounter other difficulties like occlusion and depth perception. Several possibilities exist to counteract these problems (e.g. usage of transparency).

The decision to go for a 2D or 3D representation highly depends on the characteristics of the data as well as the task. An obvious solution is the combination of both - 2D and 3D. The integration of both versions in one framework makes it possible to take advantage of the strengths of either methods. This approach can be achieved in a multiple-view user interface (see section 2.3.3 on page 23) and can be the foundation of a successful visualization system.

Applied to biochemical networks a 2D representation of the graph might be the right choice for the creation and manipulation of planar graphs. In contrast, only by the usage of the third dimension the important interdependencies and connections inside the entire metabolic network can be revealed and analyzed.

2.3.7 Linking & Brushing

Brushing

Brushing is the process of selecting data elements or groups of elements to highlight them [Martin1995]. Highlighting of a certain data portion has the objective to draw the user's attention to the data selection. The system can utilize various highlighting mechanisms. Changing color, size or shape are typical examples. Also combinations of these options are possible to amplify the visual stress effect. Selecting data with the mouse is evident but also various alternative ways for data selection are applicable (for instance the definition of regions of interest (ROIs) by using slider widgets).

Linking

For exploring complex data effectively the brushing technique is not sufficient. Especially when working in a multiple view environment (see section 2.3.3 on page 23) interactive linking to other views is essential. Hence linking is the tight coupling of one or more views.

A special application of linking could be the process of broadcasting brushing information to other views [Kosara2003].

Combined methods

The combination of these methods leads to the powerful visualization technique of *Linking & Brushing (L & B)*. Especially when operating on high-dimensional data *Linking & Brushing* yields very good results.

A trivial example is the selection of a single point in a 2D scatterplot by using the mouse. Consequently, if the point is depicted in other plots, it changes the color.

The same data can be depicted in different views but in a different style. For example a certain data point in a 2D scatterplot can correspond to a multi-dimensional connection line in a parallel coordinates view. Only the *Linking & Brushing* technique enables the user to visually investigate the identical data tuples among different projections.

In [Hauser2004] it is discussed how the *Linking & Brushing* approach in a multiple view setup fits in the concept of *Focus + Context*. Therefore a brushed piece of information in a detailed view can be highlighted in an overview map. An example of *Linking & Brushing* in a multiple view application is shown in figure 2.12.

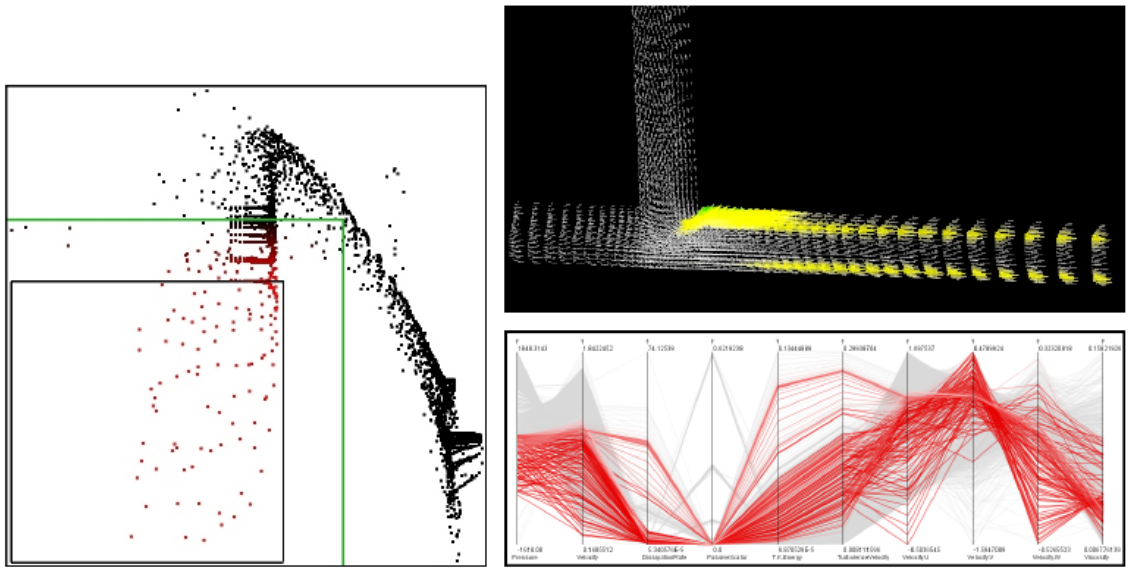


Figure 2.12: *Linking & Brushing in a multiple view environment (see figure 8 in [Hauser2002]). The data is selected by a rectangular region of interest on the 2D scatterplot (left view) using the smooth brushing method [Doleisch2002]. Consequently the brushed selection is published to the parallel coordinate view and the 3D plot.*

2.4 Pathway visualization

2.4.1 Metabolic pathways as graphs

Pathways are sub-divisions of a complex cellular network. This non-planar network is flattened to planar graphs. Each metabolic pathway is a bipartite graph. The set of nodes is divided into two disjoint subsets of nodes whereas no node inside one group is connected [Jourdan2003]. In the case of metabolism that means that enzymes are not directly connected to other enzymes. The same applies for chemical compounds.
Metabolic pathways are partially cyclic graphs. Directed and undirected edges are present. Nodes can occur multiple times in the same graph as well as in foreign graphs. A good overview over metabolic pathways with respect to graph theory is given in [Bourqui2006].

Due to their historical development, metabolic pathways are often handmade. That means experts incorporate their meta-knowledge in the specific domain to position nodes and route edges. Nowadays this approach is still widely used in the community due to the complexity of the graphs. Therefore graph drawing algorithms are needed for an automated generation of pathways.

Common graph drawing algorithms do not provide satisfactory results with respect to metabolic pathways because they focus on different constraints [Becker2001]. Becker and Rojas stated that common graph layouting algorithms are optimized to fulfill the following criteria:

- Planarity
- Minimal edge crossings
- Minimal drawing area
- Maximal symmetry

As far as metabolic graphs are concerned it is hard to define a simple set of criteria and constraints because the complexity and diversity of the cellular network. Therefore special algorithms need to be developed which are capable of handling the complex biochemical coherences.

Graph drawing algorithms

One of the first attempts to dynamically model metabolic graphs was done by Karp et al. starting in 1993 [Karp1993; Karp1994; Karp1994a]. As metabolic graphs became more and more diversified and complex over time, graph drawing approaches needed to be adapted and enhanced. [Becker2001] proposed an algorithm that builds up on the ideas of
Kerp et al. and enhanced them by including the topological structure. As a starting point the method searches for the longest cycle inside the graph. The rest of the nodes are classified as inner and outer components which are positioned using an embedded spring algorithm [Kobourov2005].

[Bourqui2006] recently published a graph drawing algorithm that addresses the visual identification of reaction cascades among distinct metabolic pathways by forming a single concatenated graph.

Systems Biology Markup Language (SBML)

SBML [Hucka2003][18] is a XML based markup language designed to model biochemical reaction networks (e.g. metabolic pathways, regulatory pathways; see section 2.1 on page 12). Since the creation of the initial version of the standard in 2003, SBML has become widely spread and is currently used in over 100 software systems.

Up to now SBML models the topology of the network but does not support the encoding of graphical representations.

2.4.2 2.5D pathway visualization

[Brandes2004] proposed a 2.5D method to visualize differences across pathways among different species. This method enables the user to see evolutionary developments and relations between organisms. In figure 2.13 several graphs are visualized as a stack. This representation gives the user the ability to grasp discrepancies.

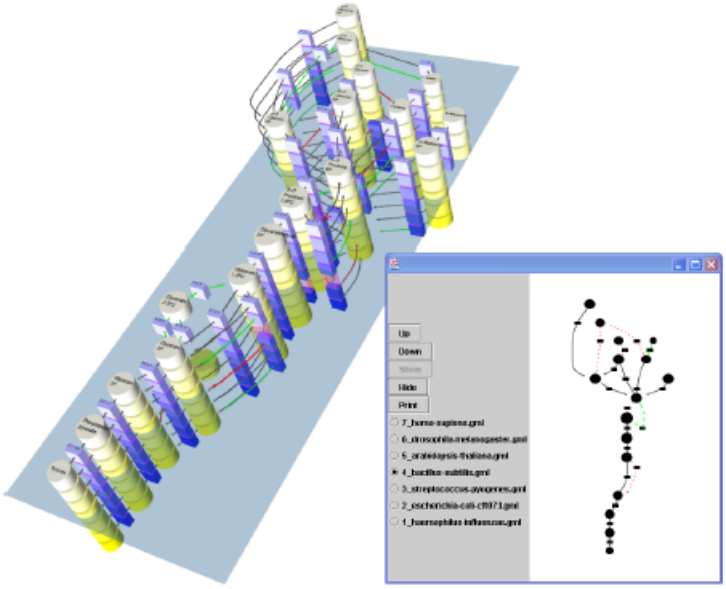


Figure 2.13: *A screenshot of a program where slight variations of a dummy graph are put on top of each other (see figure 5 (b) in [Brandes2004]).*

[18] `http://sbml.org`

2.4.3 Pathway visualization using virtual reality

Virtual Reality (VR) deals with computer animated, immersive and interactive applications. An extensive definition can be found in [Burdea2003].

In [Rojdestvenski2002; Rojdestvenski2003] a VR application called *Metabolic Network Visualizer (MNV)* is proposed. The tool integrates metabolic pathways in a VR environment by using the *Virtual Reality Modeling Language (VRML)* [Pesce1995]. They also discussed advantages and possible problems when using a real 3D representation of the biomedical network. Finally a hierarchical positioning of the graphs in 3D space as flat 2D graphs is proposed to keep the 2D graphs the biologists are familiar with.

In [Dickerson2003; Yang2005; Yang2006] the *MetNetVR* application is presented which focuses on hierarchical relationships in pathways. The root node of the hierarchy is the whole metabolic network. Pathways are child nodes of the network's root node. Sub-nodes of the pathways are in turn the molecules. For exploring the hierarchical information space the details on demand technique (see section 2.3.4 on page 25) is applied. By using a tracked input device, the orientation and position are used for the ray-picking of objects. Therefore the user can expand a single node to show all contained objects which are subsequent in the hierarchy. *MetNetVR* is implemented as a CAVE application [Cruz-Neira1993]. Figure 2.14 shows a user who interacts with the network inside the 3D CAVE environment.

Figure 2.14: *Screenshot of the MetNet3D CAVE application (see figure 1 in [Yang2006])*.

The application of VR for the visualization of metabolic pathways suffers from various drawbacks. No added value is generated due to the virtual environment. Rather the opposite is the case - the application loses the flexibility to be executed on a standard workstation. Therefore the framework's target user group is decimated dramatically.

2.5 Gene-expression analysis

Since the human genome as well as the genome of many other species is completely sequenced the main effort can move towards the identification of gene functions [Pellegrini2001].

The basis for achieving the goal of identifying gene functions is given by the invention of DNA chips (also called micro arrays) [Schena1995]. Micro array analyses result in a full profile of a genome from a tissue sample at a certain point in time. Therefore it is a snapshot that holds information about the regulation of thousands of genes. Another frequently used term for gene regulation is *gene-expression* which rates a gene's activation with a numeric value.

High-throughput methods (e.g. DNA chips) rapidly entail a downright data explosion. This exponential increase reinforces the visualization problem. Imagine an experiment for testing the human genome of one patient under different conditions (e.g. normal DNA and tumor DNA). If a visualization method takes one pixel per expression value the analysis does not fit on a modern screen [Seo2002]. As the screen space is a limited resource the challenge for *Information Visualization* solutions will even increase in future.

When doing research in the field of genomics and proteomics it is common practice to perform the same experiment several times under the same conditions for validating the results [Wolf2000]. Differences in the expression of genes could point to problems during the procedure (e.g. chip hybridization, image processing, etc). However, even when the experiment was completely valid it is nearly impossible to reproduce a result, even by using identically fabricated chips. Therefore biological replications as well as technical replications of an experiment are frequently made with the aim to generate more reliable results.

Clustering

Due to the character of the genomic data the application of statistical methods became very popular over the last years [Cavalieri2005]. Clustering of genes is intended to group genes that might share same or similar biological functions in the organism [Seo2002; Eisen1998]. By searching for well-explored genes in the clusters, biologists can try to deduce the functions of other genes in that particular group (i.e. the gene homology is utilized). Of course, this leads only to scientific hypotheses that are hard to verify. Often hundreds of experiments need to be carried out to finally find out the real gene function.

Widely used clustering algorithms are:

- k-means
- hierarchical
- Kohonen-map

It is important to mention that by using the clustering approach the gene function of a considerable amount of genes cannot be determined. In the case of a false annotation of a gene function the error can quickly propagate to other newly annotated genes [Lindroos2002].

Pathways have a great potential to overcome theses problems. [Pellegrini2001] proposed strategies how to accomplish this task by comprehension of cellular pathway knowledge. Publications on the determination of gene functions by using gene-expression data visually incorporated with metabolic pathways will be discussed later on (see section 2.6 on page 34).

Example: Hierarchical Clustering Explorer [2002]

[Seo2002] introduced the *Hierarchical Clustering Explorer (HCE)*[19]. The tool aims at combining gene clustering algorithms with visualization techniques. The hierarchical clustering is straight forward. The genes with the most similar expression value in the dataset are grouped together. Consequently this strategy applies also for already grouped entities. The result of the clustering is a binary tree that is illustrated in a dendrogram. The distance of a subtree from the root node is the similarity measure for the compared datasets.

Figure 2.15 shows a screenshot of the multiple view environment in HCE visualizing the clustering. The dataset consists of over 3600 genes that are measured in 38 different experiments. The diagram at the top shows the whole dataset with the dendrogram attached to the heatmap. The yellow highlighted area in the overview map is zoomed out in detail at the bottom.

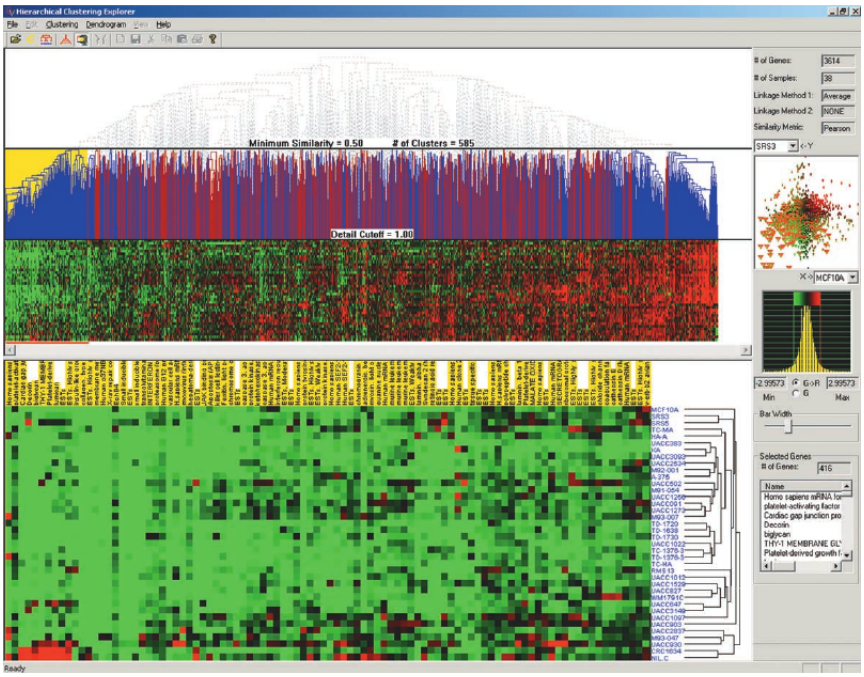


Figure 2.15: *Screenshot of the Hierarchical Clustering Explorer (see figure 2 in [Seo2002]).*

[19] http://www.cs.umd.edu/hcil/hce/

From information visualization point of view the HCE environment is a powerful tool because of the integration of a wide range of visualization techniques (e.g. multiple-views, *Linking & Brushing*, *Focus + Context*, etc.).

2.6 Application of gene-expression data onto metabolic pathways

One gene can be involved in the encoding of several distinct enzymes [Wolf2000]. Vice versa, a specific enzyme can be coded by several genes. Thus a bidirectional mapping between genes and enzymes is needed. [Wolf2000] described a system that is able to color code genes in pathways by using the keggcolor service[20] which is provided on the KEGG webpage.

Color Pathway [2002]

In [Lindroos2002] an interesting tool is introduced that deals with the mapping of gene-expression data onto pathways. They apply time-series data (see section 3.4 on page 54) of an experiment onto KEGG pathways by dividing the enzyme nodes into columns and color these sections according to their gene regulation value. Also multiple genes that are involved in the encoding of one enzyme can be represented by a horizontal splitting of the nodes. The outcome of this is a sort of color matrix. Figure 2.16 shows their approach applied to a KEGG pathway.

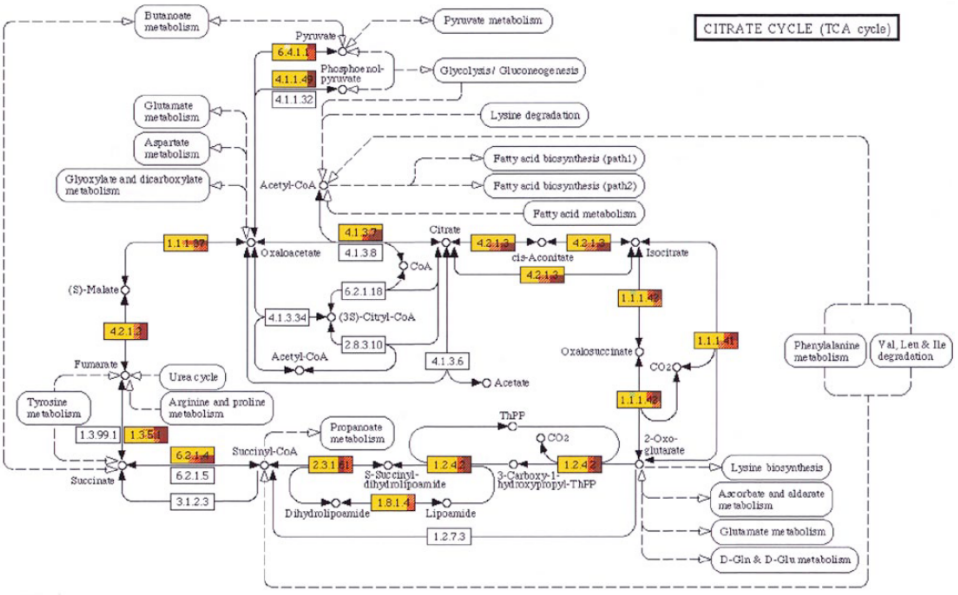


Figure 2.16: *Screenshot from the ColorPathway tool (see figure 3 in [Lindroos2002]).*

[20] http://www.genome.jp/kegg/tool/color_pathway.html

GScope [2003]

In 2003 a promising approach of visualizing DNA micro array data onto pathways was proposed by [Toyoda2003]. Their tool named *GScope* visualizes the bio-molecular network as a fish-eye view. The gene data is attached to the cellular network in the form of tiny colored heatmaps by using hyperbolic projection. The screenshot in figure 2.17 shows how this combination looks.

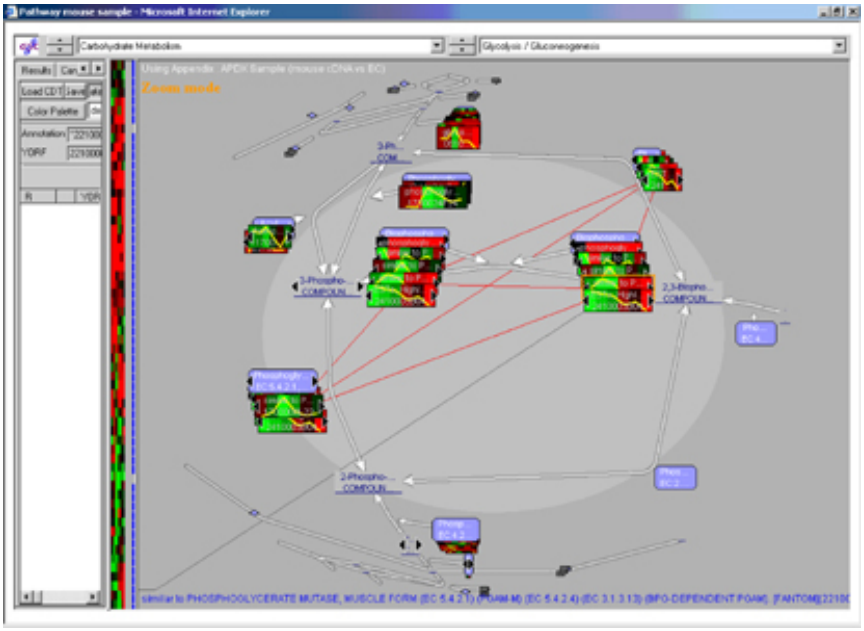


Figure 2.17: *Screenshot of the GScope fish-eye visualization (see figure 5 in [Saraiya2005]). The gene data is applied to the elements in the pathway as kind of heatmap diagrams.*

Micro array data applied to metabolic pathways are also integrated in numerous state-of-the-art frameworks which are introduced in the next section.

2.7 State-of-the-art pathway visualization frameworks

Many tools for the visualization and creation of metabolic models emerged. In this section we pick out some popular systems. After a short discussion of each, a comparison of their features follows. Most of the selected frameworks are also capable of mapping genomic information onto pathways.

2.7.1 Exemplarily selected state-of-the-art frameworks

GeneSpring [2000]

GeneSpring is focused on the analysis and exploration of expression data and is probably the market leader in gene-expression analysis software. The professional suite is written in Java and is available for purchase from Agilent Technologies, Inc. (Santa Clara, CA, USA)[21]. However, a free trial version can be downloaded. The software is powerful and feature-rich but also very expensive.

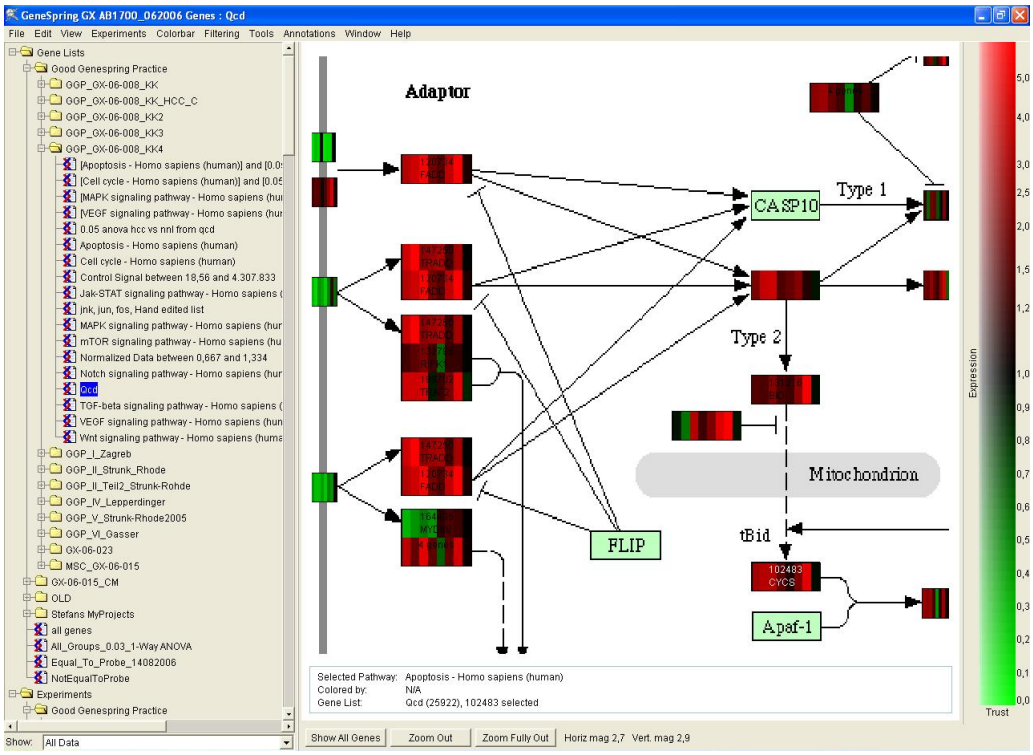


Figure 2.18: *Screenshot of GeneSpring.*

The screenshot of GeneSpring in figure 2.18 depicts a pathway with a gene-expression time-series experiment applied to the proteins (see section 3.4 on page 54).

[21] `http://www.agilent.com/chem/genespring`

Patika [2002]

Patika [Demir2002] stands for *Pathway Analysis Tools for Integration and Knowledge Acquisition* which states the main focus of the project. It consists of a server-side database and a client-side editor. For layouting the pathways the application adopts a specially designed approach described in [Dogrusoz2004]. The tool is designed for manipulation and visualization of signaling pathways (see section 2.1 on page 12) rather than their interactive exploration. The full version has not been released yet.

PATIKAweb [Dogrusoz2006] is a web service based on Java Server Pages (JSP) that is freely accessible on the project website[22]. The thin client gives read-access to the database. A simple implementation of micro array analysis integration is supported. A screenshot of the tool with a sample model loaded is shown in figure 2.19.

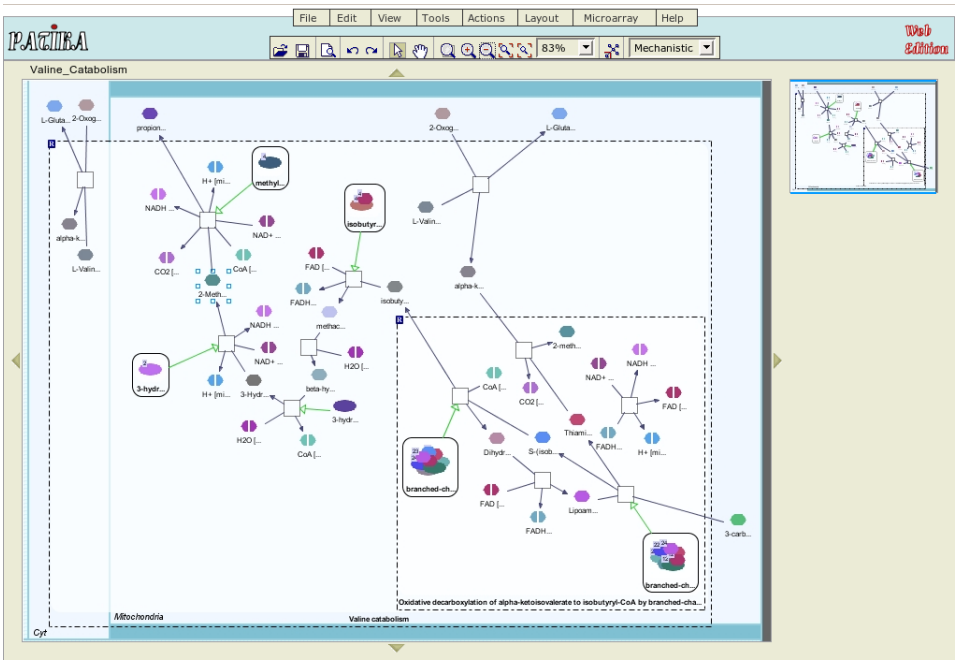


Figure 2.19: *Screenshot of the PATIKAweb application.*

[22] http://web.patika.org

GenMAPP [2002]

GenMAPP (Gene MicroArray Pathway Profiler) [Dahlquist2002][23] is a free standalone software application for creating new pathways or manipulating shared pathways from other researchers, stored on the *GenMAPP* server. The application draws pathways in a sticks-and-balls style. A sample pathway from *GenMAPP* is provided in figure 2.20.

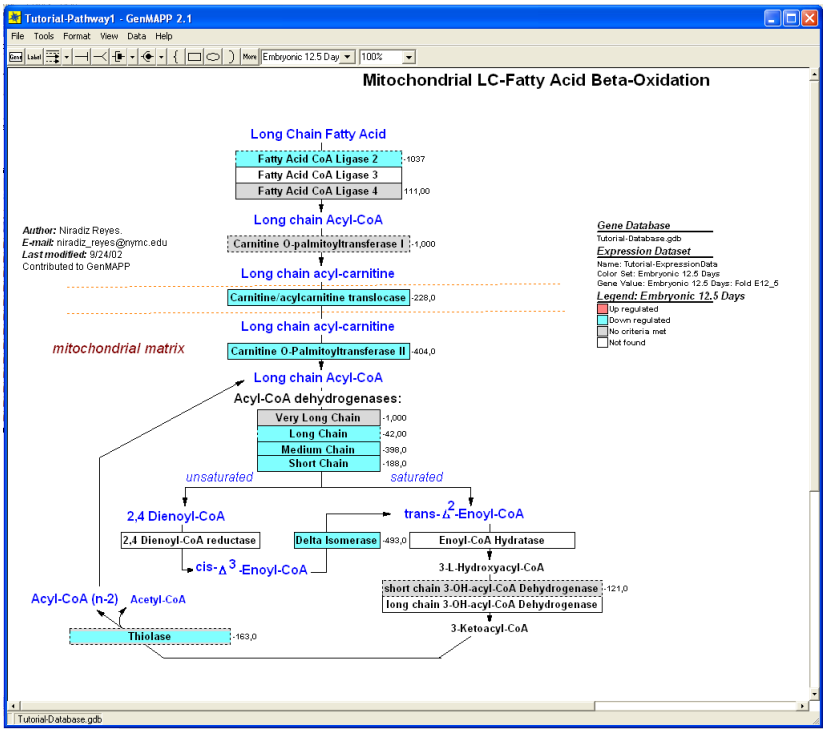


Figure 2.20: *Screenshot of GenMAPP.*

The tool supports the application of DNA micro array data onto pathways. The evaluation of time-series experiments (see section 3.4 on page 54) is not supported yet. GenMAPP lacks of interactive exploration techniques for related pathways [Saraiya2005]. It is only possible to operate on one graph at the same time. One positive aspect is the integrated tool for gene annotation mapping (see section 2.2.3.2 on page 19).

[23] `http://www.genmapp.org`

Pathway Studio [2003]

Pathway Studio [Nikitin2003][24] (formerly known as *PathwayAssist*) is a proprietary Software developed by Ariadne Genomics, Inc. (Rockville, USA). Standard features like the building of networks and gene-expression mapping are supported. For the map drawing a predefined set of nodes (e.g. for proteins, molecules, diseases) and edges (e.g. for regulation, expression) is available. In contrast to other pathways the shapes and colors of the map elements are fixed which increases the recognition value of the pathways.

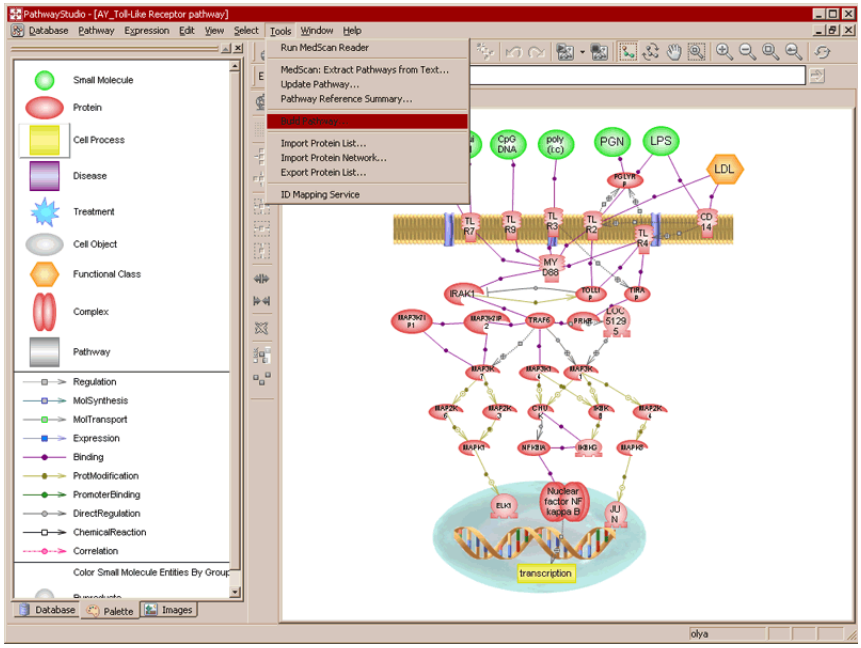


Figure 2.21: *Screenshot of Pathway Studio.*

Worth mentioning is the integrated text-mining tool MedScan Technology [Novichkova2003] which is able to scan and extract information from scientific literature (e.g. from PubMed). The method works with *Natural Language Processing (NLP)* algorithms [Kantor2001]. Consequently, the gathered information can be incorporated into the network. The extracted information as well as links to external resources can be directly attached to elements in the pathway.

[24] http://www.ariadnegenomics.com/products/pathway-studio/

PathwayExplorer [2005]

The *PathwayExplorer* [Mlecnik2005] is a web server application which is able to map genes onto enzymes in pathways from various databases (e.g. KEGG, BioCarta, GenMAPP). The system is able to handle input data in various formats and annotations. The mapping is performed on a server-side database called *PathwayDB*. Both, a web service and a standalone client, are available to the public[25]. The loading of pathways and the mapping takes a while. The mapping result is stored as PNG image.
The horizontal and vertical splitting approach of the enzyme nodes is basically the same as in section 2.6 on page 34. The main contribution of the *PathwayExplorer* is the integration of the mapping database. Figure 2.22 shows a screenshot of the browser content with the loaded PathwayExplorer application.

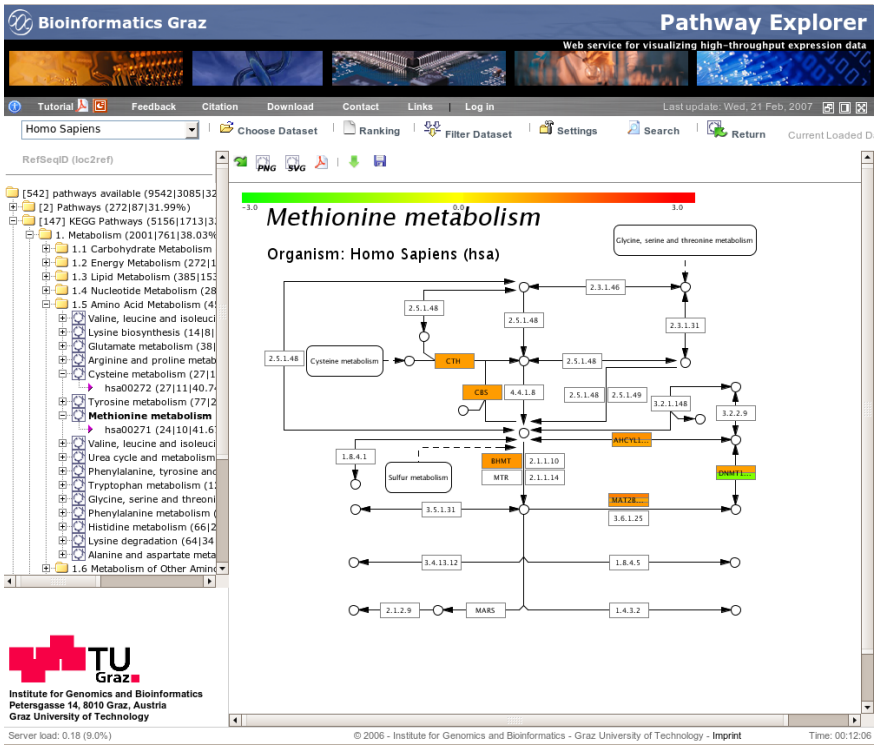


Figure 2.22: *Screenshot of the PathwayExplorer with an applied gene-expression profile to the KEGG Methionine Pathway.*

[25] https://pathwayexplorer.genome.tugraz.at/

GeneView [2006]

GeneView[26] is a visualization framework which is developed at the Institute of Computer Graphics and Vision (ICG) at the Graz University of Technology, Austria. The system is written in Java and supports various standard visualization techniques which can be flexible configured via configuration files:

- *Multiple views (see section 2.3.3 on page 23)*
- *Linking & Brushing (see section 2.3.7 on page 27)*
- *Details on demand (see section 2.3.4 on page 25)*
- *Focus + Context (see section 2.3.5 on page 26)*

In figure 2.23 a user interacts with a *GeneView* application by using an optical magic lens [Waldner2007].

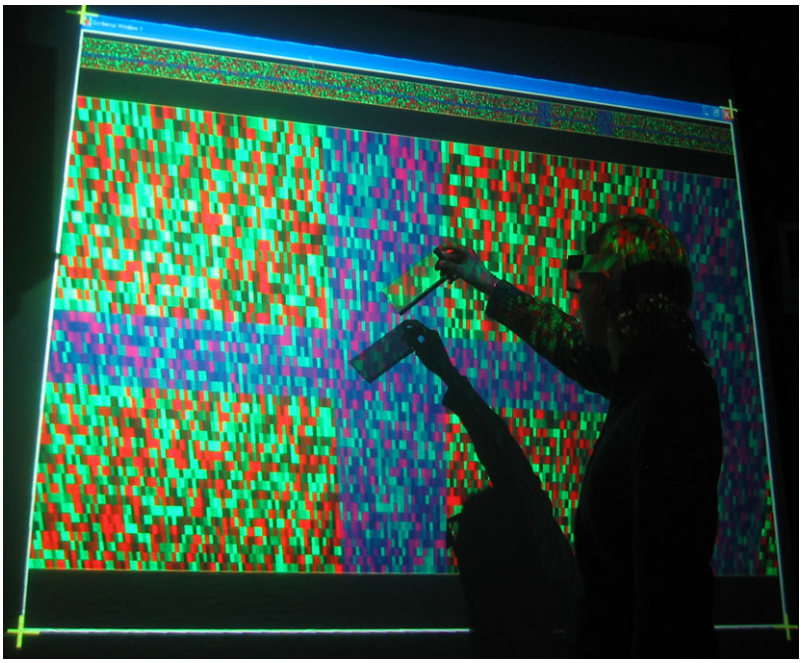


Figure 2.23: *GeneView application showing a 3D heatmap. The user stands in front of the projection wall and holds an optical magic lens. By using the magic lens the user can see a second visual representation.*

The framework is designed to allow an easy integration of new data and visual representations. It is optimized for interactive visualization coupled with multi-modal user input. The focus of the framework lies on interactive visualization which increases the memory requirements. Therefore *GeneView* uses memory management concepts proposed by [Kalkusch2005; Kalkusch2006]. Distributed visualization across multiple workstations linked via local area network (LAN) is supported.

[26] `http://www.GeneView.org`

Several different kinds of input data for biomedical research are supported:

- Micro array data
- Pathway data
- Micro array time-series data
- Clinical data

The pathway visualization is integrated as a module and uses the API of the *GeneView* framework. During this thesis the framework was continuously enhanced and improved. Details about the design of *GeneView* are discussed in section 3.2 on page 46. *GeneView* is available on request for academic use.

2.7.2 Evaluation and conclusion of the current state

Due to the diversity of the discussed frameworks it is hard to compare them.
Table 2.1 roughly summarizes the software suites.

Table 2.1: *Evaluation of selected pathway visualization frameworks.*

Application name	Available	License	Source code	Platform	Pathway creation	Interactive linked exploration	Gene-expression mapping
GeneSpring	✔	Proprietary	⊘	Win/Mac OS X	⊘	+ +	✔
PATIKAweb	✔	Own license agreement	⊘	Web-based	⊘	−	✔
GenMAPP	✔	Apache Software License	✔	Win	✔	+	✔
Pathway Studio	✔	Proprietary	⊘	Win/Linux/Solaris	✔	+	✔
PathwayExplorer	✔	Free	⊘	Web-based	⊘	+	✔
GeneView	✔	Academic use	⊘	Win/Linux	⊘	+ + +	✔

The previously presented tools are powerful but more or less static. Some of them deal with statically generated color maps - others apply the information only in a static manner. The main point of criticism is the lack of interactivity with respect to the whole network. If a tool supports interactive switching between pathways, which is not a standard feature, it is impossible to see influences of selections in related pathways at the same time. For example the most common way to show identical nodes in foreign pathways is to generate a textual listing. From that point the user can open a resulting pathway and investigate the connection to the original one. This solution is static and slow.

Without doubt, there are very powerful and comprehensive tools on the market. Often highly important features as for example the integration of gene-expression data is only

provided as a kind of add-on feature. Frameworks often provide a basic implementation which is then neglected.

To fill this gap we implemented a flexible and highly interactive visualization system which is capable of exploring metabolic pathways in detail without losing the context inside the network.

Chapter 3

System architecture

This chapter starts with a discussion of the system's concept. Due to the heavy dependency of the pathway visualization to *GeneView*, the framework and its modules will be described. This consideration is followed by a rather detailed presentation of the pathway data management. Finally the mapping process of gene-expression data onto the metabolic pathways is conceptually outlined.

3.1 Concept

A vast amount of knowledge is currently available from online databases. The data preparation and presentation of these services are far from perfect and therefore has lots of room for improvement. One example would be the web-portal presentation of the KEGG pathways, which uses hyperlinks to interconnect the information. This solution is limited in use and the data mining process is very time consuming for the scientist. Furthermore the user is always confined to the representation of one pathway at the same time. Extra-information is linked using the same static mechanism. The example illustrates that web browsers are inappropriate for exploring this kind of data - but as most tools use similar techniques they all suffer from the same problems.
To overcome these drawbacks, we aim for an integration of the pathway information in a data mining and data exploration tool. The user must be able to get a clue about the relations inside the network. We strive for an integrated solution that combines state-of-the-art visualization techniques (described in section 2.3 on page 20) in one system. Multi-monitor and multi-view methods are a necessity. The visual data exploration process needs to be supported by a consequent supply of meta-knowledge.

3.1.1 2.5D pathway visualization

Well established pathway visualization systems like for example GeneSpring present the pathways in 2D. Due to the planar design of the graphs this approach seems to be reasonable. The restriction to 2D allows only the display of one pathway. By using a workstation with multiple monitors attached the maximum of shown pathways could be slightly increased. However, for a in-depth investigation of the network and the dependencies between the graphs this is not enough. In this paper a hybrid solution is proposed: 2D in combination with 3D. By placing the planar pathways in a 3D scene we are able to simultaneously present multiple related pathways. This 2.5D approach is similar to the one proposed by [Brandes2004] (see section 2.4.2 on page 30). The difference is that

our solution supports arbitrary positioning of pathways in the 3D scene. This high degree of flexibility allows the creation of powerful exploration setups (see section 5.1 on page 71).

3.1.2 Identical node highlighting

Enzymes and compounds can be involved in several pathways (see section 2.4.1 on page 29). One enzyme can for instance catalyze multiple reactions inside the same pathway as well as in other pathways inside the network. The highlighting of these identical nodes in the graphs is of vast importance for the exploration of the metabolic network.

3.1.3 Neighborhood visualization

In graphs, where the layout aims at the positioning of nodes and edges to circumvent intersections, neighborhood visualization plays an important role. The geometric position of nodes is independent from their relational distance. Hence nodes positioned far away can be directly related to a node in focus whereas topographically close nodes can be several indirections away. Therefore the user could get a false impression about the topographic distances in the graph. A flexible visualization of neighborhoods can solve this problem. The distance can be depicted by color coding the nodes. We propose an algorithm which works with enzymes as well as chemical compounds as starting point in the pathways.

3.1.4 Details on demand

We implemented the details on demand technique (see section 2.3.4 on page 25) for presentation of all kind of meta-knowledge inside the framework. For example in the case of chemical compounds the user might be interested in the mass weight, the chemical formula, the structure and more. More information about the meta-data is given in section 3.3.3 on page 54. However, providing these additional information permanently, would cause an overkill for the user. Thus, we decided to provide special information only on demand. An information request is triggered by a mouse-over event or a picking action.
Our framework supports two distinct approaches to supply the user with information:

- Information browser
- Info-area integrated in a 3D view

Information browser

A standard HTML browser is well suited for the purpose of displaying detailed information. URLs can be pushed into the widget and the corresponding page (local or remote) gets automatically loaded.

Info-area in 3D view

The info-area is placed statically at the bottom of a 3D view. It is intended to view small plain text information pieces. The piece of information is updated on mouse-over events. A picking action is automatically triggered after a small time period that starts when the mouse cursor is moved to a new position. If the mouse cursor's position remains for a predefined time span (e.g. 0.3 seconds) the picking is performed (see section 4.4.3.2 on page 68). If an object is qualified as a positive hit, additional information is looked up in internal data storages. Finally the resulting information is served via the info-area to the user.

3.2 GeneView framework

All methods and system design issues which are covered in this chapter are integrated in the *GeneView* visualization framework (see figure 2.7.1 on page 41). *GeneView* was the basis for the development of the pathway visualization. During this thesis the framework has been continuously enhanced.

3.2.1 GeneView design

The block chart in figure 3.1 shows the modular structure of the *GeneView* visualization framework.

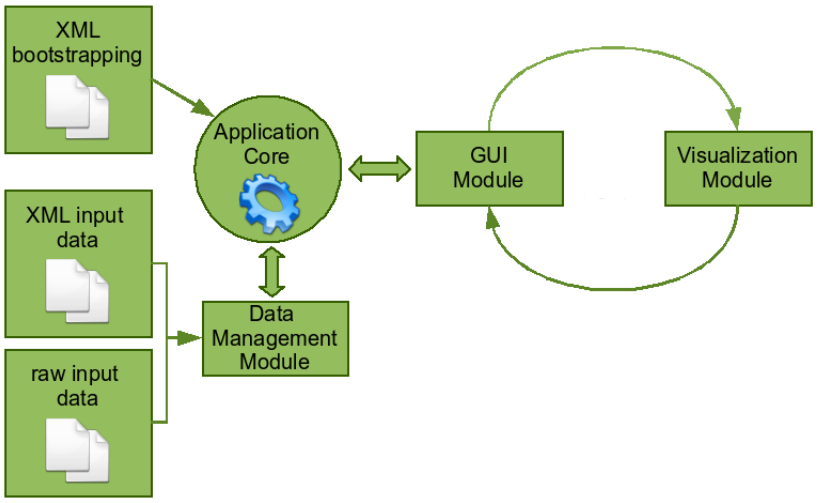


Figure 3.1: *Application block chart.*

The core application in the center of the chart takes the role of a controller. After the start of the application it triggers the XML bootstrapping. At this point data needed to set up the system is loaded from the XML configuration files. Information about the construction and the layouting of the GUI (for details see section 3.2.2 on page 47) is parsed as well as which data should be displayed in which view container. Also the

connection between the views and how notification messages should flow is determined in these files.

After the initialization of the system the management module starts parsing the actual data (see section 3.3). After this task is completed the control is handed over to the GUI module which sets up the user interface. At the same time the visualization module renders the data in the view. Finally the interaction loop begins to run. At this point the user actively controls the application. This process is similar to the action cycle described in section 2.3 on page 20.

3.2.2 Graphical user interface

The graphical user interface (GUI) of the *GeneView* framework is independent from a specific GUI library. Due to its good performance SWT is the preferred library (see section 4.1.1 on page 58)). The system is designed to support fast creation of new visualization setups. A setup is a special purpose combination of views. Views are canvas areas which are designed to act as a container for 2D and 3D visualizations.
It is possible to create arbitrary layouts (e.g. row layout, grid layout, etc.) by using XML configuration files.
The *GeneView* framework is capable of visualizing data in different representations:

- 2D pathway graph
- 3D pathway graph
- Scatterplot
- Heatmap
- Parallel coordinates
- Histogram

The UML class design in figure 3.2 clearly shows the separation of the view module and the GUI module. The GUI part manages the creation and the layouting of windows and containers while the view module is in charge of rendering the data in the canvas areas. For the sake of clarity interface classes are omitted in the UML diagram.

Note: According to the *GeneView* naming convention the letter “A“ at the beginning of class names indicates abstract classes. An ”I“ at the beginning of class names indicate interface classes.

GUI module

The `SWTGuiManager` creates SWT widgets (`ASWTWidget`) and stores them. The widgets are layouted in windows and containers as specified in the XML configuration file. The abstract widgets are further specialized in native SWT widgets and embedded widgets. Native widgets (`SWTNativeWidget`) are for example sliders, combo-boxes, tables, etc. - basically everything that is included in the standard SWT library. Embedded widgets (`ASWTEmbeddedWidget`) are containers for widgets that are using another GUI library than SWT. The current implementation differentiates between JGraph drawing areas (see

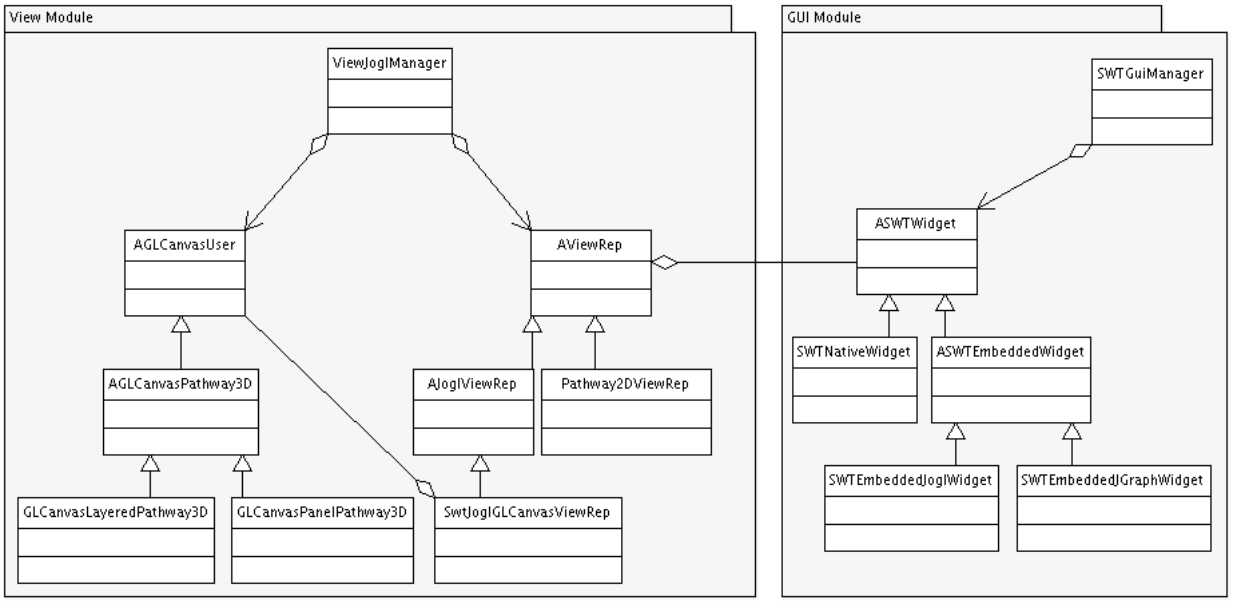


Figure 3.2: *UML class diagram showing the GUI and view management of the GeneView framework.*

section 4.1.3 on page 59) and Java OpenGL (JOGL) render areas (see section 4.1.2 on page 58). JGraph uses the Swing library. JOGL uses the AWT library. Therefore this design enables the integration of all three major Java GUI libraries (SWT, AWT, and Swing) into the *GeneView* framework.

View module

Central point of this module is the `ViewJoglManager` which creates and manages the views. Views are canvas areas whereas it does not matter if they are 2D or 3D.

All views are inherited from the base class `AViewRep`. Sub-classes can be divided into 2D and 3D views. 2D views are for example scatterplots, pathways, etc. 3D views need a special implementation and are consolidated in the `AJoglViewRep` class. Each 3D view holds its OpenGL canvas because this is the point where the OpenGL rendering loop is controlled.

`AGLCanvasUser` is the base class for 3D OpenGL views. Sub-classes are all OpenGL views that are supported by the *GeneView* framework. In this UML model only the pathway related part is modeled in detail. Pathways are specialized into the implemented showcase setups - the layered pathway visualization and the panel pathway visualization. The setups are presented in section 5.1 on page 71.

3.2.3 Command orientation

Every action in the *GeneView* framework is executed as a command. The command design pattern is described in [Gamma1995]. Figure 3.3 shows the command pipeline in *GeneView*.
Commands can be triggered by three sources:

- **XML file**
  Commands which are stored serialized in XML files can be processed on startup or dynamically reloaded at runtime.
- **GeneView API**
  The programmer can trigger commands via calling methods in the interface of the *GeneView* API.
- **Muddleware server**
  In the first instance *GeneView* is a single-user system but it is also designed to act in multi-user scenarios as a distributed application. By using the command mechanism actions can be transferred over the network in a serialized way. The system can be connected to a Muddleware server [Wagner2007] that stores commands in XML format. This approach allows the saving and reconstruction of complete program states on demand. Also the distribution of commands inside a workstation cluster is feasible. In the current implementation the Muddleware server is not fully integrated yet.

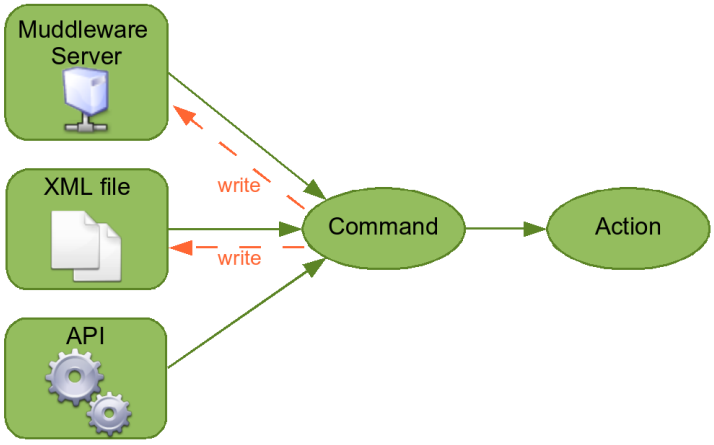


Figure 3.3: *Command pipeline in GeneView. Commands can be written back to XML files and to the Muddleware server.*

3.2.4 Update mechanism

A data update is the process of changing data and in turn notifying related parties. Objects of the systems are qualified to receive update notifications if they somehow act on this data. The receiver can decide which actions will be triggered and what to do with the received data.

Two kind of updates are distinguished in the *GeneView* framework:

- **Data update**
  Data updates are changes in the raw data. For example a regulation value of a specific gene that is changed by dragging a slider.
- **Selection update**
  A selection is a data portion that is chosen by the user.

For the implementation of the update mechanism the *GeneView* framework employs an enhanced version of the Observer design pattern, also known as Publish/Subscriber design pattern [Gamma1995]. The clear distinction between model and view is incorporated in the design of the update mechanism (see figure 2.3.3 on page 24).

Extending the update mechanism

The *GeneView* framework provides an interface for the communication with the event publisher module. For the involved parts the event publisher is a black box. Implementation details are encapsulated in the *GeneView* framework. At the beginning the involved views (`View A`, `View B`, and `View C`) need to register at the event publisher module. This registration step is shown in figure 3.4. At this point the views must declare whether they act as sender, as receiver or as both. Therefore bidirectional connections between views can be realized without difficulty. Furthermore senders and receivers can also be added at runtime.

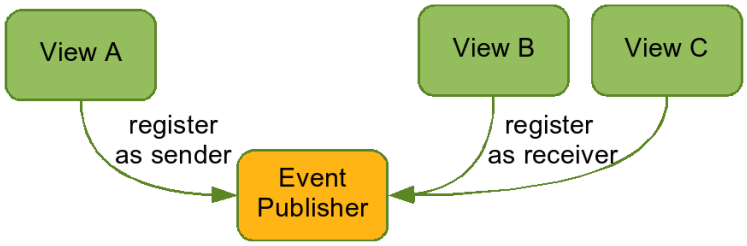


Figure 3.4: *GeneView sender and receiver update registration.*

After the initialization the publisher is ready to receive and forward events. In the example in figure 3.5 `View A` changes the data and sends an update notification to the event publisher. The event publisher looks up if the receivers, `View B` and `View C`, are registered to the events from `View A`. If this is the case both views get an update notification message. Consequently each view can start a read action to refresh the data.

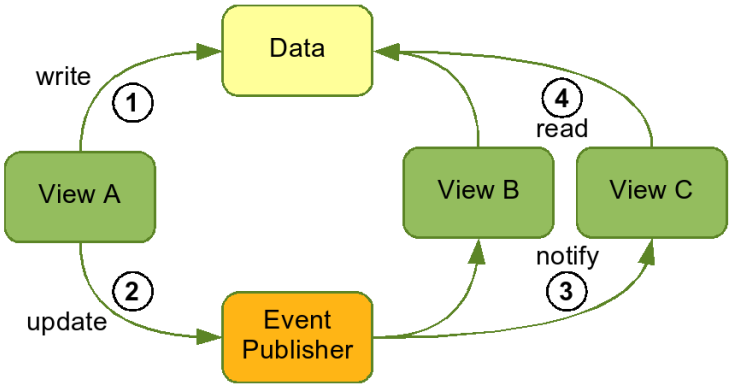


Figure 3.5: *GeneView data update mechanism. All views operate on the same piece of information.*

3.3 Pathway data management

The *GeneView* framework is able to handle data in various formats and from different sources. Possible sources are external and local databases as well as local and remote data files. We utilize the Proxy design pattern of the *GeneView* framework to integrate pathway data as well as gene-expression data.

Availability vs. performance

The idea of the data proxy is to treat the data management module as a black box that can only be accessed via a uniform programming interface. Therefore the data origin is not exposed to the accessing objects and the complex internal data handling is encapsulated. This approach is similar to the Proxy design pattern described in [Gamma1995]. Figure 3.6 presents the proxy concept.

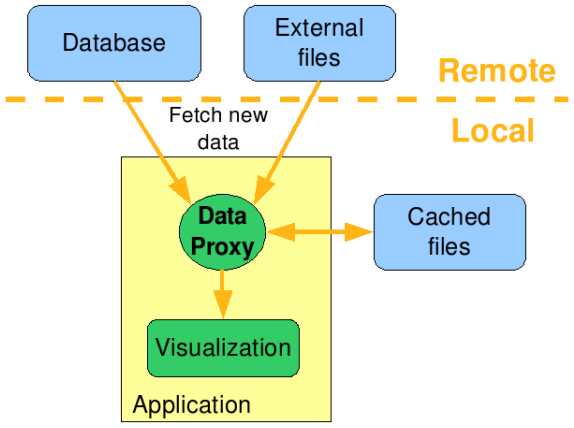


Figure 3.6: *Data block diagram showing the data loading process.*

Types of data

The following data sources are integrated:

- Pathway data: see section 3.3.1 on page 52
- Gene-expression data: see section 3.3.2 on page 54
- Meta-information: see section 3.3.3 on page 54

3.3.1 Pathway data

The pathway input data is taken from the KEGG project (see section 2.2.1.1 on page 15). The files exist in XML format and can be downloaded from the KEGG FTP server.

The design of the pathway data handling can be separated into the actual data representation and its internal management. The UML diagram in figure 3.7 shows the pathway data class model.

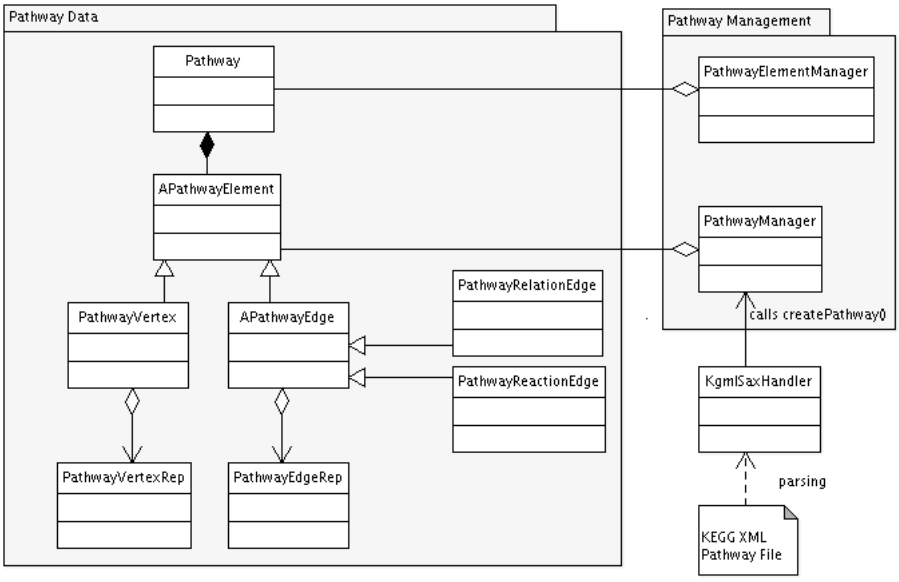


Figure 3.7: *UML class diagram showing the pathway data design.*

Pathway creation and management

The pathway creation is handled by the `PathwayManager` in close collaboration with the `PathwayElementManager`. The input data is parsed by the `KgmlSaxHandler` from a KEGG XML pathway file. The `PathwayElementManager` is in charge of the on the fly instantiation of the new pathway, edge and vertex objects during parsing of the XML file. When the parsing process has finished a reference to the new pathway is stored in a hash map by taking the pathway ID as a key. The hash map is kept in the `PathwayManager`. Vertices and edges are stored in the `PathwayElementManager` by using the same strategy. The element hashing allows fast access to all objects at runtime.

Pathway graph data

The design of the pathway graph is inspired by its natural representation. An object from type `Pathway` holds instances of `APathwayEdge` and `PathwayVertex`. Edges are further specialized into relation edges (`PathwayRelationEdge`) and reactions (`PathwayReactionEdge`). As both basic elements of a graph need to hold partly the same attributes, edges and vertices are inherited from the abstract element class `APathwayElement`. Examples for element attributes are the title, the ID, and so on.

Model data vs. view data

During the design of a visualization framework it is of great importance to clearly distinguish between two types of data - view data and model data [Heer2006; Tang2004]. In case of pathway graphs the topological information are the input data which are vertices and edges. An example would be an edge that connects vertex A with vertex B. When drawing the graphs each data element needs visual attributes. Applied to the example of an edge the visual properties are parameters like the color and the width. Figure 3.8 illustrates the abstract division of model data from viewing information in the *GeneView* framework.

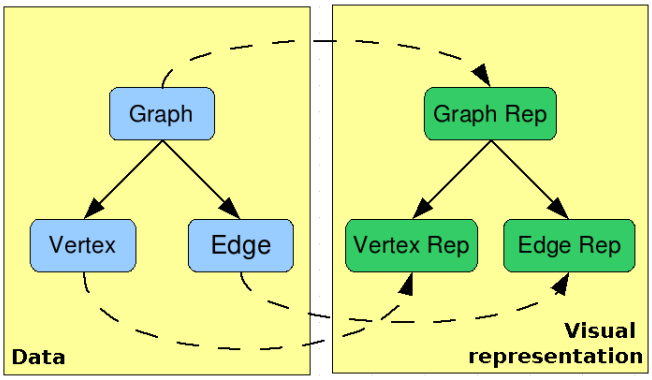


Figure 3.8: *Strict differentiation between model data and view data.*

This decoupling of view data and model data is considered in the *GeneView* class design. An `APathwayElement` instance holds references to an arbitrary number of pathway vertex representation (`PathwayVertexRep`) objects. These objects hold view-specific data. In the case of nodes this would be the x/y position in the graph, the color, and so on. The same design applies for edges and edge representations. Only the strict compliance to this principle enables a clean implementation of a multiple-view framework (see section 2.3.3 on page 23).

We utilize this approach to address the problem of multiple occurring entities in pathways. When a certain node is parsed for the first time a new `PathwayVertex` object is created. A vertex object stores its title, ID number, and type. Consequently a view representation is assigned to the vertex that holds the position in the graph, the color, the shape, etc. As the parser continues processing, every new node is looked up in the hash map to check if a node with the same ID already exists. If this is the case the `PathwayVertex` object is retrieved from the hash map and only a new vertex representation is added to

the array(`PathwayVertexRep`). Therefore it is possible that a specific node XY occurs in the same network multiple times, but with different viewing parameters (typically at least the x/y position in the graph).
We take advantage of this clean design in the selection and highlighting of pathway elements. This makes it possible to store a flag that indicates whether a node is highlighted. This mechanism helps implementing identical node highlighting.

3.3.2 Gene-expression data

The result of DNA micro arrays is exported to a raw comma-separated text file. This is the format in which we get the data from our medical partner. Each spot on the micro array is represented by one line in the file. The files usually contain a lot more information than needed for our purposes, hence the raw information needs to be filtered in a preprocessing step. For the further processing of the data two parameters per line are required: the *gene identifier* and the *gene-expression value*.

3.3.3 Meta-information

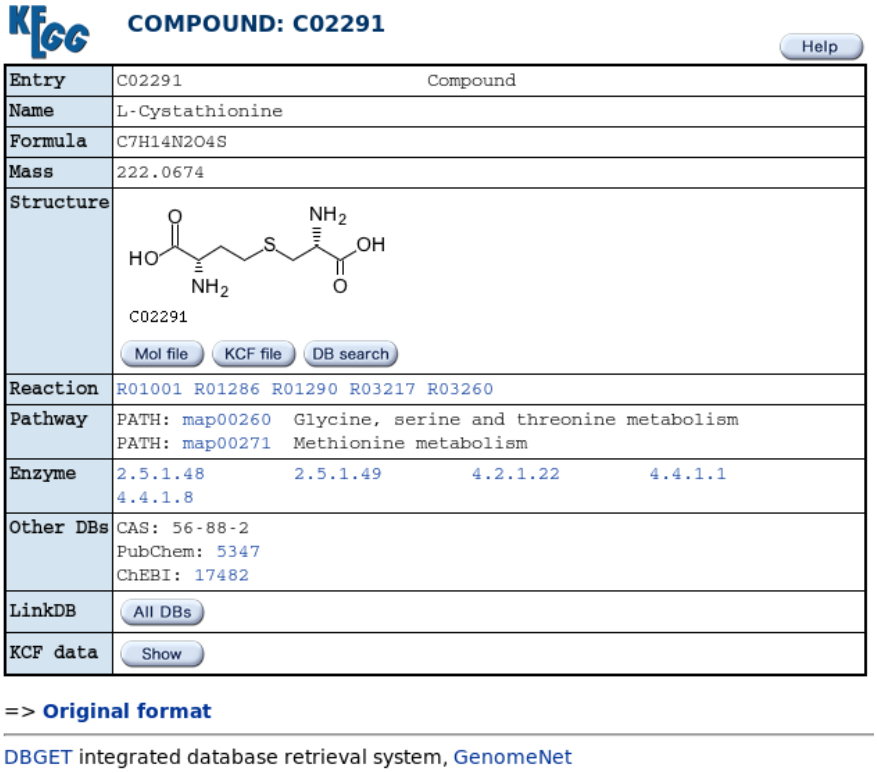
For data exploration and data analysis the system needs to supply the user with a wide range of additional information. Due to the vast amount of data, we access the information on demand from online databases. It is hard to decide which particular portion is important to the user, hence it makes no sense to filter the meta-knowledge that we received from the online resources. The integration of the data in HTML format is one possible way to address this issue. Figure 3.9 shows the meta-information for a chemical compound in KEGG presented in HTML. We need access to contextual information about gene data as well as pathway data. Therefore it is indispensable to integrate several databases. The gene information is retrieved from Entrez (see section 2.2.2.1 on page 18) and INSDBC (see section 2.2.2.2 on page 18). The extra pathway knowledge is read from KEGG (see section 2.2.1.1 on page 15)

3.4 Enzyme-gene mapping

For implementing the mapping of enzymes onto the pathway we had to face the question how to store the data. Considering the fact that the framework has to handle several thousands of enzymes and up to a factor of 10 more genes, the design decision has to consider these massive amounts of data.

The complete mapping tables are loaded into RAM at startup and are held at runtime. Of course, this approach considerably increases the working memory consumption of our application. However, we consciously accepted the high memory consumption because the application clearly focuses on short mapping times instead of memory-saving. Due to the data management of *GeneView* it is possible to dynamically unload data at runtime.

For each external identification number (e.g. enzyme codes, gene accessions, etc.) that is



COMPOUND: C02291

Help

Entry	C02291 Compound
Name	L-Cystathionine
Formula	C7H14N2O4S
Mass	222.0674
Structure	C02291
Reaction	R01001 R01286 R01290 R03217 R03260
Pathway	PATH: map00260 Glycine, serine and threonine metabolism PATH: map00271 Methionine metabolism
Enzyme	2.5.1.48 2.5.1.49 4.2.1.22 4.4.1.1 4.4.1.8
Other DBs	CAS: 56-88-2 PubChem: 5347 ChEBI: 17482
LinkDB	All DBs
KCF data	Show

Mol file KCF file DB search

=> **Original format**

DBGET integrated database retrieval system, GenomeNet

Figure 3.9: *Screenshot of the meta-information for the chemical compound "C02291" that is provided by KEGG in HTML format.*

fed into the system an internal ID is generated. The idea behind that approach is that in every stage of the system the data type can quickly be determined by investigating the ID. The disadvantage of this method is that an external to internal ID mapping table needs to be accessible at any time.

The mapping mechanism must take care of the fact that a bidirectional conversion of annotation types is required. For instance if an enzyme inside a pathway is selected, the user might want to know which genes produce that particular protein (i.e. mapping from enzymes to genes). On the other hand in a scatterplot that visualizes gene-expression results the user might be interested in information which enzymes are encoded by which genes (i.e. mapping from genes to enzymes). Those mappings from genes to enzymes and vice versa are depicted in figure 3.10. A further optional indirection between the mapping from gene accession numbers to gene-expression values can be necessary. Whether this indirection is needed depends on the result files from the micro array analysis. Different institutions use different annotations to map genes to regulation values. A sample mapping for real data is provided in section 4.5 on page 70.

Different types of associations must be considered for the design of the data structure. Possible annotation mappings are one to one (1:1), one to many (1:n), and many to many (n:m) relations. Usually all three types are mixed in the mapping process of enzymes to genes. A flexible data structure must take care of this fact. The 1:1 associations cannot be resolved in a preprocessing step because of the involved databases. The data in this

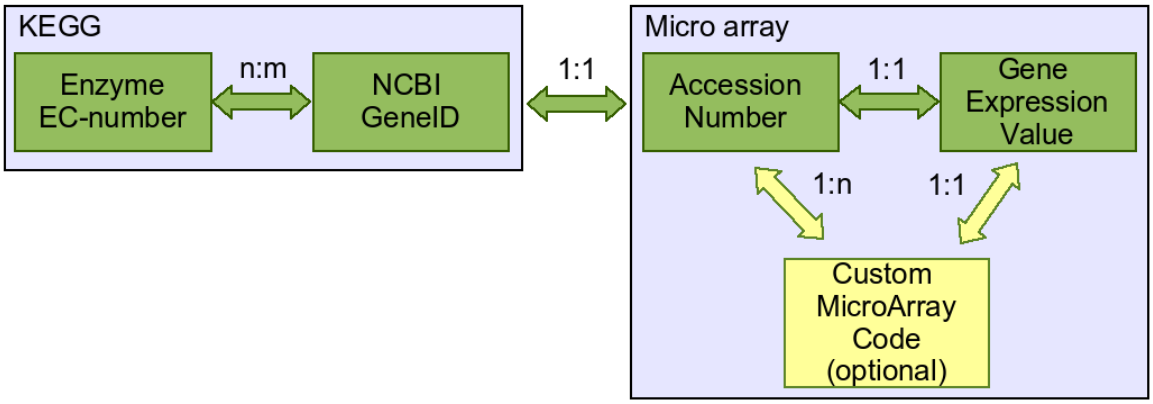


Figure 3.10: *Enzyme-gene mapping. The yellow part of the mapping is optional. A mapping sample is provided in figure 4.9.*

resources is very dynamic. Mappings that are present one day could be removed on the next. Associations and mappings can continually be added, removed or changed. If the association chain (see figure 3.10 on page 56) is partially incomplete the whole path must be discarded.

Our framework is capable of visualizing different gene mapping methods:

- **Single gene mapping**
  One gene corresponds to one enzyme in the pathway.
- **Multiple gene mapping**
  Often several genes are involved in the production of a specific enzyme. Each gene has its own regulation. Hence multiple genes need to be mapped onto one enzyme.
- **Time-series gene-expression visualization**
  A time-series experiment aims at revealing the temporal development of the gene regulation in a cell probe. Applied to pathways this knowledge can give clues about what happens in the cell over the time. The gene's regulation value is mapped to enzyme nodes in the pathway. This approach is shown in figure 3.11. If multiple genes correspond to an enzyme the current implementation calculates the mean expression value. For an example of a time-series experiment mapped onto a pathway in the GeneSpring application see section 2.7.1 on page 36.
- **Comparison between species**
  As explained in section 2.1 on page 12 pathways are often the same among species. Hence the pathway graphs are the same. They differ only in the genes that are mapped onto the enzymes in the pathway. Therefore species can be compared by mapping the genes for each species onto the enzyme nodes.

Basically the implementation of all of these gene mapping approaches is similar. Enzyme nodes are split up into one or more sections and a color coded gene-expression value is applied to each subsection. The difference lies only in the underlying data storage.

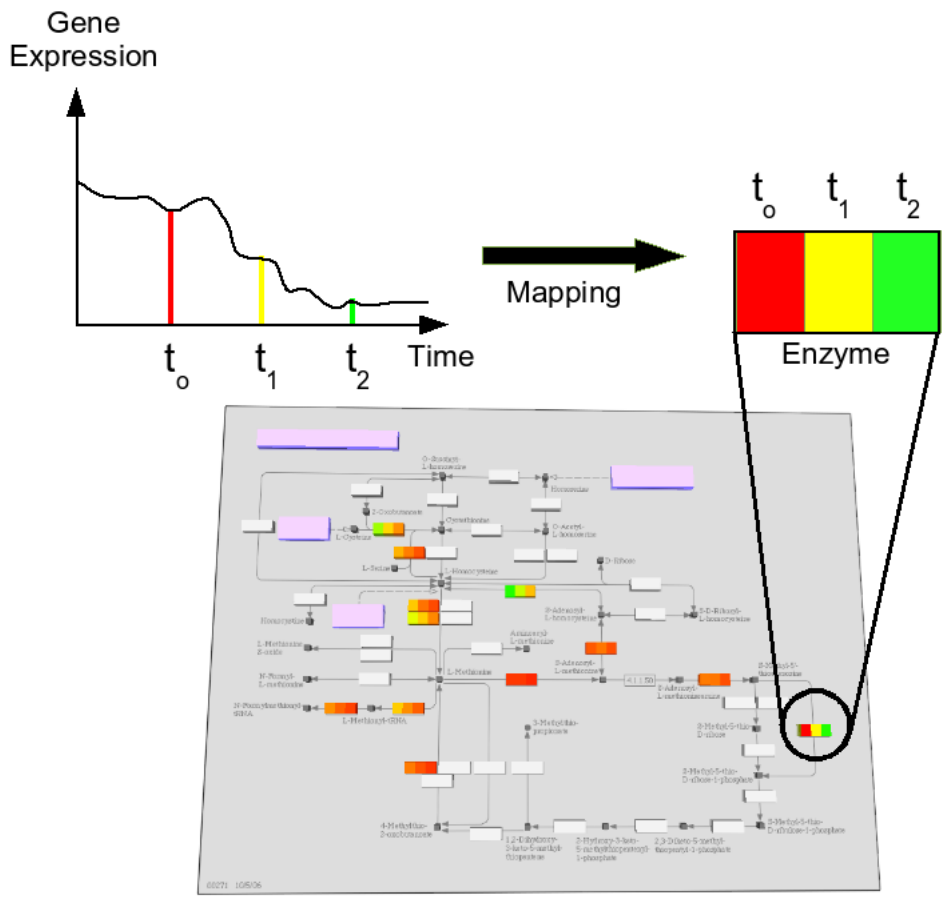


Figure 3.11: *Gene-expression mapping of a time-series experiment. The same experiment is carried out in three different points in time (t1, t2, t3). A transfer function maps the three gene-expression float values to the appropriate color. Finally the colors are applied to the corresponding enzyme in the Methionine Metabolism graph.*

(Pathway texture taken from `http://www.genome.jp/kegg/pathway/map/map00271.html` *)*

In case of transferring data between multiple connected views, the mapping process takes place on the receivers side in the update callback-method. The origin of this policy is the consideration that only the target view knows how to map the data because this is the point where the data is going to be visualized. This is also the perfect place for filtering the data (e.g. ignore data that is useless for the visualization in that view).

Chapter 4

Implementation

In the previous chapter the underlying basic concepts of the pathway visualization module in the *GeneView* framework were introduced. The realization of these architectural concepts and modules will be described in the following.

4.1 Used technologies

The framework is written in Java[1]. The development is done under Linux and Windows. *GeneView* is tested on both platforms.

4.1.1 Standard Widget Toolkit

The *Standard Widget Toolkit (SWT)*[2] is an open source library for creating graphical user interfaces [Harris2004]. The project was initiated by IBM within the scope of the Eclipse framework[3] in 2001. In contrast to the standard Java toolkits like the Abstract Widget Toolkit (AWT) and Swing, the SWT designers took a less abstract approach which is closer to the operating system. This low-level implementation results in better performance and a native look of the widgets. Therefore different packages of the toolkit are published which are adapted to the particular platform. Packages for a wide range of platforms are available.

4.1.2 Java OpenGL Library

The *Java OpenGL Library (JOGL)*[4] was released by Sun Microsystems in 2003. It is an open source library that integrates 3D hardware-rendering support for Java applications [Wolff2005; Xu2005]. The project is published under the *Berkeley Software Distribution (BSD)* License. The OpenGL 2.0 specification is fully supported by the JOGL API. Packages for Windows, Linux, Mac OS X, and Solaris are available.

[1] `http://java.sun.com`
[2] `http://www.eclipse.org/swt/`
[3] `http://www.eclipse.org`
[4] `http://jogl.dev.java.net/`

4.1.3 JGraph

JGraph[5] is a graph drawing library written in Java. The product is offered under a dual license - one is commercial and the other one is the *GNU Lesser General Public License* (LGPL). The free library contains the full feature set but lacks technical support, documentation and examples. Nevertheless the free version of the library is a good choice for the integration of graphs into an application. The API is well designed and provides features like zooming, undo, drag & drop, and much more.

The *GeneView* visualization framework uses SWT as GUI library. Although JGraph builds on Swing components we decided to take JGraph for implementing the 2D pathway visualization. The Swing graph widget is embedded in a SWT container widget. Therefore the two competing libraries (SWT and Swing) can be combined in the same application.

4.2 Selection interchange implementation

In the multiple view setup, in which 2D and 3D representations are mixed, the propagation of selections to other views is required. The data is synchronized by employing the *GeneView* update mechanism (see section 3.2.4 on page 50).
A selection object consists of three arrays:

- **ID array**
  For each node in the selection the internal ID is transferred. It is up to the receiving view to decide whether an element will be handled or not. The common decision criterion is the visibility of the elements in the current view.
- **Group array**
  In the case of multiple independent selections this field holds the classification to a selection group. This array is designated for future use and not used in the current implementation.
- **Optional array**
  An optional array is part of each selection object. This additional array increases the flexibility of the selection mechanism. One example for a possible usage is the forwarding of the neighborhood information (see section 4.4.1.3 on page 64).

[5] `http://www.jgraph.com/`

4.3 Pathway data loading

KEGG provides the data in XML file format. For parsing the data in Java, we had to face the choice between several ways that the Java Standard API offers.
Most relevant methods are [Niemeyer2002]:

- **Simple API for XML (SAX)**
  SAX is an event oriented parsing model. The XML document is sequentially parsed and predefined tags trigger callback functions during parsing. Only the filtered information resides in memory. Therefore SAX is well suited for large XML files that are read once.
- **Document Object Model (DOM)**
  DOM reads in the document and stores a representation of the model in a hierarchical structure. The hierarchy is held in the memory. This approach enables a navigation in the tree. DOM is suitable for multiple access of the data. The major drawback of DOM is the memory consumption in case of huge XML models.

GeneView natively uses SAX for parsing its own configuration files. Furthermore we decided to use SAX for reading in the pathways. After this preprocessing step the graph information is held in internal objects. Another determining factor is the potentially huge size of complex pathways.

KEGG pathways are defined in the KEGG Markup Language (KGML)[6]. In contrast to SBML (see section 2.4.1 on page 30) KGML mixes information of the map structure and the visual representation. Examples for the visual representation of nodes are parameters like colors, shapes, etc. The alternative to the XML data would have been the usage of the KEGG application programming interface (API). The Java programmed API employs SOAP for the transfer of XML information over HTTP. Due to the fact that real-time behavior is the main focus of the application, SOAP turned out to be too slow.

The following is an example of a pathway of the *Methionine Metabolism* in KGML format:

```
<pathway name="path:map00271" org="map" number="00271"
title="Methionine metabolism"
image="http://www.genome.jp/kegg/pathway/map/map00271.gif"
link="http://www.genome.jp/dbget-bin/show_pathway?map00271">

  <entry id="1" name="ec:2.6.1.-" type="enzyme" reaction="rn:R07396"
  link="http://www.genome.jp/dbget-bin/www_bget?enzyme+2.6.1.-">
    <graphics name="2.6.1.-" fgcolor="#000000" bgcolor="#FFFFFF"
    type="rectangle" x="335" y="571" width="45" height="17"/>
  </entry>
  ...
```

[6] http://www.genome.jp/kegg/docs/xml/

```
  <entry id="2" name="cpd:C08276" type="compound"
  link="http://www.genome.jp/dbget-bin/www_bget?compound+C08276">
    <graphics name="C08276" fgcolor="#000000" bgcolor="#FFFFFF"
    type="circle" x="466" y="494" width="8" height="8"/>
  </entry>
  ...
  <reaction name="rn:R01402" type="irreversible">
    <substrate name="cpd:C00170"/>
    <product name="cpd:C04188"/>
  </reaction>
  ...
</pathway>
```

The root node of the XML tree is the pathway tag which keeps information like name, identifier, online link for meta-information, and a path to the handmade graph image. The image is used for the texture overlay feature (see section 4.4.1.1 on page 62). The web links are provided for all graph objects and are utilized by the integrated web-browser for viewing extra information (see section 3.1.4 on page 45).

"Entry" tags contain information about the nodes (i.e. compound, enzyme, and maps). Embedded in the "entry" tag is the "graphics" tag which holds the graphical representation (e.g. color, x/y-position, width, height, etc.). By using this information a node can be exactly drawn as in the static images from KEGG. Chemical reactions that occur in the pathway are specified by the "reaction" tag and include a list of substrates and products that are involved in the reaction. In contrast to nodes, where the graphical representation is added to the XML file, the edges are stored without this information. This makes it impossible to automatically reconstruct the pathway graphs in the way they are provided as static images. Jourdan illustrates in [Jourdan2004] how the KEGG images differ from the generated reaction using the KGML information (see figure 4.1). Obviously all connections which belong to the same reaction are collapsed. This would be the desired result. How the *GeneView* framework overcomes this problem is described later on in section 4.4.1.1 on page 62.

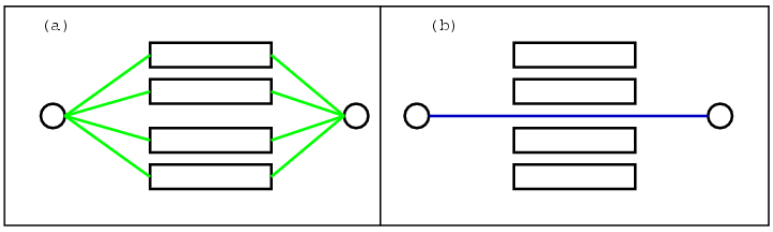


Figure 4.1: *KGML versus KEGG edge representation [Jourdan2004]. Both graph portions show a reaction that is catalyzed by multiple enzymes. (a) is the result from the generated representation by using the KGML information as input. (b) shows the KEGG version of the same reaction.*

4.4 Pathway visualization in GeneView

In the upcoming section the used information visualization methods are described. Furthermore the integration of these visualization techniques in the *GeneView* visualization framework is discussed. The methods are subdivided in 2D and 3D.

4.4.1 2D pathway implementation

The drawing of the planar graphs is implemented using the JGraph library (see section 4.1.3 on page 59). The x/y-positions of enzymes and compounds which are contained in KEGG XML files are used for absolute positioning of the nodes. A similar approach is proposed by Jourdan et al. where the layouting of the nodes is taken from the KGML files [Jourdan2003; Jourdan2004]. This approach has the advantage that researchers are used to that style.

The problem lies in the nonexistent data for the graphical edge representation in the KEGG XML files (see section 4.3 on page 60). Figure 4.2 shows an automatically drawn *Methionine Metabolism* pathway that uses the graphical information from the KGML files. In this representation the edges are straight connected to each other. By using this simple style the graph becomes cluttered in some areas which reduces the readability.

4.4.1.1 Pathway texture overlay

Because of the lack of edge routing information Jourdan [Jourdan2004] described an algorithm that aims for reproducing a similar edge routing as in the static KEGG pathway images. However, the *GeneView* framework avoids the implementation of an edge layouting algorithm by using the texture overlay method. The static handmade KEGG images that can easily be retrieved from the KEGG server are placed in the background of the graph. Concurrently the internal edges are hidden.
Figure 4.3 shows the same pathway as figure 4.2 but with application of the texture overlay in the background of the graph. The usage of this method results in a representation that corresponds to the version that the community is used to while preserving the interactivity of the graphs.

4.4.1.2 Hierarchical pathways

Due to the biological function KEGG added two abstraction levels above the metabolic pathways. The pathways are categorized in 10 groups which represent the highest layer. Figure 4.4 shows the three KEGG abstraction levels. In the web-based KEGG version each abstract level has an image-map in the back. In the image-map rectangular portions are defined and provided with a hyperlink to the next layer below.

We use the same mechanism for linking pathways between abstraction levels. In contrast to KEGG we store the image-map information in XML files which are parsed at startup.

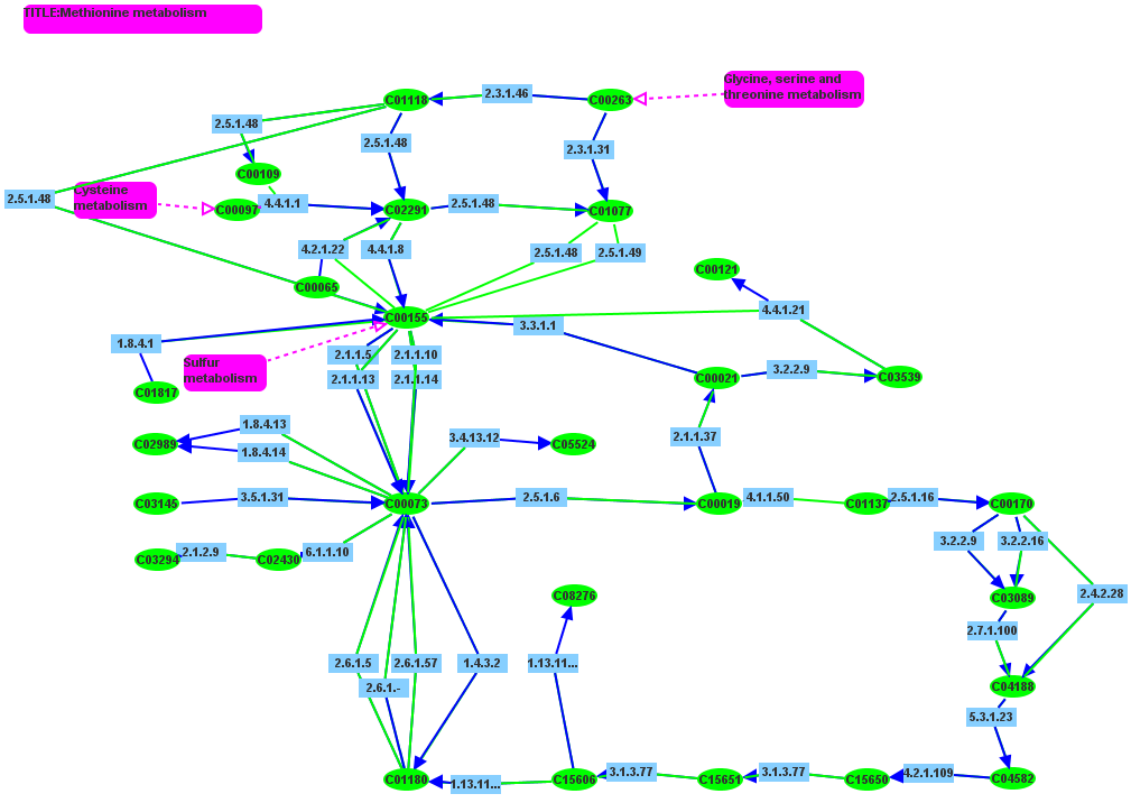


Figure 4.2: *Sample pathway showing the Methionine Metabolism without any layouting modification. The node placement is determined in the KEGG XML pathway file. The node layouting is the same as in figure 4.3.*

The following is a sample image-map in XML format:

```
<imagemap image="map01100.gif">
  <area shape="rect" coords="194,326,297,361"  link="map01110.xml" />
  <area shape="rect" coords="349,471,441,505"  link="map01150.xml" />
  ...
</imagemap>
```

When a mouse event is triggered on the 2D pathway at runtime, the clicked position is checked if it is contained in a predefined area. If a positive match is found the link for this area is returned and the requested next layer map will be loaded.

The context information about the location in the hierarchical KEGG levels would lead to a better orientation inside the metabolic network [Jourdan2003]. This feature could be a valuable extension for the future.

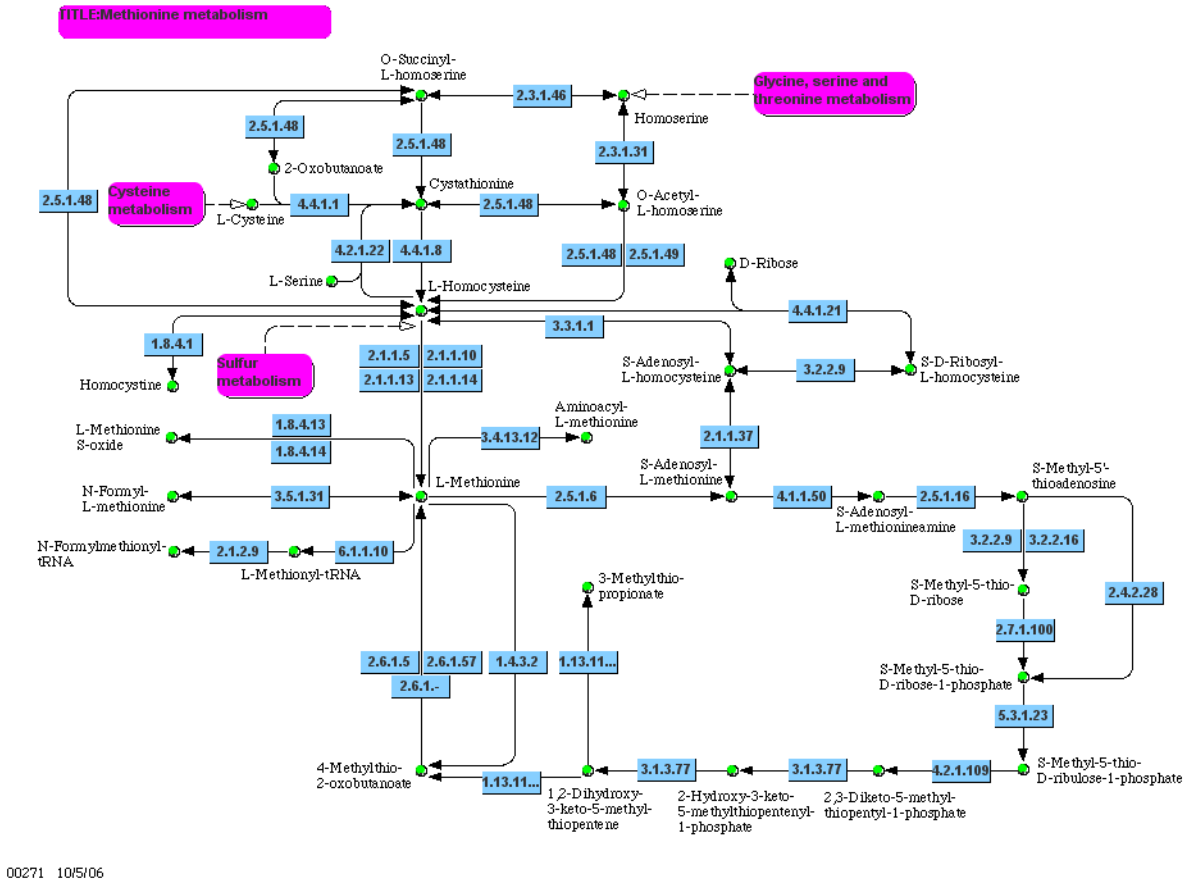


Figure 4.3: *Sample pathway showing the Methionine Metabolism with background texture overlay. The node layouting is the same as in figure 4.2.*
(Pathway texture taken from `http://www.genome.jp/kegg/pathway/map/map00271.html`*)*

4.4.1.3 Neighborhood implementation

For a in-depth exploration and understanding of the metabolic network the visualization of neighborhoods in the graphs is vital. We implemented the neighborhood visualization using a modified breadth-first-search (BFS) approach [Cormen2001]. The BFS algorithm visits all direct neighbors before descending recursively. If an already visited node is inspected it will be ignored. This assures that only the shortest path to a node is considered in an unweighted graph. The algorithm can be executed for arbitrary depth. Due to the current GUI implementation the user can perform a depth search up to a distance of 3.

The neighborhood algorithm for a distance of 2 is applied to the sample *Methionine Metabolism* pathway in figure 4.5. The selected node is colored red. The neighboring nodes are shaded from orange (distance 1) to yellow (distance 2).

Neighborhood coloring up to the depth of 3 seems to be the limit. The application of farther neighborhoods entails confusion at user's side because of the cyclic character of the graphs.

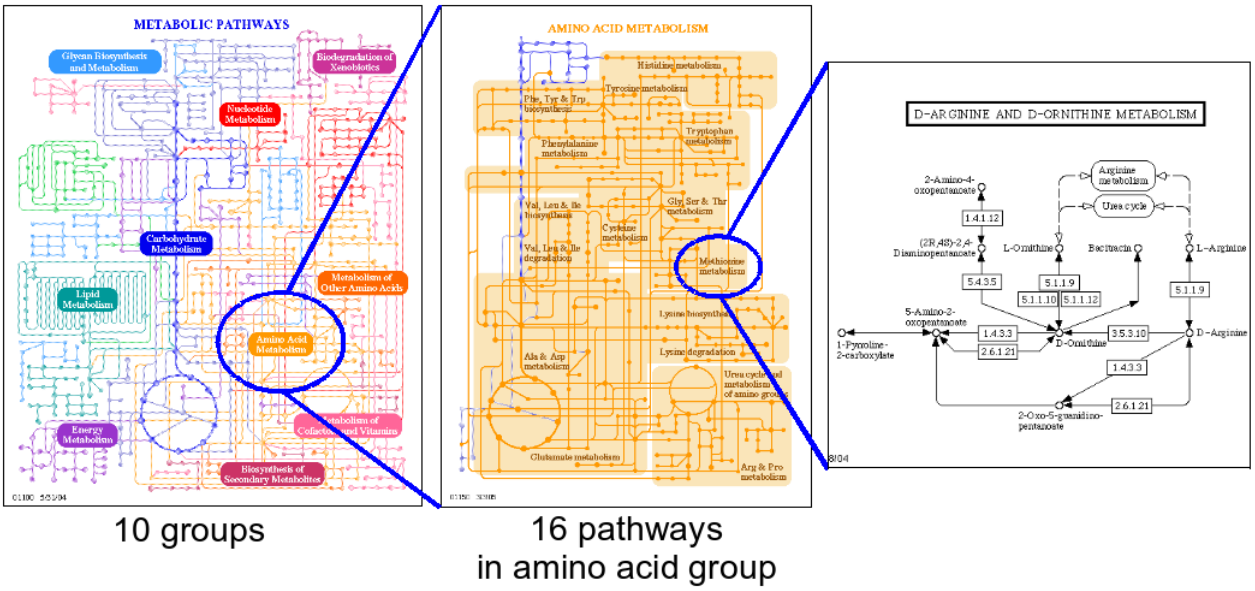


Figure 4.4: *KEGG abstraction hierarchy.*

Neighborhood selection forwarding

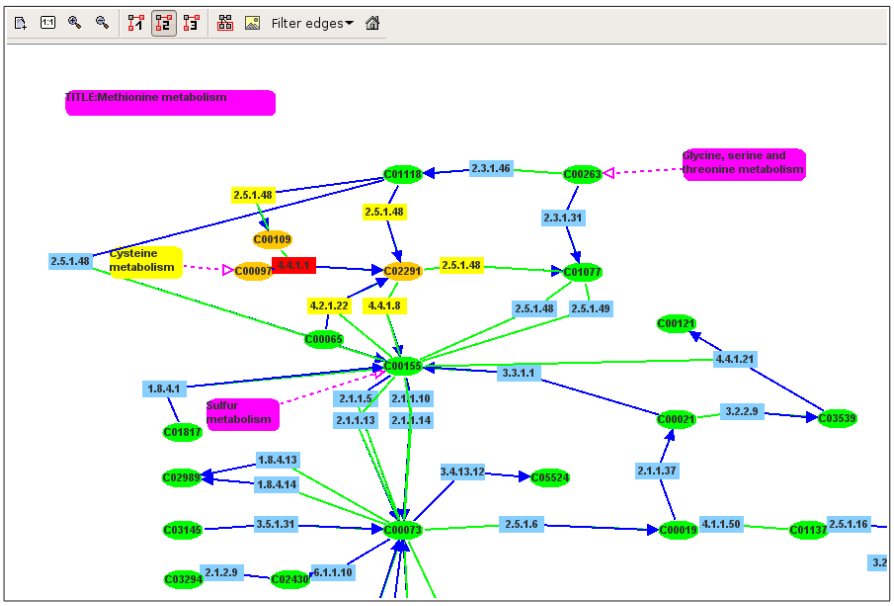
Neighborhoods that are shown in one view need to be published to other views that contain the same pathway. The example in figure 4.6 shows a color coded neighborhood. The red highlighted node is the selected one. The table in the example shows the object that is transferred to the receivers. The update object structure is described in section 4.2 on page 59. The optional array is used for storing the distance to the selected node which is the root node for the BFS algorithm.

4.4.1.4 Overview map

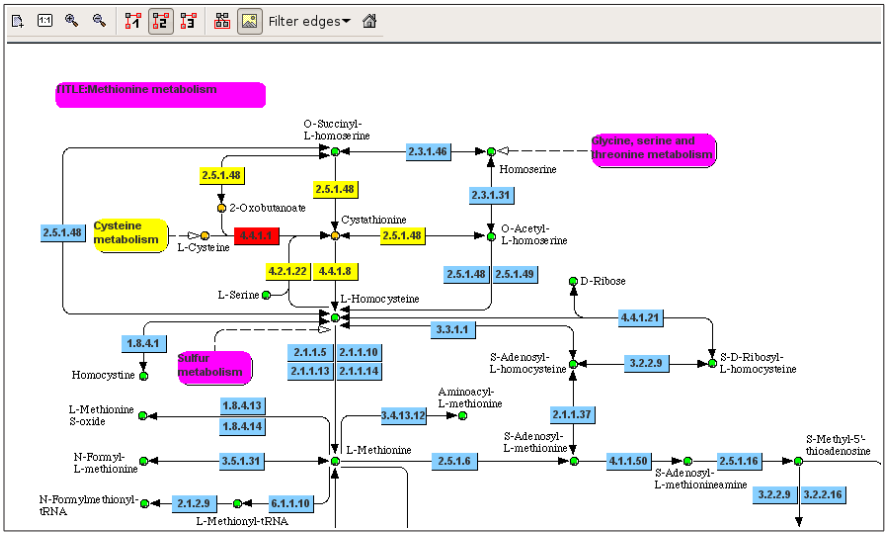
In order to retrieve detailed information about a specific area in the graph it is useful to implement some sort of zooming mechanism. When providing such a feature it is highly important to supply a visual way to keep the user oriented inside the network at any time. One way to achieve such a bird's eye view on the scene is the integration of an overview map. This approach can be assigned to the *Focus + Context* principle (section 2.3.5 on page 26). We use a separate window or area that reports the current position of the cut-out area inside the whole graph back to the user.
Figure 4.7 shows how the *Focus + Context* technique is combined with the *Linking & Brushing* approach in the *GeneView* framework.

4.4.2 2D pathway switching

The KEGG pathway representation supports the embedding of related pathways. In the KEGG drawing style linked maps are represented by round-rectangular nodes inside pathway graphs. Therefore it is important to support fast switching between them. In the online KEGG pathway version a click on this node loads the embedded pathway image.



(a) Neighborhood algorithm applied in 2D.



(b) The same pathway as in figure (a) but inclusive background overlay.
(Pathway texture taken from `http://www.genome.jp/kegg/pathway/map/map00271.html`)

Figure 4.5: *Color coded neighborhood visualization with distance 2 applied to the KEGG Methionine Metabolism pathway. The enzyme "4.4.1.1" is the starting point for the algorithm.*

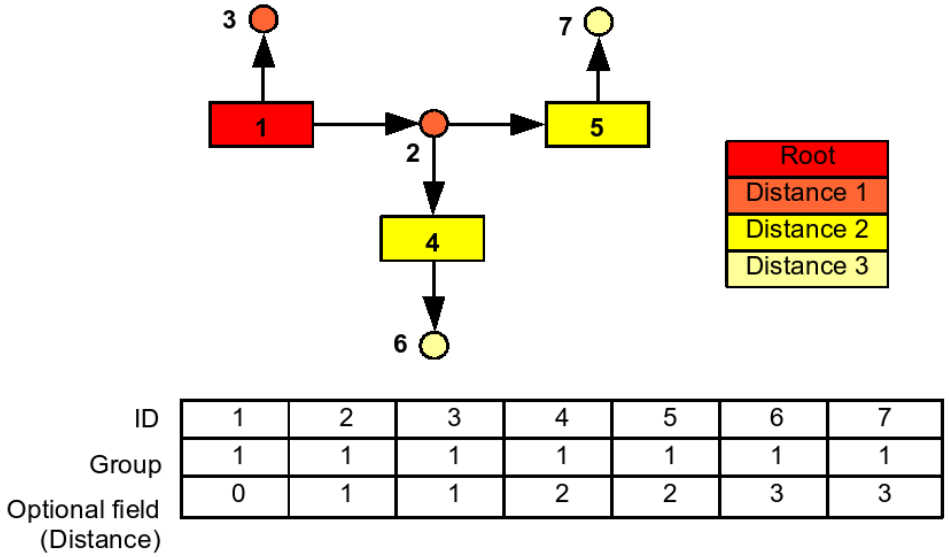


Figure 4.6: *Example of a neighborhood propagation object. The selected node is highlighted in red. The BFS algorithm marks the neighboring nodes according to their distance (orange = distance 1, yellow = distance 2, light yellow = distance 3). This neighborhood information is coded in the arrays of the update object.*

We implemented the switching action by replacing the currently shown graph which is triggered by a mouse event. This mechanism enables the user to freely navigate inside the metabolic network.

4.4.3 3D OpenGL pathway implementation

Pathway graphs in 2D views (as described in section 4.4.1 on page 62) allow the interaction of a single pathway at the same time. Even if the user can switch between the pathways this restriction holds up. For instance the task of identifying identical nodes in related pathways cannot be solved visually. This problem can be addressed by integrating the planar pathway graphs in 3D. Figure 4.8 shows the *Methionine Metabolism* pathway implemented in OpenGL. The texture overlay method as described for 2D pathways in section 4.4.1.1 on page 62 is also applied in the 3D version. The user can freely navigate in the 3D space. Zoom, pan, and rotation actions can be performed via mouse interaction.

4.4.3.1 Hierarchical display lists

The purpose of display lists in OpenGL is to group primitive objects for later usage. The group of objects is transferred to graphics card memory in the initialization phase (see chapter 7 in [Shreiner2005]). Thus the object can be reused without the expensive transfer operation. Especially for compound objects that occur multiple times in the scene the application of display lists can significantly boost the performance. A disadvantage

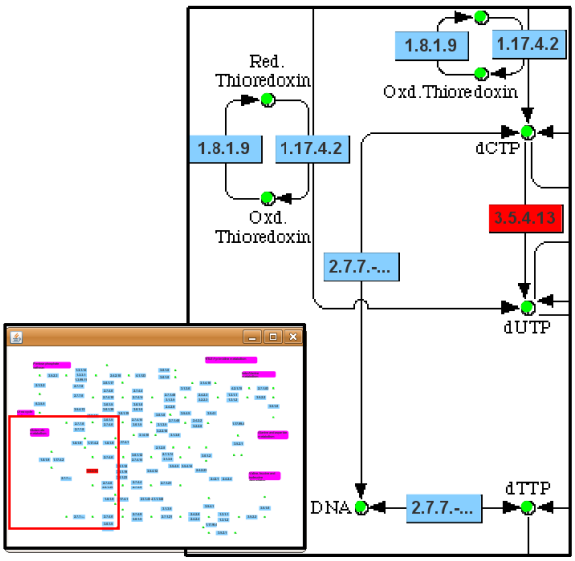


Figure 4.7: *The overview map depicts the position of the currently detailed shown area inside the graph. Even current selections are propagated to the overview map.*
(Pathway texture taken from `http://www.genome.jp/kegg/pathway/map/map00271.html` *)*

is that the lists cannot be altered after their creation. If the content changes the whole list must be recreated. In the case of pathways the scene is built up of hundreds of enzymes and chemical compounds which are simply depicted as differently colored cubes. A complete pathway can be created out of these primitive objects which are again stored in a display list. Using this approach results in hierarchical display lists where display lists are nested in higher level display lists. This empowers us to position a specific pathway more than once in the scene without sending all primitives through the graphics pipeline.

4.4.3.2 Object picking

Picking is the operation of selecting objects in the scene. In our case we employ the picking action to select nodes inside the graphs. In OpenGL element picking can be achieved by different methods. The straight forward approach uses the OpenGL build in selection mode (see chapter 13 in [Shreiner2005]). The user starts the picking operation by triggering a mouse-click event. The selection mode is entered and an area around the clicked x/y screen-coordinates is defined by applying a picking matrix on the current matrix stack. Accordingly this region is then used by the selection render mode to restrict the rendered area. In selection mode the render method returns a hit record which contains the information about the intersecting objects inside the region. The size of the region determines the sensitivity of the selection.

In addition to the described method we use hierarchical object picking. In that case multiple names are returned for each hit. Applied to our use case the scene is made up of several pathway graphs which represent the top-level elements in the hierarchy. In turn vertices

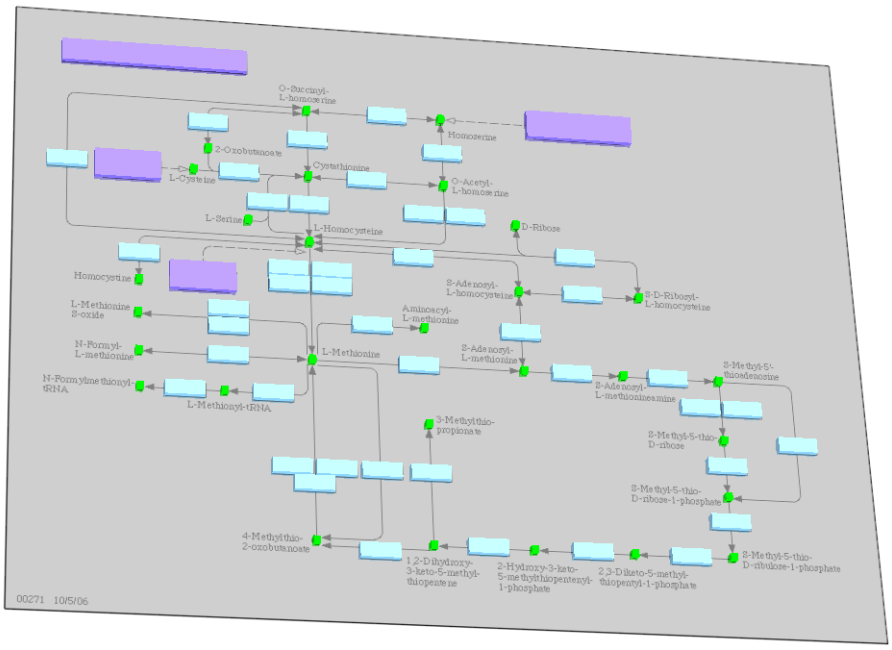


Figure 4.8: *Single 3D KEGG Methionine Metabolism pathway with texture background.*
(Pathway texture taken from `http://www.genome.jp/kegg/pathway/map/map00271.html` *)*

inside the graph are on the second level. Hence the picking result gives exact information about the node and the pathway the user has selected.

Increasing the number of visualized pathways in one scene also raises the probability of multiple objects under the cursor. A picking action returns a set of hits. As each hit comes with a depth value the nearest one to the viewport area is taken.

4.4.4 3D pathway switching

Navigating through the metabolic network by exchanging pathways is similar to the 2D version (see section 4.4.2 on page 65). The implementation is different. In a view containing a 2D pathway a switching operation clears the old pathway and paints the new one onto the canvas. As only one pathway can be shown at the same time this strategy is limited in use. In contrast to the 2D approach in 3D space an arbitrary amount of pathways can be placed in the scene. The planar graphs can differ in their rotation, orientation, perspective, and/or scaling factor. It is important that the visualization features (e.g. element picking, highlighting, etc.) are fully working without any modification in the code. We are able to achieve this by storing the model-view matrix. This approach makes it easy to replace pathways without knowing their viewing parameters and allows arbitrary layouting of pathways in 3D.

4.5 Enzyme-gene mapping

The transformation process of enzymes to genes (and vice versa) is described in section 3.4 on page 54. The mapping information is stored in hashing data structures. Due to the fact that arbitrary associations (1:1, 1:n, and n:m) need to be modeled, multi hash maps are the proper data structure. In contrast to usual hash maps, where one value is stored per key, multi hash maps allow the assignment of many values per key.
Figure 4.9 depicts the mapping process for a sample enzyme. The illustration also shows the mixing of conventional hash maps and multi hash maps. The map at the end of the pipeline holds the index of the value in the array that contains the gene-expression values. This flexible mechanism allows easy switching between storage arrays. In particular for time-series experiments the index in the storage array remains while only the reference to the array needs to be changed. This is possible due to the memory management of *GeneView*.

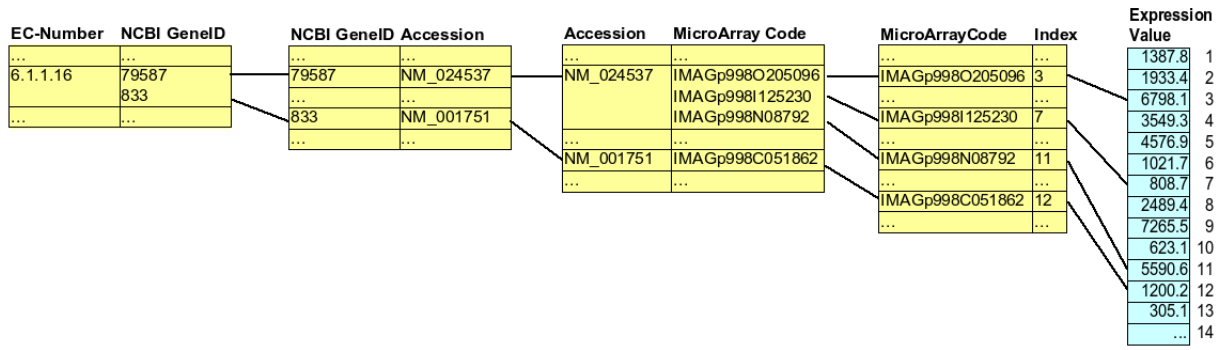


Figure 4.9: *Gene-enzyme mapping sample.*
The mapping process in general is described in figure 3.10.

Note: The mapping in figure 4.9 does not reflect the actual mapping process in *GeneView*. The principle is the same but the framework internally transforms all external IDs to local IDs. This enables a fast identification of the ID's type.

Chapter 5

Results

The last chapter discusses the implementation of the concepts and visualization techniques. At this point the findings of this thesis will be presented.

5.1 Showcase 3D pathway setups

Caching the model-view matrix per pathway allows a free arrangement of the planar graphs in 3D space. This approach makes it easy to create new setups.

Two exemplary setups are implemented:

- Layered pathway setup
- Panel pathway setup

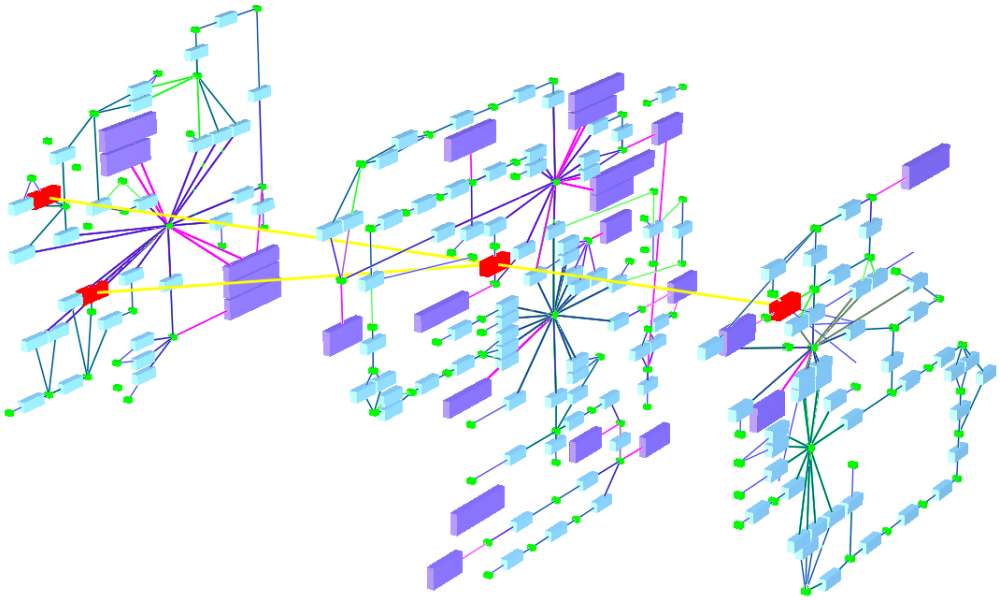
5.1.1 Layered pathway setup

The layered configuration arranges the loaded pathways as 2.5D planes in a stacked representation. An example with three layered pathways is shown in figure 5.1. In this figure the plain graph version is directly compared to the representation with enabled texture overlay. In both scenes the same selection of nodes is highlighted.

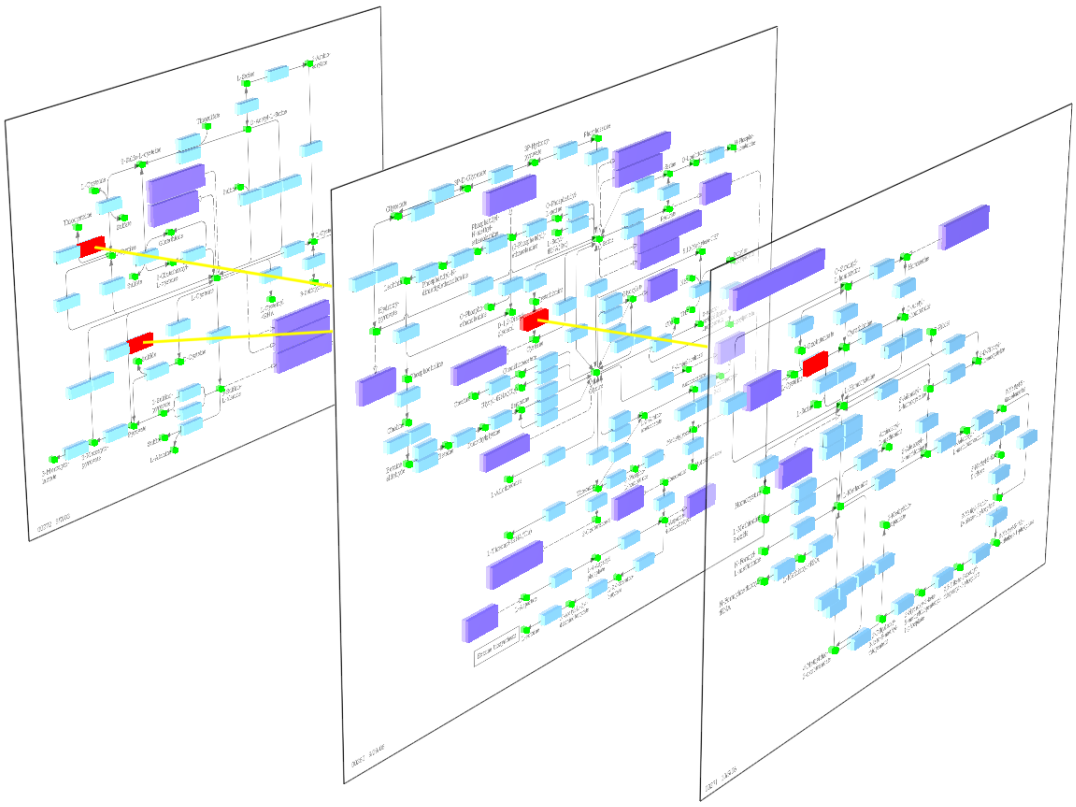
For cushioning the occlusion problem the 3D graphs support transparency. This is only useful for a small number of pathways because too many planes make the scene unmanageable for the user. Figure 5.2 shows a scene with two semi-transparent pathways.

5.1.2 Panel pathway setup

In the panel setup the user actively interacts with a pathway in the center of the view. Related graphs are positioned in a sort of gallery view in the background. The pathway in the center is slightly tilted towards the user. Therefore this configuration is called panel view. An example is shown in figure 5.3. The graphs in the background are too far away to recognize detailed structures. However, if the user selects an object in the tilted graph, identical nodes in related pathways are highlighted (see section 5.2 on page 74). To provide more insight into a specific graph in the back the user can perform a switch action to bring the desired pathway to the tilted position in the center.



(a) Directly connected edges.



(b) Blended texture background. Internal edges are removed.
(Pathway textures taken from `http://www.genome.jp/kegg/pathway.html`)

Figure 5.1: *2.5D OpenGL layered pathway scene. The enzyme node "4.4.1.1" in the middle pathway for Glycine, Serine and Threonine Metabolism is picked by a mouse-click event. Identical nodes in dependent graphs are interactively highlighted and connected to the selected one.*

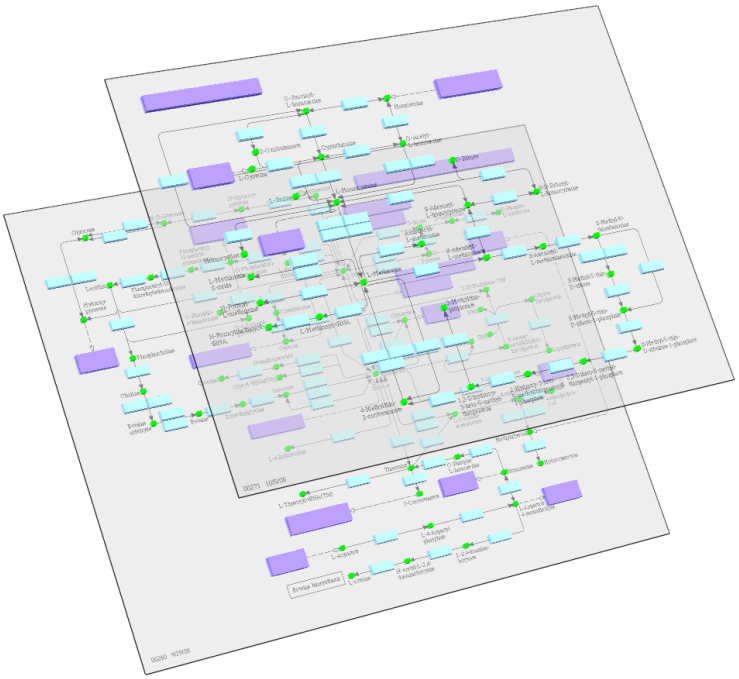


Figure 5.2: *OpenGL layered setup with transparent KEGG pathway textures.* *(Pathway textures taken from* `http://www.genome.jp/kegg/pathway.html` *)*

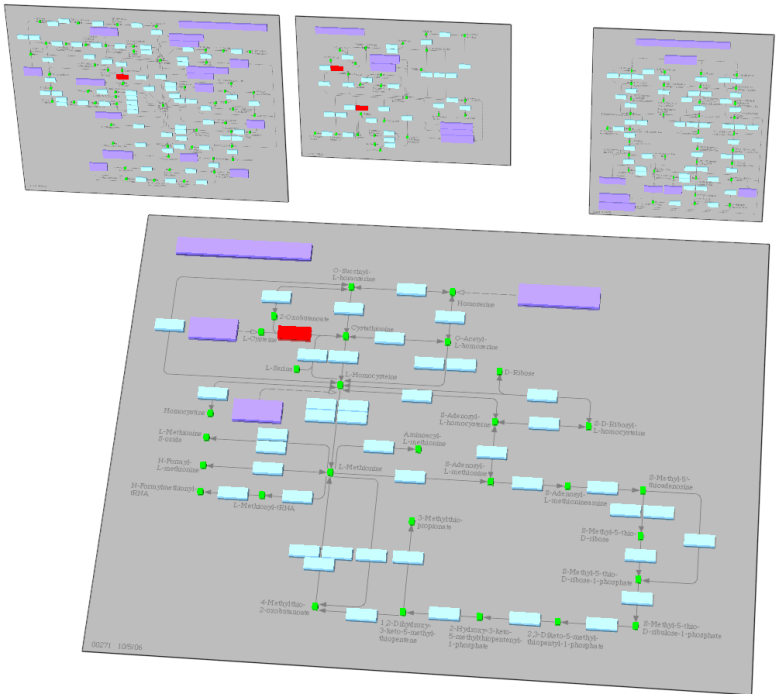


Figure 5.3: *OpenGL panel setup showing the Methionine Metabolism pathway which is in the center of the user interaction. The enzyme "4.4.1.1" is picked and highlighted. Identical nodes of that enzyme inside the graphs in the background are highlighted too.* *(Pathway textures taken from* `http://www.genome.jp/kegg/pathway.html` *)*

5.2 Identical node highlighting

The approach of highlighting multiple occurring nodes in pathways is described in section 3.1.2 on page 45. Two methods are combined to visually emphasize the nodes: changing the color and a slight pulsation in size. To support a fast perception of identical nodes in the layered pathway setup, the nodes are additionally linked by a line among the layers. Figure 5.4 shows an example of two stacked pathways in which the line linking approach is depicted.

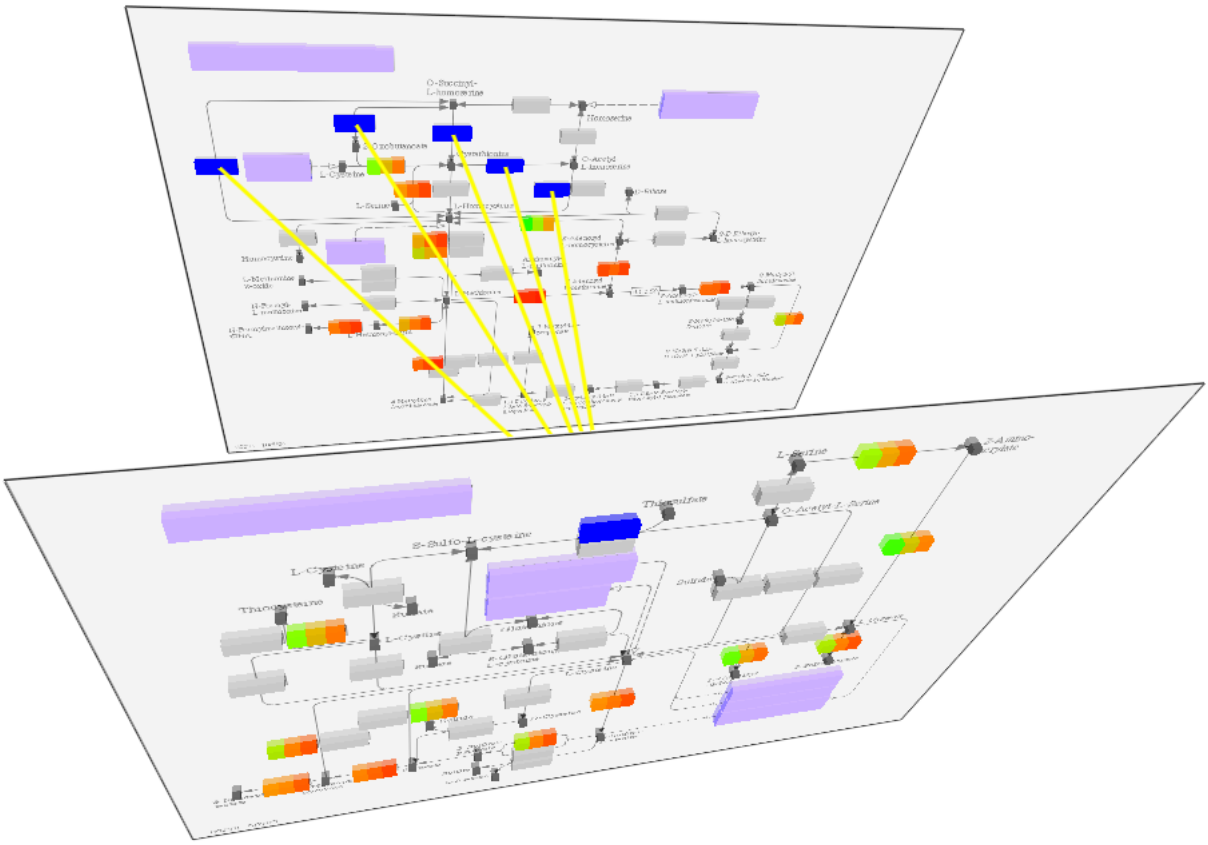


Figure 5.4: *Line linking among different pathways.*
(Pathway textures taken from `http://www.genome.jp/kegg/pathway.html` *)*

The example in figure 5.9 shows the synchronized highlighting of duplicate nodes inside the metabolic network. The highlighting of nodes and concurrent linking to foreign views is known as *Linking & Brushing* (see section 2.3.7 on page 27). An element that is selected in a pathway is highlighted system wide in all currently shown representations. Hereby it does not matter if the node is in the same view (e.g. in another layer in the layered representation) or in other visible 2D and 3D views.

5.3 Details on demand

The idea behind the details on demand approach is outlined in section 2.3.4 on page 25. Section 3.1.4 on page 45 describes how this strategy is incorporated in our system. Figure 5.5 shows an information HTML browser integrated in the framework.

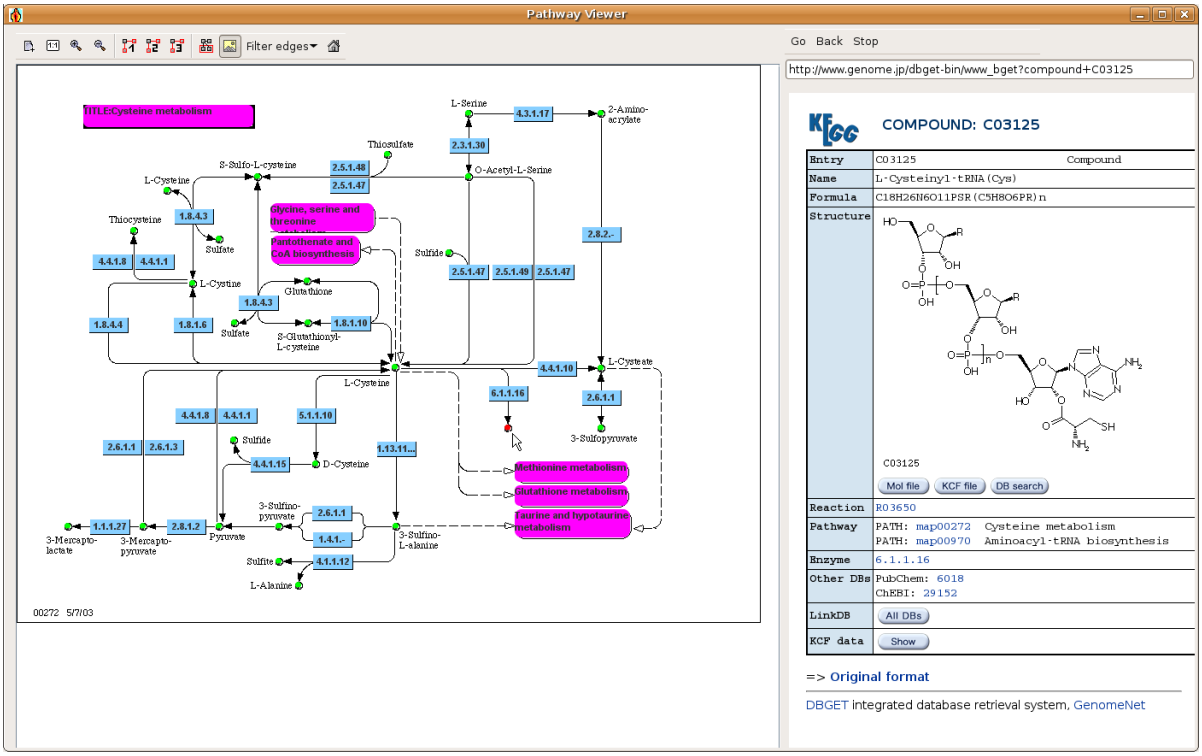


Figure 5.5: *A screenshot of the integrated information browser with loaded information on the chemical compound "C03125". On the left side of the window the Cysteine Metabolism pathway is loaded in a 2D view.*

An example of an info-area in 3D is shown in figure 5.6. The browser loads the information from the online database when the mouse pointer remains for a predefined time span over an element in the pathway. This mechanism works for elements in 2D as well as in 3D.

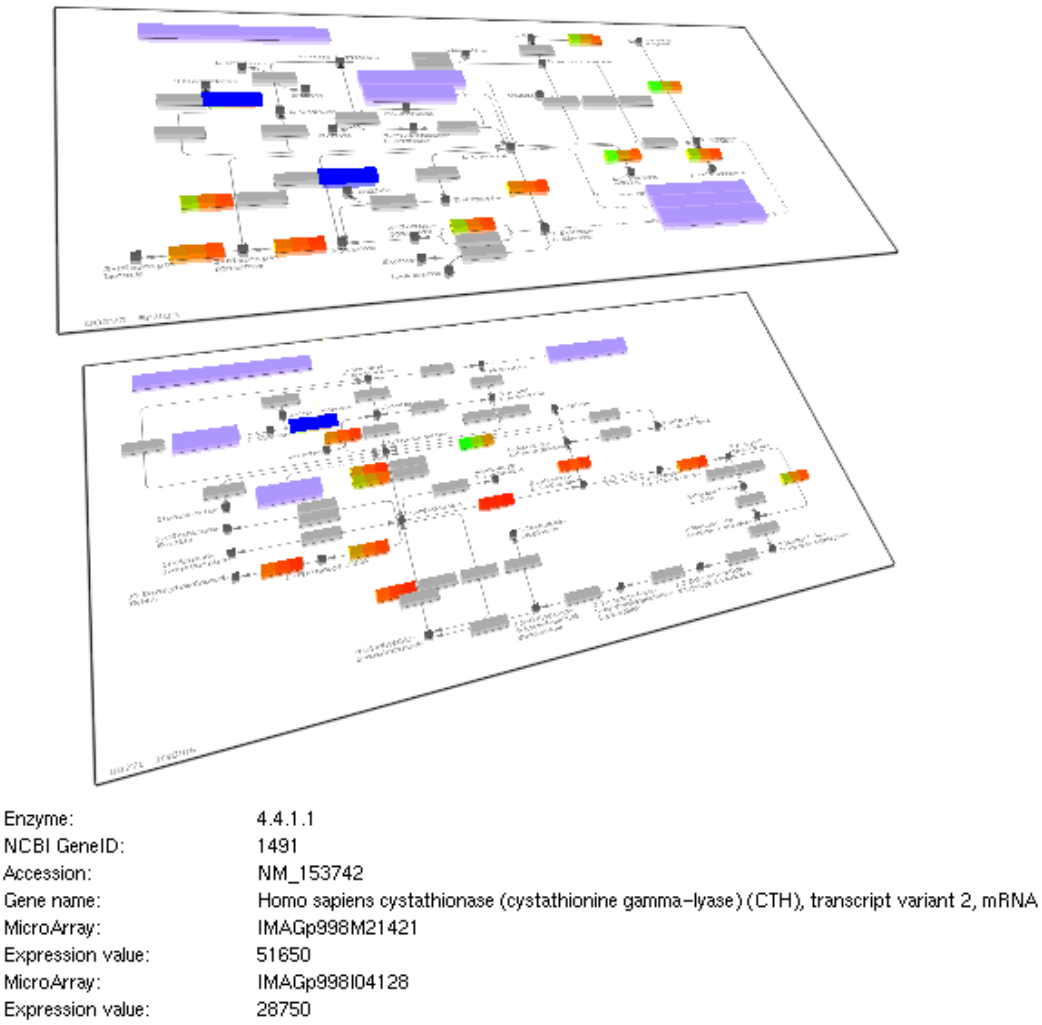
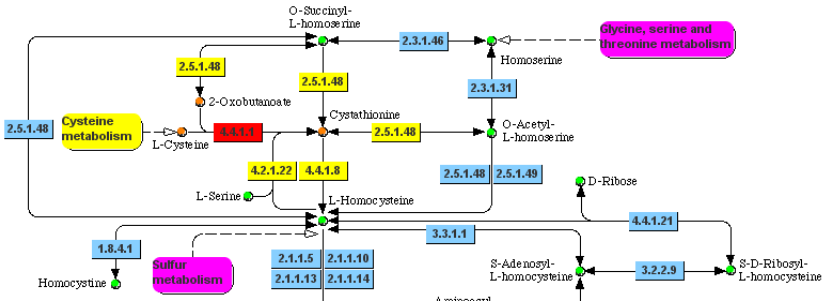


Figure 5.6: *Info-area in 3D view. The information gets updated when the user stays with the mouse cursor at least 0.3s over an object.*

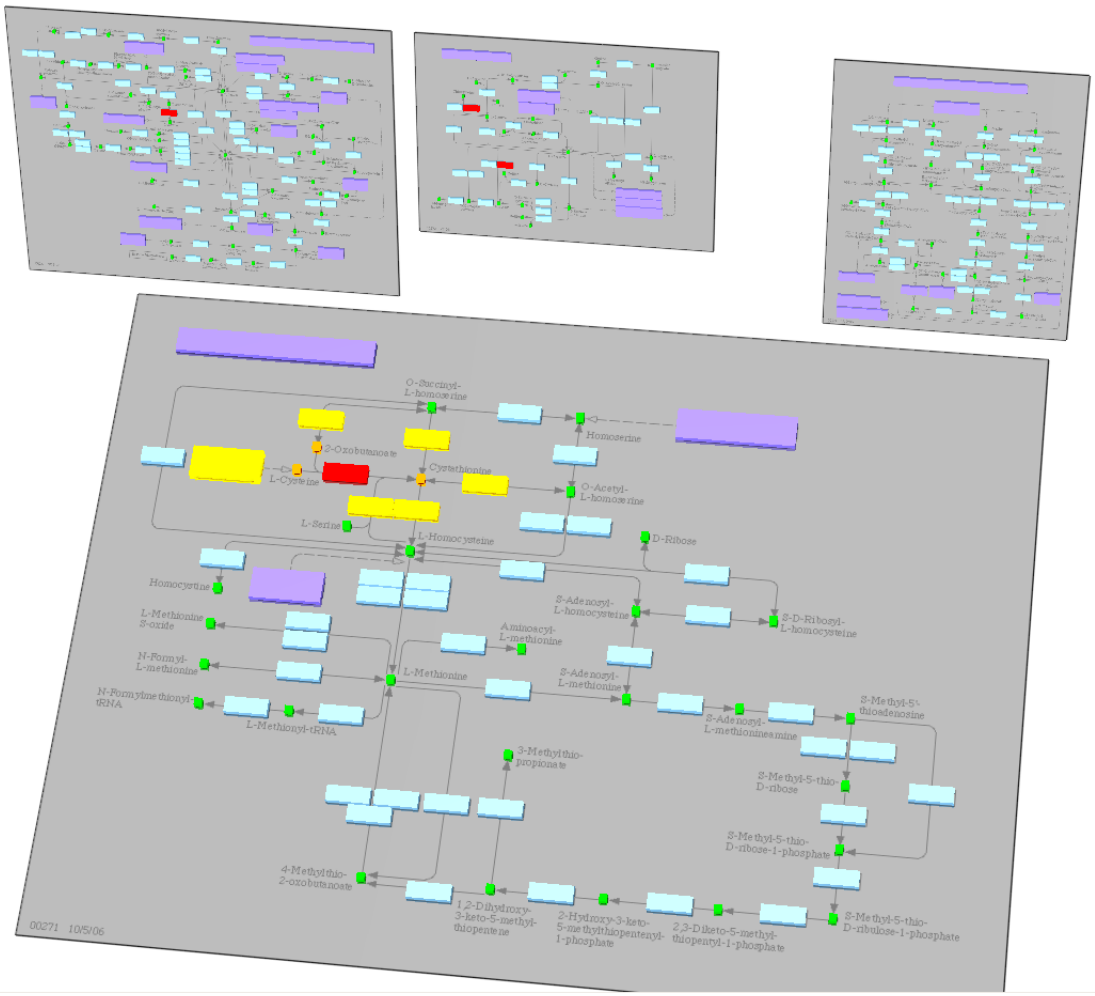
(Pathway textures taken from `http://www.genome.jp/kegg/pathway.html` *)*

5.4 Neighborhood forwarding

The synchronization between views (see section 3.2.4 on page 50) enables forwarding of the neighborhood selection to other views that contain the same pathway. Figure 5.7 shows the color coded neighboring nodes in a 3D panel setup. The 3D representation also highlights identical nodes of the selected one in related graphs.



(a) 2D pathway for *Methionine Metabolism* with a distance 2 neighborhood. The selected enzyme "4.4.1.1" is highlighted in red. The three compounds with distance 1 are highlighted in orange. Distance 2 enzymes are depicted in yellow.



(b) 3D panel view with highlighted neighborhood linked to the 2D pathway in (a). The neighborhood has been selected in the 2D pathway representation. The selected enzyme shown in red in the primary pathway (front) is also highlighted in red in the smaller pathways in the back.

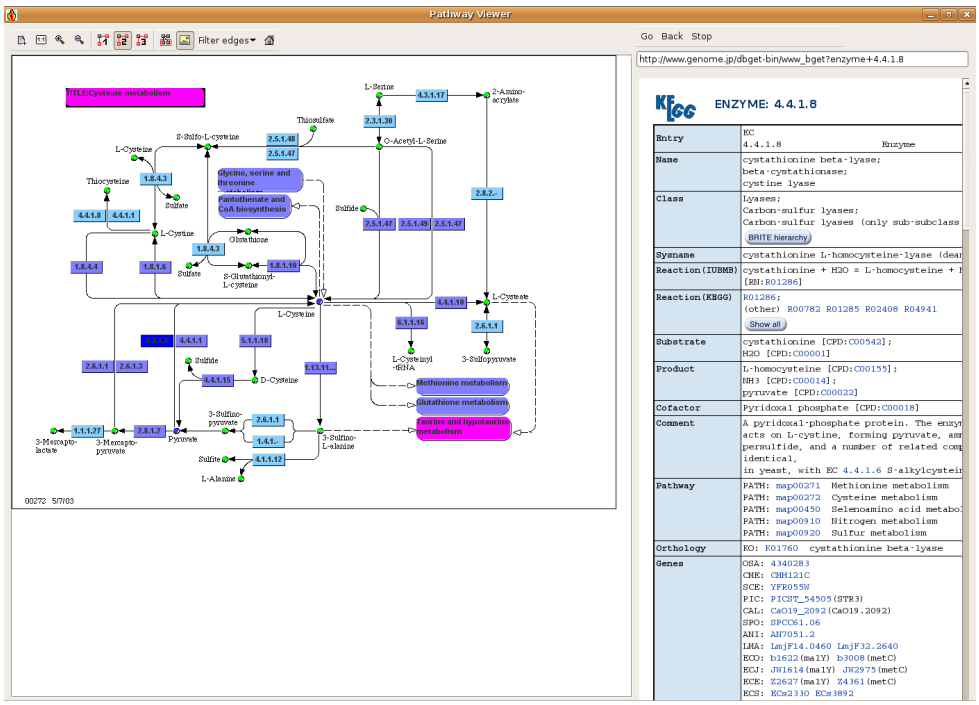
Figure 5.7: *Neighborhood synchronization example between 2D and 3D views. The selection of enzymes and compounds from the 2D view is forwarded to the 3D panel view.*
(Pathway textures taken from `http://www.genome.jp/kegg/pathway.html` *)*

5.5 Multi-monitor setup

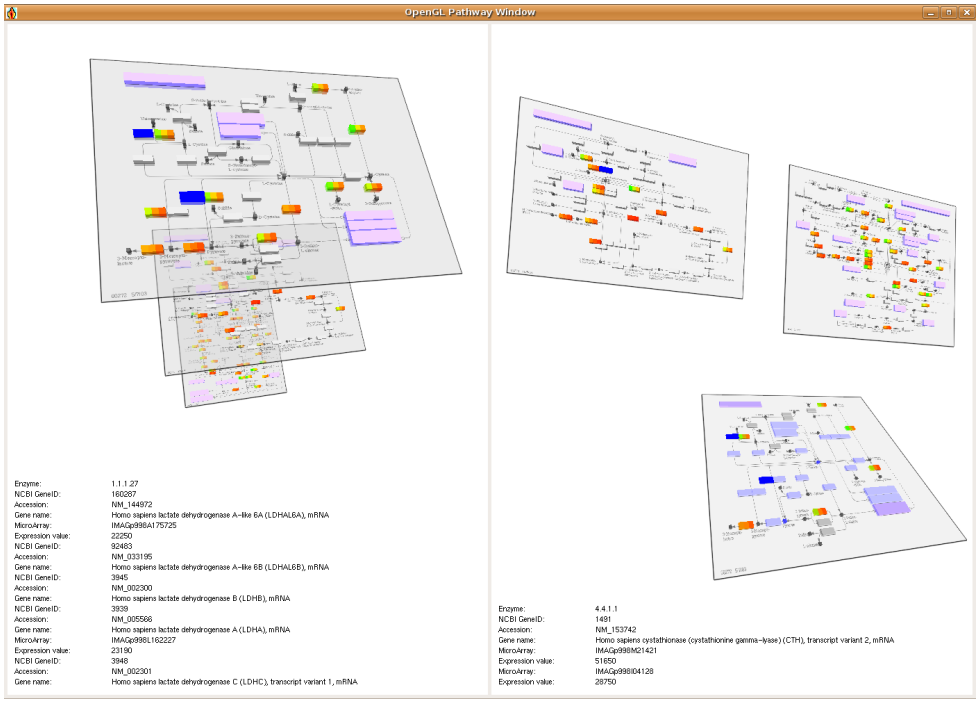
An application that employs multiple-views shows its strengths best in a multi-monitor environment. As an example we provide a setup consisting of two monitors side by side. They are connected to a dual-head graphics card. The monitors are merged to a big virtual screen. Therefore the user can drag windows from one monitor to the other. Figure 5.8 presents a picture of the multi-monitor workstation with the sample setup running.

Figure 5.8: *Multi-monitor setup with the running pathway visualization application.*

The first monitor contains the 2D interaction with the planar KEGG pathways. Additionally the information browser is integrated in a row layout. The second monitor shows the 3D layered setup and the 3D panel scene arranged in a vertically split window. All views are synchronized. Therefore a picked object in the layered view is published to the 3D panel setup and the 2D pathway view. Neighborhood groups are forwarded the same way. The information HTML browser always shows the information about the element that is currently selected by the user. Moving the mouse over an object in 3D triggers an update of the information in the browser and the text in the 3D info-area. Figure 5.9 shows the window's content of the two monitors. In the test environment a time-series gene-expression experiment is additionally mapped onto the graphs in 3D. Switching between pathways in 3D, replaces the graph at that position and automatically maps the gene data immediately onto the newly shown graph.



(a) Monitor 1 shows a 2D view with the *Cysteine Metabolism* pathway loaded in combination with a HTML browser that presents the loaded information about a selected node.



(b) Monitor 2 contains an OpenGL split view. The left side is a layered representation of three pathways. The right part depicts the same pathways in a panel setup.

Figure 5.9: *Screen content of the multi-monitor setup. The enzyme "4.4.1.8" is selected in the 2D view. The neighborhood algorithm is applied to the 2D view shown in (a) and the 3D panel view illustrated in (b). Enzymes in the neighborhood selection are colored from dark blue to light blue.*

(Pathway textures taken from `http://www.genome.jp/kegg/pathway.html` *)*

5.6 User feedback

We provided the prototype of the application to our medical partners and asked for their opinion. To support the evaluation process we prepared a list of questions. The questions mainly aimed at getting insight into which features the users appreciated and which weak points they identified in the solution. In addition we inquired for suggestions for future work. The interviews should result in a rough evaluation of the application. Extensive user studies must be carried out to validate the results from the interviews.
The users found the fast switching between the pathways very useful. Obviously this feature can significantly boost their current workflow. Also the integration of meta-information in the form of the HTML browser and the info-area in 3D appealed to the users. In special they pointed out that the visualization is clear and the display is not overloaded.
According to the feedback, the features in the pathway visualization framework fulfilled their intended purpose. This outcome is not surprising because the interviewed user group was involved in the design process of the application. Therefore an independent user group is eligible for future evaluations.
In some situations the free navigation in 3D confused the users. A possible solution to overcome this problem could be the restriction of the navigation to zooming and panning only.
The users wish for a simulation of the metabolic network. However, to perform a meaningful simulation, knowledge about the activity of metabolites in the pathways is necessary. Up to now the integration of gene-expression data gives only information about the regulation of enzymes. The data on the activity of metabolites will be available from another medical partner.

Chapter 6

Conclusion

Pathway visualization is an important field in proteomics and genomics. This thesis summarizes the current state of research from the view of information visualization. The proposed pathway visualization framework *GeneView* tries to integrate multiple metabolic graphs in an interactive environment. The user should be empowered to explore the complex metabolic network by interacting with multiple planar pathways. The framework is well adapted for the special characteristics of the cellular pathways. The solution incorporates modern visualization techniques in 2D as well as 3D. The 2D view on the network facilitates the hierarchical navigation inside cellular networks. The 3D environment enables a flexible presentation of multiple pathways at the same time. Data selections are system wide synchronized to allow an integrated examination of the pathways. During this interactive visual data mining process, the user is accompanied by an extensive support of meta-information about the elements and relations of the network.
In addition the metabolic pathway visualization can be further enhanced by the integration of gene-expression data. This additional information helps the biologist or medical doctor to gain a better understanding of the processes in the pathways. In terms of visualizing genetic data embedded in pathways, the solution presented in this thesis can already compete with *GeneSpring* (see section 2.7.1 on page 36).

Future work

Relations in pathway graphs can be investigated by the visualization of neighborhoods. The current implementation allows only the application of neighborhoods restricted to one pathway. Enzymes and compounds appear multiple times in the same graph and in related graphs. This fact could be incorporated into the neighborhood visualization. Figure 6.1 illustrates a mockup with several related pathways in a panel setup.

The selected enzyme in the pathway in the center is highlighted. Identical enzymes in foreign graphs are highlighted too. The neighborhood color coding is applied to all neighbors of the picked enzyme in all shown pathways. This approach has potential to increase the understanding of relations in the complex network.

An integration of the functional hierarchy from KEGG in 3D would further lift the visual data mining capabilities in the metabolic network to a higher level. Alternative layouting techniques of the pathways in 3D needs to be investigated.
The quality of visualization results highly depends on the data. As the KEGG database is the sole data source in the current version, the integration of other pathway databases would be desirable. Due to the fact that our proposed visualization approach utilizes the KEGG pathway layout the inclusion of other data sources would pose the problem of layouting the new data.

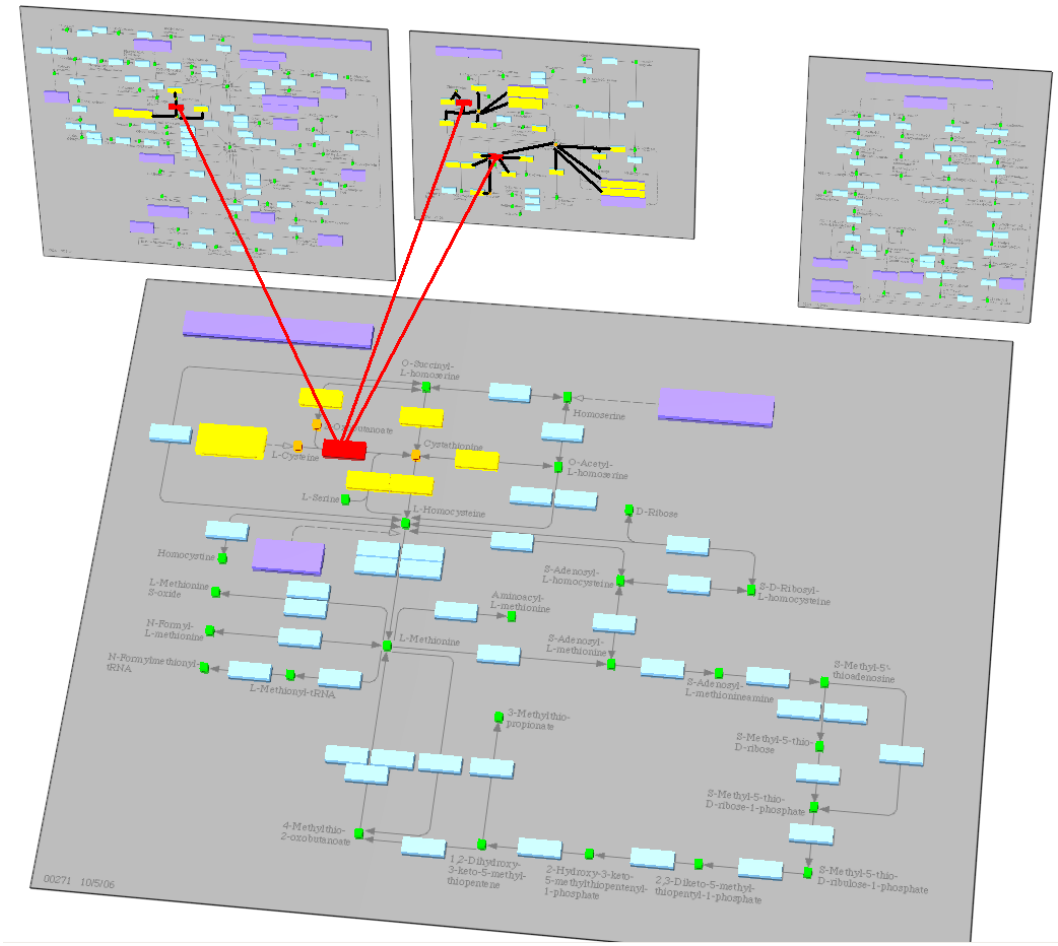


Figure 6.1: *Mockup of the neighborhood visualization in multiple pathways. The selected enzyme "4.4.1.1" in the Methionine Metabolism is highlighted system wide in red. The neighborhood is applied to all pathways in that the enzyme occurs. Connection lines between the selected enzyme and its representatives in related pathways support the visual impression.*

(Pathway textures taken from `http://www.genome.jp/kegg/pathway.html` *)*

Due to the diversity of the databases, the merging process is not trivial. The information is stored in different formats and using different annotations.

The resulting data of a gene-expression experiment are several thousands of genes. For displaying the gene data of numerous experiments the heatmap representation is widely used. However, heatmaps occupy a huge screen space. One popular approach to address this problem is the gene clustering in the heatmap. The process of clustering groups data and therefore discards or at least hides single values. This consideration leads to the conclusion that a reasonable filtering of the gene dataset would help. The mapping of genes to enzymes in the pathway is a qualified procedure. The investigation of the cellular processes which are modeled in the pathways has potential to filter and consequently reduce the amount of genes currently visualized.

This procedure is illustrated in figure 6.2. The data reduction applied on the heatmap would result in a significant heatmap representation. Since the resulting heatmap is specialized to the examined task, this approach could speed up the gene analyses.

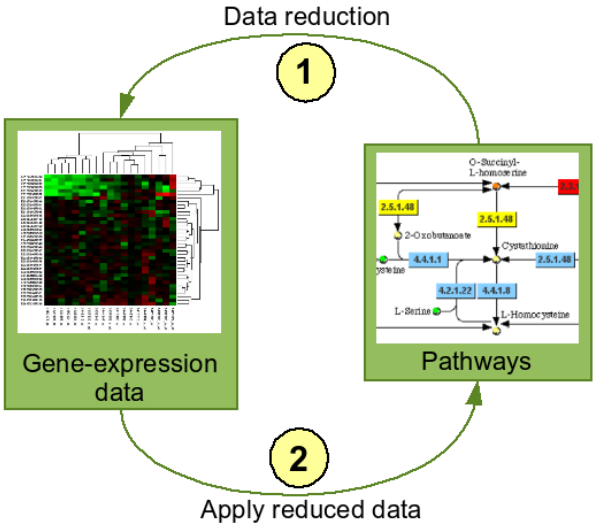


Figure 6.2: *Data reduction cycle.*

List of Figures

Bibliography

[Ahlberg1995] Christopher Ahlberg and Erik Wistrand. *Ivee: An information visualization and exploration environment.* In *INFOVIS '95: Proceedings on Information Visualization*, pages 66–73. IEEE Computer Society, Washington, DC, USA, **1995**.

[Alibes2007] Andreu Alibes, Patricio Yankilevich, Andres Canada, and Ramon D. Uriarte. *Idconverter and idclight: Conversion and annotation of gene and protein ids. BMC Bioinformatics*, 8(1):page 9, **2007**.

[Bairoch2000] Amos Bairoch. *The enzyme database in 2000. Nucleic Acids Research*, 28(1):pages 304–305, **2000**.

[Baldonado2000] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. *Guidelines for using multiple views in information visualization.* In *AVI '00: Proceedings on Advanced visual interfaces*, pages 110–119. ACM Press, New York, NY, USA, **2000**.

[Becker2001] Moritz Becker and Isabel Rojas. *A graph layout algorithm for drawing metabolic pathways. Bioinformatics*, 17(5):pages 461–467, **2001**.

[Bourqui2006] Romain Bourqui, David Auber, Vincent Lacroix, and Fabien Jourdan. *Metabolic network visualization using constraint planar graph drawing algorithm.* In *IV '06: Proceedings on Information Visualization*, pages 489–496. IEEE Computer Society, Washington, DC, USA, **2006**.

[Brandes2004] Ulrik Brandes, Tim Dwyer, and Falk Schreiber. *Visualizing related metabolic pathways in two and a half dimensions.* In Giuseppe Liotta (editor), *Graph Drawing, Perugia, 2003*, pages 111–122. Springer, **2004**.

[Burdea2003] Grigore C. Burdea and Philippe Coiffet. *Virtual Reality Technology.* John Wiley & Sons, Inc., New York, NY, USA, **2003**. ISBN 0471360899.

[Card1999] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman (editors). *Readings in information visualization: using vision to think.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, **1999**. ISBN 1-55860-533-9.

[Cavalieri2005] Duccio Cavalieri and Carlotta De Filippo. *Bioinformatic methods for integrating whole-genome expression results into cellular networks. Drug Discov Today*, 10(10):pages 727–734, **2005**.

[Cormen2001] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction To Algorithms, Second Edition.* MIT Press, Cambridge, MA, USA, **2001**. ISBN 0-262-03293-7.

[Craft2005] Brock Craft and Paul Cairns. *Beyond guidelines: What can we learn from the visual information seeking mantra?* In *IV '05: Proceedings on Information Visualisation*, pages 110–118. IEEE Computer Society, Washington, DC, USA, **2005**.

[Cruz-Neira1993] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. *Surround-screen projection-based virtual reality: the design and implementation of the cave.* In *SIGGRAPH '93: Proceedings on Computer graphics and interactive techniques*, pages 135–142. ACM Press, New York, NY, USA, **1993**.

[Dahlquist2002] Kam D. Dahlquist, Nathan Salomonis, Karen Vranizan, Steven C. Lawlor, and Bruce R. Conklin. *Genmapp, a new tool for viewing and analyzing microarray data on biological pathways. Nature Genetics*, 31(1):pages 19–20, **2002**.

[Demir2002] E. Demir, O. Babur, U. Dogrusoz, A. Gursoy, G. Nisanci, R. Cetin-Atalay, and M. Ozturk. *Patika: an integrated visual environment for collaborative construction and analysis of cellular pathways. Bioinformatics*, 18(7):pages 996–1003, **2002**.

[Dickerson2003] Blom K Reinot A Lie J Cruz-Neira C Wurtele ES Dickerson JA, Yang Y. *Using virtual reality to understand complex metabolic networks.* In *Atlantic Symposium on Computational Biology and Genomic Information Systems and Technology.* **2003**.

[Dogrusoz2006] U. Dogrusoz, E. Z. Erson, E. Giral, E. Demir, O. Babur, A. Cetintas, and R. Colak. *Patikaweb: a web interface for analyzing biological pathways through advanced querying and visualization. Bioinformatics*, 22(3):pages 374–375, **2006**. ISSN 1367-4803.

[Dogrusoz2004] Ugur Dogrusoz, Erhan Giral, Ahmet Cetintas, Ali Civril, and Emek Demir. *A compound graph layout algorithm for biological pathways.* In János Pach (editor), *Graph Drawing, New York, 2004*, pages 442–447. Springer, **2004**.

[Doleisch2002] Helmut Doleisch and Helwig Hauser. *Smooth brushing for fo-*

cus+context visualization of simulation data in 3d. In *WSCG*, pages 147–154. **2002**.

[Eisen1998] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. *Cluster analysis and display of genome-wide expression patterns. Proc. Nat'l Academy of Science USA*, 95(25):pages 14863–14868, **1998**. ISSN 0027-8424.

[Eisenberg2006] David Eisenberg, Edward Marcotte, Andrew McLachlan, and Matteo Pellegrini. *Bioinformatic challenges for the next decade(s). Philosophical transactions-Royal Society of London*, 361:pages 525–527, **2006**.

[Furnas1986] George W. Furnas. *Generalized fisheye views.* In *CHI '86: Proceedings on Human factors in computing systems*, pages 16–23. ACM Press, New York, NY, USA, **1986**.

[Furnas1995] George W. Furnas and Benjamin B. Bederson. *Space-scale diagrams: understanding multiscale interfaces.* In *CHI '95: Proceedings on Human factors in computing systems*, pages 234–241. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, **1995**.

[Gamma1995] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, **1995**. ISBN 0-201-63361-2.

[Harris2004] Bobby Harris, Rob Warner, and Robert Harris. *The Definitive Guide to Swt and Jface.* APress, **2004**. ISBN 1590593251.

[Hauser2004] Helwig Hauser and Robert Kosara. *Interactive analysis of high-dimensional data using visualization.* In *Workshop on Robustness for High-dimensional Data (RobHD 2004)*. Vorau, Austria, **2004**.

[Hauser2002] Helwig Hauser, Florian Ledermann, and Helmut Doleisch. *Angular brushing of extended parallel coordinates.* In *INFOVIS '02: Proceedings on Information Visualization*, pages 127–130. IEEE Computer Society, Washington, DC, USA, **2002**.

[Heer2006] Jeffrey Heer and Maneesh Agrawala. *Software design patterns for information visualization. IEEE Transactions on Visualization and Computer Graphics*, 12(5):pages 853–860, **2006**. Student Member-Jeffrey Heer.

[Hucka2003] Michael Hucka, Andrew Finney, Herbert M. Sauro, H. Bolouri, John C. Doyle, Hiroaki Kitano, Adam P. Arkin, Benjamin J. Bornstein, D. Bray, A. Cornish-Bowden, et al. *The systems biology*

markup language (sbml): a medium for representation and exchange of biochemical network models. Bioinformatics, 19(4):pages 524–531, **2003**.

[Jacobson1994] Allan S. Jacobson, Andrew L. Berkin, and Martin N. Orton. *Linkwinds: interactive scientific data analysis and visualization. Commun. ACM*, 37(4):pages 42–52, **1994**. ISSN 0001-0782.

[Jourdan2004] Fabien Jourdan. *Information Visualization : graphical representations, structural indices, visual cues and navigation. Application to biological and social networks.* Ph.D. thesis, **2004**.

[Jourdan2003] Fabien Jourdan and Guy Melancon. *Tool for metabolic and regulatory pathways visual analysis.* volume 5009, pages 46–55. **2003**.

[Kalkusch2005] Michael Kalkusch. *Cash Flow - A Visualization Framework for 3D Flow Data.* Master's thesis, Vienna University of Technology, **2005**.

[Kalkusch2006] Michael Kalkusch and Dieter Schmalstieg. *Extending the scene graph with a dataflow visualization system.* In *VRST '06: Proceedings on Virtual reality software and technology*, pages 252–260. ACM Press, New York, NY, USA, **2006**.

[Kanehisa2006] Minoru Kanehisa, Susumu Goto, Masahiro Hattori, Kiyoko F. Aoki-Kinoshita, Masumi Itoh, Shuichi Kawashima, Toshiaki Katayama, Michihiro Araki, and Mika Hirakawa. *From genomics to chemical genomics: new developments in kegg. Nucleic Acids Res*, 34(Database issue):pages 354–357, **2006**. ISSN 1362-4962.

[Kantor2001] Paul Kantor. *Foundations of statistical natural language processing. Inf. Retr.*, 4(1):pages 80–81, **2001**. ISSN 1386-4564.

[Karp1994] Peter D. Karp and Suzanne M. Paley. *Automated drawing of metabolic pathways.* In H. Lim, C. Cantor, and R. Robbins (editors), *Proceedings on Bioinformatics and Genome Research.* **1994**.

[Karp1994a] Peter D. Karp and Suzanne M. Paley. *Representations of metabolic knowledge: Pathways.* In *In ISMB'94: Proceedings on Intelligent Systems for Molecular Biology*, pages 203–211. **1994**.

[Karp1993] Peter D. Karp and Monica Riley. *Representations of metabolic knowledge.* In *In ISMB'93: Proceedings on Intelligent Systems for Molecular Biology*, pages 207–215. AAAI Press, **1993**. ISBN 0-929280-47-4.

[Kobourov2005] Stephen G. Kobourov and Kevin Wampler. *Non-euclidean spring em-*

bedders. IEEE Transactions on Visualization and Computer Graphics, 11(6):pages 757–767, **2005**. ISSN 1077-2626.

[Kosara2003] Robert Kosara, Helwig Hauser, and Donna L. Gresh. *An interaction view on information visualization.* In *State-of-the-Art Proceedings of EUROGRAPHICS 2003 (EG 2003)*, pages 123–137. **2003**.

[Krasner1988] Glenn E. Krasner and Stephen T. Pope. *A description of the model-view-controller user interface paradigm in the smalltalk-80 system. Journal of Object Oriented Programming*, 1(3):pages 26–49, **1988**.

[Lindroos2002] Hillevi Lindroos and Siv G. E. Andersson. *Visualizing metabolic pathways: comparative genomics and expression analysis.* In *Proceedings of the IEEE*, volume 90, pages 1793–1802. **2002**.

[Mackinlay1991] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. *The perspective wall: detail and context smoothly integrated.* In *CHI '91: Proceedings on Human factors in computing systems*, pages 173–176. ACM Press, New York, NY, USA, **1991**.

[Martin1995] Allen R. Martin and Matthew O. Ward. *High dimensional brushing for interactive exploration of multivariate data.* In *VIS '95: Proceedings on Visualization '95*, page 271. IEEE Computer Society, Washington, DC, USA, **1995**.

[Michal1999] Gerhard Michal. *Biochemical Pathways. Biochemie-Atlas.* Spektrum Akademischer Verlag, **1999**.

[Mlecnik2005] Bernhard Mlecnik, Marcel Scheideler, Hubert Hackl, Juergen Hartler, Fatima Sanchez-Cabo, and Zlatko Trajanoski. *Pathwayexplorer: web service for visualizing high-throughput expression data on biological pathways. Nucleic Acids Research*, 33(Web Server issue):pages 633–637, **2005**. ISSN 1362-4962.

[Niemeyer2002] Pat Niemeyer and Jonathan Knudsen. *Learning Java.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, **2002**. ISBN 0596002858.

[Nikitin2003] Alexander Nikitin, Sergei Egorov, Nikolai Daraselia, and Ilya Mazo. *Pathway studio - the analysis and navigation of molecular networks. Bioinformatics*, 19(16):pages 2155–2157, **2003**.

[North2000] Chris North and Ben Shneiderman. *Snap-together visualization: can users construct and operate coordinated visualizations? Int. J. Hum.-Comput. Stud.*, 53(5):pages 715–739, **2000**. ISSN 1071-5819.

[Novichkova2003] Svetlana Novichkova, Sergei Egorov, and Nikolai Daraselia. *Medscan,*

a natural language processing engine for medline abstracts. Bioinformatics, 19(13):pages 1699–1706, **2003**. ISSN 1367-4803.

[Kanehisa2000] Hiroyuki Ogata, Susumu Goto, Kazushige Sato, Wataru Fujibuchi, Hidemasa Bono, and Minoru Kanehisa. *Kegg: Kyoto encyclopedia of genes and genomes. Nucleic Acids Research*, 28(1):pages 27–30, **2000**. ISSN 0305-1048.

[Pellegrini2001] Matteo Pellegrini, Michael Thompson, Joseph Fierro, and Peter Bowers. *Computational method to assign microbial genes to pathways. J. Cell. Biochem. Suppl.*, 37:pages 106–109, **2001**.

[Pesce1995] Mark Pesce. *VRML Browsing and Building Cyberspace: Browsing and Building Cyberspace*. New Riders Publishing, Thousand Oaks, CA, USA, **1995**. ISBN 1562054988.

[Roberts2000] Jonathan C. Roberts. *Multiple-view and multiform visualization.* In Robert Erbacher, Alex Pang, Craig Wittenbrink, and Jonathan Roberts (editors), *Visual Data Exploration and Analysis VII, Proceedings of SPIE*, volume 3960, pages 176–185. IS&T and SPIE, **2000**.

[Rojdestvenski2003] Igor Rojdestvenski. *Metabolic pathways in three dimensions. Bioinformatics*, 19(18):pages 2436–2441, **2003**.

[Rojdestvenski2002] Igor Rojdestvenski and Michael Cottam. *Visualizing metabolic networks in vrml.* In *In IV'02: Proceedings on Information Visualisation*, volume 00, page 175. IEEE Computer Society, Los Alamitos, CA, USA, **2002**. ISSN 1093-9547.

[Saraiya2005] Purvi Saraiya, Chris North, and Karen Duca. *Visualizing biological pathways: requirements analysis, systems evaluation and research agenda. Information Visualization*, 4(3):pages 191–205, **2005**. ISSN 1473-8716.

[Schena1995] Mark Schena, Dari Shalon, Ronald W. Davis, and Patrick O. Brown. *Quantitative monitoring of gene expression patterns with a complementary dna microarray. Science*, 270(5235):pages 467–470, **1995**. ISSN 0036-8075.

[Seo2002] Jinwook Seo and Ben Shneiderman. *Interactively exploring hierarchical clustering results. Computer*, 35(7):pages 80–86, **2002**. ISSN 0018-9162.

[Shneiderman1996] Ben Shneiderman. *The eyes have it: A task by data type taxonomy*

for information visualizations. In *VL '96: Proceedings on Visual Languages*. IEEE Computer Society, **1996**. ISBN 081867508X.

[Shreiner2005] Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis. *OpenGL Programming Guide Fifth Edition: The Official Guide to Learning OpenGL, Version 2.* Addison-Wesley Professional, 5 edition, **2005**. ISBN 978-0321335739.

[Spence2007] Robert Spence. *Information Visualization: Design for Interaction (2nd Edition).* Prentice-Hall, Inc., **2007**. ISBN 978-0-132-0655-04.

[Tang2004] Diane Tang, Chris Stolte, and Robert Bosch. *Design choices when architecting visualizations. Information Visualization*, 3(2):pages 65–79, **2004**. ISSN 1473-8716.

[Toyoda2003] Tetsuro Toyoda, Yoshiki Mochizuki, and Akihiko Konagaya. *Gscope: a clipped fisheye viewer effective for highly complicated biomolecular network graphs. Bioinformatics*, 19(3):pages 437–438, **2003**.

[Wijk2003] Jarke J. van Wijk and Wim A. A. Nuij. *Smooth and efficient zooming and panning. INFOVIS*, 00:pages 15– 23, **2003**.

[Wagner2007] Daniel Wagner and Dieter Schmalstieg. *Muddleware for prototyping mixed reality multiuser games.* In *VR'07: Proceedings on Virtual Reality*, pages 235–238. IEEE, **2007**.

[Waldner2007] Manuela Waldner, Michael Kalkusch, and Dieter Schmalstieg. *Optical magic lenses and polarization-based interaction techniques.* In *IPT-EGVE 2007 (In Press).* IPT-EGVE 2007 symposium, **2007**.

[Wolf2000] Detlef Wolf, Christopher P. Gray, and Antoine de Saizieu. *Visualising gene expression in its metabolic context. Briefings in Bioinformatics*, 1(3):pages 297–304, **2000**.

[Wolff2005] David Wolff. *Using opengl in java with jogl. J. Comput. Small Coll.*, 21(1):pages 223–224, **2005**.

[Xu2005] Zhigen Xu, Yusong Yan, and Jim X. Chen. *Opengl programming in java. Computing in Science and Engg.*, 7(1):pages 51–55, **2005**. ISSN 1521-9615.

[Yang2005] Yuting Yang, Levent Engin, Eve Syrkin Wurtele, Carolina Cruz-Neira, and Julie A. Dickerson. *Integration of metabolic networks and gene expression in virtual reality. Bioinformatics*, 21(18):pages 3645–3650, **2005**. ISSN 1367-4803.

[Yang2006] Yuting Yang, Eve Syrkin Wurtele, Carolina Cruz-Neira, and Julie A. Dickerson. *Hierarchical visualization of metabolic networks using virtual reality*. In *VRCIA '06: Proceedings on Virtual reality continuum and its applications*, pages 377–381. ACM Press, New York, NY, USA, **2006**. ISBN 1-59593-324-7.

List of abbreviations

ADP	Adenosine Diphosphate
API	Application Programming Interface
ATP	Adenosine Triphosphate
CAVE	Cave Automatic Virtual Environment
CT	Computed Tomography
DNA	Deoxyribonucleic Acid
FTP	File Transfer Protocol
GO	Gene Ontology
GUI	Graphical User Interface
HTML	Hypertext Markup Language
JOGL	Java OpenGL Library
KEGG	Kyoto Encyclopedia of Genes and Genomes
KGML	KEGG Markup Language
MRI	Magnetic Resonance Imaging
MVC	Model-View Controller
NLP	Natural Language Processing
PCD	Programmed Cell Death
PNG	Portable Network Graphics
RAM	Random Access Memory
SBML	Systems Biology Markup Language
SOAP	Simple Object Access Protocol
SWT	Standard Widget Toolkit
UML	Unified Modeling Language
URL	Uniform Resource Locator
VR	Virtual Reality
XML	Extensible Markup Language