Institute for Computer Graphics and Vision
Graz University of Technology
Graz

Visual Analytics for Gene Expression Data

Master's Thesis

Bernhard Walter Schlegl, BSc
bernhard.schlegl@student.tugraz.at

December 2009

Supervisor:
Univ.-Prof. Dipl.-Ing. Dr. techn. Dieter Schmalstieg
Advisor:
Dipl.-Ing. Bakk. Marc Streit
Dipl.-Ing. Bakk. Alexander Lex

Abstract

The analysis of biomolecular gene expression data sets is a research area which results are used by a wide range of life science experts. Biologists and geneticists are only two sample users who are interested in analyzing gene expression profiles. To support users in terms of visualization, the Caleydo InfoVis framework provides several visualization techniques for gene expression data and pathways. The combination of both data sources is a major field of interest because it provides better insights into the biological processes occurring inside a cell into the context of patients.

To help experts extract information from the data visual analytics is sometimes used. In this thesis we apply data mining methods and visualize the results. We set a focus on clustering algorithms because they are well suited for finding co-expressed genes. Co-expressed genes are relevant because experts assume that they are responsible for similar functions inside a cell.

Keywords: Heat Map, dendrogram, clustering, gene expression, visual analytics, Caleydo

Zusammenfassung

Die Analyse von biomolekularen Genexpressions Datensätzen ist ein Forschunsgebiet, dass von einer großen Anzahl von Experten angewandt wird. Unter anderem sind Biologen und Genetiker an der Analyse von Genexpressions Profilen interessiert. Zur Unterstützung im Bereich der Visualisierung stellt das Caleydo InfoVis Framework mehrere verschiedene Darstellungsmöglichkeiten für Genexpressions Daten und Stoffwechselzyklen zur Verfügung. Vor allem die Kombination aus diesen beiden Datenquellen ist ein wichtiger Interessensbereich, da es einen besseren Einblick in biologische Prozesse innerhalb einer Zelle erlaubt.

Um die Möglichkeiten für Benutzer so viel wie möglich Information aus einem gegebenen Datensatz zu extrahieren zu verbessern, wird der Ansatz der visuellen Analyse oft angewendet. Zu diesem Zwecke werden wir in dieser Diplomarbeit Data Mining Methoden auf Genexpressions Datensätze anwenden und die resultierenden Information visualisieren. Der Fokus der Arbeit wird auf der Integration von Cluster Algorithmen liegen, da diese sehr gute Ergebnisse bei der Suche nach ähnlich regulierten Genen liefern. Ähnlich regulierte Gene sind relevant, da Experten annehmen, dass solche Gene für ähnliche Funktionen innerhalb einer Zelle verantwortlich sind.

Schlüsselwörter: Heat map, Dendrogram, Clustering, Genexpression, Visuelle Analyse, Caleydo

Pledge

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 danke ich meinen Eltern, Cäcilia und Walter Schlegl für das Ermöglichen meiner Ausbildung. Ebenfalls danke ich meiner ganzen Familie für die langjährige Unterstützung in allen Lebenslagen.

Weiters möchte ich mich ganz herzlich bei meinen Betreuern bedanken. Dieter Schmalstieg möchte ich für die Möglichkeit danken, die Diplomarbeit im Rahmen eines interessanten Projekts zu erarbeiten. Marc Streit und Alexander Lex danke ich für die durchgehend konstruktive Zusammenarbeit in den letzten Monaten, und dass sie immer Zeit für meine Fragen gefunden haben und mir mit Rat und Tat zur Seite standen.

Danke auch an meine liebe Freundin Barbara und an alle meine Freunde (Martin, Flo, Christian, Gernot und wie sie alle heißen) mit denen ich immer etwas unternehmen konnte und die immer Zeit für mich gefunden haben.

Contents

Chapter 1

Introduction

The exploration of biological processes in different life forms is a challenging research field for a variety of scientists. Among others, biologists, genetics, and physicians are interested in understanding the behavior and the function of distinct genes in an organism as well as their role in diseases. Two basic approaches became popular in the last three decades. The first one is the survey of pathways, which illustrate biological processes in a cell. The second one is the exploration of gene expression data sets. The exploration of gene expression profiles in combination with pathways exploration is an enormously important feature because the behavior of cells is influenced by their gene expression profile.

Due to the development of microarray DNA chips the expression profiles of thousands of genes can be analyzed in parallel, which generates a huge amount of data. The Caleydo[1] framework supports viewing pathways and gene expression data with several information visualization methods, which supports experts in investigating the data sets available.

To extend the currently available framework we want to integrate a *visual analytics* [Thomas2005] approach. Visual analytics is a research field which may be used to handle almost any kind of data source.

"The basic idea of visual analytics is to visually represent information, allowing the human to directly interact with it, to gain insight, to draw conclusions, and to ultimately make better decisions" [Keim2006a].

Further, Keim et al. [Keim2006a] stated that visual analytic can be seen as an integrated approach and as an interdisciplinary field as illustrated in Figure 1.1.

1.1 Problem Statement and Contribution

Caleydo is a visualization framework which focuses on the analysis of biological data. Caleydo visualizes pathways and gene expression data sets. The gene expression data is visualized with parallel coordinates and heat maps, which are visualization techniques suitable for tabular data.

The handling of group structures is essential in terms of providing any kind of clustering or user-driven group formation. The implementation of structures to handle, store, and manipulate groups/clusters is the first task which was implemented in the course of the

[1] www.caleydo.org

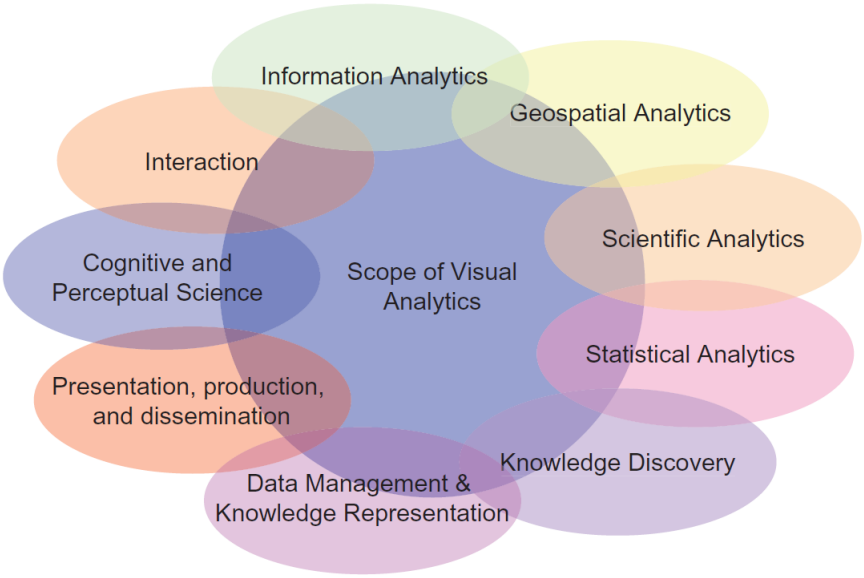


Figure 1.1: *Visual analytics. [Keim2006a].*

thesis. Next, to automatically assign elements to groups, the integration of different kinds of cluster algorithms and distance measurements is necessary. Two different types of algorithms are possible: hierarchical and partition-based clustering. While the former builds a tree structure which contains all elements according to their similarity the latter builds distinct clusters and assigns each element to a specific cluster.

Finally the clustering results and the possibilities of user interactions with the results were integrated in the existing heat map view. Therefore two different approaches, one for partition-based and one for hierarchically clustered data, are needed.

In combination, this provides a visual approach supporting the user when investigating large data sets. The visual approach in combination with user interaction methods may advance the human understanding of complex biological processes. Such an method allows users who are not well-trained in statistics to draw conclusions from their data.

1.2 Biological Background

All living things (beginning with viruses and up to human beings) are built of cells. A large part of these individuals are unicellular (built from only one cell), but all share the same structure of cell building, and are therefore, in some sense comparable. Each cell contains all the (hereditary) information [Alberts2002] required to construct a new cell also containing the same information. All the information needed for building a cell and for regulating processes is coded in Deoxyribonucleic acid (DNA) which is built up with simple subunits, called nucleotids [Alberts2002]. Nucleotids consist of two sub parts: a sugar-phosphate molecul and a base attached to it. The base may either be adenine (A), guanine (G), cytosine (C), or thymine (T). A simplified illustration of the hereditary

process is shown in figure 1.2.

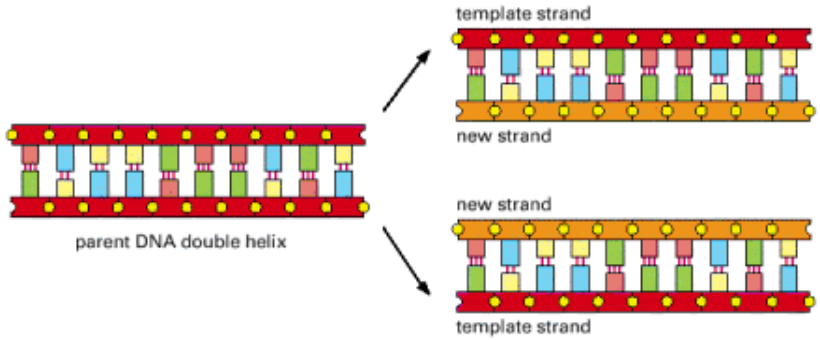


Figure 1.2: *DNA replication. The double stranded DNA is pulled apart, and each strand serves as a template for synthesis of a new complementary strand [Alberts2002].*

As mentioned above, genes code all the information to build up cells and, further, whole creatures. An important feature of genes is their expression ratio, which influences the behavior of a biological process inside cells. This expression ratio allows inference over the function of a gene inside a cell. Therefore the gene expression ratio is a very important feature for biologists, virologists, and genetics.

The measurement of gene expression ratios is accomplished with the help of microarray chips (see Figure 1.3). DNA microarrays are glass slides spotted with DNA fragments [Watson2008] which allow the measurement of tens of thousands of gene expression values in parallel. Each of these fragments correspond to one gene. To determine the expression value for a specific cell sample one has to bring the DNA of the cell onto the microarray. Due to a biochemical process the DNA strands of the cell to be analyzed bind to the corresponding spots on the array. A fluorochrome label, which indicates the expression of a special gene, can be scanned after washing the microarray [Alberts2002]. The most commonly used color scheme uses red for up regulated genes, green for down regulated genes, and black for normally regulated genes.

1.2.1 Gene Expression Data

Gene expression ratios allow conclusions to be drawn about the function of genes. The rationale behind this idea is that genes with similar expression ratios may have the same or similar tasks inside a cell. Therefore we want to group genes corresponding to their expression values by clustering.

Gene expression data may be comprised of big text-files with thousands of samples. Because of this a user without the aid of a computer is not able to draw any useful conclusions from a given data set.

Alberts et al. [Alberts2002] state that coexpressed genes may "work in concert in the cell", which means that genes with similar expression values may encode proteins serving the same functions inside a cell. Therefore [Alberts2002] proposed cluster analysis as a good method for gathering information about coregulated genes. Figure 1.4 shows a heat

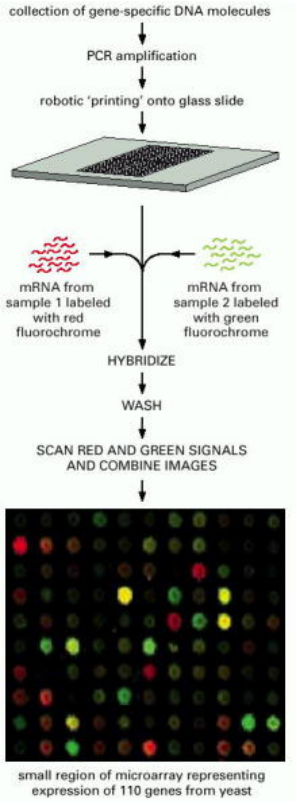


Figure 1.3: *A DNA microarray simultaneously monitors the expression of thousands of genes [Alberts2002]*

map as used for cluster analysis. Cluster analysis will be described in detail in Section 2.1.

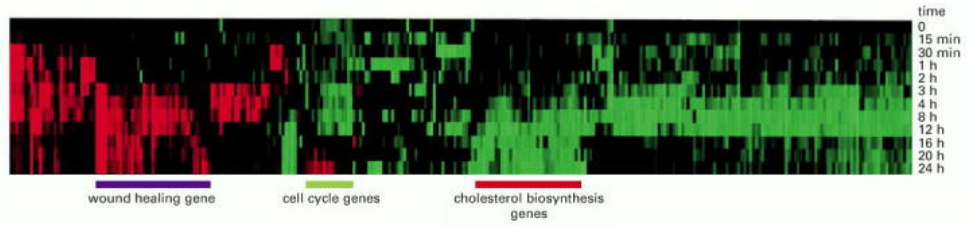


Figure 1.4: *Cluster analysis of gene expression data [Alberts2002]*

1.2.2 Pathways

Pathways illustrate chemical reactions (metabolic pathways) and signal input/output (regulatory pathways) inside a cell. KEGG[2], one common pathway database, focuses on the visualization of metabolic pathways because they can be modelled as reference pathways. Such reference pathways are well conserved among most organisms from "mammals to

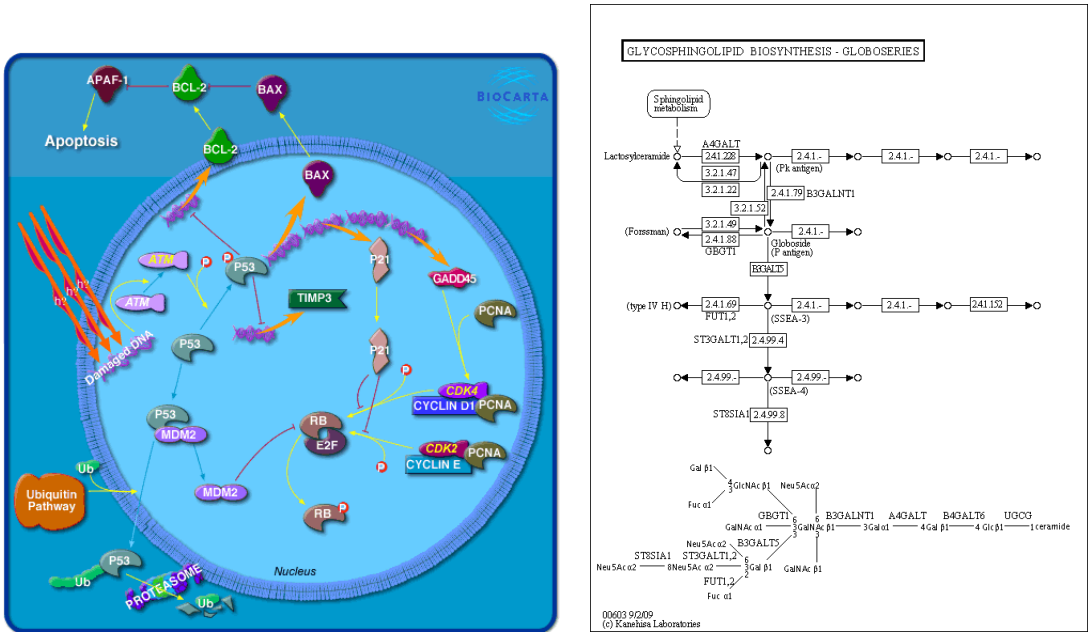
[2]http://www.genome.jp/kegg/pathway.html

bacteria" [Kanehisa2000]. Due to this it is possible to retrieve specific pathways for many organisms. This is not the case with regulatory pathways. This group of pathways diverges strongly with each kind of organism [Kanehisa2000].

All the biochemical reactions (metabolic pathways) in a cell are interconnected and may be visualized in a metabolic network [Lindroos2002]. Hence, because the complete network of a single cell is an enormously big structure, the exploration of single metabolic pathways and their connected neighbours is seen as the best approach.

There are several approaches for building online data bases to handle and to provide pathways. Two widely known online databases are KEGG[3] and BioCarta[4].

Two examples for pathway visualization can be seen in Figure 1.5.



(a) Biocarta Pathway (b) Kegg Pathway

Figure 1.5: *BioCarta and Kegg Pathway. (a) shows the Biocarta "p53 Signaling Pathway", while (b) illustrates the Kegg pathway "Glycosphingolipid biosynthesis" (available at* `http://www.biocarta.com/` *resp.* `http://www.genome.jp/kegg/pathway/`*)*

1.3 Structure of this Document

Section 2 will be an overview of related work corresponding to cluster algorithms, gene expression visualization, and visual analytics. Section 3 will provide information about the concept behind the implementation of this thesis and Section 4 will focus on the software design and the implementation. The results of this work are discussed in Section 5. Section 6 concludes the thesis and summarizes potential future work.

[3] http://www.genome.jp/kegg/pathway.html
[4] http://www.biocarta.com/genes/index.asp

Chapter 2

Related Work

This work builds upon two fields of computer science. On the one hand, data mining and, on the other hand, information visualization and human computer interaction. The first is treated in terms of clustering algorithms and will be discussed in Section 2.1. Section 2.2 describes important issues in terms of information visualization. Finally, Section 2.3 provides information about comparable information visualization and data mining frameworks. Because Caleydo focuses on the visualization of biological and clinical data, especially gene expression profiles and pathways, this work focuses on these data sources as well.

2.1 Data Mining & Clustering

Data mining is a process [Hsu2006] consisting of four distinct phases: data collection, data preprocessing, data mining, and information interpretation.

Data collection is the first step in a data mining process. It describes the acquiring of raw information. Each problem needs a distinct approach to gain information without making errors (e.g. measurement errors).

Data preprocessing has to be done after the collection of data samples. This step includes methods like the removal of erroneos data [Hsu2006], transformation, normalization, and the representation of the raw data in a manner such that it may be handled in the next processing steps. Due the nature of gaining gene expression data from microarray chips the usage of normalization methods is an essential technique. [Sherlock2001] lists several normalization methods especially feasible for gene expression data.

Data mining is the core phase of the whole process. Here different algorithms and techniques may be used to form a model for the data [Hsu2006].

Information interpretation is the last phase of the data mining process. Information interpretation is, for the most part, a challenge for human beings. However, with the support of visualization techniques provided by a InfoVis framework, the user should get an idea of trends and facts from the data set.

In this thesis we assume the first two phases as given and focus on the third and fourth. The data mining technique used in this work will be restricted to clustering algorithms. The information interpretation phase will be described later, in Section 2.2.

Clustering is a common data mining technique where data sets are divided into subsets. Each subset is a group of data samples with common features. The clusters/groups consist of objects which are similar to one another. Data samples belonging to different clusters

are dissimilar [Berkhin2002].

The data samples have to be given in vector form, where each entry in the vector represents one dimension (feature) of the sample. There are two major types of clustering algorithm: partition-based (see Section 2.1.2) and hierarchical clustering (see Section 2.1.3).

Supervised Clustering

As described in [Eisen1998], supervised cluster algorithms determine rules to assign data samples to groups according to a given reference vector. Therefore supervised clustering is closely related to classification.

After creating rules for a given data set, the algorithms are able to cluster another data set, with very similar features, satisfactory. But if the data set to be clustered is not very similar to the training data set the cluster result will not be good. Because a priori knowledge is commonly lacking these algorithms are seldomly used in practical and therefore also not in the project.

Unsupervised Clustering

Unsupervised clustering algorithms do not need a reference vector for a data set. Such an algorithm is able to return a feasible cluster result without any a priori knowledge. The grouping of data samples is done by forming a model of the given data set.

Another approach is to use unsupervised and supervised methods in one framework. Such hyprid approaches [Eisen1998] cluster the data set in two steps: unsupervised followed by supervised cluster algorithm. The result of the unsupervised clustering process is used as input for a supervised cluster method.

2.1.1 Distance Measurements

The ordering of data samples to a cluster in a partition-based cluster or to a node in a hierarchically clustered tree is influenced by the distance measure chosen.

A distance measure is a mathematical notation which gives a numerical value for the similarity of two distinct multi-dimensional data samples. An n-dimensional data sample is a vector with n elements. If the distance measure between two vectors returns a small value, the data samples are considered to be similar and therefore will potentially fall in the same cluster. If the distance measure function returns a high value, the data samples may have low similarity and therefore are likely different.

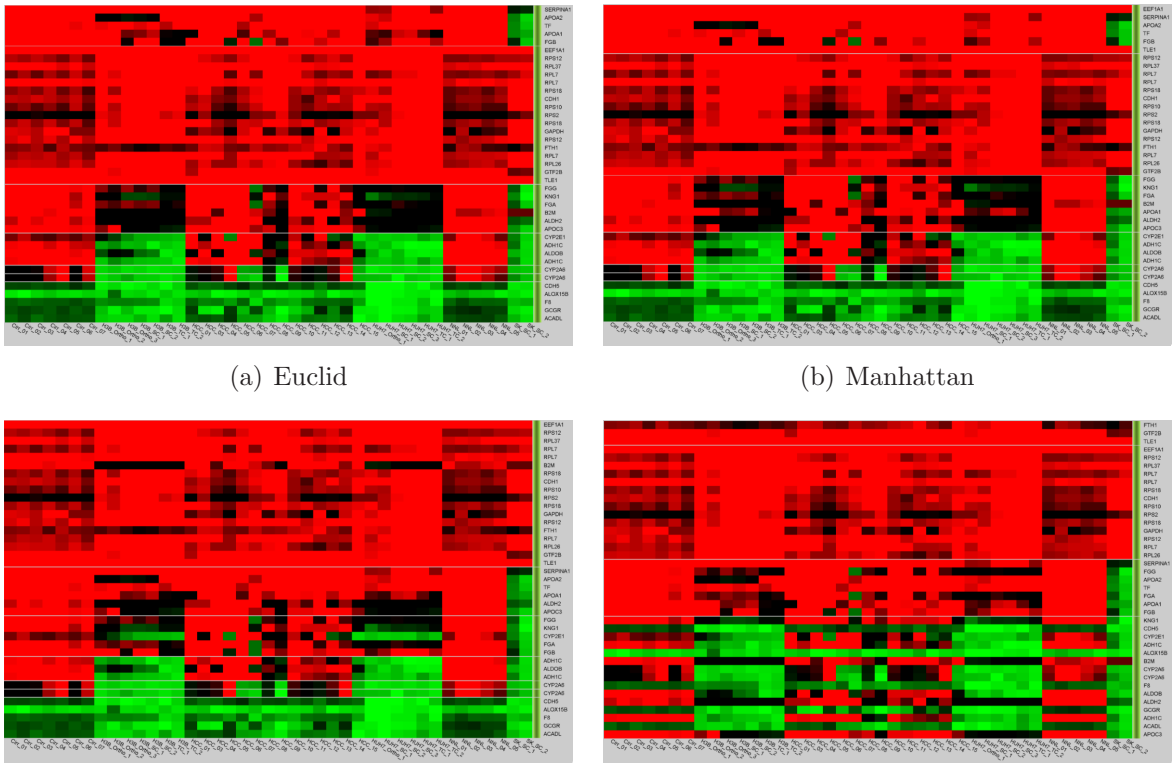
One important feature for distance measurement is the ability to handle outliers in such a way that they do not cause large measurement errors. For example, [Heyer1999] describes a problem where an outlier effect may occur when a data vector is very dissimilar in most dimensions but in one dimension the data samples are very close to each other. This may be avoided by applying normalization methods or by choosing outlier-insensitive measurements.

According to the requirements of a user, different distance measurements may be used. Because of the dispersive requirements it is not possible to automatically determine the

best distance measurement for a given data set. In most cases it may be necessary to rerun a cluster algorithm several times with different distance measure functions until the result is suitable.

In the next sub-section we will briefly discuss five interesting distance measurements. We will start with measurements which lay the focus on finding similar patterns between data samples. Afterwards we will discuss correlation based measurements which are mainly used for finding similar trends across data samples.

Figure 2.1 shows the results of four cluster runs with four different distance measurements. A data set with 39 genes and 41 experiments is clustered with affinity propagation.



(a) Euclid (b) Manhattan

(c) Chebyshev (d) Pearson Correlation

Figure 2.1: Comparison of distance measurements. A data set with 39 genes and 41 experiments is clustered four times with the affinity propagation algorithm. Each figure illustrates the result for a distinct distance measurment. The result for euclidean (a) and manhattan (b) distance measurement do not differ significantly for small data sets. Chebyshev (c) is also similar to the first two. The result for pearson correlation, shown in figure (d), depicts significant differences even for a small data sets.

Euclidean Distance

Euclidean distance (Equation 2.1) measures the distance between two points in euclidean space with the pythagorean formula: hence it is also called the pythagorean metric. It is

a commonly used distance measure for gene expression data sets, because of the "good looking" results it provides.

$$d(n,m) = \sqrt[2]{\sum_{i=1}^{N}(n_i - m_i)^2} \tag{2.1}$$

Manhattan Distance

The Manhattan distance (Equation 2.2) is not a computationally intensive measurement but it also provides good looking results. It computes the sum of the differences in each dimension.

$$d(n,m) = \sum_{i=1}^{N} |(n_i - m_i)| \tag{2.2}$$

Chebyshev Distance

The Chebyshev distance (Equation 2.3) allows the clusterer to group data samples which do not have great expression differences in any dimension. Because of the definition of this measure it returns the greatest distance between two points in any dimension. Further, it should be mentioned that outliers in any dimension may influence the result enormously.

$$d(n,m) = \max |(n_i - m_i)| \tag{2.3}$$

Pearson Correlation

This measurement was chosen, among others, by [Eisen1998] because it does not lay emphasis on the magnitude of data vectors. Rather, it gives good results for finding genes whose expression patterns have a similar shape [Jiang2004]. The distance is given by 1 - the correlation value of two data vectors.

$$c(n,m) = \frac{\sum_{i=1}^{N}(n_i - \overline{n})(m_i - \overline{m})}{\sqrt[2]{\sum_{i=1}^{N}(n_i - \overline{n})^2} * \sqrt[2]{\sum_{i=1}^{N}(n_i - \overline{m})^2}} \tag{2.4}$$

$$d(n,m) = 1 - c(n,m) \tag{2.5}$$

Jackknife Correlation

The Jackknife correlation [Heyer1999] is an enhancement to the Pearson correlation in terms of robustness to single outliers. It was driven by the jackknife procedure described in [Efron1987]. The idea is to compute the Pearson correlation for two data vectors for each dimension, while one attribute is deleted in each step. Afterwards, the minimum correlation across all the dimensions is returned. This approach avoids the "dominance effect" caused by single outliers [Zhang2006b]. To avoid effects caused by more than one outlier the Jackknife approach may also be modified to handle this task.

Because this measurement is computationally costly it is rarely used [Zhang2006b]. The distance is given by 1 - the correlation value of two data vectors.

$$c(n,m) = \min \{p_{nm}^{(1)}, \ldots, p_{nm}^{(l),}, \ldots, p_{nm}^{(p)}\} \tag{2.6}$$

$$d(n,m) = 1 - c(n,m) \tag{2.7}$$

2.1.2 Partition-based Clustering

Partition-based cluster algorithms divide the given data set into several sub classes and provide a label for each data sample. Each data sample can be assigned to a specific cluster corresponding to this label. One important feature of a partition-based cluster algorithm is whether they are able to determine the number of clusters on their own, or if they need the number of clusters as an input. Without any prior knowledge it is not possible for a user to give a feasible value for the number of clusters. Hence algorithms which are able to determine a suitable number of clusters, are preferable [Dubitzky2007]. A representative of the more intelligent first type is, for example, affinity propagation.

An example for the second type is K-means, which will be described below.

K-means Clustering

K-means [Tavazoie1999] clustering is a common and widely used algorithm. The k in the name stands for the number of clusters that will be built by this clusterer. Therefore it needs a value for k as an input argument. The cluster algorithm searches for k data samples which are used as a reference for k clusters. One essential disadvantage is the fact that the user has to choose a number of clusters without any prior knowledge. Because of this it may be necessary to cluster a given data set several times to find a satisfying result. An advantage of the algorithm is the comparatively fast performance. This fact decreases the disadvantages. Because of the good performance one can run the algorithm several times with different cluster numbers to determine a feasible cluster number for a given data set.

Affinity Propagation

Affinity propagation [Frey2007] solves the problem of determing the number of clusters in a data set. By using affinity propagation the user is only asked to provide a factor (from 1 to 10) which is used to indicate whether the number of examples (representative element for one cluster) should be small or large. After this, all the other data samples are assigned to one of those examples.

The algorithm operates with messages (see Figure 2.2) sent from one data sample to all the others and vise versa. Hence the algorithm is known as "Clustering by passing messages between data points" [Frey2007]. Data points sent "responsibilities" to candidate exemplars which indicate the likelihood for the candidate exemplar to be well-suited to act as a cluster representative. "Availabilities", sent from candidate exemplars to data points, reflect the evidence for a data point to accept an other point as an example.

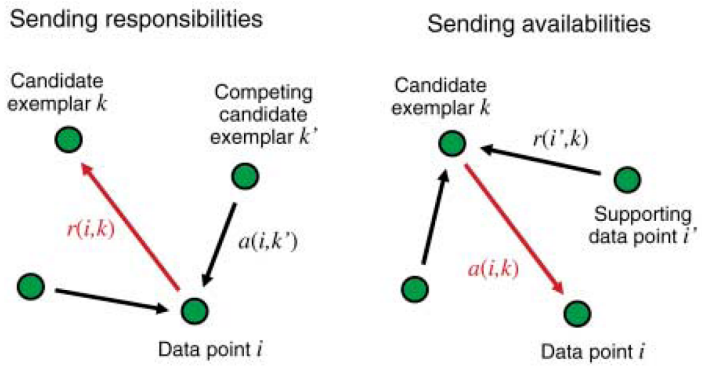


Figure 2.2: *Passing messages between data points is done in two directions. [Frey2007]*

After several iterations (a maximal number can be defined) the algorithm either converges or an exception is thrown. The cluster process for a two-dimensional data set is illustrated in Figure 2.3. There, after six iterations the algorithm converges and three data points are identified as cluster representatives (examples).

By changing the cluster factor mentioned before, a user may indirectly regulate the number of clusters formed. A decreasing value for the cluster factor results in a higher number of clusters. As shown in Table 2.1, the number of clusters determined for a data set with 1191 genes varies from eight to 35.

Self-organizing Maps

Self-organizing maps (SOMs) [Kohonen1982, Kohonen2000], also kwown as Kohonen networks, are an unsupervised method for the analysis of high-dimensional data sets. Thereby a high-dimensional data set is mapped to a low-dimensional grid according to a distance measure. Each position in the grid represents a reference vector and all the data samples similar to a specific reference vector are assigned to a position in the grid. At startup each grid unit gets a random vector. After each iteration all the data samples

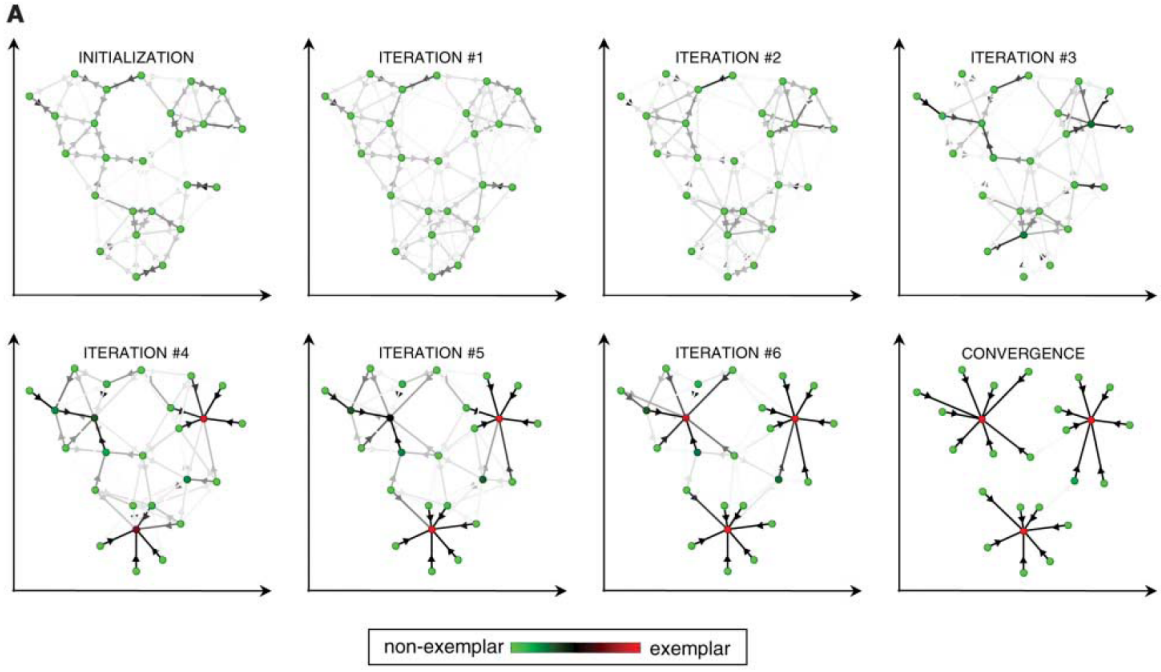


Figure 2.3: *Affinity propagation is illustrated for a two dimensional data set. The data points pass messages between each other until the algorithm converges and the corrsponding exemplar samples are found. [Frey2007]*

are assigned to a grid unit and the reference vectors are updated. This iterative approach may also be seen as an abstraction of the given data set to a low-dimensional form [Kohonen2000].

There are two distinct approaches for using SOMs as a clusterer. The first, very intuitive one is to define the number of units in a grid as the number of clusters the algorithm returns. Due this assumption a user predefines the number of clusters by defining the dimensions of the SOM. For example, a SOM with a grid dimension of three times two units returns six clusters. The second approach is more computationally intensive. Thereby, after the algorithm has converged and all data samples are assigned to a grid unit, the distance between a unit and all its members is computed. In Figure 2.4 one can see a SOM with ten times ten units and the distance coded in the color of each hexagon (unit). The information gained about the distances may be used as the input for a further algorithm (e.g. K-means [Yano2003]) which determines the clusters according to the result of the SOM.

Fuzzy K-means Clustering

Fuzzy K-means clustering [Dunn1974, Bezdek1981] is an extension to the standard K-means approach described above. In contrast to the standard K-means approach this algorithm determines possibilities for each data sample. These possibilities encode a degree of "membership" [Gasch2002] for a data sample to belong to each cluster. The more similar a data sample is to a cluster centroid, the higher is the possibility for the data sample to

Cluster factor	#clusters	#iterations	runtime[ms]
6	8	158	10016
4	12	146	8703
3	16	139	8547
2	20	115	7078
1.5	28	101	6078
1	35	116	7172

Table 2.1: *Affinity propagation results for a data set with 1191 genes. The number of determined cluster varies according to the cluster factor chosen. The higher the factor, the smaller the number of clusters. The euclidean distance was used as the similarity measurement. The test system is a dualcore Intel Pentium D with 3GHz and 2GByte RAM.*

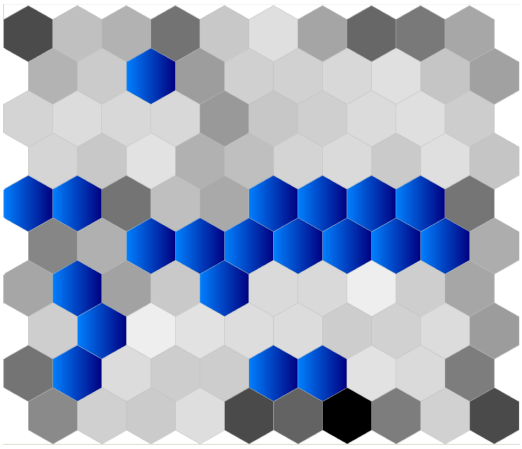


Figure 2.4: *Self-organizing map implemented by [Sturn2000]. The distance of each unit to all its members is coded in the color of each cell. Dark cells indicate a great distance and bright ones small distance. Units with no assigned data sample are colored blue.*

be assigned to this cluster. All the possibilities for a single data sample summarize to 1.0. When using only the highest similarity of each data sample the result is closely related to the result of a standard K-means algorithm.

[Gasch2002] implemented a fuzzy K-means clustering approach which returns a unique list of cluster centroids and a table of cluster memberships for each data sample. An illustration is shown in Figure 2.5.

2.1.3 Hierarchical Clustering

A hierarchical cluster algorithm, in contrast to a partition-based clusterer, does not use a class assignment for each data sample. Instead, all the data samples will be arranged in a tree structure according to their similarities/distances between each other.

There are two major types of hierarchical cluster algorithms: bottom-up (agglomerative)

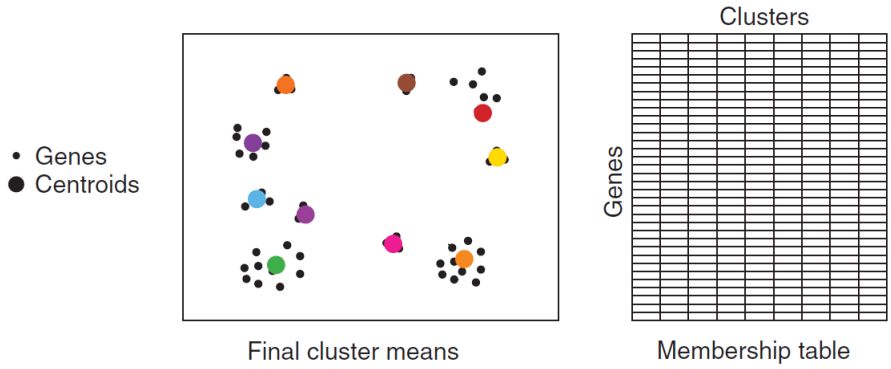


Figure 2.5: *Fuzzy K-means result consists of a list of cluster centroids and a table of cluster memberships for each data sample [Gasch2002].*

and top-down (divisive). The first starts with each data sample as an individual cluster and builds up a tree from leaf to root. The similarity of each data sample to any other has to be provided in, for example, a similarity matrix. Depending on the biggest value in this similarity matrix, the most correlated data samples will be merged to a cluster. The similarity matrix has to be updated after this operation. This procedure will be repeated until all the data samples are registered in the tree.

Divisive algorithms take the whole dataset as one big cluster and split this cluster up step by step according to a given similarity matrix to form a tree structure. This approach splits the tree until all leaf nodes have only one data sample.

To provide a cluster assignment, similar to a partition-based result, one can generate groups of data samples according to their position inside the tree structure. Hence samples which are placed in a subtree are assumed to have similar values and/or patterns a user may want to form a cluster containing all samples related to a specific hierarchy level in the tree. Such an approach is for example a cutoff bar, which will be discussed in Section 2.2.4. With this tool a user is able to adjust a variable amount of clusters.

Tree Clusterer

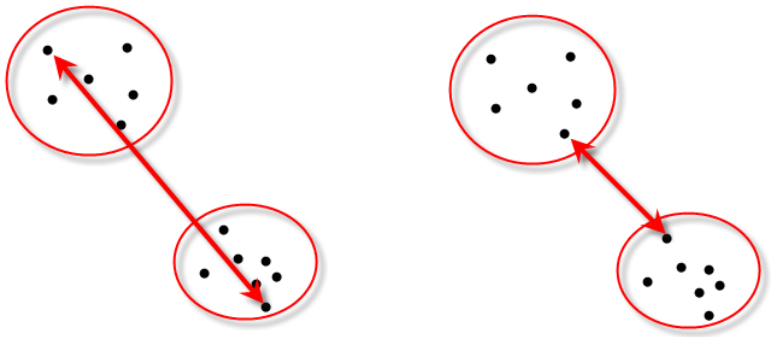
The Tree Clusterer (by [Eisen1998]) is an agglomerative method. At startup, all the data samples are seen as distinct clusters and are hierarchically grouped to form a single tree. Because the algorithm observes exactly two data samples/clusters at a step it results in a binary tree. Each node has exactly two siblings in a binary tree.

The Tree Clusterer is the umbrella term for several different approaches to build a tree from a given similarity matrix. Such approaches are called linkage rules. These rules describe how data samples are sorted according to their similarity/distance between each other and how clusters are arranged inside the tree. The three most commonly used linkage rules are briefly described below. For further information about features of linkage methods see [Sturn2000b].

- *Complete linkage* (furthest neighbour): Clusters whose members have the smallest maximum similarity are merged, see Figure 2.6 (a). This yields an intuitive cluster

result in most cases.

- *Single linkages* (nearest neighbour): Clusters whose members have the greatest minimum similarity are merged, see Figure 2.6 (b). This may yield unbalanced trees (chains).
- *Average linkage*: The distance of two clusters is determined by the distance of each element in the first cluster to each element in the second cluster. Due this approach average linkage is more computationally intensive than the two methods mentioned above, but provides well balanced results.



(a) Complete linkage (b) Single linkage

Figure 2.6: Complete and Single Linkage. (a) illustrates the furthest neighbour approach of complete linkage, (b) illustrates the nearest neighbour approach of single linkage (available at `http://stirner.wiwi.hu-berlin.de/mediawiki/mmstat_de`).

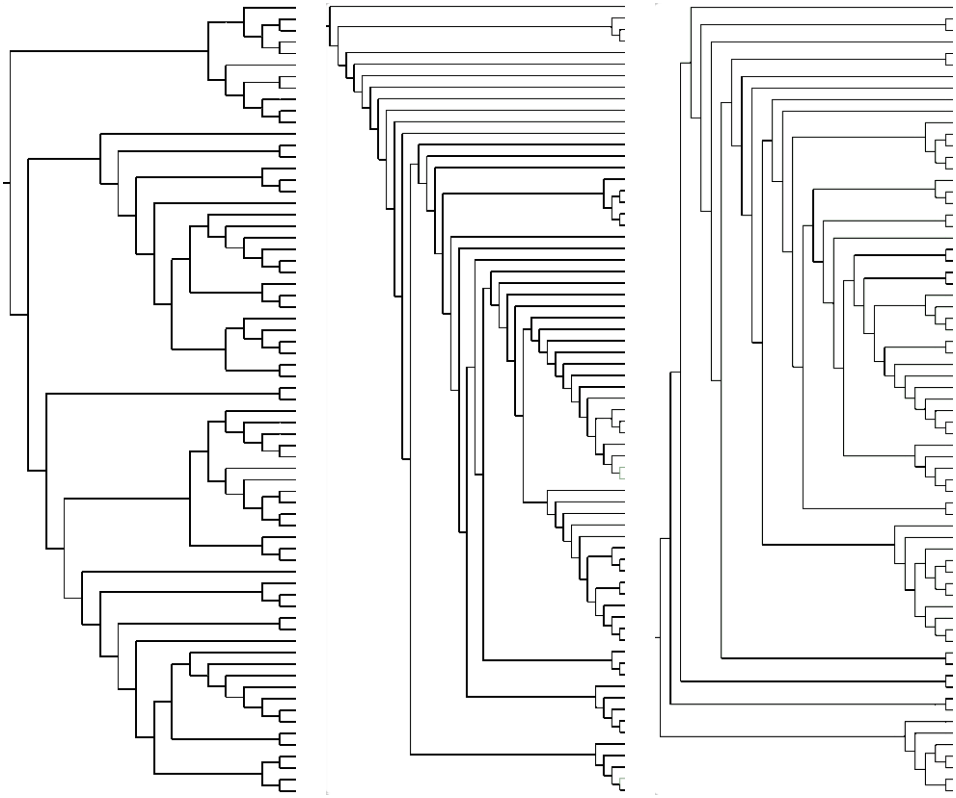
In Figure 2.7 one can see the results of three cluster processes with three different linkage rules.

Cobweb

Cobweb [Fisher1987] is an incremental conceptual system for hierarchical clustering. It builds a classification tree where each node is a concept representing an object class. Depending on the concepts stored in each node, the insertion of a new data sample can be handled. Four major operations are applied to the tree structure to build up the tree and to handle new samples.

Cobweb Operations

- Merging two classes to a single class. Merging is a method also used in agglomerative algorithms.
- Splitting one class into several classes. Splitting is used in divisive algorithms.
- Creating a new class. A new class in this context is a new concept which describes the data samples inside the class.
- Placing an object in an existing class. The existing tree will be traversed from top to bottom until a concept is found that matches the current data sample.



(a) Complete linkage (b) Single linkage (c) Average linkage

Figure 2.7: Linkage Rules used to cluster a data set with 69 samples. (a) complete linkage, (b) single linkage, and (c) average linkage.

Because a single concept can be matched to several data samples, the number of siblings is not fixed. Therefore cobweb also returns non-binary trees.

2.1.4 Clustering Gene Expression Data

When applying clustering algorithms to gene expression datasets there are two main possibilities: gene clustering or experiment clustering. Additionally a combination of the two, called bi-clustering is possible. Before we describe these three approaches we have to briefly introduce the data format. The gene expression data sets are stored in matrix form. The common approach is to fill the columns with experiments and the corresponding rows with the gene expression values. This is because, in the majority of cases, the amount of genes is bigger than the amount of experiments. A gene expression data set is illustrated in Figure 2.8. The rows are filled with n genes, while the columns include m experiments.

According to this assumption gene clustering means that the rows of a data set are seen as separate vectors and the columns are the dimensions. In terms of gene clustering, the experiment ordering stays the same but the ordering of rows (genes) will be different according to the clustering result. Experiment clustering is the corresponding approach

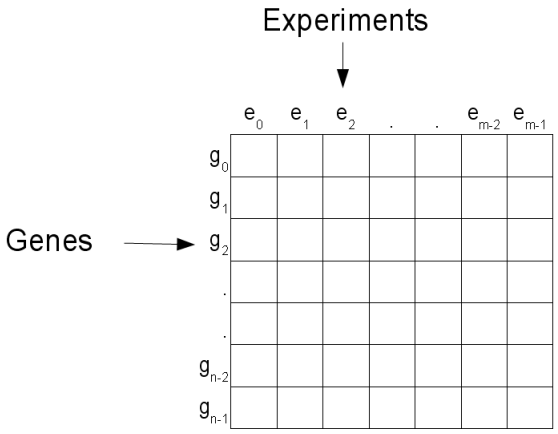


Figure 2.8: *Gene expression data set with n genes and m experiments placed in a n times m matrix.*

where the experiments will be reordered.

Bi-clustering re-orders both dimensions. There are two main approaches: the first (computationally intensive) approach is to include both dimensions in the cluster algorithm (e.g. [Cheng2000, Getz2000, Zhao2009]) also referred to as "true bi-clustering". The second idea is to cluster the data set in two consecutive passes. The first pass builds the clustering of experiments and the genes are clustered afterwards.

2.1.5 Discussion and Comparison of Cluster Algorithm

Due their individual features the six algorithms mentioned above and three linkage rules cover a broad field of cluster algorithms. Table 2.2 summarizes the algorithms properties. Affinity propagation is an especially interesting method, because it determines the number of clusters according to a given cluster factor. An advantage of K-means is the good performance, provided by Weka, which allows the user to rerun the cluster process with different values for the cluster number. The strict assignment to a single cluster is in some cases non satisfying, due to the nature of genes belonging to several function inside a cell. To overcome this disadvantage, fuzzy K-means clustering provides a possibility-matrix as a cluster result. In this matrix each gene has a membership value belonging to each cluster.

Because there is no knowledge about a feasible cluster number, hierarchical cluster results combined with a cutoff bar are a very useful technique. By moving the cutoff bar the user is able to determine the number of clusters according to the structure of the tree and the position of data samples in the tree.

We restrict our work on algorithms to cover only one dimension in one pass. We achieve a result partially comparable to bi-clustering by applying a cluster algorithm in both dimensions to a given data set.

Algorithm	partition-based	hierarchical	Special feature
Affinity Propagation	yes	no	determines number of clusters according to a given parameter
K-means	yes	no	simple and fast
Fuzzy K-means	yes	no	identification of overlapping groups
SOM	yes	no	builds abstract model of a high-dimensional data set
Complete Linkage	no	yes	good results for gene expression data
Single Linkage	no	yes	unbalanced trees (chain-like)
Average Linkage	no	yes	computational intensive
Cobweb	no	yes	builds non-binary trees

Table 2.2: *Cluster algorithms.*

2.2 Information Visualization

The field of visualization is related to computer graphics and human computer interface (HCI). Visualization is often divided into scientific visualization and information visualization. Unlike scientific visualization InfoVis (information visualization) handles "unnatural data", like databases of arbitrary information or high dimensional data [Kosara2003]. Scientific visualization handles the visualization of natural processes which may be defined by a mathematic formula in some way. Examples are visualization of volume data sets, of mathematical formulas, or scientific concepts and results.

[Ferreira2003] proposed that the interaction with visual representations may support user in dealing with several drawbacks occuring when visualizing big, complex, multidimensional data sets. Among others, visual clutter and object overlay may be reduced by allowing the user to interact with the visualization.

2.2.1 Visual Information-Seeking Mantra

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

Ben Shneiderman's [Shneiderman1996] visual information-seeking mantra has become one of the most cited papers in terms of guidelines for designing InfoVis frameworks since its publication in the year 1996. The short version includes four tasks an information visualization framework has to handle. These are overview first, zoom and filter, and details on demand. Additionally, there are three more features which Shneiderman mentiones: relate, history, and extract. All these seven tasks were evaluated by Craft and Cairns [Craft2005],

who conclude that the utility of such guidelines justifies their use.

Overview First

In most cases the user needs to see an overview of all the available data to be able to gain an insight in the data at hand. Therefore it often is preferable to present the whole data set at start-up. After the user has an overview and an idea of the size of the data she may wish to have a closer look at a distinct part of the data set. How the user selects an interesting part will be discussed in the next sections.

Zoom

Zooming [Craft2005, Furnas1995] is a common interaction method integrated in many visualization frameworks. One should distinguish zooming-in from zooming-out. These are two similar operations but provide the user with two diverse functionalities. Zooming-in means that a sub-area or the whole view is enlarged. Hence it may happen that areas at the border of the view disappear. Zomming-in provides the user with a closer view of a specific part of the data set. Therefore he may browse parts of the data which were not visible because of their limited size. Zooming-out is the inverse operation, where the view is rendered smaller which may allow the user to see more parts of data. This means that smaller features may not be visible, but an overview is provided.

Filter

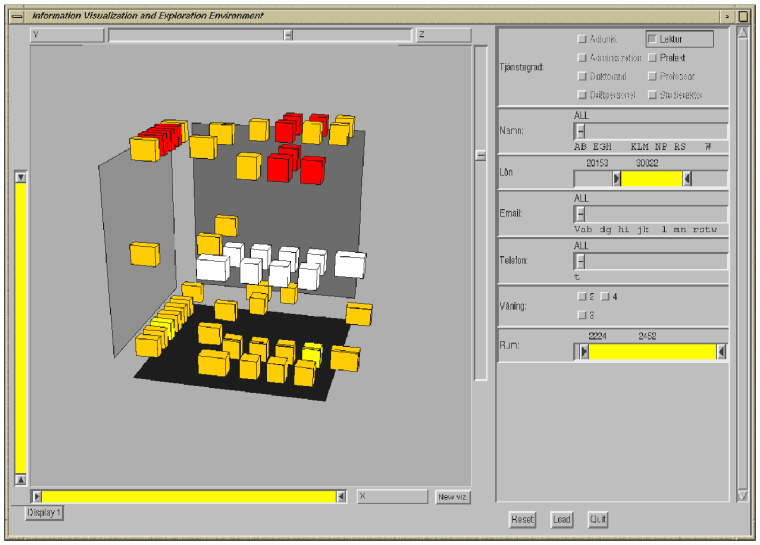


Figure 2.9: Dynamic queries used as filter operation. By adjusting the sliders the filter is updated and the view only visualizes samples matching the filters [Ahlberg1995].

Filtering is the removal of a subset of features based on an aperture adjustable for user interaction. All the data samples which match given filter are shown and all the others are hidden from the user. This restricted amount of data samples may allow the user to more easily determine trends in the data set by removing uninteresting samples. One important point in terms of user interaction is that filtering operations may be applied

dynamically and that filter operations have immediate effects on the visualization. An approach for applying filter operations "online" is described by [Ahlberg1995]. They provide the possibility of adjusting filter operations with sliders and the result of the filter operation is visualized immediately (see Figure 2.9).

Details on Demand

Details on demand is a widely used method (e.g. [Ahlberg1995]) for representing additional information about an element corresponding to a large view. Because of the limited screen space it is impossible in most cases to visualize all the information available about all the entities directly on the screen. Therefore some information is only rendered on user request (see Figure 2.10). This request may be triggered by a mouse event (e.g. mouse-over or click) or comparable interactions. The information is then visualized for example in a pop-up window, a tool-tip, or is integrated in a context menu.

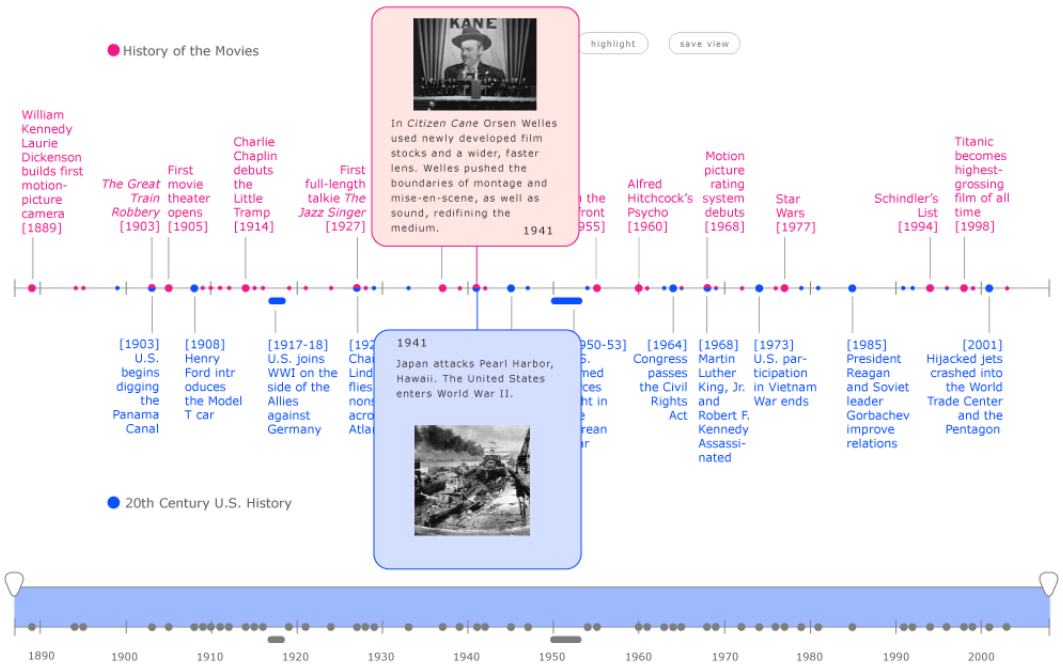


Figure 2.10: Details on demand approach integrated in a timeline visualization (available at `http://www.raccoonandzebra.com/timelinecompare/`.

Relate

As Shneiderman mentioned [Shneiderman1996], one important aspect in understanding big, interrelated data sets is the visualization of relationships between two or more data items. The relationship should be visualized in such a way that the user receives as much information as possible both about a specific data sample and related samples.

History

Providing a history supports a user in cases of errors caused by a wrong user interaction. [Shneiderman1996] reveals that a history approach is necessary to provide actions like "undo, replay, and progressive refinement". Especial progressive refinement is a very

important approach in state-of-the-art InfoVis frameworks.

Extract

The user should be able to extract some information out of the data represented by the InfoVis framework [Craft2005]. The extracted data may be used in other views of the same framework, copied to other programs, or stored in a text file for later usage.

2.2.2 Methods Combined

In addition to the mantra invented by Shneiderman [Shneiderman1996], there are several further methods which are combinations of above mentioned methods or improvements in terms of user interaction.

Linking & Brushing

Linking & brushing [Schumann2000] is a method used in multi-view systems [Baldonado2000]. This means that the information is split into several views and visualized with different visualization techniques (see Figure 2.11). Due the fact that no visualization method is optimal for all possible data sets, frameworks providing several visualization methods will achieve the best performance.

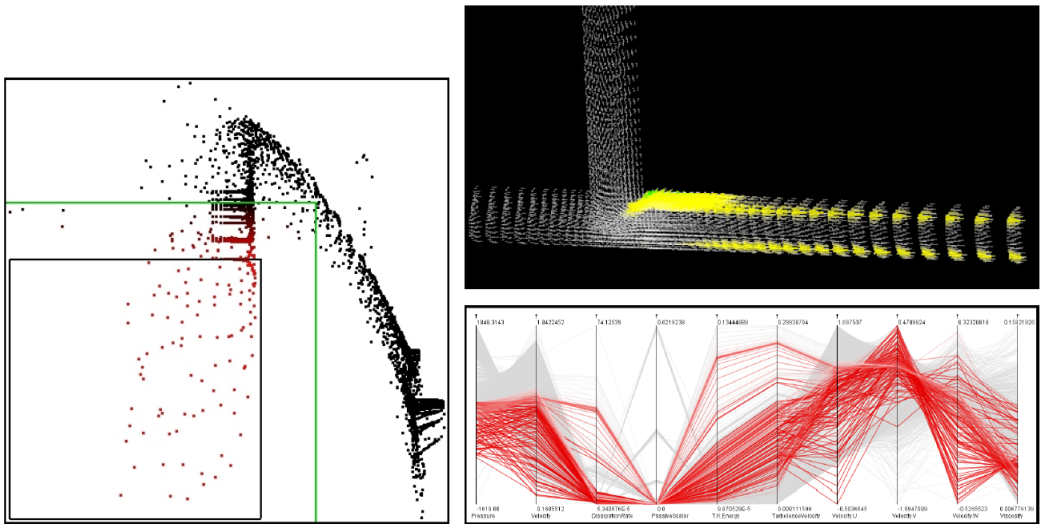


Figure 2.11: Multi-view visualization of a multidimensional data set. In the left scatterplot view, several data points are marked by using a brush. These points are highlighted in the views on the right.[Hauser2004].

Brushing is the interactive selection. It determines those parts of the data set which are marked in all views of a framework (a 3D approach is described in [Doleisch2002]). Linking is used to provide the user with a kind of relationship, or rather, connection between the location of data samples in the framework. Linking visualizes all the instances of a brushed sub data set in distinct views. To provide an interactive framework

changes in one visualization have to be automatically reflected in the other visualizations [Keim2002].

Focus+Context

When visualizing large data sets it is difficult to enable a user to explore some data in detail (focus) and still be able to relate the data to the whole data set context. Focus+context techniques [Kosara2003, Hauser2004] strive to do exactly that.

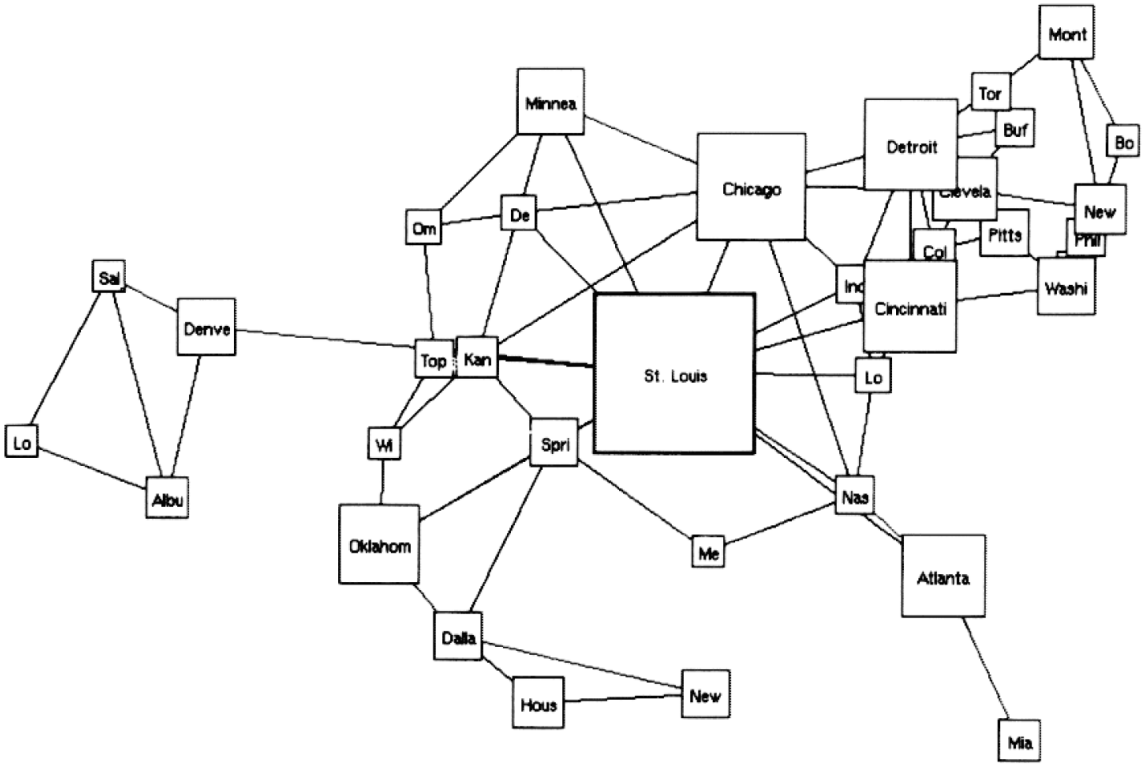


Figure 2.12: *Fisheye concept applied to a map of cities in the USA. The focus lies on St. Louis [Sarkar1994].*

Several ways of providing focus+context approaches exist. [Kosara2003] lists four groups of techniques which are usually used. Distortion orientated approaches like the fisheye concept (see Figure 2.12) set a subpart of the data in focus by rendering the interesting part bigger while the rest of view is visualized smaller. Overview methods are applied in multiview setups. While one view provides the whole data set, another view visualizes only a sub part of the data. [Ball1996] implemented a three level visualization, see Figure 2.13. Filtering techniques provide additional information about a data sample. [Bier1993] introduced a magic lens. With this tool a user may focus a part of the view and gains additional information about it. Finally [Kosara2003] lists in-place techniques. These methods do not need an additional view or a tool. Instead, interesting parts are highlighted directly in the view and are, therefore, set to focus.

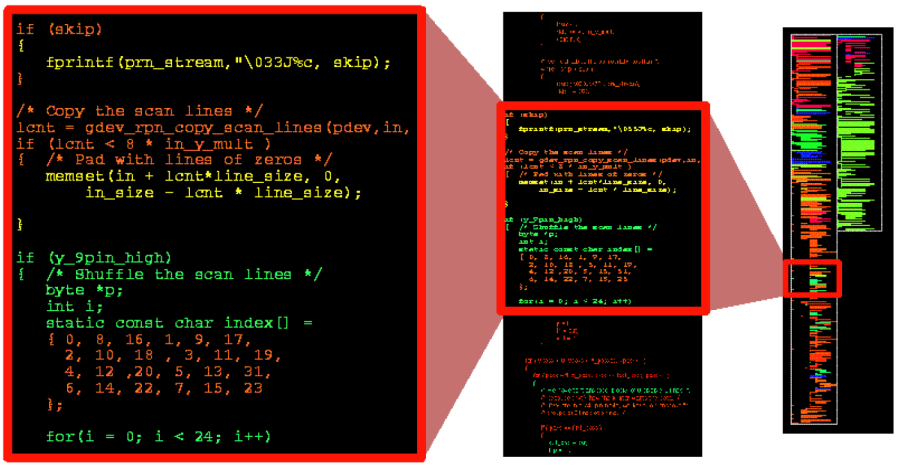


Figure 2.13: *The line representation uses a three level visualization of text with a varying scale. Due this approach the rightmost level builds an overview about the whole file while the leftmost level provides a subpart of the file in focus. [Ball1996].*

Composite Views

Streit et al. [Streit2007, Streit2008] proposed two concepts for the arrangement of several views in a single window. Both approaches are integrated in Caleydo. The first is the *jukebox* approach, it provides a textual list where a user is asked to select pathways. Afterwards, the selected pathways are placed in a 2.5D layer and elements occurring in several layers are connected by visual links [Shneiderman2006, Aris2007, Collins2007]. The curent selected pathway is rendered a second time standalone which allows the user to explore the pathway in detail.

Based on the jukebox approach they proposed an additional arrangement system called *Bucket* (see Figure 2.14). In a Bucket up to five views are arranged on the walls and the bottom of a bucket. For further information about the jukebox and Bucket approach see [Lex2008, Streit2009a].

2.2.3 Visualizing (clustered) Gene Expression Data

In this section we want to talk about visualization techniques for multi-dimensional datasets. A gene expression dataset can be seen as a such a set of data samples, where each experiment represents a dimension. As a data set may contain up to 50 or even more dimensions only a handful of visualization techniques are suitable.

After this, the integration of cluster assignment and cluster border visualization is discussed for each view. We will start with parallel coordinates as a widely used method for numerical and textual data sets, followed by scatterplots. Finally heat maps, which are the most commonly used views for gene expression data, will be examined.

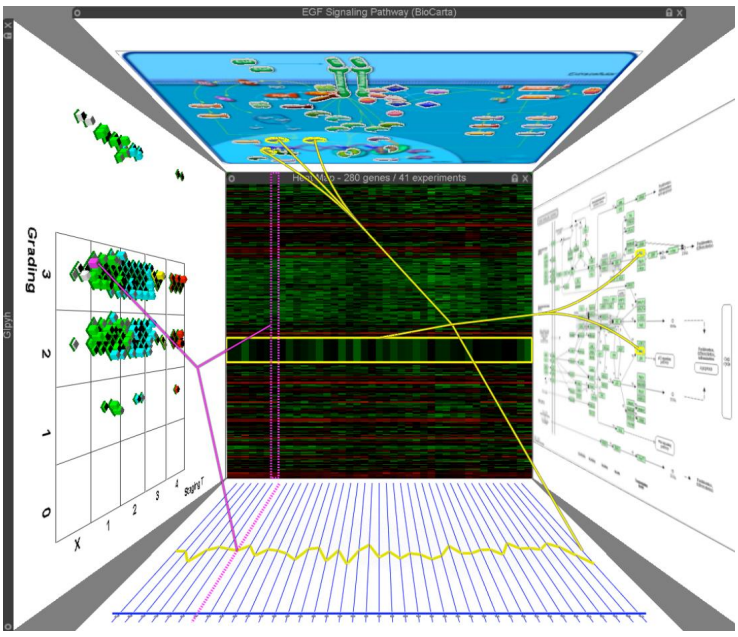


Figure 2.14: *Bucket visualizes up to five views. Visual links are used to connect elements in the distinct views. [Streit2008]*

Parallel Coordinates

Parallel coordinates [Inselberg1985] got their name because of the parallel placement of axes, where each axis represents a dimension in a high-dimensional data set. An n-dimensional point in the data set is visualized as a polyline connecting all n axes (see Figure 2.15). The positioning on the axes is determined by the value of the data point at the corresponding dimension. One important advantage of parallel coordinates is the possibility to visualize nominal data by mapping nominal values to numerical ones.

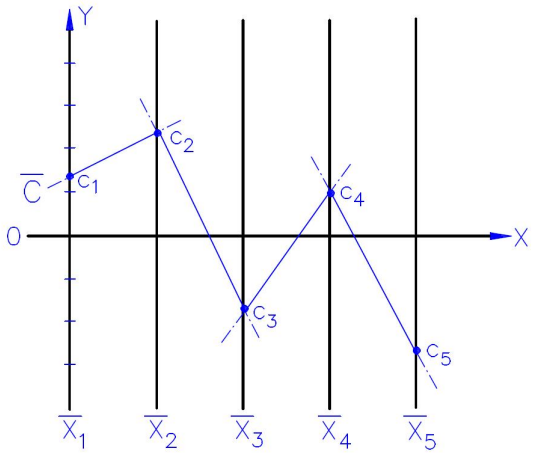


Figure 2.15: *Parallel coordiantes visualizing a 5-dimensional data vector (Alfred Inselberg, available at* `http://www.math.tau.ac.il/~aiisreal/` *).*

Several approaches for cluster visualization in parallel coordinates have been invented

([Fua1999, Johansson2004]). One commonly used technique invented by [Fua1999] is to visualize a cluster by printing one line representing the center of the cluster and several lines with decreasing alpha values for the rest of the data samples in the cluster. In Figure 2.16 one can see the visualization of one cluster in a four dimensional data set.

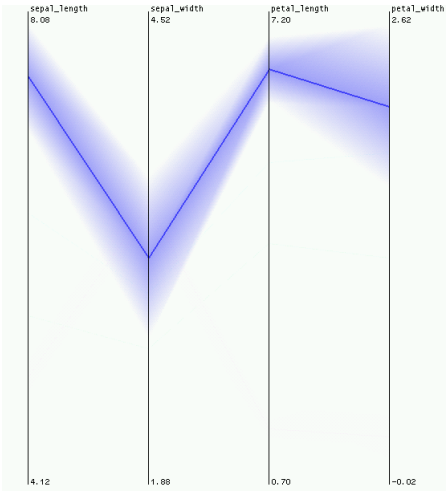


Figure 2.16: *Parallel coordinates view visualize one cluster [Fua1999].*

[Johansson2005b] described an approach for visualizing clusters with transfer functions operating with high-precision textures. Several transfer functions are provided. Figure 2.17 shows a 7-dimensional data set. A transfer function called "Feature animation" was used to visualize statistics about clusters [Johansson2005b].

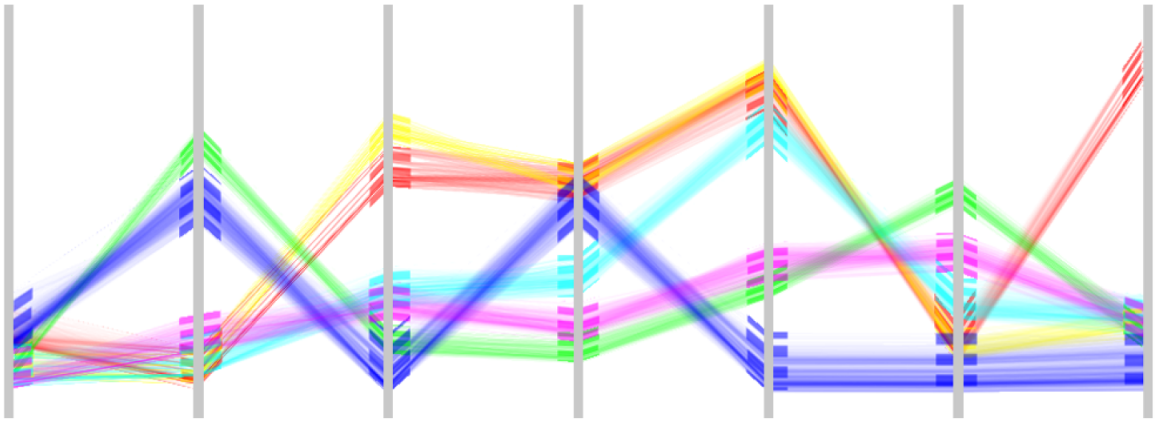


Figure 2.17: *Feature animation transfer function applied to a 7-dimensional data set with about 7000 data samples divided into 6 clusters [Johansson2005b].*

Scatterplot

In scatterplots each data point is directly mapped to a point in a coordinate system. Figure 2.18 shows a 2D scatterplot where the variable *SPO 30M* is mapped to the x-axis and *SPO*

2H to the y-axis. Because of the restriction to a maximum of three dimensions, datasets with higher dimensions can not be directly visualized. They need a further preprocessing step to reduce the number of dimensions [Thomas2005]. This could be done manually by choosing two or three dimensions which were mapped to the coordinate system afterwards.

Since a user may not be able to choose representative dimension for a high-dimensional data set, another approach that can be used is an algorithmic method. PCA (principal component analysis) or a comparable technique may be used to determine the most interesting dimensions in a high-dimensional data set.

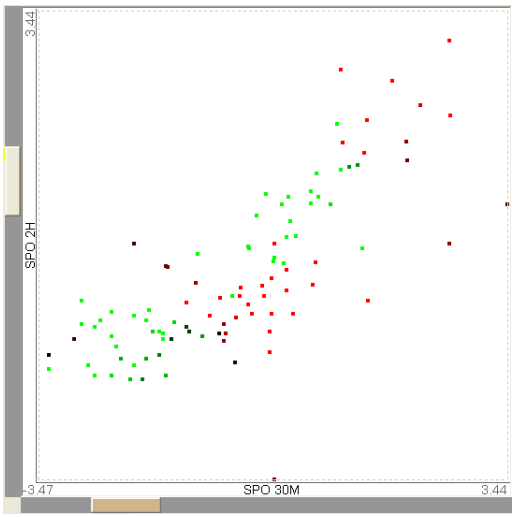


Figure 2.18: *2D Scatter plot [Seo2002]*

The integration of cluster visualization can be done by assigning a seperate color to each cluster. All the data samples lying in a cluster have the same color. Figure 2.19 shows a data set with 12 clusters in 3D coordinate system.

Heat map

Heat maps are one of the most common techniques for visualizing gene expression data sets. Normally gene expression data sets are given in text files with rows representing genes and experiments in columns. Each gene expression ratio is mapped to a color value visualized in a quad in the heat map. The domain specific color coding is green, black, and red but other codings are also used. Green is returned in cases where a gene is down-regulated. If a gene is up-regulated it is coded in red. Due to the fact that many people are not able to distinguish red from green (color blindness) many heat map implementations provide the possibility of changing the coloring scheme. A widely used alternative color combination is blue and yellow.

Cluster assignments and dendrograms are easily integrated in a heat map view. [Eisen1998] was the first to combine hierarchical clustering algorithms and visualization of the gene expression cluster results (see Figure 2.20).

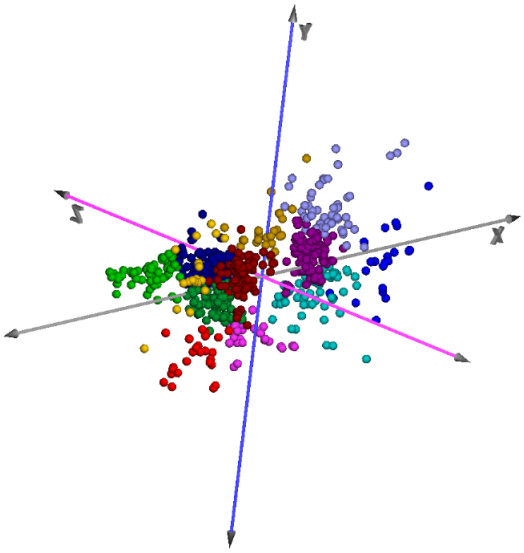


Figure 2.19: *3D Scatter plot view which visualizes 12 clusters implemented by Sturn [Sturn2000].*

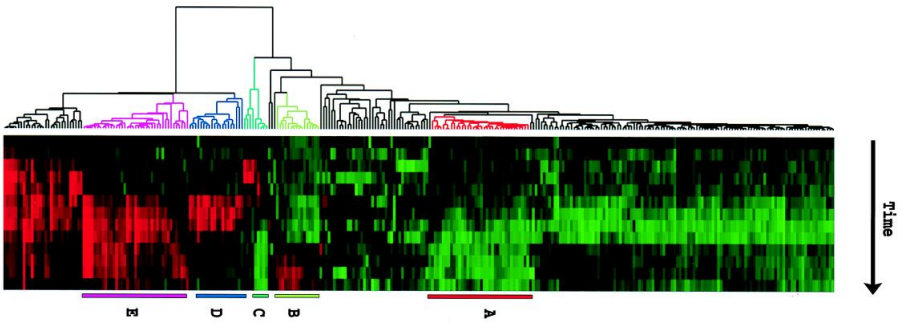


Figure 2.20: *Heat Map implementation by Eisen et al. [Eisen1998]*

A few years later [Seo2002] combined hierarchical cluster result visualization (using a dendrogram) with a hierarchical approach (see Section 2.3) to visualize the information in a more user friendly way. They proposed a multi-view environment, where the whole dataset is visualized in one view and a subpart of the data samples is rendered in another view. The user is able to determine clusters according to their position in the tree with a tool called similarity bar. The subtrees determined are then handled like the result of a partition-based algorithm.

2.2.4 Tree Visualization

A tree is a data structure consisting of nodes and edges. Edges connect nodes to form a graph without loops. There are several examples where data fits perfectly to tree structures: family trees, computer file systems, hierarchical cluster results, etc.

There are four commonly used visualization methods for tree structures (see, for examples, [Telea2007]): dendrogram, radial hierarchy, treemap, and hyperbolic tree. These four will

be discussed below.

Dendrogram

The dendrogram view is the most common visualization method for hierarchical clustered gene expression data sets (e.g. [Eisen1998, Seo2002]). However, it is equally applicable to any other data which may be mapped to a tree structure.

The dendrogram visualization technique has one feature [Ovaska2008] which makes it preferable for cluster result visualization. This is the cutoff value, shown in Figure 2.21 as a blue line in the left dendrogram. The user is able to dynamically adjust cluster numbers and sizes with this handle, depending on the hierarchy determined by the cluster algorithm.

The cutoff bar is also known as a minimum similarity bar [Seo2002].

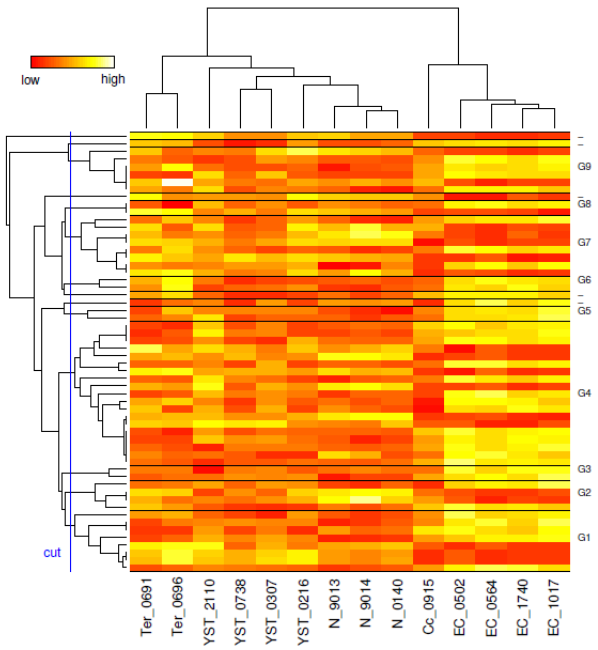


Figure 2.21: *Dendrogram visualizing a hierarchical clustering result (tree structure) [Ovaska2008]*

Koren and Harel [Koren2003] illustrate a significant disadvantage: a dendrogram does not provide information about symmetrics, outliers, and the shape of the clusters. To overcome this problem, they mentioned an algorithm which reorders the data samples inside a dendrogram and combines the dendrogram visualization with a two-dimensional scatterplot (see Figure 2.22).

Radial Hierarchy Layout

Radial hierarchical layouts were part of research contucted by Andrews [Andrews1998] and Chuah [Chuah1998]. Additional work in terms of user interaction and visualization of

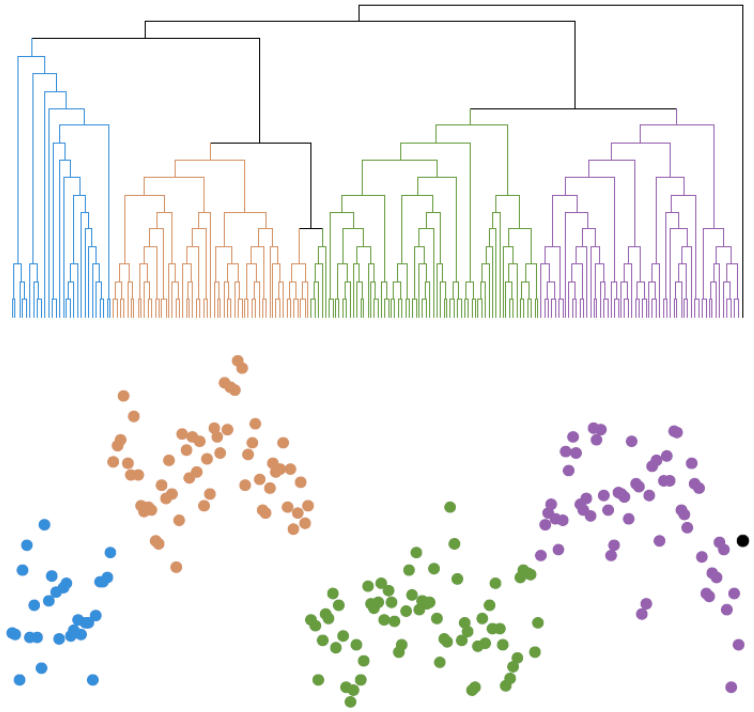


Figure 2.22: *Reordered dendrogram places similar elements near to each other [Koren2003]*

large hierarchies was done by [Stasko2000].

A radial hierarchical layout positions the root node of a tree structure as a dish in the center of the screen space and places the children concentrically around the parent node. Figure 2.23 shows an example. One feature of the nodes (like the number of children, hierarchy level, etc.) is been coded in the angle of a partial disc. Other aspects can be encoded by using coloring.

Treemap

One very important limitation of the methods mentioned above is that they require a considerable amount of space [Telea2007]. A tree map [Johnson1991] does not waste any screen space in terms of unused areas between nodes in the tree. Instead, any pixel on a screen is used to visualize a given hieararchy (see Figure 2.24).

Each element is encoded in a rectangle. The area of the rectangle encodes the most important attribute of the data (e.g., size of a folder in a file system). The screen space is subdivided into several rectangles, corresponding to the structure of the tree. One rectangle represents a node in the tree and contains nested sub-rectangles, which correspond to the children of the node.

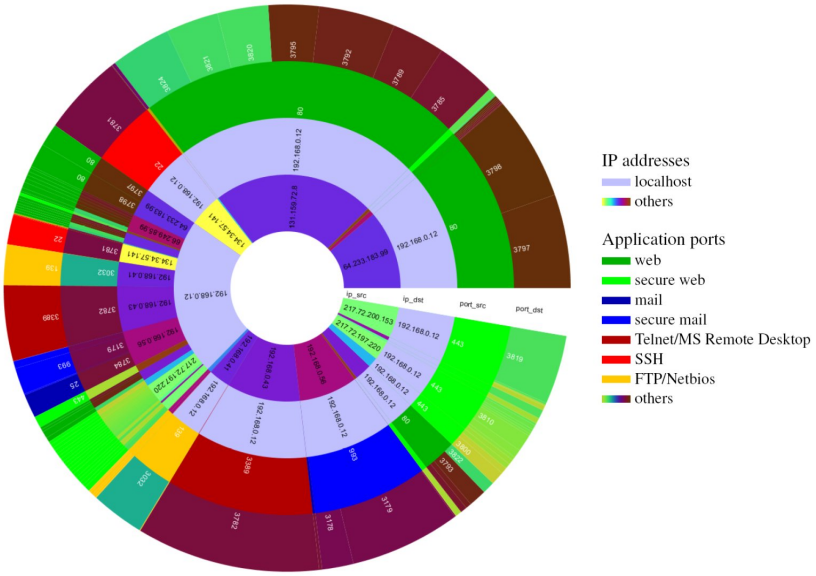


Figure 2.23: *Radial Hierarchy visualizing network traffic online* (available at `http://www.infovis-wiki.net/index.php?title=Radial_Hierarchical_Visualization`

Hyperbolic Tree

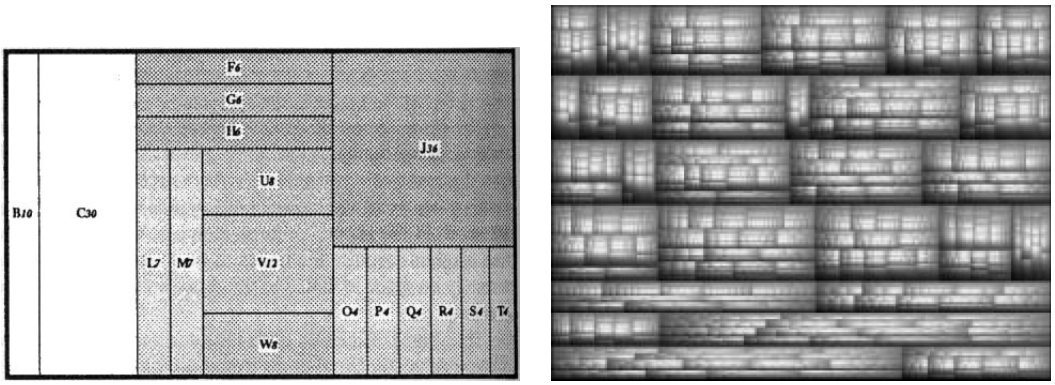
[Lamping1995] described a framework which combines a focus+context approach with visualization for large hierarchies. They visualized a tree structure by projecting a tree onto a sphere. The node in focus is placed at the center of the sphere, the rest of the nodes are placed surrounding it. Using a fisheye concept the node in focus is visualized bigger while the rest of the nodes are shown with decreasing size. This means that nodes near the center are bigger than nodes near the border of the circle. Excess nodes are hidden. To change the focus the user may click on a node in the tree or drag a node to a new position.

A hyperbolic tree visualization is able to visualize big hierarchies (see Figure 2.25). However it is restricted in terms of visibility, because nodes near or beyond the border are rendered small respectively not at all.

2.2.5 Discussion

The most common view for gene expression data sets is the heat map. The equivalence of the color mapping used on the microarray chip and the color of the elements in a heat map is one reason for its wide spread adoption. Thus the heat map visualization method is very intuitive for biologists. A further advantage of heat maps is the possibility to visualize hieararchically clustered data sets by combining a heat map view with a dendrogram. A direct mapping from leaf nodes of the tree to data samples inside the heat map is possible by placing the dendrogram next to the heat map.

The dendrogram is probably the best tree visualization method for this domain. For example, hyperbolic tree visualization has major drawbacks when the depth of a tree structure



(a) Treemap [Johnson1991] (b) Cushion Treemap [Wijk1999]

Figure 2.24: Treemap implementations. (a) shows an early treemap implementation by [Johnson1991]. The treemap shows a cheese partioned in 17 pieces based on their weights. (b) shows a cushion treemap [Wijk1999].

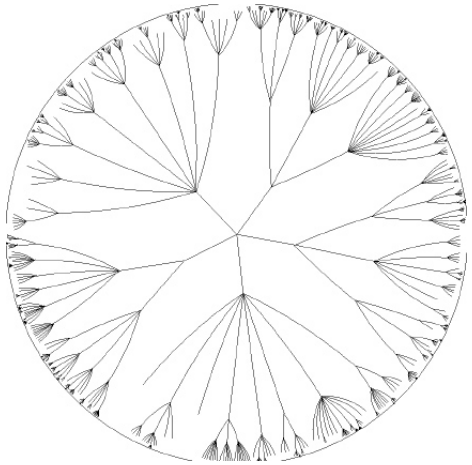


Figure 2.25: *Hyperbolic tree shows a tree with its origin in the center and 1004 nodes [Lamping1995]*

reaches a critical value. A further key advantage of the dendrogram is the possibility to integrate a cutoff bar. With this feature the user is able to determine the number of clusters according to the tree structure.

2.3 Comparable Frameworks

Caleydo is a framework which focuses on the visualization of biological data, especially gene expression data and pathways. Due to the importance of visualization for the field there are several comparable frameworks. Some of those have richer functionality in areas like data mining methods or number of visualization methods for gene expression data. But hardly any other framework combines pathway visualization and gene expression visualization methods as Caleydo does. The combination of these biological data sources allows a user

to retrieve a lot of information out of a data set by analyzing special genes according to their expression ratio and their occurence in several pathways.

Several visualization frameworks which provide cluster algorithms and visualization of cluster results will be discussed in the following pages. As the focus of this work lies on data mining methods for gene expression data we will focus on frameworks for gene expression ratios. Programs handling the visualization of pathway data will not be discussed. For further information about pathway visualization in general, and especially in Caleydo, see [Streit2007] and [Streit2008].

We will begin with four very common frameworks in terms of visual analysis of gene expression data sets. The very first, and propably most important, framework is the one provided by Eisen et al. [Eisen1998]. Then we will discuss four more, in the broadest sense, open source frameworks. Finally three commercial products are listed.

Cluster 3.0 and Treeview

One of the first approaches using cluster analysis on gene expression data was investigated by Eisen et al. [Eisen1998]. Therefore it is one of the most often cited papers in this domain. The framework provided by Eisen consists of two separate programs. The first one is the Cluster 3.0 which is, as the name suggests, responsible for the data mining part. It provides a couple of clustering algorithms with several distance measurements. For example, the framework provides hierarchical and k-means clustering, as well as SOM and PCA analysis.

The results obtained by the clusterer are stored in a text file in tabular form. This stored cluster assignments are then used by the second program called Treeview (shown in Figure 2.26).

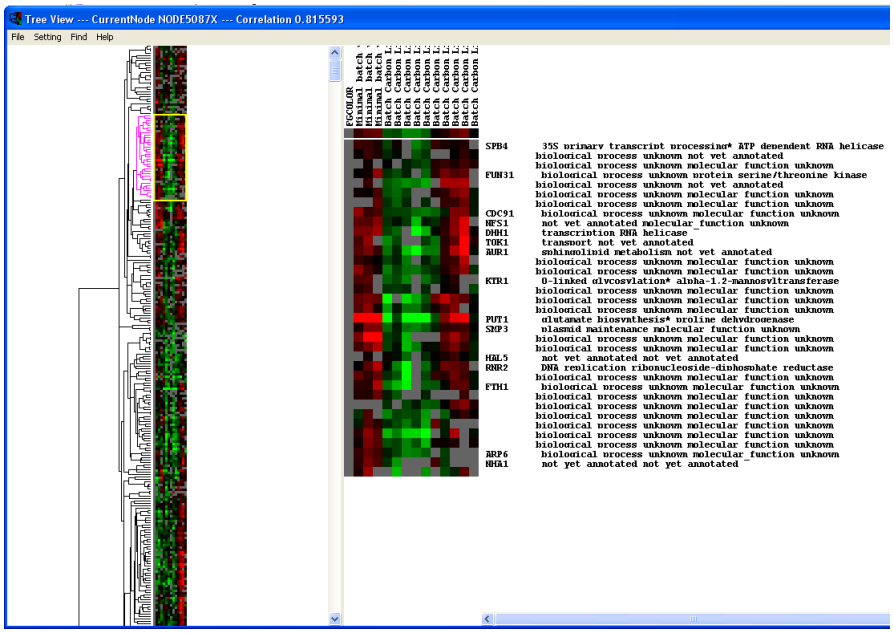


Figure 2.26: *Treeview implementation by Eisen et al. [Eisen1998]*

Treeview is responsible for the visualization. It allows a user to load a previously clustered

data set and to visualize the gene expression ratios in a heat map with a dendrogram. As can be seen in Figure 2.26, the left side of the window provides an overview of the whole data set. In the middle one can see the visualization of a part of the heat map selected by the user in the overview part. The labels corresponding to each gene are visualized on the right side.

Aside from the selection of nodes in the tree structure this tool does not provide further user interactions.

Hierarchical Cluster Explorer

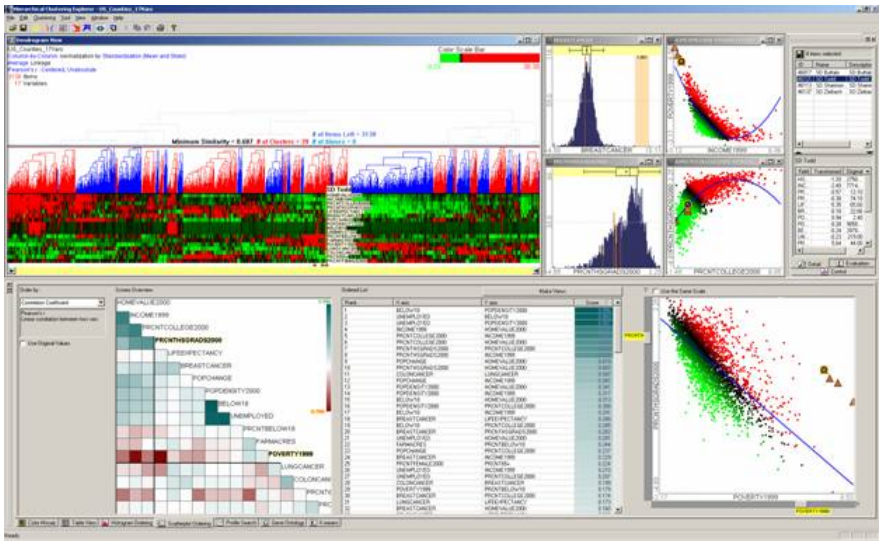


Figure 2.27: *Hierarchical Cluster Explorer. Gene expression cluster result visualization by Seo and Shneiderman [Seo2002].*

The hierarchical cluster explorer [Seo2002, Seo2005] is in some ways an enhancement to Eisens approach. Seo and Shneiderman combined the visualization of hierarchical and partition-based cluster results with several additional views and interaction techniques, for example, a scatterplot, a histogram, and a table view (see Figure 2.27). One very interesting feature invented by Seo and Sheiderman is the minimum similarity bar. This may be interpreted as a cutoff value which determines clusters corresponding to the similarity of distinct items. Using drag&drop the similarity bar may be placed inside the dendrogram and a number of clusters will be formed corresponding to this position. Additionally, HCE provides a hierarchical visualization approach which means that a subpart of the whole data set may be selected by the user and rendered in a separate view.

An additional feature is the "Compare Two Hierarchical Cluster Results" [Seo2002] which is integrated in the framework. By using this option the user is asked to choose two linkage rules and, if desired, two different distance measurements. The loaded data set will be clustered twice with the previously chosen algorithms and the results are visualized next to each other in a single view. When selecting one data sample it is highlighted in both heatmaps. Thus the user is able to compare the differences in the cluster result for a single (specific) data sample (see Figure 2.28).

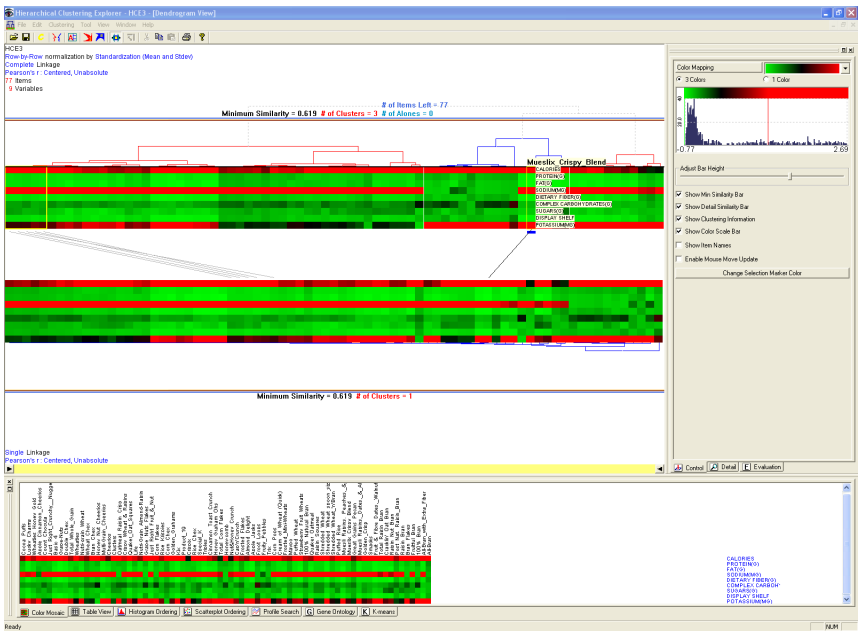


Figure 2.28: *Hierarchical Cluster Explorer. A data set was clustered twice with two different hierarchical cluster algorithm and both results are visualized in one view [Seo2002].*

GeneVAnD

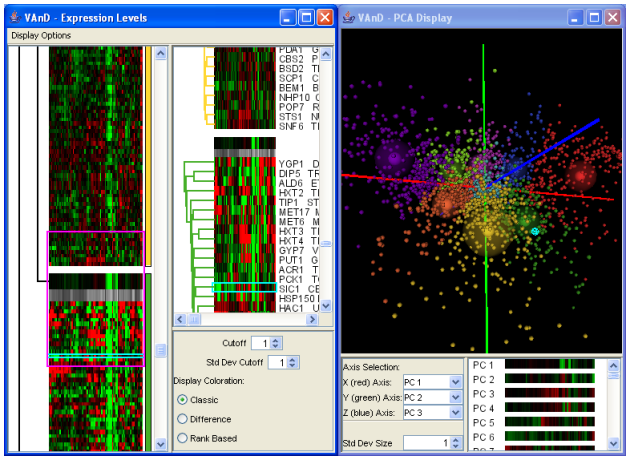


Figure 2.29: *GeneVAnD [Hibbs2005]*

GeneVAnd [Hibbs2005] (see Figure 2.29) is another framework providing data mining and visualization methods for gene expression data. As GeneVAnd provides additional views like scatterplots it is a very interesting approach. However, is does not handle large data sets well. The framework provides no support for visualizing large data sets in a hierarchical approach. It is therefore not possible to fit the data into a single view. Two possibilities to overcome this disadvantage are to browse the whole data with a scrollbar or to use large high resolution displays to render as much information as possible into a single view.

Genesis

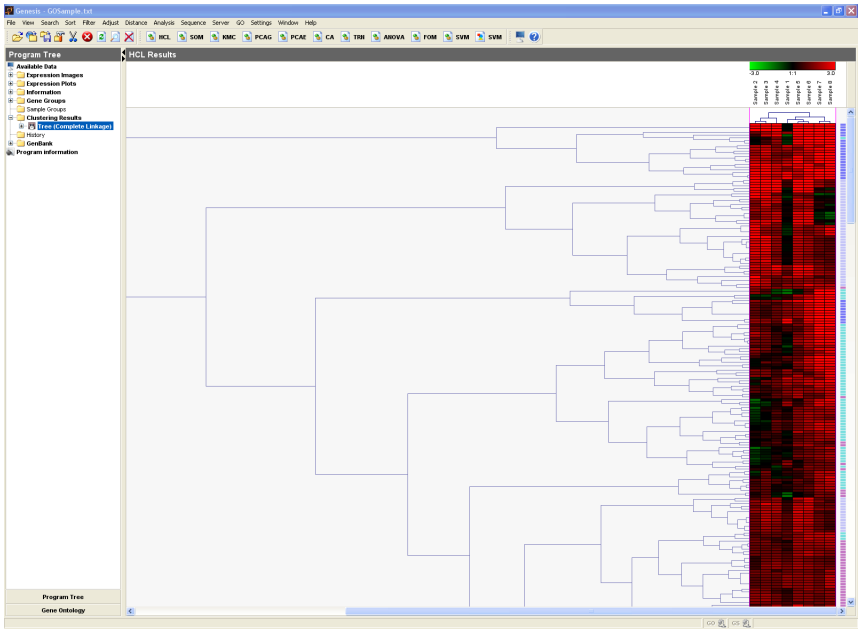


Figure 2.30: *Genesis [Sturn2000]*

Genesis [Sturn2000] is a framework written in Java which provides a couple of cluster algorithms (hierarchical, SOM, K-means, PCA, SVM (support vector machines), etc.) for analyzing large gene expression data sets. It was developed by Alexander Sturn during his diploma thesis. It additionally provides several tools for normalization, filtering, cluster visualization, and distance measurements. It is therefore a very useful framework but has, as many others, the problem of visualizing big data sets. Figure 2.30 shows a data set clustered hierarchically in both dimensions. Due the limited amount of screen space it is not possible to visualize the whole data set at a time. The application uses scrollbars to browse the data set.

Other Open-Source Tools

GeneXplorer [Rees2004] is a web application, under the permissive MIT Open Source license, for interactive microarray analysis. As it is available online with a standard web browser it may be used with the majority of operating systems and even out of date hardware.

TM4 [Saeed2003] is a suite containing four distinct tools for data analysis. The most important in terms of visual analytic of gene expression data is the Multiexperiment Viewer (MeV). Among other features MeV contains hierarchical clustering, SOMs, PCA, and self-organizing trees.

Mayday [Dietzsch2006] is a graphical tool for the analysis of microarray data, implemented in Java. The main part of the framework is a light-weight core which provides functionality for data structures and plug-in handling. It provides several plug-ins for data mining and data visualization. Among others, it offers a connection to the statistical environment R and Weka. Mayday has the focus on the fast implementation of new analysis

methods.

JTreeview [Saldanha2004] is an open source Java implementation of Treeview [Eisen1998]. Because of the implementation in Java it is not bound to a specific platform and therefore may be used with any operating system which provides a Java Virtual Machine (JVM). There is no data mining support integrated in JTreeview. A user needs an additional program like Cluster 3.0 which provide a clustered data set.JTreeview supports the user by providing additional views like the scatterplot view.

Commercial Tools

GeneSpring GX[1] is a commercial tool for the interpretation and evaluation of gene expression data sets. It provides several visualization methods (scatterplots, dendrograms, etc.) and a couple of data mining methods. It includes, among others, SVM, bayesian statistic, and neural networks. Due this tremendous support in terms of analyzing gene expression data sets it is a recommendable framework. However, the one big drawback for academical use, is the price.

Spotfire[2] is a business intelligence tool applicable to a wide range of data sources. Due to its great support for data mining methods and visualization techniques it is also interesting for gene expression data analysis. It supports, among others, 2D and 3D scatterplots. Data mining methods like PCA and several cluster algorithms are available.

GeneLinker[3] provides the user with a great many useful functionalities. For example, they include methods for pre-processing, normalization, and filtering. Further, GeneLinker supports several cluster algorithm like SOMs, K-means, and hierarchical clustering. Scatterplots, dendrograms, and color matrix plots (heat maps) are available for the visualization of data sets. Additionally, the program provides linkage of data samples across several views and an export functionality.

2.3.1 Discussion

There are several frameworks for the analysis of multidimensional gene expression data sets. Most of them focus on the data mining part by providing different cluster algorithms and many different distance measurements. All of the discussed frameworks do not visualize large data sets well. In heat maps the reduction of size of a single quad (single data value) is limited. To deal with these problems most programs provide a simple scrollbar approach which allows the whole data set to be browsed in a single view by providing a minimum size for the heat map elements thereby violating the focus+context paradigm. One interesting approach to overcome this is to visualize the data set hierarchically. This means that the framework shows the whole data set to give an overview, and subparts of the data to show the focus. Table 2.3.1 shows an overview of all the above mentioned frameworks and their features.

[1] http://www.chem.agilent.com/
[2] http://spotfire.tibco.com/
[3] http://www.improvedoutcomes.com

Application name	PCA	Partition-based clustering	Hierarchical clustering	License	Special features
Treeview & Clusterer [Eisen1998]	yes	yes	yes	free for acadmic use, commercial licenses available	filtering
Hierarchical Cluster Explorer [Seo2002]	yes	yes	yes	free for acadmic use, commercial licenses available	2 level hierarchical visualization
GeneVAnD [Hibbs2005]	yes	yes	yes	for non-commercial research	
Genesis [Sturn2000]	yes	yes	yes	free for acadmic use, commercial licenses available	provides several data-mining methods
GeneXplorer [Rees2004]	no	no	yes	freely available under MIT license	web application
TM4 [Saeed2003]	yes	yes	yes	free, open-source	software suite for microarray data, consists of 4 programs
Mayday [Dietzsch2006]	yes	yes	yes	GNU Public License	data mining plug-ins for R and Weka
jTreeview [Saldanha2004]	no	no	no	GNU GPL Version 2	visualization tool for hierarchical clustered data
GeneSpring GX	yes	yes	yes	commercial tool	high-end application
Spotfire	yes	yes	yes	commercial tool	business intelligence tool
GeneLinker	yes	yes	yes	commercial tool	integrates scripting tools

Table 2.3: *Comparable frameworks for visual gene expression data analysis.*

Chapter 3

Concept

The main objective of this work is to provide a framework for the interactive analysis of large gene expression data sets. The user should be able to analyze gene expression data sets and to extract as much information as possible from a given data set. Useful information in this context includes finding similarities and patterns which could lead to new insights about genes and their functions. To handle this the system should provide several cluster algorithms, methods to visualize the achieved cluster results, and methods to interoperate with data.

To enable the handling of large data sets we want to integrate a hierarchical approach into the existing heat map view following the focus+context paradigm.

3.1 Hierarchical Heat Map

Caleydo integrates two major views for the visualization of multi-dimensional data sets. The first is a parallel coordinates view which may used for numerical and for nominal data, the second is the heat map view which is especially suitable to gene expression data sets. This is because the data set, given in matrix form, can easily be mapped to color values which are visualized in the heat map view. Because of the limited number of data samples which can be visualized with the standard implementation of a heat map without using scrollbars, we come up with a hierarchical approach.

The idea of a hierarchical heat map is to use a three level hierarchy (see Figure 3.1), where the first level provides an overview of the whole data set by rendering all data samples. The second level visualizes only a subpart of the samples. The last level is limited to about 50 data samples. Level three provides information about each included gene and experiment by rendering captions next to the data vector. Additionally, each data sample may be selected (brushed) in the third level and all occurrences of the sample (in level one and two) are highlighted as well following the linking & brushing paradigm.

The three level approach is very useful for big data sets, but becomes irrelevant for smaller ones. Due to this we propose three different states. The first one is used for big data sets and renders all three levels as described above (see Figure 3.1). The second state is used for data sets of medium size (50 - 200 data samples). Hereby the first level is skipped and only level two and three are rendered. For small data sets (below 50 samples) only the third level is rendered. This three state approach provides the user with a good ratio between overview and detail for all types of data sets.

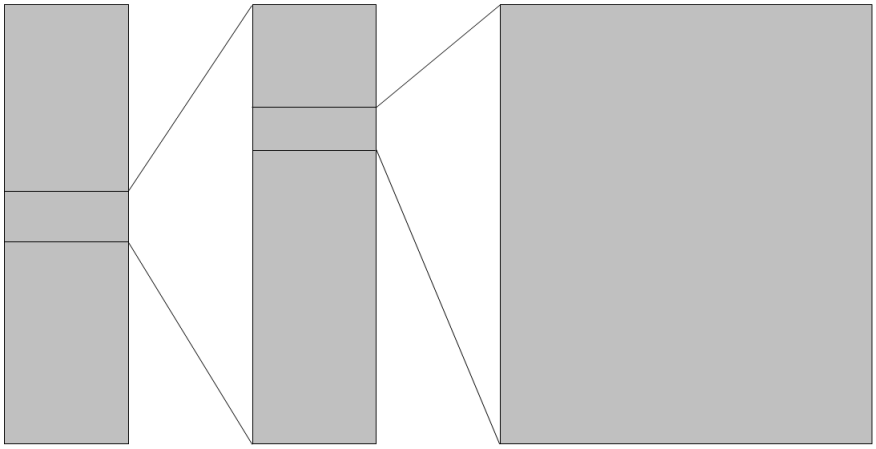


Figure 3.1: *Hierarchical heat map concept. The first level includes all available data samples. The second and the third level include only a limited number of data samples but therefore provide the user with the possibility to select distinct genes and/or experiments.*

As the number of experiments and with it the required horizontal space may also vary we propose a method to set level three in focus, which means that level two will be rendered smaller, which provides more screen space for level three. The user should be able to trigger the states manually.

3.2 Clustering

Clustering is one of several data mining techniques, comparable with sorting, normalization, or classification. Many researchers ([Eisen1998, Seo2002, Jiang2004]) have shown, that clustering is a good approach to gene expression data analysis. Therefore several cluster algorithms are going to be integrated into the framework.

During the cluster process data samples (gene or experiment vector) are grouped together corresponding to their expression values. Biologists assume that genes with similar gene expression values may be involved in the same processes inside a cell. Because of this a grouping of similarly expressed genes is essential in terms of data analysis.

An important point, which influences the cluster result immensely, is the choice of a feasible distance measure. A distance measure is a mathematical notation which computes the similarity of two data vectors against one another. We found four distance measures potentially useful for gene expression analysis. One type of these measurements focuses on finding vectors with the same patterns (euclidean, chebyshev, and manhattan distance), but they differ in their computational intensity and handling of outliers. Another kind of measurements (pearson correlation) builds a correlation value of two given data vectors. The correlation of two vectors does not directly focus on the values of each data vector but on the shape of two data samples.

3.3 Cluster Result Visualization

An essential aspect of this work is to visualize the cluster assignment in the hierarchical heat map view. To do this one has to take the outcome of the distinct cluster algorithms into consideration. As described in Section 2.1, two types of cluster algorithms are common. On the one hand, there is the partitional-based clusterer, which divides the data set into several groups by assigning each data sample a label. Since the order of resulting clusters produced by these algorithms is arbitrary the gene or experiment vectors have to be sorted to achieve a visually pleasing result. The visualization of such a cluster assignment is depicted in Figure 3.2.

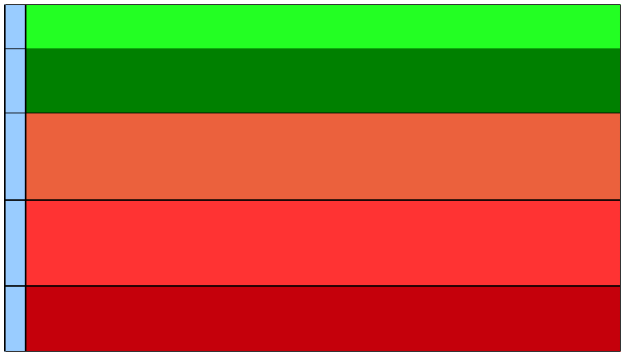


Figure 3.2: *Partitional cluster result visualization. The data set was clustered into five groups which are visualized with the blue bar on the left side of the heat map.*

On the other hand, a hierarchical clusterer does not divide the data set into independent groups but splits the data set hierarchically and builds up a tree structure. This tree structure contains a leaf node for each data vector (gene or experiment). The non-leaf nodes inside the tree represent the connections/similarities of the subjacent nodes. After sorting the data samples corresponding to their position in the tree, the result is comparable to those achieved with partitional clusterers. In addition, the tree structure obtained can be visualized to allow the user to draw conclusions from the similarities of the data samples. Another interesting feature discussed in Section 2.2 is the cutoff value. A cutoff value may be seen as a slider which divides the data set into clusters depending on their hierarchy level and on their position in the tree. Such a dendrogram visualization with a cutoff included may look like the example in Figure 3.3.

To visualize hierarchically clustered data sets we propose a stand-alone dendrogram view which can also be integrated into the hierarchical heat map. Because we support clustering of both dimensions (gene and/or experiments) the hierarchical heat maps should use up to two dendrograms to visualize the cluster information: one for genes, which is placed on the left side and one for experiments. The experiment dendrogram is placed on the top of level three in the three level hierarchical heat map. Gene dendrograms have to be rendered from left (root) to right (leaves) while dendrograms for experiments are rendered from top (root) to bottom (leaves).

As an additional feature the view should provide a cutoff bar, which allows the user to determine clusters according to the tree structure. By moving the cutoff bar inside the

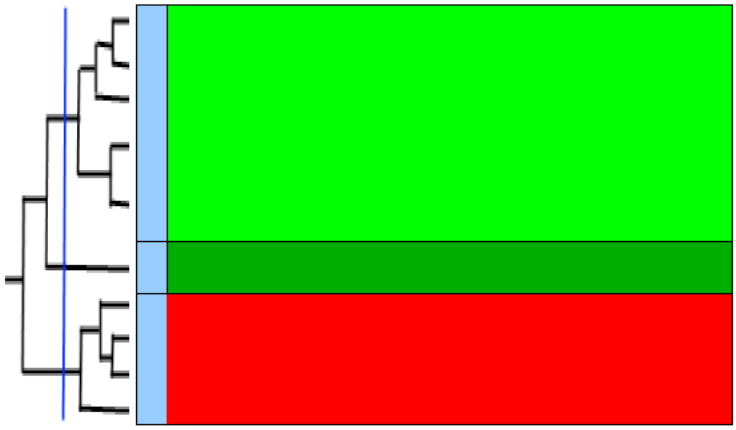


Figure 3.3: *Hierarchical cluster result visualization. The data set was clustered with a hierarchical clusterer and three clusters were determined with the cutoff value which are visualized with the blue bar on the left side of the heat map.*

dendrogram the number of clusters changes and the hierarchical heat map visualizes the newly generated cluster assignments.

Since the dendrogram needs a significant amount of screen space is reduces the area left for the hierarchical heat map. We conceived an approach for saving screen space. The attempt was to provide two states for rendering the dendrogram. The initial state renders the whole dendrogram from the root node down to the leaf nodes, including the cutoff bar. In this state the user is able to determine clusters with the cutoff bar. If the user afterwards wants to explore the gene expression data in detail, he can switch the renderstate for each dendrogram. In the second state only a subpart of the whole dendrogram will be rendered showing only the relations above the cutoff. The renderstate should be switched on user request.

Keeping in line with linking & brushing, the dendrogram view should be synchronized with the radial hierarchy view which is also used to visualize cluster results. This means that selections in one of the views should be highlighted in both views.

To provide the user better inside into hierarchically clustered data we propose to visualize the similarity information in each level of the hierarchical heat map. Hence, the user is able to easily analyze the dependencies of the elements visualized in level three (embedded heat map) without loosing the focus. The same is right for level two which also provides the similarity information for all included data samples.

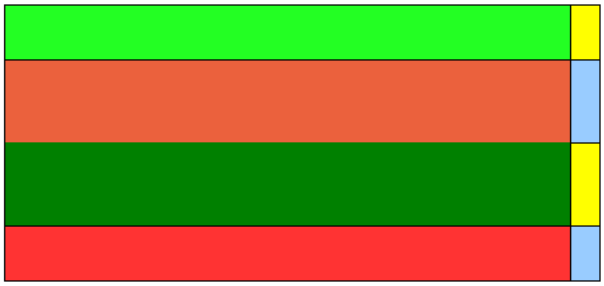
3.4 User Interaction

One important point, as discussed in Section 2.2 is the possibility to allow the user to interact with the cluster result. This is not handled well in previous approaches. Therefore we want the user to be able, for example, to reorder clusters determined by a partitional clusterer. Some of the operations included are splitting, interchanging, and the merging of clusters. All of this will be described briefly below.

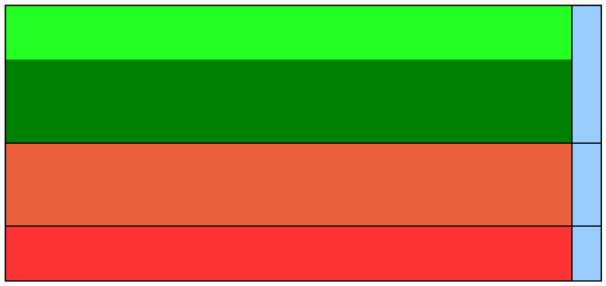
The interaction methods should be accessible via a context menu or by drag&drop, which will be available in each level of the hierarchical heat map.

Merging Cluster

Merging two or more clusters is one of the most intuitive operations. A user may not be satisfied with the result of a clusterer, but may also not want to rerun the algorithm. Thus the user may wish to influence the cluster result in such a way that clusters containing defined data samples are merged into one cluster. The interactions concerned with these operations are best managed with a context menu. This context menu should appear when a user right-clicks on the cluster visualization in the heat map. In Figure 3.4 (a) one can see a cluster visualization integrated into the heat map view, where two of four clusters are highlighted (yellow). The selection was made by clicking the corresponding quads next to the clusters. The result of the merge operation is shown in Figure 3.4 (b).



(a) Cluster visualization before merge.



(b) Cluster visualization after merge.

Figure 3.4: Illustration of a merge operation on a partitional cluster result. In (a) one can see that the first and the third cluster are selected (yellow). After the merge operation the clusters are merged and reordered as can be seen in (b).

Split Cluster

Splitting is an operation which may be useful in two cases. The first idea is to split a non-clustered data set into distinct groups/clusters. This may be interesting in cases where a data set contains different types of experiments, for example samples from healthy or diseased tissue. The second alternative is to resize the result from a clustering algorithm. In both cases the user should be able to select a part of the data set (several genes or experiments) by dragging the mouse cursor. After the mouse is released the group should

be split in two or three new subgroups. The amount of generated clusters depends on the position of the newly generated cluster inside the old one.

Interchange Cluster

Interchanging clusters/groups may be interesting when a user is not satisfied with a cluster result. Maybe she knows about special features of several experiments which could not be seen by the cluster algorithm. Then she should be able to manipulate the cluster result in terms of interchanging clusters/groups. This feature should be available for both dimensions (gene and/or experiment).

Reorder Cluster

Reordering clusters is a function similar to interchanging clusters. But there is one main difference. Reordering in this context means that the user is allowed to drag a cluster and to drop it at another position inside the data set.

Export Cluster

The user is able to select one or several clusters and can export all the data samples belonging to the chosen cluster to a .csv file. The operation is available for both dimensions (gene and experiments). If the user selects clusters for genes and for experiments only those data samples belonging to the clusters in both dimensions will be exported.

Chapter 4

Design and Implementation

In this chapter the design and implementation of the software developed in the curse of this work will be presented. The first Section 4.1 will describe the technologies used by the Caleydo visualization framework. After this, a few important facts about the framework, like the storage concept, the remote rendering approach, and the event system will be described in Section 4.2. The last Section, 4.3, provides information about design issues which have influenced the implementation of the project.

4.1 Used Technologies

Caleydo[1] is a visualization framework which focuses on genetic and clinical data visualization. It integrates views for pathway visualization and for gene expression data sets. Several state-of-the-art information visualization techniques are provided to support the user. These are linking&brushing, focus+context, visual links, and multi-view information visualization. The user may use pathways in the bucket arrangement, parallel coordinates, heat maps, glyphs or radial hierarchies as views. Several other views are in progress and will be added soon.

Java

The entire framework is written in Java[2]. Java is a high-level object-oriented programming language supported by most operating systems and used in a wide range of applications. The current version of Caleydo requires Sun Java 6.

Eclipse

Eclipse[3] is the development platform used to implement the framework. As Eclipse is under the terms of the Eclipse Public License it is freely available and therefore very popular in the academic world. Due to the fact that Eclipse may be extended by plug-ins it provides support for a broad band of functionalities and features. For example, plug-ins for using subversion[4] and bug-tracking tools like trac[5] may be included.

[1] www.caleydo.org
[2] java.sun.com
[3] eclipse.org
[4] http://subclipse.tigris.org/
[5] http://trac.edgewall.org/

RCP

One of the extensions mentioned above is the Eclipse Rich Client Platform[6]. It is a set of plug-ins needed to build a rich client application. RCP provides the possibility to write applications which are in the widest sense independent from operation systems and may be ported with little or no cost to other platforms. It includes libraries that manage things like windowing, tool-bars or preferences.

SWT

The Standard Widget Toolkit[7] is a part of RCP. It provides access to the native user-interface elements of the operating system. By abstracting the access it allows an implementation independent of the operating system.

Jogl

OpenGL[8] is used for rendering the views embedded in the visualization framework. OpenGL [Shreiner2005] is an application programming interface which provides many functions useable for rendering 2D and 3D graphics. Jogl[9] is used as a wrapper library. Jogl provides direct access to OpenGL functions inside a Java program.

Weka

Weka (Waikato Environment for Knowledge Analysis) [Witten2005] is a data mining software written in Java. It provides algorithms for the classification and clustering of multi-dimensional data sets. There are two ways to apply algorithms integrated in the Weka framework. The first is to start the Weka data mining framework [Witten2005] and directly apply algorithms on data sets. The second approach is to use the libraries provided, which is used in this work. The K-means and Cobweb algorithms, which are described in Section 2.1 are integrated this way.

4.2 Framework

The system architecture and the startup process of Caleydo are described in detail in [Streit2007] and [Lex2008]. Therefore this will not be included in this thesis. Instead, we will focus on parts of the framework with which this work interacts closely. Among others these are the view management, the storage concept, and the event system.

4.2.1 Storage Concept

Data handling in terms of loading and storage are described in detail in [Kalkusch2005] and [Lex2008]. Hence this is only a brief summary given in this thesis. We want to concentrate on several approaches which have been improved since the implementation by Michael

[6] http://wiki.eclipse.org/RCP
[7] http://eclipse.org/swt/
[8] http://www.opengl.org/
[9] https://jogl.dev.java.net/

Kalkusch.

All the views coordinate their data with `UseCases`. These are central classes shared by all views. In a `UseCase` all the information about corresponding data is available. Hence it is possible to provide loading of genetic or general data sources in Caleydo by switching the `Usecase` from genetic to general data.

`Sets` are very important as the next stage in the storage concept. A `Set` is a container for loaded data concerning, for example, a gene expression data file. A `Set` includes a number of `Storages`. The actual data is stored in such `Storages`. A `Storage` can handle different kinds of data like integer values or strings corresponding to the loaded data file. `Storages` use primitive arrays for performance reasons. `VirtualArrays` are used to get easy access to data inside `Storages`. `VirtualArrays` use highlevel collections and include indices which determine the order of accessing members inside a `Storage`. As a consequence `Storages` do not have to be modified during the runtime of Caleydo. They are filled at startup and all the other operations are done by manipulating of `VirtualArrays`. For example, the clustering changes the ordering of the `VirtualArray` and, after the cluster process, all the views with the same `UseCase` and `Set` use the newly ordered `VirtualArray`. Figure 4.1 illustrates the usage of `VirtualArrays` in combination with a cluster process.

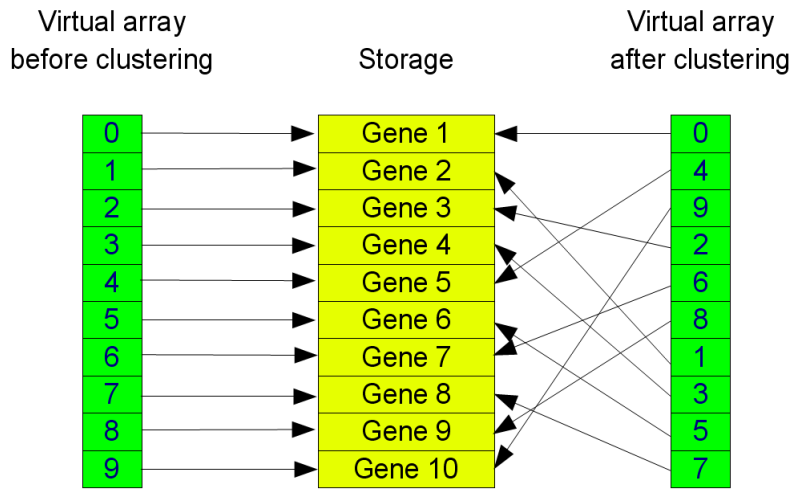


Figure 4.1: *Virtual arrays provide access to storages. On the left side a virtual array is shown with the initial values after start-up. On the right the virtual array is shown as it is stored after clustering. The storage containing the expression rates for the genes are not manipulated.*

4.2.2 View Management

Caleydo is a visualization framework permitting a software developer to easily implement new views. This is due to the software architecture used. The framework provides abstract classes (`AStorageBasedView`, `AGLEventListener`, etc.) which a newly invented view has to extend. These abstract classes provide methods which a view has to implement. Several of these methods are responsible both for the initialization of the data

and for rendering views. Additionally, the framework provides the possibility to remotely render views. This means that a view is able to handle another view remotely and to render this remote view inside its own frustum. This approach is used, for example, in the Bucket view where pathways, parallel coordinates, glyphs, or heat maps are rendered remotely.

Beside OpenGL views Caleydo is able to integrate SWT views. SWT views are used, for example, for an integrated webbrowser, which contains SWT widgets.

4.2.3 Event System

Events are triggered by an `EventPublisher` at any place in the framework. Classes that implement interfaces for `IListenerOwner` and which provide a concrete instance of an `AEventListener` to handle a specific event are able to react on the `Event`. For example, `UpdateViewEvent`, `SelectionUpdateEvent` or `ReplaceVirtualArrayEvent` are events served by the views. To receive a particular `Event` a class has to register its `Listener` with the `EventPublisher`.

Figure 4.2 illustrate the class structure responsible for the event system as an UML diagram.

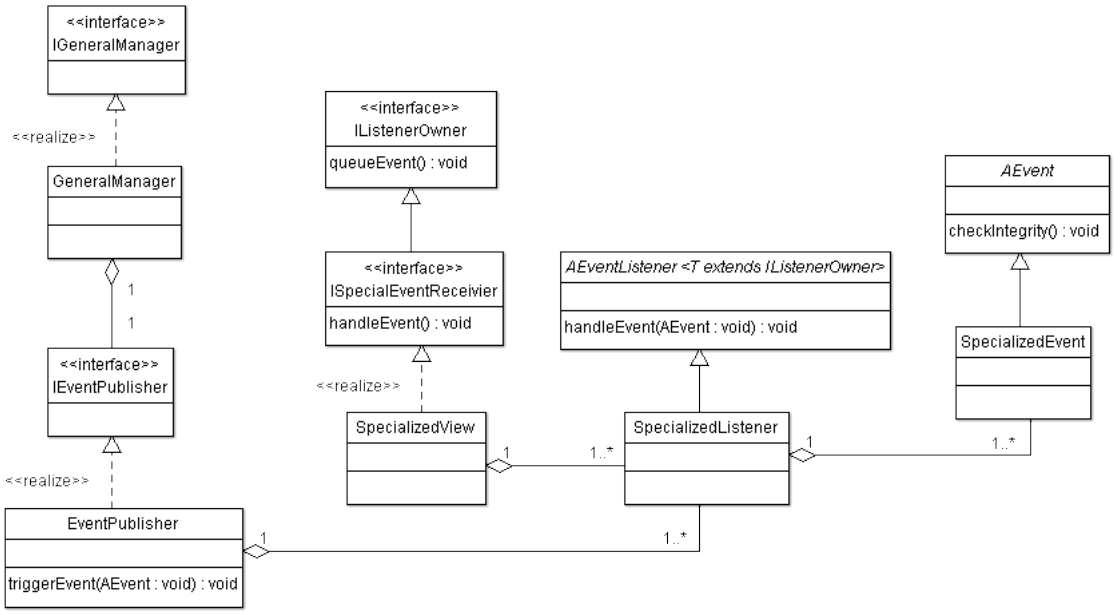


Figure 4.2: *Class structure responsible for the event system.*

4.3 Software Design

This section will cover the software design implemented during the thesis. The design consists of four major parts: clustering, cluster assignment handling, hierarchical heat map view and dendrogram view. The first Section 4.3.1 will cover the implementation of cluster algorithms and distance measurements. After this, the handling of cluster assignment information will be discussed in Section 4.3.2. Section 4.3.3 and 4.3.4 provide information about the visualization of clustering results.

4.3.1 Clustering Framework

The framework provides an interface called `IClusterer` which is located in the core part of Caleydo in the package `clusterer`. This class has several methods used to handle the cluster process. The most important one is `cluster()` which is called by the `ClusterManager` and is responsible for the main cluster process. It returns a `VirtualArray` with the new order of indexes. The interface is implemented by the abstract class `AClusterer`. Additionally, `AClusterer` is responsible for queueing events. Each implemented cluster algorithm extends `AClusterer` and provides the functionality needed for the respective process. The class diagram is shown in Figure 4.3.

The K-means and Cobweb algorithms are provided by the Weka framework[10]. The implementation for the tree clusterer based on the code by [Eisen1998][11]. The code for clustering by passing messages between data samples is based on [Frey2007] and is online available[12].

[10] http://www.cs.waikato.ac.nz/ ml/weka/
[11] http://rana.lbl.gov/eisen/
[12] http://www.psi.toronto.edu/

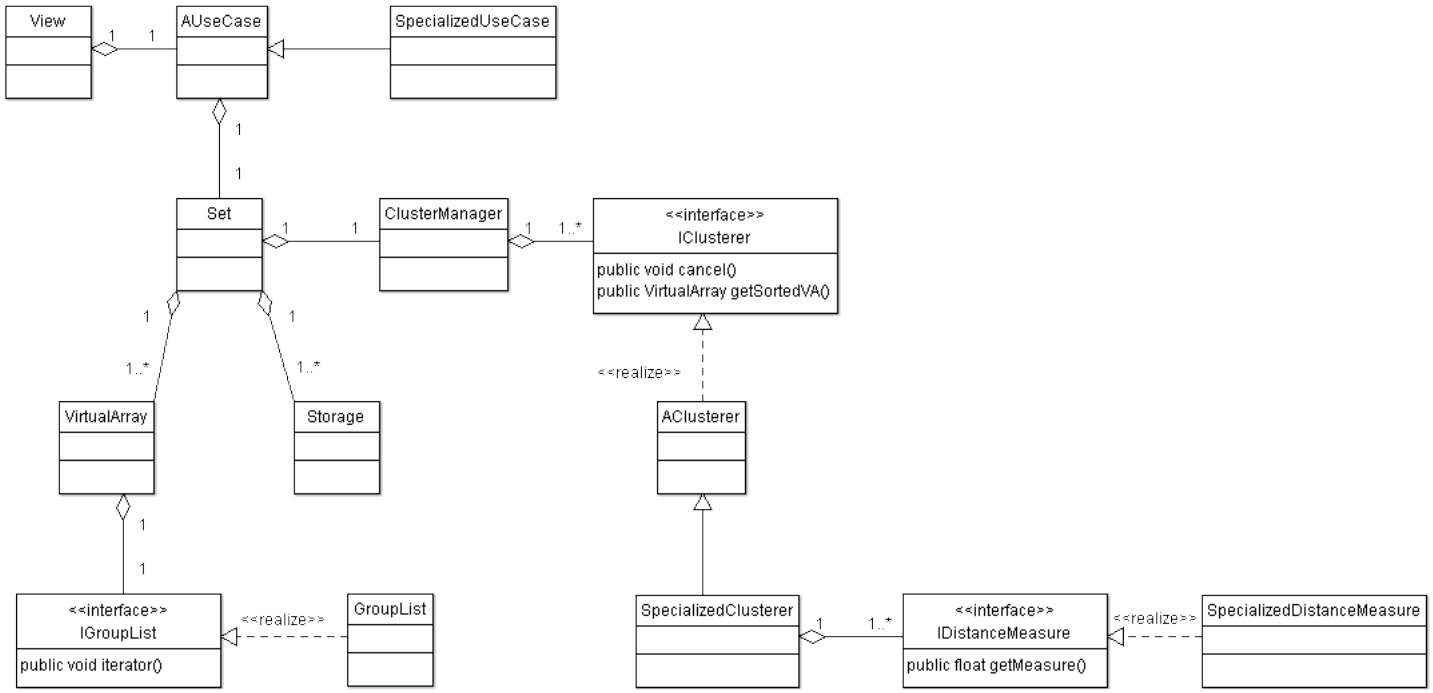


Figure 4.3: *Clusters responsible for clustering. The* `ClusterManager` *is responsible for the handling of the cluster process and updates the* `VirtualArrays` *in the* `Set`.

The cluster process is started on user request. In a pop-up window cluster parameters can be adjusted. There the user may choose a distinct cluster algorithm, a distance measure, and further options regarding the special algorithm. After adjusting the cluster features the user acknowledges his choice by clicking the OK button. Afterwards all the information needed is stored in a `ClusterState` class and a `StartClusterEvent` is triggered. The current available `UseCase` is responsible for taking care of the event and starts the clustering process. After the cluster algorithm returns the updated `VirtualArray` a `ReplaceVirtualArrayEvent` will be triggered and all the views will be updated according to the new cluster result. The process ends with the visualization of the cluster result. The states are illustrated in Figure 4.4.

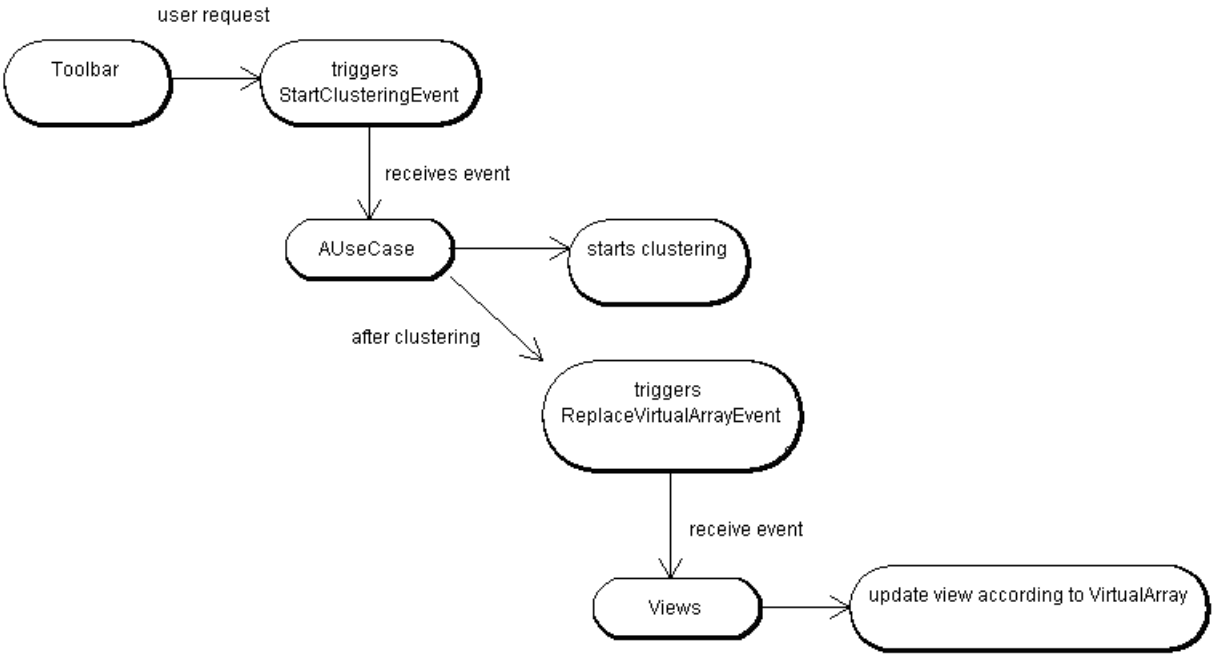


Figure 4.4: *State chart of a cluster process. It is triggered on user request. All the views will be updated after the cluster algorithm returns a new ordered virtual array.*

Distance Measurements

Each cluster algorithm requires a distance measurement to compute the similarities between two data samples. Because of the distinct requirements of various users, we implemented four kinds of distance measures. All distance measurements implement `IDistanceMeasure`. To extend the framework in terms of adding a new measurement one has to implement this interface.

4.3.2 Cluster Assignment Handling

After the clustering process we need to handle the information about the cluster assignments determined by the algorithm. To do this some enhancement in terms of storage

management to handle the cluster assignments for partition-based and hierarchical clustering were required.

We decided to implement a `GroupList` for partition-based clustering, which holds the separated `Groups` (clusters). A group in this context is a set of information needed to visualize and store the cluster assignments. The dependencies between a `VirtualArray` and the `GroupList` are illustrated in Figure 4.3. An example `GroupList` is depicted in Figure 4.5.

# elements	selection type	collapsed	index example
56	NORMAL	FALSE	44
23	SELECTED	FALSE	68
234	NORMAL	TRUE	299
9	MOUSE_OVER	FALSE	315
..	...	...	...

Figure 4.5: *Group list holds information about groups/clusters.*

Further, we needed to implement a class holding the tree structure obtained by a hierarchical clustering algorithm. This was achieved by implementing a `Tree` class which holds a `Graph` (the cluster result) and provides methods for traversing and restructuring of the tree. The restructuring methods like addChildren() or comparable ones are needed to provide the possibility of user interactions.

The framework provides a couple of very useful export functionalities. For example, it is possible to save the whole project state to an archive [Puff2009] which may be used later on or may be loaded from another user (provided the projects use the same version of Caleydo). Additionally, the user is allowed to store the clustered, filtered, and reordered gene expression data to a specified file (.csv), which may be used in other programs or may also be loaded again into the framework at a later date. We implemented a "export groups" functionality as an extension to the given framework. Hence it is possible to select several groups/clusters of genes and/or experiments and store them in a file.

4.3.3 Hierarchical Heat Map

The previously implemented heat map view [Lex2008] was enhanced to cope with large gene expression data sets and information about cluster assignments. This was achieved by integrating a three level hierarchy, with each level providing cluster information (see Section 2.2).

To manage big data sets it is infeasible to render each data point (one gene expression value for a specific experiment/patient) to a single OpenGL primitive. This approach is used in the existing heat map and works fine for a limited number of data samples. Hence the standard heat map is used in the Bucket view of Caleydo, where only genes which occur in currently loaded pathways are shown. To also provide good performance for large data sets we need a new approach to handle a couple of data samples as a single primitive.

This is achieved by building textures. The performance and the usability for large data sets is improved using texture mapping [Hearn2003] in the first and second levels of the hierarchical heat map view (see Figure 4.6). The (re-)building of the textures must be done at start up, when a cluster algorithm reorders the `VirtualArray`, or when a user filters uninteresting data samples in the parallel coordinate view.

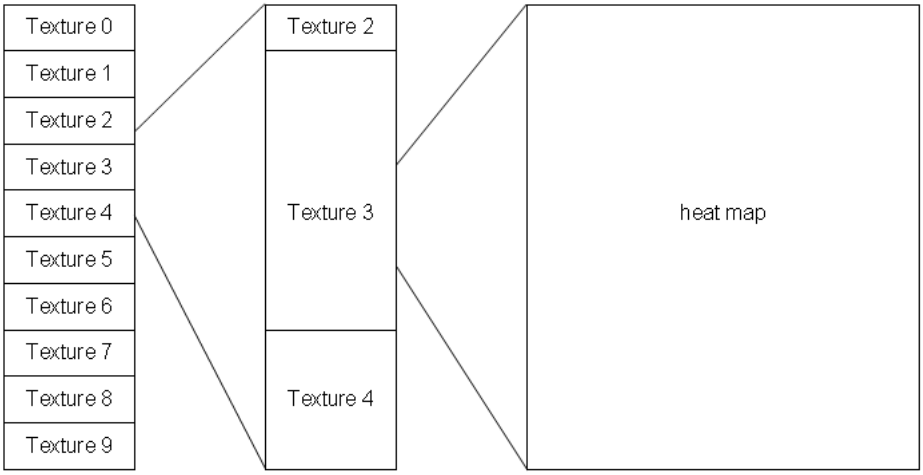


Figure 4.6: *Textures are used to render level one and two in the hierarchical heat map view.*

In the first level, all the textures are rendered one after another without any spacing or restriction. Due the fact that the number of samples in level two is adjustable and may be varied from 200 to 400 data samples with dragging cursors, we need to render a variable number of textures. Level two in Figure 4.6 is built up with three textures. Texture 3 is rendered entirely while texture 2 and 4 are truncated.

The third level of the hierarchical heat map is implemented as a remotely rendered view - the existing heat map view is used. This view provides the ability to select distinct data samples, renders captions, and visualizes selected genes/experiments.

An additional bar is rendered next to each level to visualize results from a partition-based clusterer. This cluster assignment bar visualizes the cluster borders and provides the possibility to select distinct clusters and to gain access to the context menu. With the help of either the context menu or drag&drop the user is able to interact with the cluster result. For example, one can export selected clusters, merge clusters, or load pathways which include genes integrated in the cluster.

The `GLDendrogram` view (for details see Section 2.2.4) is responsible for drawing the tree structure. The dendrogram view provides remote rendering functionality as other views also do. Because the data set can be clustered in both dimensions it is necessary that the `GLHierarchicalHeatMap` renders two dendrograms remotely (see Figure 4.7). One is for genes, which is rendered on the left side, and one for experiments which is rendered on the top of level three.

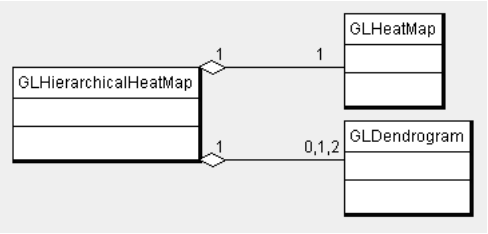


Figure 4.7: *Class diagram for remotely rendered views.* `GLHierarchicalHeatMap` *renders one* `GLHeatMap` *and up to two* `GLDendrogram` *views remotely.*

4.3.4 Dendrogram View

One part of the work during the thesis was to implement a new view to visualize the hierarchical cluster results in a manner most suitable for the user to recognize the dependencies between data samples. As described in the related work section in Chapter 2, the dendrogram is a good choice to handle this task. If there is a clustered tree stored in the `Set` (included in the `UseCase`), the view renders the tree structure inside the given frustum. The dendrogram view supports remote rendering, which is used in the hierarchical heat map view.

Chapter 5

Results

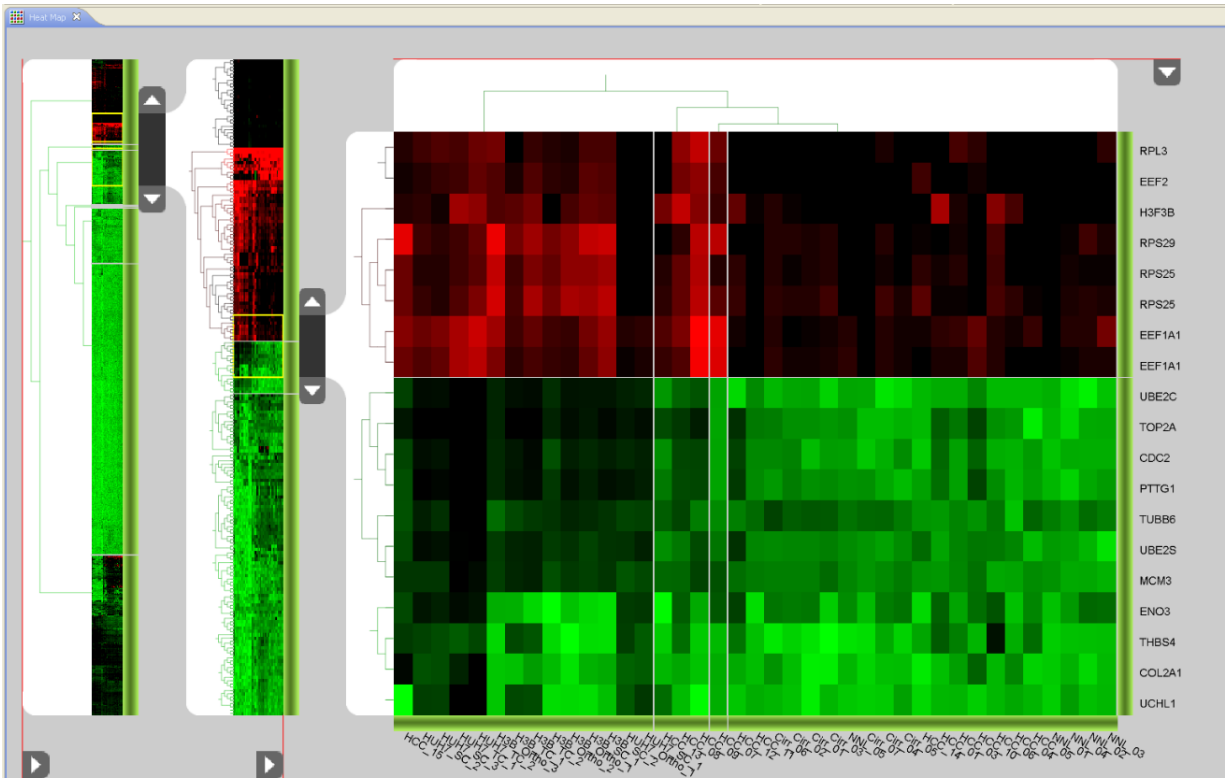


Figure 5.1: *Hierarchical heat map visualizing a data set with about 1800 genes and 41 experiments in a three level hierarchy. The data set was clustered with a hierarchical cluster algorithm and the cluster result (dendrogram) is visualized in both dimensions. Additionally one can see clusters determined by the cutoff value which are visualized with the green bars at the right side of each level and at the bottom of level three.*

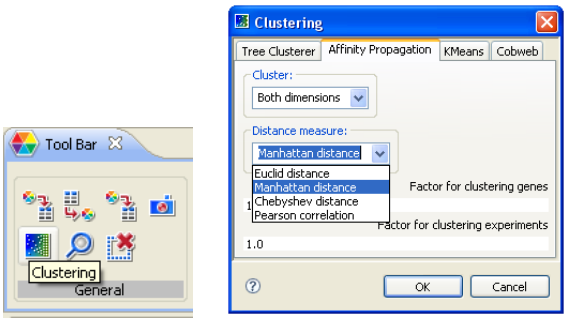
The three level hierarchy heat map view shown in Figure 5.1 enables the user to explore large gene expression data sets. This solves a significant drawback of standard heat map implementations. Because of the three level hierarchy one can visualize big data sets with up to 10000 samples without any problems in terms of performance or user interaction.

One essential idea was to integrate a group approach into the existing framework. This means that the framework is able to handle several data samples as a single group. Handling several elements as a group is necessary, especially for the integration of cluster

algorithms.

The clustering process is fully integrated in the Caleydo framework. Therefore it is available via a button in the general tool bar. By clicking on the tool bar button (shown in Figure 5.2 (a)) a dialog appears and the user is asked to choose one of the given cluster algorithms, (see Figure 5.2 (b)). Furthermore, the user can choose between clustering the data set in gene, experiment, or in both dimensions. Depending on the algorithm, the user has additional possibilities to influence the cluster result. For example, one can select one out of four distance measurements. If the user wants to cluster with the tree cluster algorithm he can choose either complete, average, or single linkage.

To support the user we provide a help button (see Figure 5.2 (b) in the cluster dialog). By clicking the button the integrated browser will be openend and the help section for clustering on the Caleydo homepage is shown. The user may then retrieve further information about the cluster algorithms available.



(a) General Toolbar.

(b) Cluster Dialog.

Figure 5.2: General toolbar and cluster dialog. (a) By clicking on the toolbar button for clustering a new dialog for clustering appears. (b) The cluster dialog is shown where a user can choose an algorithm and is able to modify several settings according to the clusterer chosen.

The visualization of cluster assignments is fully integrated in the heat map view. Due the nature of the different cluster algorithms there is the necessity to implement two different approaches for partition-based and hierarchical clustered data sets.

As mentioned in Chapter 3 our requirements are focused on the integration of user interactions in the cluster analysis stage. This is an essential feature in terms of the visual analysis of any data sets. Therefore we implemented a couple of operations with which a user is able to manipulate and restructure the results from cluster processes. The most important actions are available through a context menu, which can be seen in Figure 5.3. One interesting feature provided by the context menu is loading pathways, which contain genes included in the current selected cluster, into the Bucket. Reordering of groups may be done using drag&drop. This means that a user can drag a group and drop it at another position inside the data set. This feature is only available for partition-based cluster results, because a reordering of data samples would destroy the tree structure determined by a hierarchical clusterer.

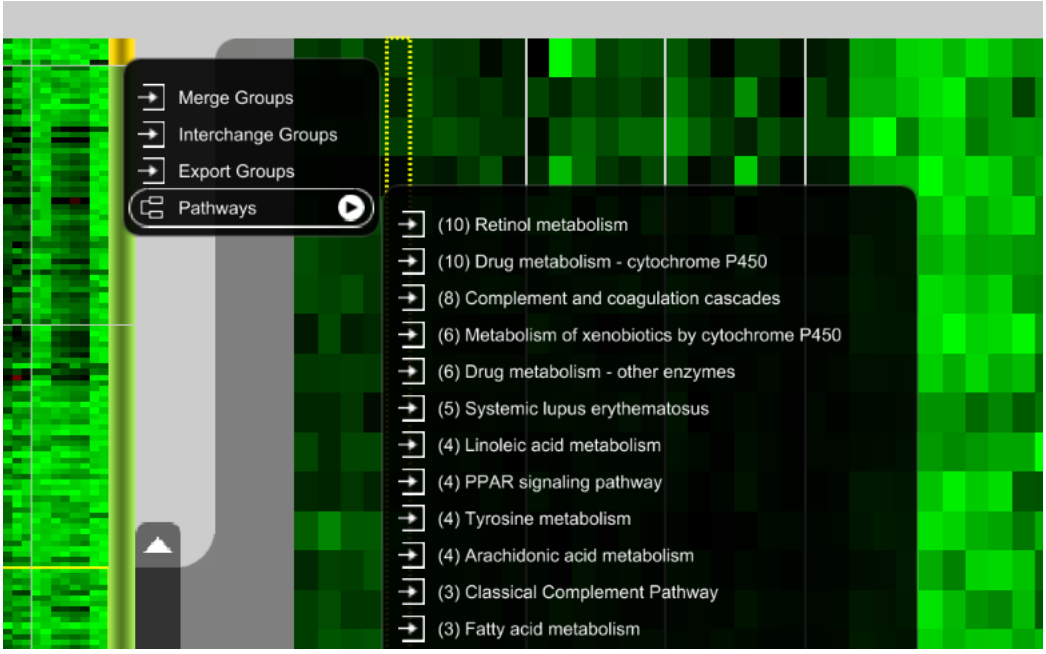


Figure 5.3: *Context menu for the usage of interactions according to cluster manipulation. An operation provided by the conext menu is to load pathways, which contain genes included in the current selected cluster, into the Bucket.*

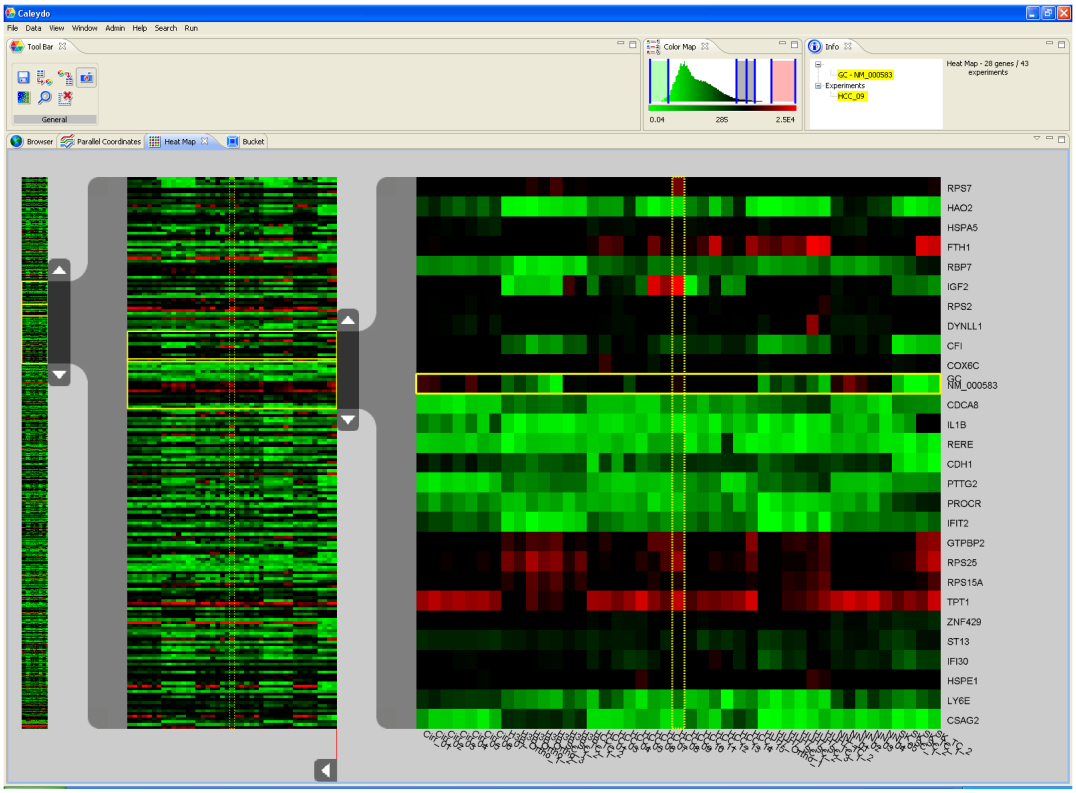
5.1 Hierarchical Heat Map in Detail

The existing heat map view was enhanced both in terms of visualizing big data sets and in terms of visualizing clustered data sets. This was achieved by integrating a three level approach, as described in Section 3.1. The hierarchical approach is very helpful when visualizing clustered data. Figure 5.4 (a) shows the newly implemented hierarchical heat map view visualizing a data set with 1,341 genes. Figure 5.4 (b) shows the same data set after clustering in both dimensions.

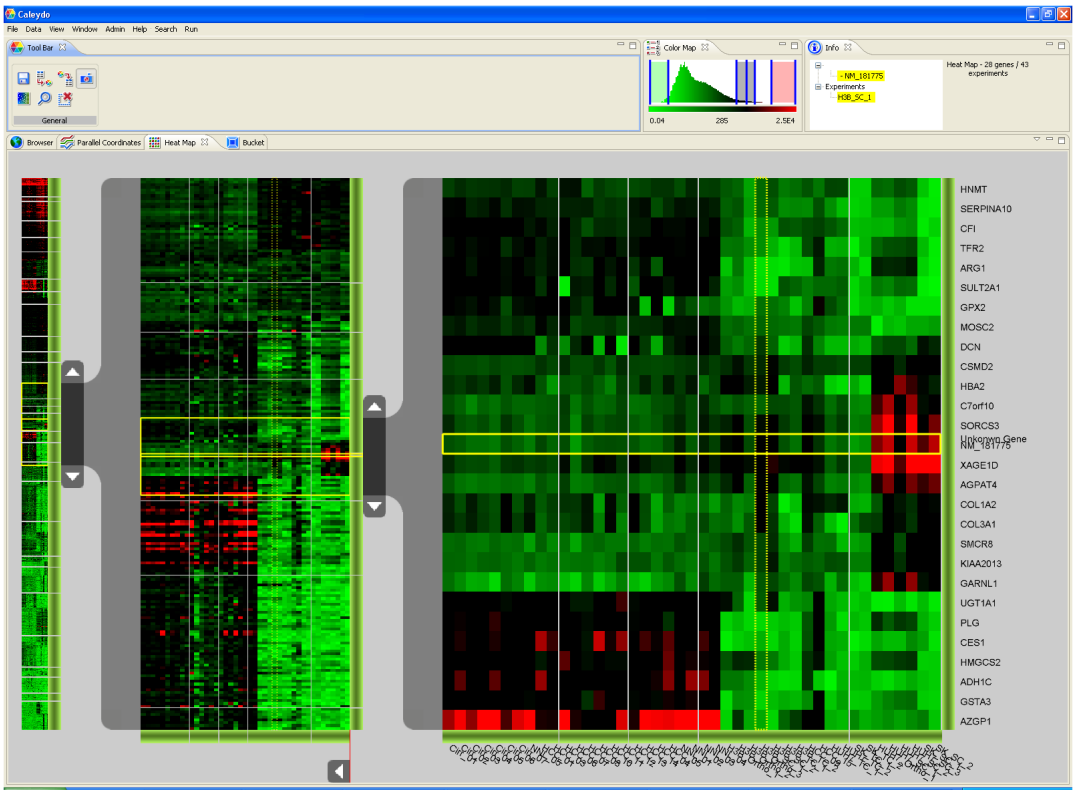
We built the hierarchical heat map view to be adaptable in order to support any kind of data set sizes . This means the number of levels used in the view depends on the number of data samples in the given file. Additionally, the view reacts to filter actions determined in the parallel coordinate view and, should the situation arise, updates the number of levels. In figure 5.5 one can see the hierarchical heat map view with (a) three, (b) two, and (c) only one level active.

Additional bars, which represent the clusters, are rendered on the right side of each level in the heat map. The groups are seperated by lines which are continued through the heat map. In Figure 5.4 (b) one can see a clustered data set with about 1,300 genes and 39 experiments which was clustered in both dimensions using a partition-based algorithm.

The visualization of the tree structure is accomplished with dendrogram views, which are rendered remotely. After a cluster process the dendrogram views appear inside the heat map view.

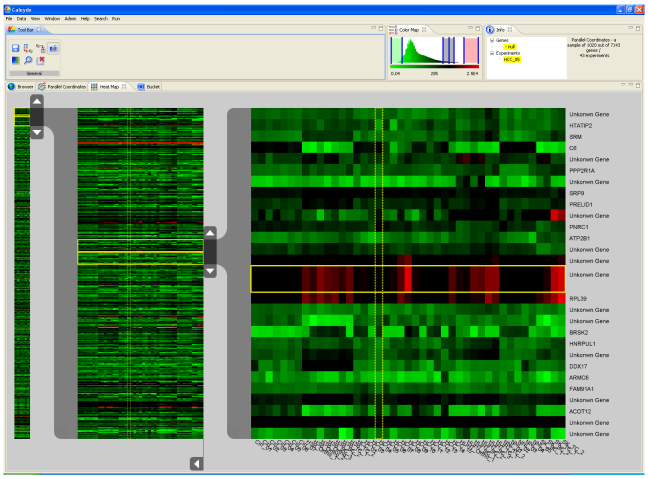


(a) Hierarchical heat map view with an unclustered data set.

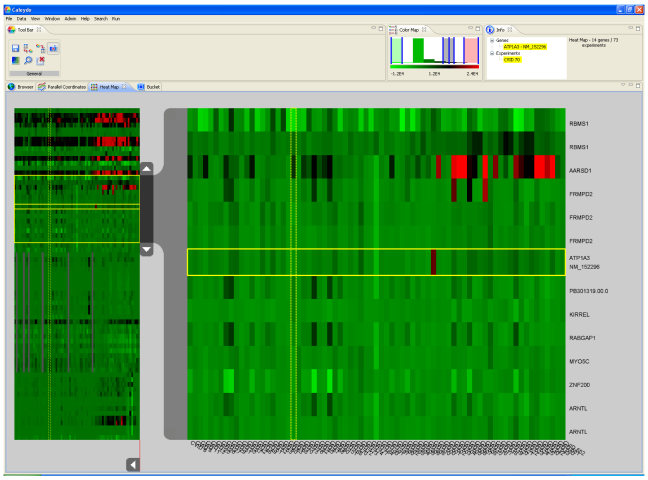


(b) Hierarchical heat map view visualizing the data set clustered in both dimensions.

Figure 5.4: A data set with 1,341 genes and 39 experiments. In (a) the data set is visualized with the newly implemented hierarchical approach, and in (b) the same data set is shown after clustering in both dimensions.



(a) Hierarchical heat map view visualizing about 7,000 genes in 3 levels.



(b) Hierarchical heat map view visualizing 69 genes in 2 levels.

(c) Hierarchical heat map view visualizing 29 genes in 1 level.

Figure 5.5: Hierarchical heat map view with 3 different data sets showing the different number of rendered levels.

If both dimensions are clustered hierarchically, two embedded dendrogram views are available (see Figure 5.6). With the cut integrated in the dendrogram the user is able to set a number of clusters, depending on the position of the cut. Such a use case is shown in Figure 5.6. The cut is visualized with a line which can by dragged and moved horizontally (in the case of genes) or vertically (in the case of experiments). By clicking on the buttons beside the dendrogram, the tree is only rendered up to the cutoff value, as can be seen in Figure 5.1.

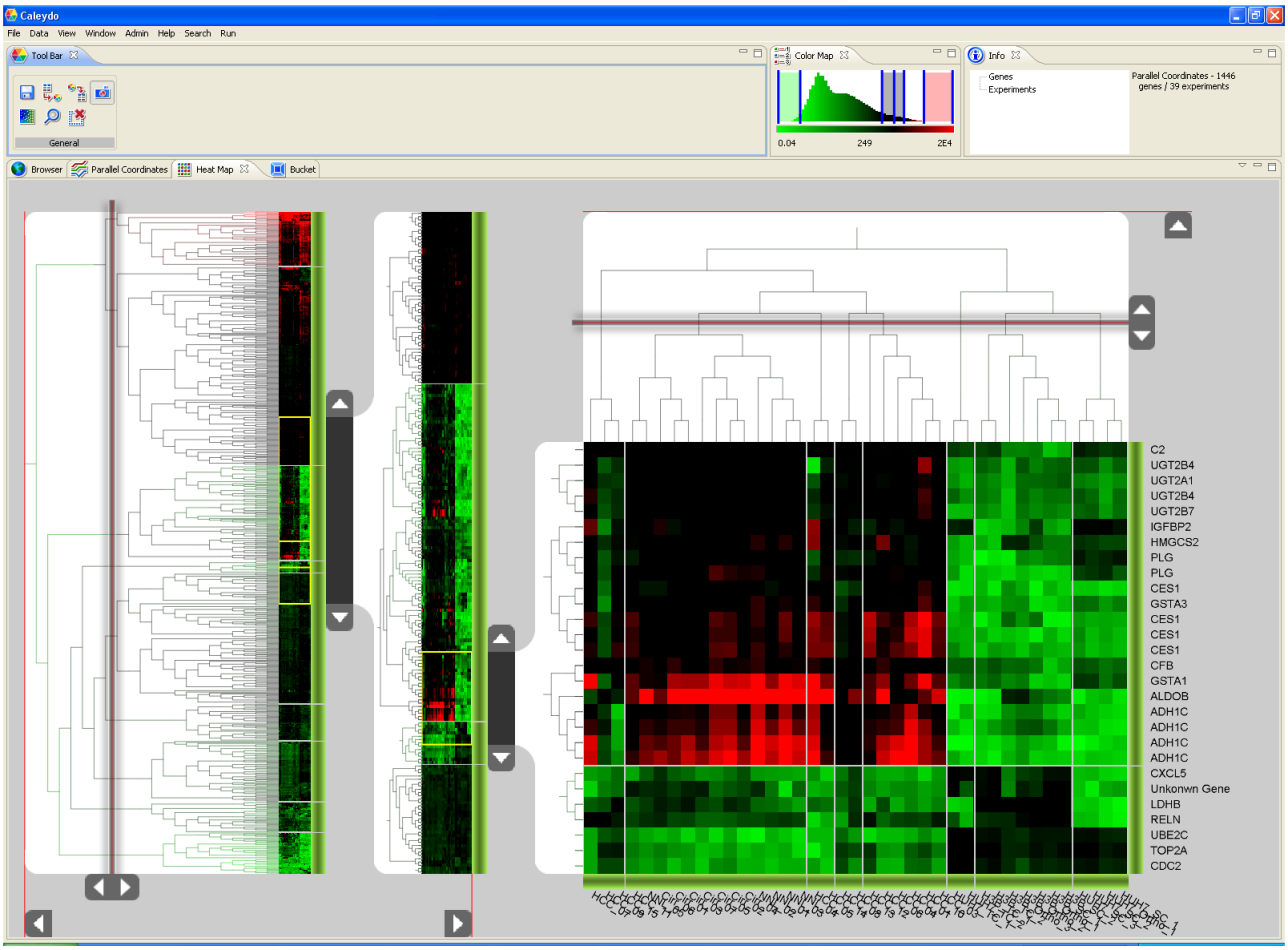


Figure 5.6: Hierarchical cluster visualization of a data set with about 700 genes and 39 experiments.

Additionally, the activation of the cutoff value is also controllable by key events. The dendrogram for clustered gene data appears or fades out by pressing 'g'. The dendrogram corresponding to experiment data is managed by pressing 'e'.

As a significant feature the dependencies of data samples in level two and three are visualized by rendering sub-dendrograms next to the levels. This can be seen in figures 5.6. This feature provides information about similarities of data samples in each level of the three level hierarchical heat map approach. Without this approach a user has to search for this information in the big dendrogram next to the overview bar. Especially for large data sets this would be problematic, because of visual clutter.

5.2 Linked Cluster Visualization

Caleydo integrates two views for exploring a tree structure: dendrogram and radial hierarchy. The views exchange information about selections using the newly implemented event synchronizing the views. In Figure 5.7 one can see a screenshot of Caleydo where the hierarchical heat map and the radial hierarchy view are shown. By selecting a cluster in the radial hierarchy view the same cluster is highlighted in the remotely rendered dendrogram, and vice versa.

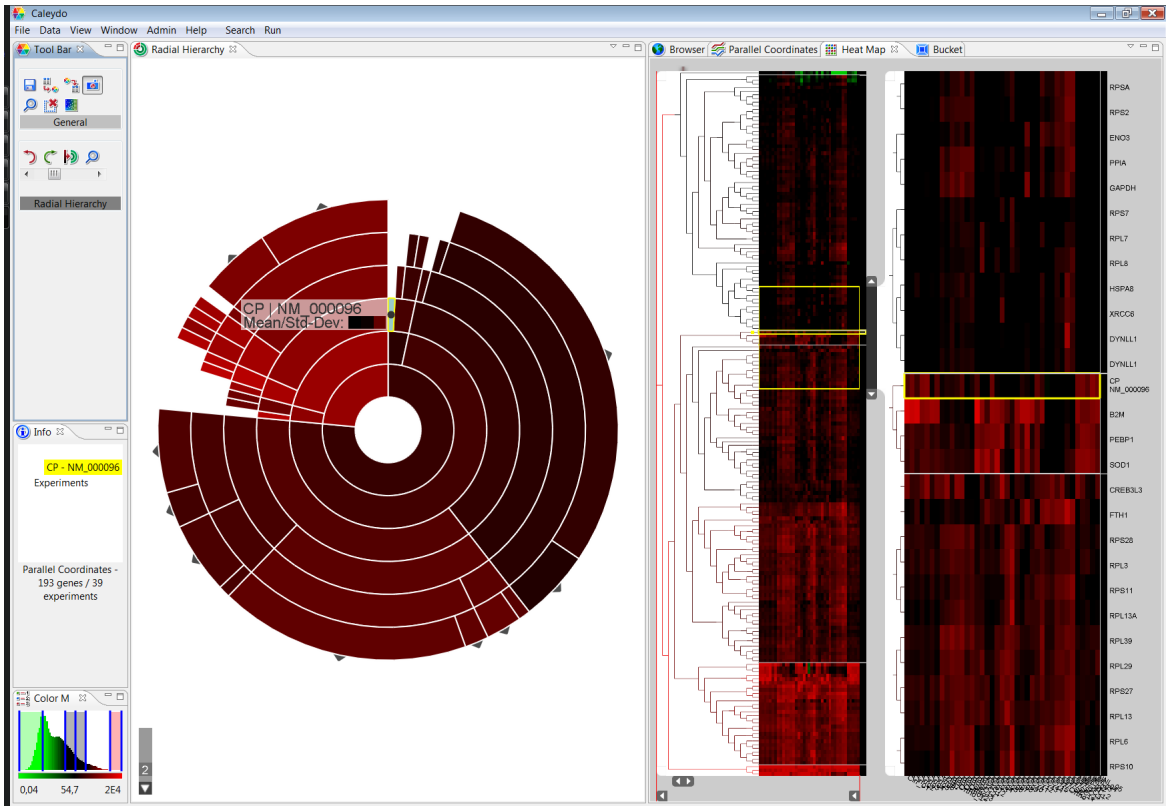


Figure 5.7: *Tree visualization in Caleydo. The dendrogram and the radial hierarchy view are synchronized. Selections in one view are visualized in both views and vice versa.*

The implementation of the radial hierarchy view was part of the work of another group member [Partl2009] and hence will not be described in detail in this document.

5.3 Bucket

The previously integrated standard heat map [Lex2008] is still used in the Bucket [Streit2007]. Hence the amount of data samples and the available screen space is limited. As an additional feature, the data samples visualized in the heat map view are clustered on the fly. This means that as soon as new genes are added to the view (by loading new pathways)

the data samples included are clustered. A screenshot of the Bucket, including a heat map visualizing a clustered set of genes, is shown in Figure 5.8.

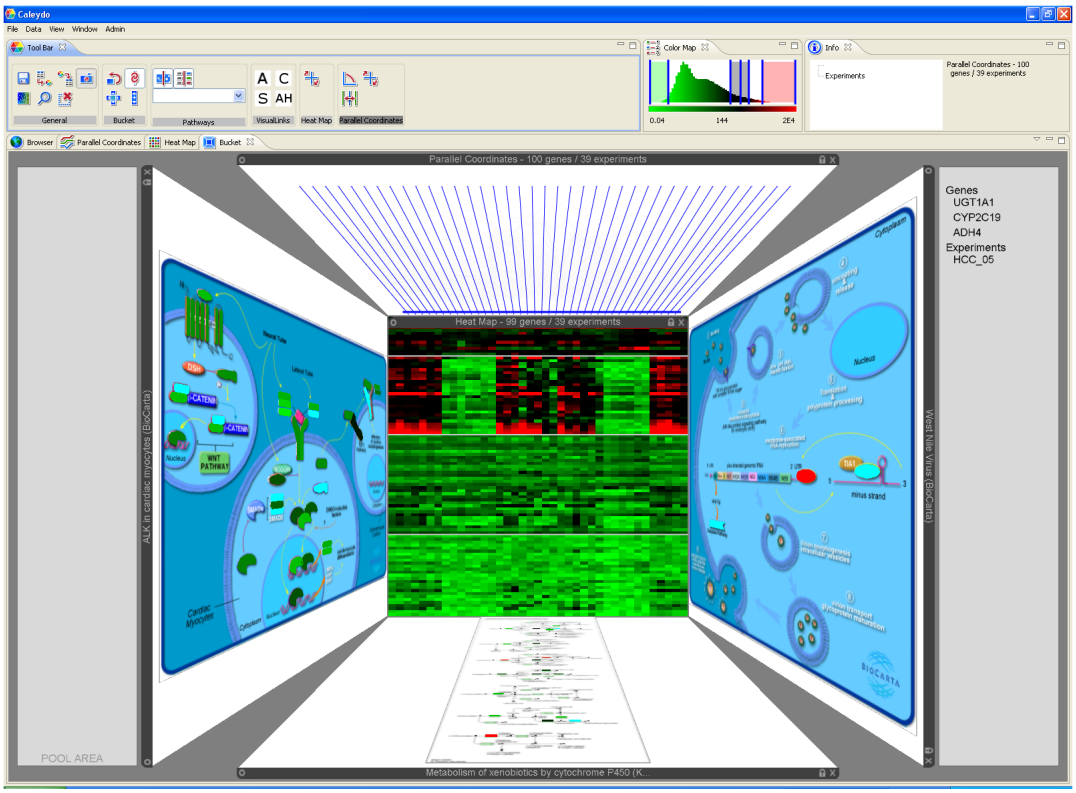


Figure 5.8: *Bucket view with three loaded pathways. The data samples inside the heat map are clustered.*

5.4 Use Cases: Gene Expression Centric Analysis

In this section we want to demonstrate a use case for the developed application. The use case is a gene expression centric analysis where the user wants to gather information about genes and pathways which may have an impact on diseases. The use case is described in figure 5.9.

An user (e.g. a biologist) loads a set of experiments. The experiments may include several samples of healthy and several samples of diseased tissue. During the analysis the user wants to collect information about which pathways, including specially expressed genes, may play a role in a specific disease.

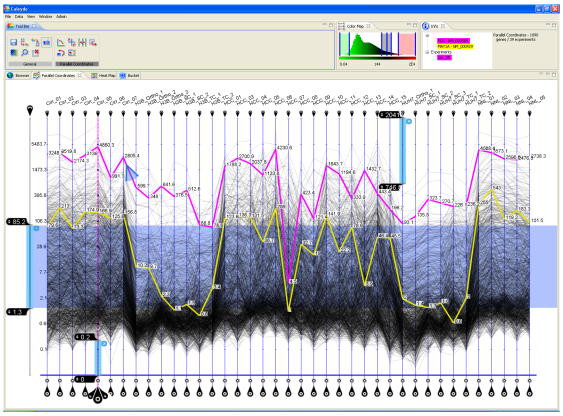
After loading the data set the user filters uninteresting data samples in the parallel coordinate view. This is done by using different types of filters, e.g., an angular brush. By saving the selected samples (an action available in the tool bar), all the other views receive the reduced number of samples.

To order the genes according to their expression value one starts a cluster algorithm. Because of the non-existent prior knowledge about a feasible cluster number the user will potentially not choose an algorithm for which he needs to provide the number of clusters. Instead, the expert will choose affinity propagation. This algorithm determines the number of cluster automatically. Alternatively he could use a hierarchical clusterer. Because of the integrated cutoff value the user may define the granularity and the number of clusters by himself.

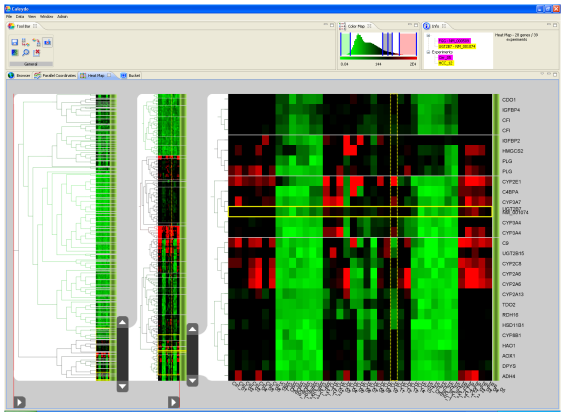
By checking the cluster result in the hierarchical heat map view the user detects an interesting cluster whose genes display great difference between the distinct experiments. The user is able to load pathways with the help of the context menu which include genes of a specific cluster, into the Bucket. During the exploration of several pathways he may find one or more pathways which include several genes from the previous cluster. As a consequence of looking more closely at specific genes, the user detects that the genes included in the pathway and in the cluster belong to a specific family.

The user may check biological databases (like PubMed[1]) online with the help of the integrated browser. During this investigation the user may find out that the gene family which falls in a cluster is responsible for a specific function in a biological process inside a cell.

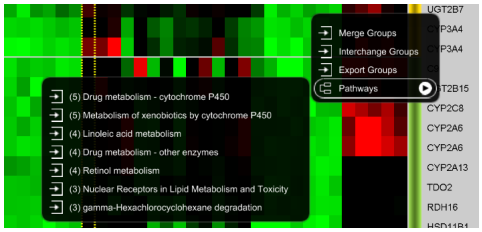
[1] http://www.ncbi.nlm.nih.gov/pubmed/



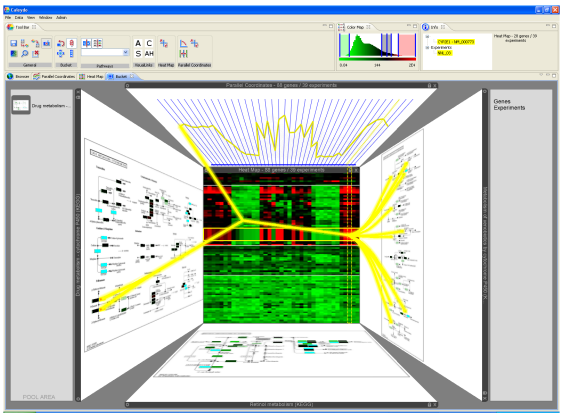
(a) Parallel Coordinates



(b) Heatmap



(c) Context Menu



(d) Bucket

Figure 5.9: Gene expression centric analysis. After loading a data set with experiments containing healthy and diseased tissue the user filters out uninteresting samples with brushes in the parallel coordinates view (a) and starts a cluster algorithm. Because the expert has limited pre-knowledge a hierarchical clusterer is chosen and a couple of clusters are determined by the use of the integrated cutoff bar (b). With the help of the context menu (c) the user loads pathways containing genes of an intersting cluster inside the Bucket (d). After further investigation, and with the help of online databases, the user finds out that the CYP gene group is a known catalyst for reactions in the drug metabolism.

Chapter 6

Conclusions and Future Work

The algorithms chosen for gene expression data clustering are tree clustering, Cobweb, affinity propagation and K-means clustering. The first two clusters algorithms are hierarchical clusterers which yield a dendrogram (tree structure). The other two algorithms are partition-based clusterers. These four cluster algorithm seem to be a good choice in terms of covering a broad field and, additionally, are not too computationally intensive.

The main focus of this thesis was to provide a framework for visual analytics for large gene expression data in combination with pathway visualization. The pathway visualization part was previously integrated by [Streit2007]. To handle large data sets (up to 10,000 genes and up to 50-100 experiments) we integrated the cluster visualization into a three level hierarchy. This means that the cluster border information is available in each of the three hierarchy levels. The first (leftmost) level visualizes the whole data set and builds a kind of overview bar. The second and the third level visualizes smaller parts of the data set. In the rightmost level less than 50 genes are included. Hence it is possible to identify each distinct gene and retrieve information about a specific gene. This is due to a caption rendered for each gene and each experiment and a context menu providing additional actions like loading pathways.

As mentioned above, the cluster border (assignment) visualization is integrated into a three level hierarchy in the newly implemented heat map view. This is true for partition-based clustering as well as for hierarchical clustering results. Therefore we provide information about the cluster assignment of a data sample in each level and allow the user to gather information about clusters in the overview (level one) as well as in the detailed view (level three). If the number of samples falls under a given threshold, the first or the first and the second level are suppressed. This ensures that the screen space is used effectively.

The work implemented in this thesis is published in [Mueller2009]. The paper provides an overview of the workflow of the Caleydo framework.

Future Work

As an additional feature one can implement further cluster algorithms. This should be relatively easy due the modality of the implemented work. Some conceivable algorithms could be, for example, fuzzy c-means clustering ([Gasch2002]). Fuzzy cluster algorithms in general do not provide a strict assignment to only one cluster. Instead, such algorithms return probabilities for a data sample to belong to one or more clusters. If fuzzy cluster algorithms will be added to the framework new methods to visualize the clus-

ter result will be needed. The current implementation is not intended for such information.

Further implementations of data mining methods, like Self-Organizing Maps [Yano2003, Nikkilae2002], Minimum Spanning Trees [Speer2003], Continuous Markov Models [Marshall2005], etc., for gene expression data sets should be surveyed. If one of these approaches achieves very good results for biological data sets, especially gene expression data, these algorithms should be implemented.

To speed up the computation and to allow the analysis of even larger data sets several approaches taking advantage of parallelism [Du2005] or using graphics hardware [Zhang2006] have been introduced. Such methods may be integrated in the framework as an improvement.

Some users may not be satisfied with the distance measurements chosen. Therefore, the necessity may arise to implement further distance measures with special features.

Additional views are required for the visualization of cluster results. A possible view for tree structures (hierarchical cluster result) would be the treemap approach. For the visualization of partition-based cluster visualization the parallel coordinate view may be extended or an additional view like scatterplots may be integrated in the Caleydo framework.

An everlasting problem in terms of cluster algorithms is the validation of the results. Due the nature of this problem there is, in general, no optimal solution. This is a highly complex issue especially for biological data. One approach to validate the significance of cluster assignments may be a manual test with a BLAST (Basic Local Alignment Search Tool) program. Blast is an algorithm for comparing biological sequences, like amino-acid or nucleotide sequences. There are several online tools which provide different Blast algorithms, for example NCBI[1], EMBL-BI[2], or Genome-Net[3]. With one of these tools a user may compare several genes assigned to a specific cluster. If parts of the sequence of genes are highly correlated one can assume that the cluster result is useful. As an extension Caleydo may be enhanced to automatically validate cluster assignments with the help of BLAST algorithms. The degree of correspondence returned by the online tool may be visualized next to the clusters.

The improvement for the visualization and handling of group/clusters may be a further interesting point for enhancing the framework. The visualization of various features of groups, like differences or similarities between groups, would be an important point.

[1] http://blast.ncbi.nlm.nih.gov/Blast.cgi
[2] http://www.ebi.ac.uk/Tools/blast2/index.html
[3] http://blast.genome.jp/

List of Figures

List of Tables

Bibliography

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

[Alberts2002] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell, Fourth Edition.* Garland Science, Taylor & Francis Group, New Yorks, fourth edition, **2002**. ISBN 0815332181.

[Andrews1998] Keith Andrews and Helmut Heidegger. *Information slices: visualising and exploring large hierarchies using cascading, semicircular discs.* In *InfoVis'98: Proc. IEEE Information Visualization Symposium, Carolina, USA*, pp. 9–12. **1998**.

[Aris2007] Aleks Aris and Ben Shneiderman. *Designing semantic substrates for visual network exploration. Information Visualization*, volume 6(4):pp. 281–300, **2007**. ISSN 1473-8716.

[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*, pp. 110–119. ACM Press, New York, NY, USA, **2000**.

[Ball1996] Thomas Ball and Stephen G. Eick. *Software visualization in the large. Computer*, volume 29(4):pp. 33–43, **1996**. ISSN 0018-9162.

[Berkhin2002] P. Berkhin. *A survey of clustering data mining techniques.* pp. 25–71. **2006**.

[Bezdek1981] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, **1981**. ISBN 0306406713.

[Bier1993] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. *Toolglass and magic lenses: the see-through interface.* In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 73–80. ACM Press, New York, NY, USA, **1993**.

[Cheng2000] Yizong Cheng and George M. Church. *Biclustering of expression data.* In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pp. 93–103. AAAI Press, **2000**. ISBN 1-57735-115-0.

[Chuah1998] Mei C. Chuah. *Dynamic aggregation with circular visual designs.* In

INFOVIS '98: Proceedings of the 1998 IEEE Symposium on Information Visualization, pp. 35–43. IEEE Computer Society, Washington, DC, USA, **1998**. ISBN 0-8186-9093-3.

[Collins2007] Christopher Collins and Sheelagh Carpendale. *Vislink: Revealing relationships amongst visualizations. IEEE Transactions on Visualization and Computer Graphics*, volume 13(6):pp. 1192–1199, **2007**. ISSN 1077-2626.

[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*, pp. 110–118. IEEE Computer Society, Washington, DC, USA, **2005**.

[Ferreira2003] Maria Cristina Ferreira de Oliveira and Haim Levkowitz. *From visual data exploration to visual data mining: A survey. IEEE Transactions on Visualization and Computer Graphics*, volume 9(3):pp. 378–394, **2003**. ISSN 1077-2626.

[Dietzsch2006] Janko Dietzsch, Nils Gehlenborg, and Kay Nieselt. *Mayday–a microarray data analysis workbench. Bioinformatics (Oxford, England)*, volume 22(8):pp. 1010–1012, **April 2006**. ISSN 1367-4803.

[Doleisch2002] Helmut Doleisch and Helwig Hauser. *Smooth brushing for focus+context visualization of simulation data in 3d.* In *WSCG*, pp. 147–154. **2002**.

[Du2005] Z. Du and F. Lin. *A novel parallelization approach for hierarchical clustering. Parallel Computing*, volume 31(5):pp. 523 – 527, **2005**. ISSN 0167-8191.

[Dubitzky2007] Werner Dubitzky, Martin Granzow, and Daniel Berrar. *Fundamentals of Data Mining in Genomics and Proteomics*. Springer, **2007**.

[Dunn1974] J. C. Dunn. *A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters*, **January 1974**.

[Efron1987] Bradley Efron. *The Jackknife, the Bootstrap, and Other Resampling Plans (CBMS-NSF Regional Conference Series in Applied Mathematics)*. Society for Industrial & Applied Mathematics, **January 1987**. ISBN 0898711797.

[Eisen1998] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. *Cluster analysis and display of genome-wide expression patterns. Proc. Natl. Academy of Science USA*, volume 95(25):pp. 14863–14868, **December 1998**. ISSN 0027-8424.

[Fisher1987] Douglas H. Fisher. *Knowledge acquisition via incremental conceptual clustering. Mach. Learn.*, volume 2(2):pp. 139–172, **1987**. ISSN 0885-6125.

[Frey2007] Brendan J J. Frey and Delbert Dueck. *Clustering by passing messages between data points. Science*, volume 315(5814):pp. 972–976, **January**

2007. ISSN 1095-9203.

[Fua1999] Ying-Huey Fua, Matthew O. Ward, and Elke A. Rundensteiner. *Hierarchical parallel coordinates for exploration of large datasets*. In *VIS '99: Proceedings of the conference on Visualization '99*, pp. 43–50. IEEE Computer Society Press, Los Alamitos, CA, USA, **1999**. ISBN 0-7803-5897.

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

[Gasch2002] Audrey P. Gasch and Michael B. Eisen. *Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. Genome biology*, volume 3(11), **October 2002**. ISSN 1465-6914.

[Getz2000] G. Getz, E. Levine, and E. Domany. *Coupled two-way clustering analysis of gene microarray data. Proc Natl Acad Sci U S A*, volume 97(22):pp. 12079–12084, **October 2000**. ISSN 0027-8424.

[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, **May 2004**.

[Hearn2003] Donald D. Hearn and M. Pauline Baker. *Computer Graphics with OpenGL*. Prentice Hall Professional Technical Reference, **2003**. ISBN 0130153907.

[Heyer1999] Laurie J. Heyer, Semyon Kruglyak, and Shibu Yooseph. *Exploring expression data: Identification and analysis of coexpressed genes. Genome Res.*, volume 9(11):pp. 1106–1115, **November 1999**.

[Hibbs2005] M. A. Hibbs, N. C. Dirksen, K. Li, and O. G. Troyanskaya. *Visualization methods for statistical analysis of microarray clusters. BMC Bioinformatics*, volume 6:p. 115, **2005**. ISSN 1471-2105.

[Hsu2006] Hui-Hsang Hsu. *Advanced Data Mining Technologies in Bioinformatics*. Idea Group Publishing, **2006**.

[Inselberg1985] Alfred Inselberg. *The plane with parallel coordinates. The Visual Computer*, volume 1(4):pp. 69–91, **1985**.

[Jiang2004] Daxin Jiang, Chun Tang, and Aidong Zhang. *Cluster analysis for gene expression data: a survey. Knowledge and Data Engineering, IEEE Transactions on*, volume 16(11):pp. 1370–1386, **2004**.

[Johansson2005b] Jimmy Johansson, Patric Ljung, Mikael Jern, and Matthew Cooper. *Revealing structure within clustered parallel coordinates displays*. In *INFOVIS '05: Proceedings of the 2005 IEEE Symposium on Information Visualization*, p. 17. IEEE Computer Society, Washington, DC, USA, **2005**. ISBN 0-7803-9464-x.

[Johansson2004] Jimmy Johansson, Robert Treloar, and Mikael Jern. *Integration of*

unsupervised clustering, interaction and parallel coordinates for the exploration of large multivariate data. In *IV 2004: Proceedings on the Eighth International Conference on Information Visualisation*, pp. 52–57. **2004**. ISSN 1093-9547.

[Johnson1991] B. Johnson and B. Shneiderman. *Tree-maps: a space-filling approach to the visualization of hierarchical information structures*. In *Visualization, 1991. Visualization '91, Proceedings., IEEE Conference on*, pp. 284–291. **1991**.

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

[Keim2002] Daniel A. Keim. *Information visualization and visual data mining. IEEE Transactions on Visualization and Computer Graphics*, volume 8(1):pp. 1–8, **2002**. ISSN 1077-2626.

[Keim2006a] Daniel A. Keim, Florian Mansmann, Jorn Schneidewind, and Hartmut Ziegler. *Challenges in visual data analysis*. In *IV '06: Proceedings of the conference on Information Visualization*, pp. 9–16. IEEE Computer Society, Washington, DC, USA, **2006**. ISBN 0-7695-2602-0.

[Kohonen1982] Teuvo Kohonen. *Self-organized formation of topologically correct feature maps. Biological Cybernetics*, volume 43(1):pp. 59–69, **January 1982**.

[Kohonen2000] Teuvo Kohonen. *Self-Organizing Maps*. Springer, 3rd edition, **December 2000**. ISBN 3540679219.

[Koren2003] Yehuda Koren and David Harel. *A two-way visualization method for clustered data*, **2003**.

[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)*, pp. 123–137. **2003**.

[Lamping1995] John Lamping, Ramana Rao, and Peter Pirolli. *A focus+context technique based on hyperbolic geometry for visualizing large hierarchies*. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 401–408. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, **1995**. ISBN 0-201-84705-1.

[Lex2008] Alexander Lex. *Exploration of Gene Expression Data in a Visually Linked Environment*. Master's thesis, Graz University of Technology, **2008**.

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

[Marshall2005] Adele H. Marshall and Roy Sterritt. *Clustering gene expression data using continuous markov models. 2005 IEEE Computational Systems Bioinformatics Conference - Workshops*, volume 0:pp. 314–321, **2005**.

[Mueller2009] Heimo Mueller, Robert Reihs, Stefan Sauer, Kurt Zatloukal, Marc Streit, Lex Alexander, Bernhard Schlegl, and Dieter Schmalstieg. *Connecting genes with diseases.* In *Sixth International Conference BioMedical Visualization.* **2009**.

[Nikkilae2002] J. Nikkilä, P. Törönen, S. Kaski, J. Venna, E. Castrén, and G. Wong. *Analysis and visualization of gene expression data using self-organizing maps. Neural networks : the official journal of the International Neural Network Society*, volume 15(8-9):pp. 953–966, **2002**. ISSN 0893-6080.

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

[Ovaska2008] Kristian Ovaska, Marko Laakso, and Sampsa Hautaniemi. *Fast gene ontology based clustering for microarray experiments. BioData mining*, volume 1(1), **November 2008**. ISSN 1756-0381.

[Partl2009] Christian Partl. *Visualizing a Cluster Hierarchy using a Radial Layout.* Master's thesis, Graz University of Technology, **2009**.

[Puff2009] Werner Puff. *Information Visualization and Collaboration in a Multi Desktop Environment.* Master's thesis, Graz University of Technology, **2009**.

[Rees2004] Christian Rees, Janos Demeter, John Matese, David Botstein, and Gavin Sherlock. *Genexplorer: an interactive web application for microarray data visualization and analysis. BMC Bioinformatics*, volume 5(1), **2004**.

[Saeed2003] A. I. Saeed, V. Sharov, J. White, J. Li, W. Liang, N. Bhagabati, J. Braisted, M. Klapa, T. Currier, M. Thiagarajan, A. Sturn, M. Snuffin, A. Rezantsev, D. Popov, A. Ryltsov, E. Kostukovich, I. Borisovsky, Z. Liu, A. Vinsavich, V. Trush, and J. Quackenbush. *Tm4: a free, open-source system for microarray data management and analysis. Biotechniques*, volume 34(2):pp. 374–378, **February 2003**. ISSN 0736-6205.

[Saldanha2004] A. J. Saldanha. *Java treeview–extensible visualization of microarray data. Bioinformatics*, volume 20(17):pp. 3246–3248, **November 2004**. ISSN 1367-4803.

[Sarkar1994] Manojit Sarkar and Marc H. Brown. *Graphical fisheye views. Commun. ACM*, volume 37(12):pp. 73–83, **1994**. ISSN 0001-0782.

[Schumann2000] Heidrun Schumann and Wolfgang Mueller. *Visualisierung - Grundlagen und allgemeine Methoden.* Springer Verlag, Berlin Heidelberg, Germany, **2000**.

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

[Seo2005] Jinwook Seo and Ben Shneiderman. *A rank-by-feature framework for interactive exploration of multidimensional data. Information Visualization*, volume 4(2):pp. 96–113, **2005**. ISSN 1473-8716.

[Sherlock2001] G. Sherlock. *Analysis of large-scale gene expression data. Brief Bioinform*, volume 2(4):pp. 350–362, **December 2001**. ISSN 1467-5463.

[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.

[Shneiderman2006] Ben Shneiderman and Aleks Aris. *Network visualization by semantic substrates. IEEE Transactions on Visualization and Computer Graphics*, volume 12(5):pp. 733–740, **2006**. ISSN 1077-2626.

[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.

[Speer2003] Nora Speer, Peter Merz, Christian Spieth, and Andreas Zell. *Clustering gene expression data with memetic algorithms based on minimum spanning trees*, **2003**.

[Stasko2000] J. Stasko and E. Zhang. *Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pp. 57–65, **2000**.

[Streit2007] Marc Streit. *Metabolic Pathway Visualization Using Gene-Expression Data.* Master's thesis, Graz University of Technology, **2007**.

[Streit2008] Marc Streit, Michael Kalkusch, Karl Kashofer, and Dieter Schmalstieg. *Navigation and exploration of interconnected pathways. Computer Graphics Forum (EuroVis 2008)*, volume 27(3):pp. 951–958(8), **May 2008**.

[Streit2009a] Marc Streit, Alexander Lex, Michael Kalkusch, Kurt Zatloukal, and Dieter Schmalstieg. *Caleydo: Connecting pathways with gene expression. Bioinformatics*, **2009**.

[Sturn2000b] Alexander Sturn. *Cluster Analysis for Large Scale Gene Expression Studies.* Master's thesis, Graz University of Technology, **2000**.

[Sturn2000] Alexander Sturn, John Quackenbush, and Zlatko Trajanoski. *Genesis: cluster analysis of microarray data. Bioinformatics Application Note*, volume 18:pp. 207–208, **2000**.

[Tavazoie1999] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. *Systematic determination of genetic network architecture. Nature genetics*, volume 22(3):pp. 281–285, **July 1999**. ISSN 1061-4036.

[Telea2007] Alexandru Telea. *Data Visualization: Principles and Practice.* A K

Peters, Ltd, **2007**.

[Thomas2005] James J. Thomas and Kristin A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr, **2005**. ISBN 0769523234.

[Wijk1999] Jarke J. van Wijk and Huub van de Wetering. *Cushion treemaps: visualization of hierarchical information*. In *In Proceedings of the IEEE Symposium on Information Visualization (InfoVis99)*, pp. 73 – 78. **1999**.

[Watson2008] James D. Watson, Tania A. Baker, Stephen P. Bell, Alexander Gann, Michael Levine, and Richard Losick. *Molecular Biology of the gene, Sixth Edition*. Addison-Wesley Longman, Amsterdam, sixth edition, **2008**. ISBN 0321507819.

[Witten2005] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, **June 2005**. ISBN 0120884070.

[Yano2003] N. Yano and M. Kotani. *Clustering gene expression data using self-organizing maps and k-means clustering*. In *SICE 2003 Annual Conference*, volume 3, pp. 3211–3215 Vol.3. **Aug. 2003**.

[Zhang2006b] Aidong Zhang. *Advanced Analysis of Gene Expression Microarray Data (Science, Engineering, and Biology Informatics)*. World Scientific Publishing Company, London, **June 2006**. ISBN 9812566457.

[Zhang2006] Qiong Zhang and Yingsha Zhang. *Hierarchical clustering of gene expression profiles with graphics hardware acceleration. Pattern Recognition Letters*, volume 27(6):pp. 676 – 681, **2006**. ISSN 0167-8655.

[Zhao2009] Lizhuang Zhao and Mohammed J. Zaki. *Microcluster: Efficient deterministic biclustering of microarray data. IEEE Intelligent Systems*, volume 20(6):pp. 40–49, **2005**. ISSN 1541-1672.