Institute for Computer Graphics and Vision
Graz University of Technology

Multiform Visualization of Heterogeneous Data Spaces

Master's Thesis

Christian Partl, BSc
partl@icg.tugraz.at

Graz, March 2012

Supervision:
Prof. Dr. Dieter Schmalstieg
DI Alexander Lex
Dr. Marc Streit

Abstract

Investigators from different domains are frequently confronted with large volumes of data originating from multiple sources that need to be analyzed simultaneously. Inhomogeneities within datasets thereby often obstruct the analysis process, but can also be the source potentially interesting relationships in the data. Most existing visualization techniques do not take such inhomogeneities into account and represent the data in a one-view-fits-all fashion, thus being of limited use in the course of an analysis. The proposed approach considers the inhomogeneous nature of datasets and the need for a differentiated visualization of their subsets. It is based on VisBricks, an interactive multiform visualization that is made up of basic building blocks called bricks. Each brick visualizes a homogeneous subset of a dataset in a way that suits the subset's characteristics best. Visual linking of bricks is used to indicate relationships between different homogeneous subsets. VisBricks also provides drill-down features that allow a detailed examination of individual subsets. An important issue when dealing with several homogeneous subsets in a setup of multiple datasets is their configuration and management. For that purpose we developed the Data-View Integrator, which combines views and datasets in an abstract graph representation. It provides an overview of available datasets and their relationships, and can be used to configure and assign homogeneous subsets to VisBricks or other views for an in-depth analysis. The proposed visualization techniques were evaluated in case studies with domain experts in the context of multiple genomic datasets that are used for the characterization of cancer subtypes. The experts were able to quickly reproduce known results from the literature and also gained new insights into the data.

Keywords: Heterogeneous data, multiform visualization, VisBricks, Data-View Integrator, cancer subtype characterization, Caleydo

Zusammenfassung

Forscher aus verschiedensten Fachbereichen werden oft mit großen Mengen von Daten konfrontiert, die aus verschiedenen Quellen stammen und simultan analysiert werden müssen. Inhomogenitäten innerhalb von Datensätzen erschweren dabei häufig den Analyseprozess, sind aber oftmals die Quelle für interessante Beziehungen in den Daten. Derartige Inhomogenitäten werden in den meisten existierenden Visualisierungsmethoden nicht beachtet. Stattdessen wird ein gesamter Datensatz auf uniforme Art und Weise dargestellt, was in einem Analyseprozess nur bedingt von Nutzen ist. Der hier vorgeschlagene Ansatz berücksichtigt die inhomogene Beschaffenheit der Datensätze und das Erfordernis einer differenzierten Visualisierung ihrer Teilmengen. Er basiert auf VisBricks, einer interaktiven Multiform-Visualisierung, welche sich aus simplen Bausteinen, genannt Bricks, zusammensetzt. Jeder Brick stellt eine homogene Teilmenge eines Datensatzes mittels einer Visualisierung dar, die besonders geeignet ist für die Charakteristika, welche die Teilmenge aufweist. Beziehungen zwischen verschiedenen homogenen Teilmengen werden über visuelle Verbindungen der Bricks angezeigt. VisBricks ermöglicht auch die detaillierte Analyse von einzelnen Teilmengen über Drill-Down Features. Wenn man es mit multiplen homogenen Teilmengen aus verschiedenen Datensätzen zu tun hat, ist bereits deren Konfiguration und Handhabung komplex. Um diesen Prozess zu vereinfachen, führen wir den Data-View Integrator, welcher Datensätze und deren Visualisierungen in einer abstrakten Graphrepräsentation kombiniert, ein. Dieser wird dazu verwendet einen Überblick über die Datensätze und deren Beziehungen zu erlangen. Zusätzlich dient er der Konfiguration von Teilmengen und deren Zuweisung zu VisBricks oder anderen Visualisierungen, um sie in weiterer Folge genauer zu analysieren. Die vorgeschlagenen Visualisierungstechniken wurden in Fallbeispielen von Experten mittels verschiedener genomischer Datensätze, welche zur Charakterisierung von Krebsunterarten dienen, evaluiert. Dabei waren sie in der Lage, bekannte Resultate aus der Literatur schnell zu reproduzieren und neue Einblicke in die Daten zu gewinnen.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, Austria	**March 12, 2012**	
Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, Österreich	**12. März, 2012**	
Ort	Datum	Unterschrift

Acknowledgements

First, I would like to thank my supervisors Alexander Lex and Marc Streit for their continuous support and guidance throughout the course of this work. I also want to thank Dieter Schmalstieg for enabling me to work at the institute. Thanks to Hans-Jörg Schulz and Nils Gehlenborg for their great ideas and collaboration, and to Steffen Hadlak for providing his implementation of a graph layouting algorithm.

Furthermore, I want to thank my dear colleagues and friends Dieter, Patrick, Thomas, and Georg for their great collaboration in many assignments throughout my studies, which always was characterized by a good balance of productive work and fun.

Markus, Patritz, Michael, Lukas, and Ralph, thank you for your friendship over many years and the awesome time we had together. I also want to thank my good friend Regina. Thank you for being such a great person and for the way you enriched my life.

I would like to thank my grandma, and my sisters for being the dear persons they are. Finally, I want to thank my parents, who were always there for me when I needed them and supported me in every way throughout my studies and my whole life.

Contents

Chapter 1

Introduction

Nowadays, ever-growing amounts of data are a common phenomenon in many scientific domains. Data from different sources is collected in large-scale projects for its combined investigation. For example, *The Cancer Genome Atlas project*[1] (TCGA) accumulates genomic and clinical data of cohorts of patients for cancer characterization. The analysis of heterogeneous data has a high potential in terms of knowledge discovery, but the sheer amount of data typically overwhelms the comprehension of investigators.

Automatic methods from *data mining* represent one way to facilitate the analysis process. Data mining is commonly referred to as the extraction of patterns or models from observed data [GG99]. Likewise, interactive visualizations of the data that allow its exploration can provide great benefits for investigators. The combination of both methods is often designated as *visual data mining* [KS02], where each method compensates shortcomings of the other one. Visualization and data mining represent two major building blocks of *visual analytics*, which also includes other scientific disciplines such as data management or human perception and cognition [KKEM10].

1.1 Problem Statement and Contribution

The primary goals of simultaneously analyzing multiple datasets at a time are typically the detection of interesting patterns within datasets, the discovery of unexpected relationships between them, and the confirmation of existing hypothesis. Real-world datasets often exhibit inhomogeneities caused by properties of the data, such as different characteristics or semantic meanings that make data processing and visualization difficult. However, inhomogeneities often yield the patterns and relationships investigators are seeking.

For that reason, we propose a visualization system that acknowledges the inhomogeneous property of datasets and uses a divide and conquer strategy to unveil patterns and relationships within and between the datasets. Following this strategy, datasets are segmented into homogeneous subsets, which in turn are individually visualized. This is achieved within

[1] `http://cancergenome.nih.gov`

VisBricks, a flexible multiform visualization technique conceptually capable of integrating virtually any visualization method in one of its building blocks called *bricks*. Each of these bricks represents one of the homogeneous subsets obtained in the division step and thereby offers different visualization options suitable for that subset, its degree of homogeneity, and the current task. The overall layout of VisBricks is reminiscent of parallel coordinates: Multiple bricks, which are stacked on top each other, form a column and bricks of neighboring columns are visually linked, if their subsets are related.

An investigator is typically interested in a detailed analysis of only a selected group of subsets among different datasets instead of being confronted with all of them at a high level of granularity. Therefore we propose a second visualization technique, the **Data-View Integrator**, which is essentially a graph representation of all present datasets and views, that serves two main purposes: First, to provide an overview of how the datasets are conceptually related, and second, to specify, which of their homogeneous subsets shall be investigated in more detail within VisBricks or another view.

We implemented prototypes of the Data-View Integrator and VisBricks for the Caleydo framework [LSKS10]. Caleydo is an information visualization system that focuses on the analysis of genetical and clinical data. Thus, our prototypes are demonstrated and evaluated in that context, more precisely, by means of different TCGA datasets.

1.2 Collaboration Statement

The work presented in this thesis was developed cooperatively with different colleagues and resulted in the following publications. The contribution of the author of this thesis to each of them is explicitly stated.

- Alexander Lex, Hans-Jörg Schulz, Marc Streit, **Christian Partl**, and Dieter Schmalstieg. *VisBricks: multiform visualization of large, inhomogeneous data.* IEEE Transactions on Visualization and Computer Graphics (InfoVis '11), vol. 17, no. 12, pp. 2291-2300, 2011. [LSS*11]

  The author contributed considerably to the implementation of VisBricks.

- Alexander Lex, Marc Streit, Hans-Jörg Schulz, **Christian Partl**, Dieter Schmalstieg, Peter J. Park, and Nils Gehlenborg. *StratomeX: visual analysis of Large-Scale heterogeneous genomics data for cancer subtype characterization.* Conditionally accepted for: Computer Graphics Forum (EuroVis '12), vol. 31, no. 3, 2012. [LSS*12]

  In addition to major contributions to the concept and implementation of the Data-View Integrator, the author supported the implementation of StratomeX and made minor conceptual contributions to it.

1.3 Definitions

In the literature, many different terms have been established describing data and their properties. In this section the ones most relevant for this thesis are clarified and defined in the way they are to be understood within this document.

Data that is organized in a tabular form consisting of *rows* and *columns* is referred to as *multi-dimensional*. Rows are also called *records* or *data items* while *dimensions* or *attributes* denote columns. An *attribute value* is the value of an attribute for a specific data item. We use the term data item not only in the context of multi-dimensional data, so that it generally refers to a self-contained entity within the data that has different properties. A *graph* consists of a set of *nodes*, which are also denominated as *vertices*, and a set of *edges* or *links*, which represent pair-wise relationships between nodes. We define data to be *heterogeneous* if it is distributed across different datasets, may or may not their *data type* (e.g., multi-dimensional) be the same. *Homogeneity* and *inhomogeneity* describe how similar or different data within a dataset is with respect to certain properties.

Note, that in some cases the conceptual terms introduced in this thesis slightly differ from those used in our publications [LSS*11, LSS*12]. We also use the term VisBricks to refer to the StratomeX visualization technique, as it represents an extension of the VisBricks concept.

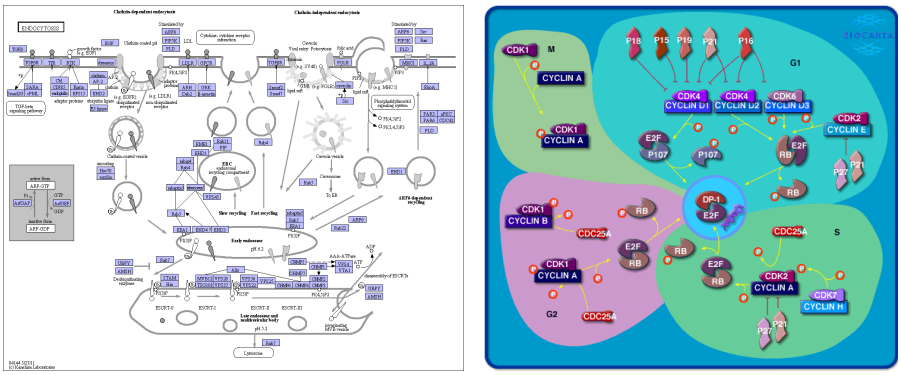
1.4 Biological Background and Data

All organisms on earth store their hereditary information in the form of a long double stranded molecule called *desoxyribonucleic acid* (DNA) within a cell. Among other functions, a single DNA strand can be used as a template to replicate itself and to synthesize *proteins* [AJL*07, cp. 1]. Both functions play an important role in cell division. The synthesis of proteins consists of two major processes: First, the DNA is *transcribed* into a shorter, but very similar molecule called *ribonucleic acid* (RNA). In a more complex process of *translation* RNA, more precisely *messenger RNA* (mRNA), is used to guide the synthesis of proteins, which is formed by stringing together amino acids. The type of protein and therefore its function within a cell is determined by its sequence of different *amino acids*, which in turn is coded in the DNA. DNA molecules contain the specification for thousands of proteins, whereas a segment of the DNA, that codes for one protein is a *gene* [AJL*07, cp. 1]. **Gene expression** refers to the synthesis of coded proteins. Cell types become different from one another by synthesizing different sets of proteins [AJL*07, cp. 7]. There are various mechanisms that control the amount of proteins being expressed, which is generally referred to as *gene expression regulation*. For example, epigenetic phenomena can influence the expression of genes. *Epigenetics* generally refers to the study of ideally heritable changes in gene expression or cellular phenotype, that occur without

modifications in the DNA [GAB07]. An example of such a change is **DNA methylation**, which suppresses the transcription of RNA if present in certain regions of the DNA. **Micro RNA** (miRNA) molecules also play a regulatory role in the process of translation by binding to mRNA molecules [CYHY09]. **Mutations** are permanent changes within the DNA [AJL*07, cp. 5]. If they occur in regions that code for proteins, their structure and function is affected. **Copy number variations** refer to large alterations within the DNA caused by duplication or deletion [FPF*06].

Cancers are diseases caused by an accumulation of molecular alterations affecting the DNA or inheritable epigenomic characteristics, such as methylation. They are typically classified by the tissue and cell type they originate from [AJL*07, cp. 20]. However, this classification does often not reflect the molecular alterations, which leads to the notion of *cancer subtypes*. Large efforts have been made to collect different sorts of data from about 10.000 cancer patients for more than 20 types of cancer to characterize and identify subtypes in the TCGA project. The collected data includes mRNA, miRNA, methylation, copy number and mutation status data and is generated using microarrays [Hel02] or next-generation sequencing technologies [SJ08].

Pathways are graphs that model cellular processes of entities such as genes or proteins in a cell. These graphs are typically represented as images (see Figure 1.1) and can be useful supplemental material when analyzing genetic data. They are made publicly available in different projects such as the *Kyoto Encyclopedia of Genes and Genomes* (KEGG)[2] [KAG*08] or *BioCarta*[3].



(a) (b)

Figure 1.1: Different representation styles of sample pathways from different sources: (a) The pathway 'Endocytosis' from KEGG and (b) the BioCarta Pathway 'Cyclins and Cell Cycle Regulation'. Available at `http://www.genome.jp/kegg/` and `http://www.biocarta.com/`.

[2] `http://www.genome.jp/kegg/`
[3] `http://www.biocarta.com/`

The amount of heterogeneous data that is involved during the analysis of TCGA data when trying to characterize cancer subtypes is typically very large, which makes the discovery of reasonable relationships among the highly interrelated data difficult. In order to prevent investigators from getting lost in the heterogeneous data spaces the application of data mining methods and suitable visual representations of the data are absolutely mandatory.

Chapter 2

Related Work

The vast amount of data that is generated by high-throughput experimental methods and collected in large-scale projects from different domains can only be understood with the help of data mining methods or suitable visualizations of the data. The focus of this work is on visualization, however, data abstraction methods do also play a role as a precursory step and are therefore briefly discussed in Section 2.1. Basic principles of Information Visualization are introduced in Section 2.2, different methods for the visualization of multi-dimensional data are discussed in Section 2.3, and graph visualization is covered in Section 2.4. Finally, existing approaches to subset visualization and the visualization of relationships between multiple datasets are described in Sections 2.5 and 2.6 respectively.

2.1 Data Abstraction

Nowadays, very large datasets that need to be analyzed are common in many scientific domains. Increasing amounts of data that need to be processed and displayed result in longer response times and cluttered visualizations, making interactive visual data analysis and exploration difficult [CWRY06]. Data abstraction addresses this problem by reducing the amount of data to be displayed. The following abstraction methods are listed by Oliveira and Levkowitz [FdOL03]:

- **Dimension reduction:** The process of transforming an n-dimensional dataset into a k-dimensional dataset, where k is much smaller than n, is called dimension reduction. Principal Component Analysis [And84, cp. 11] is an example of this method.
- **Sub-setting:** Sub-setting methods determine a representative subset of the original dataset. Random sampling [DE02], for instance, generates such a subset by randomly selecting data items.
- **Segmentation:** Segmentation techniques partition the dataset into multiple subsets. A prominent example of segmentation is *clustering*, which refers to the unsupervised classification of data items into groups [JMF99]. In clustering, *distance measures* like the *Euclidean distance* are used to quantify the similarity of data items so that similar items are assigned to the same group [JMF99]. Generally, clustering algorithms can

either be *partitional* or *hierarchical*. The former methods produce a single partition of the data whereas the latter ones produce a nested series of partitions [JMF99], i.e., a hierarchy of clusters.

- **Aggregation:** Aggregation methods group the original data into subsets and compute collective characteristics for them [AA06, p. 293]. Aggregate characteristics can be, for example, the mean value, the minimum, or the maximum [AA06, pp. 92, 334].

In the context of this work we use clustering to divide a large dataset into homogeneous subsets, which are easier to visualize. Additionally we use sub-setting methods to avoid over-plotting and aggregation methods to obtain characteristics from data that can be visually represented in a compact way.

2.2 Basic Principles of Information Visualization

Information visualization is "the use of computer-supported, interactive, visual representations of abstract data to amplify cognition" [CMS99, p. 7]. Abstract data thereby refers to data that has no obvious spatial mapping.

Ben Shneiderman provides a guideline for information visualization systems in his visual information seeking mantra:

> Overview first, zoom and filter, then details-on-demand [Shn96]

Gaining an *overview* of the data helps to identify major trends, patterns, relationships, and components [CC05]. The overview strategies suggested by Shneiderman [Shn96] include distortion-oriented and overview *Focus + Context (F+C)* techniques [KHG03]. F+C refers to the concept of focusing on a subset of data while preserving the context of the rest of the data. Distortion-oriented F+C methods such as the fisheye view [Fur86] magnify elements in focus while decreasing the space for contextual elements. Overview F+C techniques display focus and context separated from each other (e.g., in different windows).

When elements of interest have been identified, a user might want to *zoom in* on them, thus enlarging those elements while simultaneously hiding or shrinking other elements [CC05]. *Zooming out* represents the reverse operation. *Filtering* removes uninteresting elements from the display, e.g., by using widgets. Good implementations immediately reflect widget adjustments within the visualization (*dynamic filters*) [CC05].

Typically, a large amount of data items is displayed simultaneously within a visualization. The limited screen space and the visual complexity make it difficult to show all supplementary information for each data item at once [CC05]. With the *details on demand* method, additional information of data items is pointed out to the user on demand. This could be done by showing a popup-window containing the additional information upon data item selection [Shn96].

Generally, the way data is represented highly depends on its nature. Shneiderman [Shn96] discriminates in his task by data type taxonomy 7 data types to classify visualizations:

- **1-dimensional:** Linear data such as textual documents or lists.
- **2-dimensional:** Planar or map data.
- **3-dimensional:** Real-world objects.
- **Temporal:** Timelines, where data items have a start and finish time.
- **Multi-dimensional:** Each data item has n attributes.
- **Tree:** Hierarchies and trees, where each data item has a link to a parent item except the root.
- **Network:** Data items are linked in an arbitrary way.

It should be noted, that different categorizations for data types exist. Keim [Kei02], for example, differentiates between *one-dimensional data*, *two-dimensional data*, *multi-dimensional data*, *text and hypertext*, *hierarchies and graphs*, and *algorithms and software*.

If datasets of different types need to be visually explored together, usually multiple different visualizations are required. The use of multiple visualizations of data in separate windows is generally referred to as *multiple views*, whereas the term *multiform* visualizations designates a special case of multiple views, where one and the same data is represented in different ways [Rob00]. Coordinating views in a way that an action in one view also affects another one can provide the following benefits [NS97]:

- Information access time can be reduced, yielding a user performance improvement.
- Relationships within the information can be discovered, which would have been overlooked without view coordination.
- Complex information might be easier to understand when it is presented in multiple, simple views that are coordinated compared to when it is shown in one integrated visualization (unification of the desktop).

One possibility to coordinate views is *linking & brushing*. *Linked* views exchange viewing parameters such as brushing information [KHG03]. *Brushing* refers to the interactive selection of displayed elements in order to perform operations such as highlighting or masking on them [MW95]. A very common application of linking & brushing is to highlight corresponding elements in different views upon selection.

In complex scenarios, where multiple related elements have to be identified, *visual links* have shown to be more suitable to indicate these relationships than highlighting [SWS*11]. Visual links are typically realized by connecting the elements via lines, curves or surround-

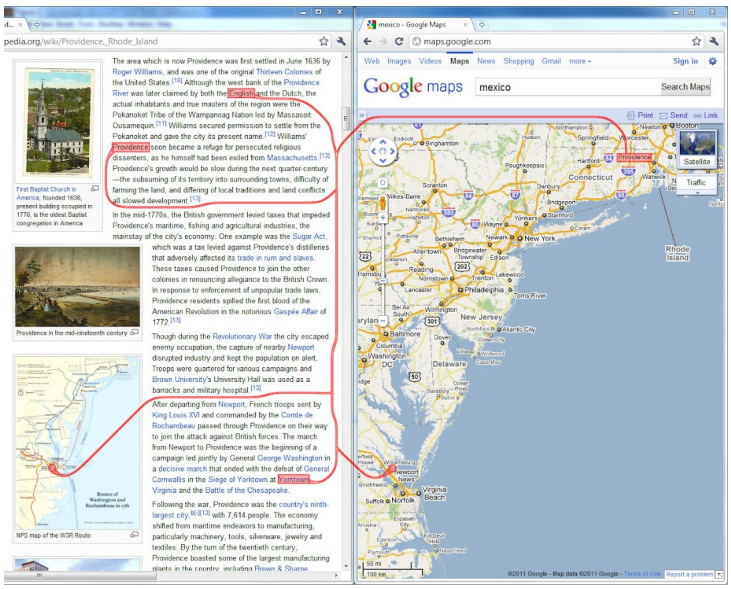


Figure 2.1: Context-preserving visual links [SWS*11] connecting elements in different windows.

ing surfaces. They are used to relate elements within a visualization [FWD*03, CPC09], across multiple views [CC07, LSKS10], or even across applications [WPL*10]. Since visual links can potentially occlude important information, Steinberger et al. [SWS*11] introduced context-preserving visual links, which minimize such occlusions (see Figure 2.1).

2.3 Visualization of Multi-dimensional Data

Data analysis tools typically assume a rectangular organization of data [Klö02], i.e., multi-dimensional data. Caleydo also mainly handles multi-dimensional data. Keim [Kei02] categorizes visualization techniques (primarily for multi-dimensional data) into standard *2D/3D displays*, *geometrically transformed displays*, *iconic displays*, *dense pixel displays* and *stacked displays*. This section gives a brief overview of existing visualization methods for multi-dimensional data, thereby focusing on those that are realized within Caleydo.

2.3.1 Parallel Coordinates

Parallel coordinates are a prominent geometrically transformed display method for multi-dimensional data that was developed by Inselberg and Dimsdale [ID90]. For an n-dimensional dataset, n parallel axes are drawn. A data item's value in each dimension is mapped to a location on the corresponding axis. Connecting these locations with a polyline creates the visual representation of the data item. Parallel coordinates were originally designed

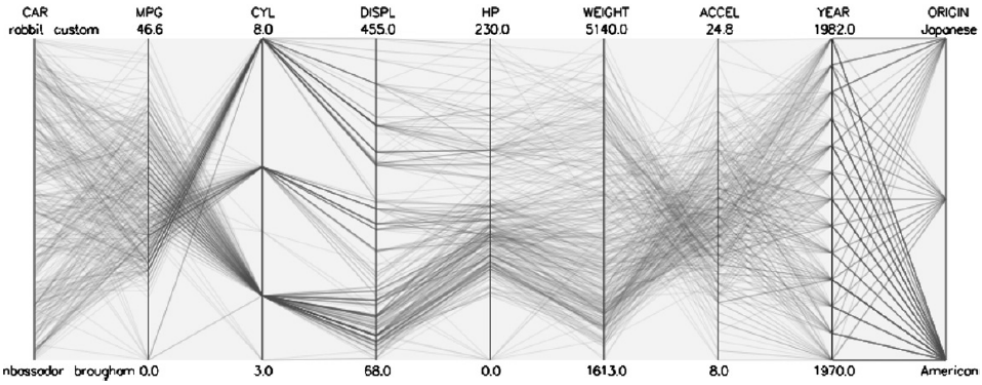


Figure 2.2: Parallel coordinates plot showing the car dataset [RD83]. Note that both numerical and nominal values are displayed. [SR06]

for visualizing n-dimensional Euclidean geometry, but by allowing the axes to have different scales and zero points, they become a more powerful tool for multi-dimensional data visualization [SR06]. It can even be used to display nominal values by mapping them to numeric data [RRB*04] (see Figure 2.2). Common interaction techniques include brushing polylines, axis duplication, deletion, and reordering. The latter ones are especially important, because relationships between values of dimensions, whose axes are not adjacent, can easily be missed [SR06].

One major problem of parallel coordinates plots is occlusion, either caused by a massive amount of polylines or by polylines that intersect axes at very similar or equal positions, making them hard or even impossible to distinguish. A simple solution to the latter case is brushing (highlighting) [SR06]. Another solution is provided by Graham and Kennedy [GK03], who suggest to use continuous curves instead of straight line segments, making the lines easier to follow at the axis intersections. If occlusion is caused by a massive amount of polylines, possible solutions are making the polylines transparent [WL96] or abstraction methods such as random sampling, clustering, or aggregation (see Section 2.1).

2.3.2 Heatmaps

Heatmaps [ESBB98] are a common method for the visualization of gene expression data. Each expression value is represented by a small colored rectangle. The expression value is mapped to a color from a color scale that traditionally ranges from green over black to red. Thus, if a gene is under-expressed with respect to a reference, its rectangle is colored increasingly green, whereas the rectangle of a gene that is over-expressed relative to the reference is colored increasingly red. Black rectangles indicate genes that have a similar expression as the reference. A problem of this color scale is that people suffering from red-green colorblindness, which are about 7% of the males with European descent [GS01],

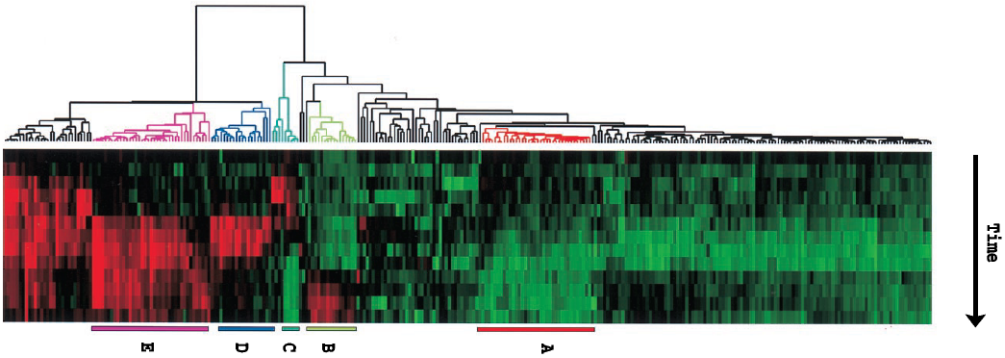


Figure 2.3: Heatmap [ESBB98] of hierarchically clustered gene expression data with an adjacent dendrogram.

will not be able to distinguish the colors. In order to avoid this problem alternative color scales, such as the colorblind safe diverging schemas from Colorbrewer [Bre09], should be used.

All the rectangles are arranged in a tabular form, where rows represent genes and columns experiments (or vice versa). Patterns in the data can be discovered by applying clustering algorithms and placing genes that share a cluster next to each other. The cluster hierarchy of a hierarchical cluster algorithm is often represented by a dendrogram next to the heatmap (see Figure 2.3).

2.3.3 Other Methods

A very basic way to visualize data are *scatterplots*, in which characters or symbols are drawn for each 2-dimensional data point in a coordinate system [Cha77, p. 220]. Tools like Polaris [STH02] use visual attributes such as shape or color of the symbols to encode additional dimensions. However, the number of dimensions in a multi-dimensional dataset is typically much higher. By arranging many scatterplots in a grid, a *scatterplot matrix* [CM88] is formed. Thereby the scatterplot in the i-th row and j-th column of the matrix maps the i-th and j-th dimension of the data onto its axes.

An iconic method by Chernoff [Che73] draws a cartoon face for each data item. The properties of the face such as nose length or curvature of the mouth represent the values of the data item in different dimensions. Other methods for visualizing multi-dimensional data include *circle segments* [AKK96], *dimensional stacking* [LWW90], and the *table lens* [RC94].

2.4 Graph Visualization

The way a graph is visualized highly depends on its class [DBETT99, p. 12]. In the context of this thesis, the class of *bipartite graphs* is especially important. Bipartite graphs can be partitioned into two sets of nodes where only nodes of different sets are connected by edges. A comprehensive overview of existing graph classes is given by Brandstädt et al. [BLS99].

Generally, graph visualizations can be categorized into three major types of representations [Sch10]: *matrix representations*, *node-link representations* and *implicit representations*. Each of these representations has certain strengths and weaknesses when it comes down to performing different tasks on the graph. A task taxonomy for graph visualizations is given by Lee et al. [LPP*06]:

- **Topology-based tasks:** These tasks are related to the structure of the graph such as finding all nodes adjacent to a given one.
- **Attribute-based tasks:** Tasks of this type focus on the attributes of edges or nodes, e.g., finding all nodes that have a specific property.
- **Browsing tasks:** Browsing tasks are associated with exploring the graph node by node, like following a given path or revisiting a node.
- **Overview tasks:** The goal of these tasks is to get estimates of the graph, such as its size, quickly.

The remainder of this section discusses the different types of representations for graphs in more detail, thereby laying emphasis on node-link representations.

2.4.1 Matrix Representations

Matrix representations are the graphical interpretation of the *adjacency matrix* of a graph [Sch10]. In an adjacency matrix, each node is represented by one row and one column. If there is an edge between two nodes, the corresponding cell is marked. As illustrated by Bertin [Ber10, pp. 36, 254, 255], finding a good permutation of rows and columns can greatly simplify the visualization and therefore makes it easier to comprehend. In matrix representations emphasis is laid on the representation of edges by granting them the most drawing space. This makes them suitable for performing tasks on the edge set [Sch10], especially if there are edge attributes. However, path-related tasks are much harder to accomplish than with node-link representations [GFC05, KEC06]. Examples of matrix representations include the visualization of phone calls made between states of the US [AK02], and call matrices, showing the calls being made between components of a large software system [vH03] (see Figure 2.4). In the former case a matrix representation is particularly useful, as the graph of phone calls is fully connected.

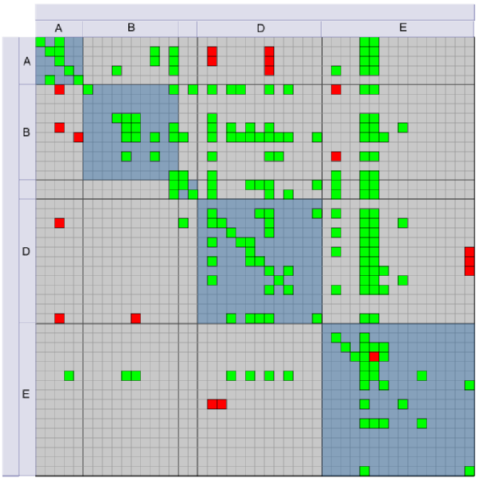


Figure 2.4: Call matrix: Rows and columns represent software components, calls between them are indicated by colored squares in the corresponding cells. [vH03]

2.4.2 Node-Link Representations

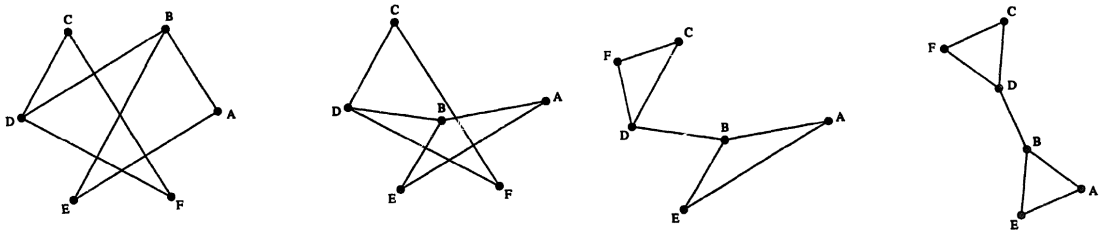
In node-link representations, nodes are depicted as points or other visual objects that are connected by lines or curves representing edges [Sch10]. A summary of aesthetic criteria for node-link layouts is given by Di Battista et al. [DBETT99, pp. 14-15] and includes amongst others the minimization of edge crossings, the minimization of edge bends, preserving uniform edge lengths and reflecting symmetries of the graph in the layout. Attempts to comply with at least some of these criteria (e.g., edge crossing minimization) often lead to NP-complete layout problems, so that solutions can only be approximated in practice [DPS02]. Generally, layouts can be classified by the degree of freedom in node placement [SS06] into *free*, *styled*, and *fixed* layouts, which will be discussed in the following.

Free Layouts

In free layouts, node placement is not restricted at all [SS06]. *Force-directed* methods are probably the most common example for this kind of layouts. In general, these methods define an energy function that map a graph layout to a number, so that low energy corresponds to layouts in which adjacent nodes are placed close to each other and non-adjacent nodes are farther apart. A final layout is determined by finding a (often local) minimum of this function [Kob12]. The probably most traditional method by Eades [Ead84] tries to meet the aesthetics criteria of uniform edge length and layout symmetry. This method imposes a physical analogy: Nodes are considered as steel rings and edges as springs, which therefore exert an attractive force on the nodes. In addition to this attractive force, a repulsive force is defined between non-adjacent nodes. Both forces depend on the Euclidean distance of the node positions. The algorithm iteratively updates the node positions ac-

cording to the forces affecting them until a certain iteration threshold is reached. An extension to this method is presented by Fruchterman and Reingold [FR91], which also try to incorporate the criterion of an even node distribution.

A slightly different approach is taken by Kamada and Kawai [KK89]. They define the desirable Euclidean distance between every pair of nodes to be dependent on the graph theoretic distance, i.e., the shortest path, between them. Thus, the force between two nodes can either be attracting or repelling, depending on their current Euclidean and graph theoretic distance. The overall energy function is defined in a way that the minimization of the difference between Euclidean and (the Euclidean mapping of) graph theoretic distances minimizes the energy in the system. Starting from an initial regular polygonal layout of the nodes, such a (local) minimum is obtained by iteratively determining the node affected by the greatest force and moving it to a stable position (see Figure 2.5).



(a) (b) (c) (d)

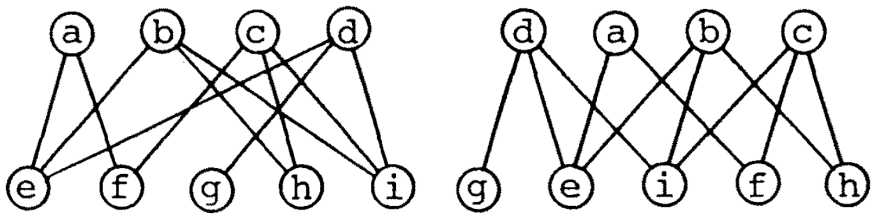
Figure 2.5: Illustration of the energy minimization process in the method of Kamada and Kawai [KK89]: (a) Starting from an initial layout, (b) node B is moved to a stable position first, (c) followed by node F. (d) The final positions are obtained after 21 movement steps. Adapted from [KK89].

These original algorithms are computationally expensive and therefore only suited for small graphs. In order to handle large graphs some modifications and optimizations are required. The approach by Quigley and Eades [QE01] reduces the number of necessary force calculations between non-adjacent nodes. This is achieved by using hierarchical force calculation [BH86]. The basic principle of this method is to approximate the forces of nodes that are farther away by grouping them into pseudo-nodes and only calculate the exact forces for nodes that are nearby. A different method by Walshaw [Wal01] creates a sequence of increasingly coarse graphs. These graphs are obtained in the following way: The nodes of the original graph are grouped into clusters, yielding a new, coarser graph. This procedure is repeated until the node size is sufficiently small. Starting from an initial layout of the coarsest graph the layout is successively refined for the graphs of the next coarseness levels until the original graph is reached. The layout algorithm that is used at each level is a modified version of the algorithm by Fruchterman and Reingold [FR91]. A comprehensive overview of other force-directed methods and optimizations is given by Kobourov [Kob12].

Styled Layouts

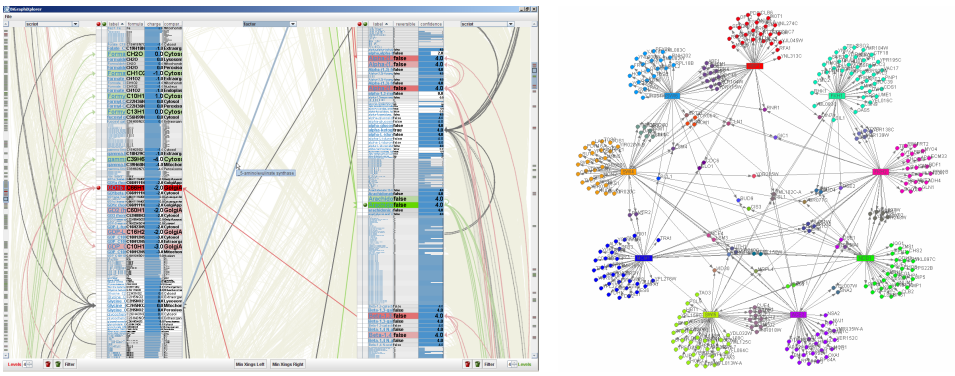
In styled layouts, nodes are arranged according to predefined schemas [SS06], which range from linear layouts [Cim06] over circular layouts [DMM97, GK07] to layered drawings [STT81]. The number of edge crossings has been shown to be one of the most important aesthetic criteria [Pur97, WPCM02] and their minimization does play an important role in most of these layouts. In the case of layered drawings edge crossings between the layers can be reduced by finding good permutations of the nodes in the layers (see Figure 2.6). For instance, the method by Sugiyama et al. [STT81] considers two adjacent layers at a time and alternately sorts the nodes in each layer according to their barycenters. They are calculated as follows: The nodes in each layer are assigned to an index according to their current position in the layer so that the first node of the layer is assigned to index 1 and the last node to index n. The barycenter of a node is given by the mean index of adjacent nodes from the other layer. A comparison of various methods for minimizing edge crossings in layered drawings is given by Jünger and Mutzel [JM97].

Two-layered drawings are often used to represent bipartite graphs. A sophisticated example for large bipartite graphs is given by Schulz et al. [SJUS08], who arrange the two node sets in opposed reorderable tables (see Figure 2.7 (a)). Besides connecting nodes from the different tables, arcs are drawn at the sides of the tables representing projection edges, where each of them basically indicates, if the two nodes it connects have common nodes in the other table. A different way to visualize bipartite graphs are *Anchored Maps* [Mis06], in which the nodes of one set are arranged in a circle (anchored nodes) and the nodes of the other set are positioned in a suitable manner relative to the anchor nodes (see Figure 2.7 (b)). Another method by Naud et al. [NUUT07] arranges the two node sets on the surfaces of two concentric spheres.



(a) (b)

Figure 2.6: The number of edge crossings in a two-layered layout (a) can be reduced by permuting the nodes in each layer (b). Adapted from [STT81].



(a) (b)

Figure 2.7: (a) A bipartite graph visualization [SJUS08] displaying nodes as rows in opposing reorderable tables. Image used with permission of Hans-Jörg Schulz. (b) An Anchored Map [Mis06] representing one node set as rectangular anchors and the other one as circles.

Fixed Layouts

In fixed layouts, the positions of nodes are predetermined, so that only edges can be routed freely [SS06]. Common examples for fixed layouts are graphs whose nodes contain geospatial information, which are therefore positioned on a geographical map. *Arc diagrams* [Wat02] also belong to the class of fixed layouts. This method points out repeating patterns in string data by connecting them via arcs.

Dealing with Large Graphs

A problem of node-link representations is that they often result in a cluttered display when the amount of nodes and edges is beyond a certain threshold. Especially when compared to matrix representations they are less suitable for many tasks on large graphs [GFC05, KEC06]. However, different methods exist for increasing the readability of graphs in node-link representations.

Edge bundling reduces the complexity of a display by combining edges that follow a similar path. A hierarchical edge bundling method by Holten [Hol06] assumes a graph with a hierarchical structure at its backbone to be laid out in a tree visualization. The non-hierarchical edges of such a graph typically cannot be shown by the tree visualization. These edges are explicitly drawn as curves between their nodes following a path that connects these nodes in the tree visualization. That way edges following the same path in the tree structure are bundled (see Figure 2.8 (a)). The edge bundling approach by Holten and van Wijk [HvW09] defines forces between edges in a way that the more parallel edges are and the lower their distance is, the more they attract each other (see Figure 2.8 (b)).

(a) (b)

Figure 2.8: (a) Hierarchical edge bundling in a circular tree visualization [Hol06]. The edges are laid out according to an invisible tree structure that is mirrored from the outside to the center. (b) Force-directed edge bundling applied to a US migration graph. Adapted from [HvW09].

Occlusions between nodes and edges can be avoided using *edge routing*. A simple, but very general method is presented by Dokulil and Katreniakova [DK08], which routes edges that intersect the bounding boxes of nodes along their corners. Phan et al. [PXY*05] also route edges in a similar fashion around bounding boxes that correspond to clusters of nodes obtained by hierarchical clustering. Of course, a grid-based energy function that has been used to avoid occlusions for context-preserving links [SWS*11] could also be used for routing edges.

2.4.3 Implicit Representations

In contrast to node-link representations, implicit representations do not explicitly draw edges. Instead, edges are indicated by the relative position of nodes, e.g., via inclusion, overlap, or adjacency [SS06, Sch10]. Most implicit representations are used for the visualization of trees. For example *treemaps* [JS91] use inclusion to encode parent-child relationships by recursively subdividing a rectangle into smaller rectangles at each level of the hierarchy. *Beamtrees* [vHvW02] represent nodes as horizontal or vertical beams, whereat their orientation switches at each level in the hierarchy. The relationship of a parent node to its child nodes is indicated through overlap: The beams of the child nodes are drawn in front of the beam of the parent node. *Sunburst* [SZ00] is a radial hierarchy visualization that represents each node as a partial disc. The partial discs of child nodes are thereby drawn further away from the center of the visualization, adjacent to their parent's partial disc, and within its sweep angle. Since the drawing space is exclusively used for the visualization of nodes, implicit representations are potentially well-suited for attribute-based tasks on the node set [Sch10].

2.5 Subset Visualization

Typically, datasets are visualized in a very uniform manner. However, subsets of their data often exhibit some special properties, which distinguish them from the rest of the dataset. Such properties can be crucial for the visual representation of a subset. Therefore it might be advantageous not to represent a whole dataset uniformly, but rather visualize each subset in a way that is most suitable for it.

2.5.1 Subset Visualization of Graphs

In the context of graph visualization the idea of individual subset visualization has been picked up in *TopoLayout* [AMA07]. In this method, a graph is hierarchically decomposed into subgraphs according to its topological features. The graph hierarchy is then drawn bottom up using a visualization method that is specifically suitable for each subgraph. These visualization methods are however limited to different node-link layouts. A hybrid visualization that combines different graph representations described in Section 2.4 in one visualization is given by *NodeTrix* [HFM07] (see Figure 2.9). It aims at the efficient visualization of social networks by using a node-link representation to show the global structure of the network, while local communities are intended to be visualized in matrix representations. Communities are subsets of nodes which are highly interconnected, so that a node-link representation of such communities would result in a cluttered display due to edge crossings or overlap. Starting from an initial layout, where the whole graph is visualized in node-link style, a user can interactively specify subsets of the graph to be shown as a matrix.

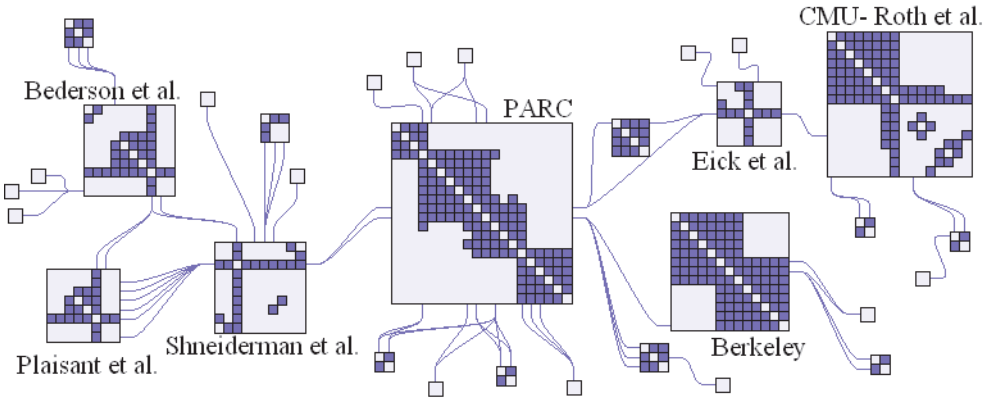
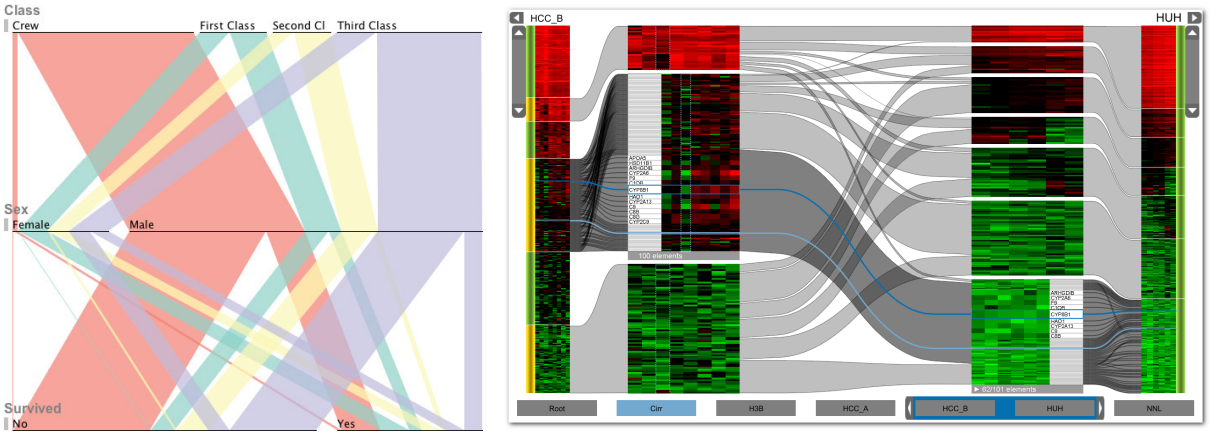


Figure 2.9: NodeTrix [HFM07]: Combining node-link and matrix representations for the effective visualization of a social network with highly interconnected communities.

Elastic Hierarchies [ZMC05] is another example for hybrid visualizations, in this case for the representation of trees. It combines the structural clarity of node-link representations and space efficiency of implicit representations, which are in this case treemaps. These representation methods can be switched at different points in the tree. A third hybrid graph visualization by Rufiange et al. [RMF09] extends matrix representations with treemaps to show hierarchical structured portions.

2.5.2 Subset Visualization of Multi-dimensional Data

In the context of multi-dimensional data *DataSplash* [WOA*01] is a system capable of visualizing subsets in a user-specified way within a main visualization. The system provides facilities for constructing different custom representations of the data which can be assigned to canvases. Additionally, portals can be created within a canvas. A portal is essentially a window that shows another canvas, so that a visualization of nested canvases can be built, where each canvas potentially provides a different representation of subsets of the data. For example, a geographical map can be used as main visualization where portals are positioned at locations of cities showing their transportation data as bar charts. *Multiform Matrices* [MXH*03] also display portions of multi-dimensional data in different ways. Such a matrix is essentially a scatterplot matrix, but instead of only displaying scatterplots in its cells, different kinds of visualizations capable of displaying two dimensions can be used for a cell.



(a) (b)

Figure 2.10: (a) Parallel Sets [Kos10] illustrating a categorical dataset of crew members and passengers that were on board of the Titanic when it collided with an iceberg. (b) The detail view of Matchmaker [LSP*10]. Selected clusters from the left and right columns are displayed as detail heatmaps inbetween the columns.

Both of the discussed methods for visualizing subsets of multi-dimensional data encode the relationships between the subsets through their relative position. A second possibility is to explicitly point out the relationships by connecting the subsets with visual links. Since the membership in a subset of a subdivided multi-dimensional dataset can be treated as an additional categorical variable, visualizations for categorical data that use visual linking could be adapted to present subsets of multi-dimensional data. For example, *Parallel Sets* [KBH06] is a visualization technique that adopts the visual linking approach in the context of categorical data. The basic layout of Parallel Sets is reminiscent of that of parallel coordinates. Each categorical variable of the data is represented by one axis, which is drawn as a sequence of bars. Every bar portrays one category with its width being proportional to the number of elements for this category. Two bars from different axes are connected by a visual link if elements correspond to both categories. Thereby the width of the links depicts the number of corresponding elements. In order to differentiate the categories and to point out relationships between categories of non-adjacent axes, the visual links are colored. A color is assigned for each category of the currently 'active' categorical variable. At any point in time the active variable can be changed. An extension to the basic method [Kos10] reduces the bars to lines thus laying emphasis on the connections (see Figure 2.10 (a)).

A system using a visual metaphor similar to Parallel Sets to display numerical multi-dimensional data was recently realized in *Matchmaker* [LSP*10]. In this system a multi-dimensional dataset is first grouped into subsets of dimensions. In a second step the records of each of these subsets are clustered individually. In Matchmaker a column is drawn for each subset of dimensions. Within a column heatmaps show the data of the subset's clusters. Visual links connect the heatmaps of related clusters from neighboring columns. Due to the fact that in the heatmaps individual data items are displayed rather small if the dataset contains many records, each cluster can be selected for detailed examination by displaying an enlarged version (a detail heatmap) of it between two columns (see Figure 2.10 (b)). In addition to showing the relationships between clusters of a dataset, MatchMaker can also be used to visually compare different clustering algorithms by applying them on copies of the same subset.

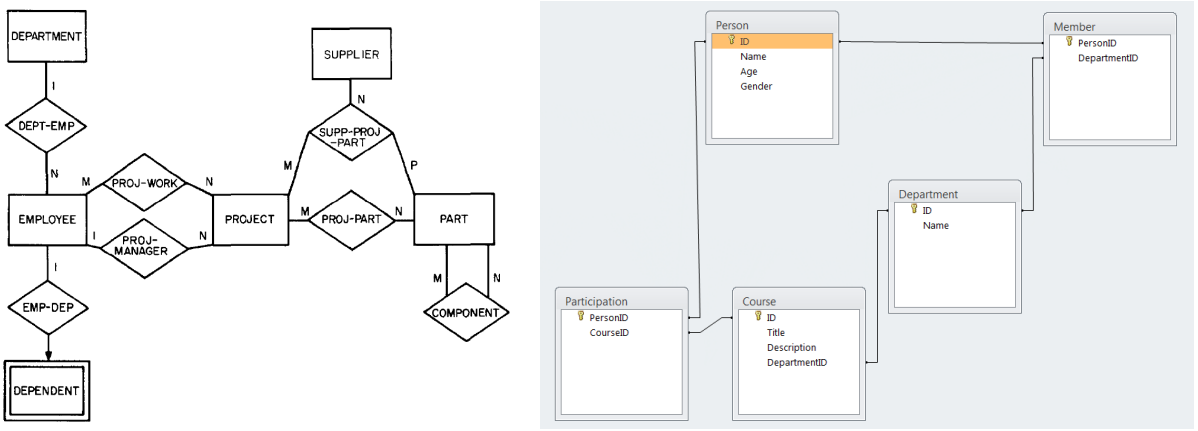
Other notable systems that adopt the Parallel Sets idea for subset visualization include *CComViz* [ZKG09] and the *Parallel Cluster View* in [TPRH11]. The former basically represents a Parallel Sets visualization for cluster comparison that is enhanced by an algorithm, which sorts the bars within the axes to reduce link crossings. The latter is used in conjunction with other visualizations to compare clusterings in a similar way as Matchmaker.

2.6 Visualization of Dataset Relationships

The investigation of multiple datasets originating from different sources requires an understanding of their relationships. This kind of information can be provided by visualizations. Early instances of such visualizations were created in the context of databases. Entity-relationship diagrams [Che76] are node-link representations that use rectangles to indicate entity-sets (e.g., persons, companies, events), which are linked with diamond-shaped boxes representing their relationships (see Figure 2.11 (a)). Nowadays many database systems also provide the ability to show schemas of relational databases [Cod70] in a similar fashion, where tables are represented as linked boxes (see Figure 2.11 (b)).

Such diagrams are great for getting an overview of the data. Therefore they can be used as a starting point for a visual analysis process in an information visualization system that follows Shneiderman's mantra [Shn96]. For example, entity-relationship diagrams fulfill this purpose in a system proposed by Derthick et al. [DRK97]. Besides of providing an overview, they are also used to initiate visual queries on the database. For all entity sets, 'dynamic aggregates' can be created in a separate window, which are indicated as interconnected boxes showing properties of the corresponding entities. A query is formulated by visual filters on the properties and the way the dynamic aggregates are connected. The subsets of data specified by these queries can finally be displayed in customized visualizations.

In analogy to database schemas, North et al. [NCS02] introduce the concept of *visualization schemas* for the *Snap-Together Visualization* [NS00], which is basically a framework that allows a user to rapidly create a custom multiple view system. A visualization

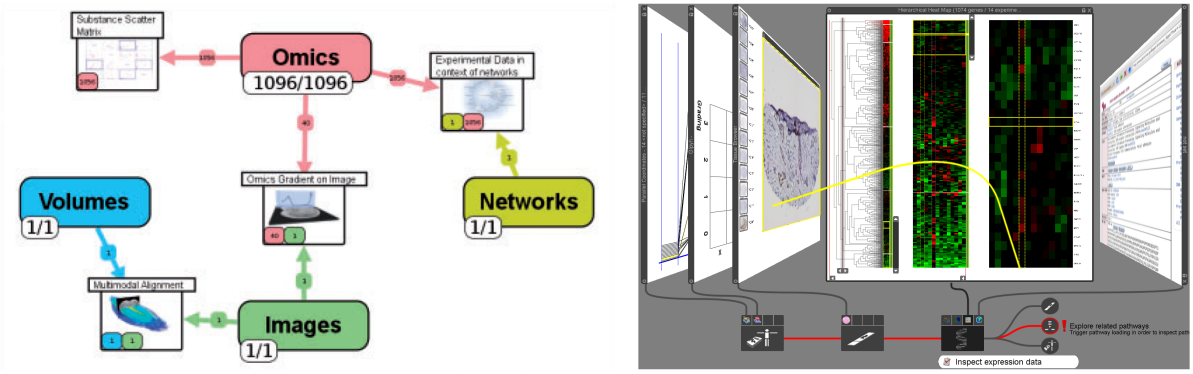


(a) (b)

Figure 2.11: (a) An entity-relationship diagram of information in a manufacturing firm [Che76]. (b) A diagrammic representation of a database schema in Microsoft Access™.

schema defines which data is shown in a view and how multiple views are coordinated. The graphical representation of a visualization schema is similar to that of a database schema: Views are depicted as nodes and links represent their coordinations. Such a diagram can be interactively created by assigning tables from the database schema to views, specifying the attributes of their data to be shown within the views, and defining the coordinations among different views. This diagram is directly reflected in a ready-to-use multiple view system. North et al. [NCS02] also envision 'DataFaces', which represents a combination of database and visualization schemas in a single graph. This graph shows which data is currently displayed in which view by connecting the corresponding nodes from the different schemas.

The general idea of a combined graph was realized by Rohn et al. [RKS11]. They consider four types of data in their system: Numerical data, networks, images, and volumetric data. For each of these types, a node, which represents all available measurements of the corresponding type, is initially placed in a *MappingGraph*. This graph can be extended by adding mapping nodes, which represent mappings. A mapping is the result of a mapping function, which basically takes measurements from one or more sources as input and maps it to a subset of these measurements or creates new measurements. A new measurement can be created, for example, by a 3D reconstruction from rotated 2D images. In the MappingGraph a mapping node A is connected to a node B, if measurements of B serve as input for the mapping function of A. Thereby B can either be one of the initial nodes or another mapping node. Each mapping node can be assigned to a visualization that displays its mapping. This assignment is indicated by a miniature representation of the real visualization in the mapping node (see Figure 2.12 (a)).



(a) (b)

Figure 2.12: (a) The MappingGraph in context of different datasets on barley grain. Adapted from [RKS11]. (b) Stack'n'flip [SSL*11] showing the steps a user has taken in an analysis process in a biomedical use case. Image used with permission of Marc Streit.

The relationships between datasets and their visual representations are also indicated in *Stack'n'flip* [SSL*11] (see Figure 2.12 (b)), a visualization that realizes a model created for guiding and orienting a user in an analysis process of multiple datasets. In this model, the process is structured into a series of tasks that may also contain branches or loops. Each task is associated with a dataset. The path a user has taken during his analysis is shown at the bottom area of Stack'n'flip as a sequence of nodes, each referring to a dataset. The nodes are visually linked to views representing their data above of the node sequence. If the user has completed the tasks associated with the current dataset he may proceed to the next one by choosing from different options. Thus, a new node is added to the sequence.

2.7 Discussion

There are only few existing visualization techniques that take inhomogeneities in data into account by individually visualizing homogeneous subsets. Few of these techniques are real multiform visualizations that can represent subsets in different ways. Techniques that emphasize the comparison of subsets typically do not provide any multiform features. Additionally, most comparative subset visualizations are restricted to the analysis of subsets that originate from a single dataset. Our approach overcomes the limitations of existing subset visualization techniques by providing a system that enables the comparison of homogenous subsets from multiple datasets, which can individually be visualized in various forms. The concept of this approach will be discussed in the following chapter.

Chapter 3

Concept

The integration of datasets originating from different sources for visual analysis poses a wealth of challenges. The information landscape created by these datasets has to be presented in a comprehensive way so that a user does not get lost in the large amount of data she is confronted with. The relationships between datasets also need to be indicated at different levels of granularity, ranging from the way datasets are conceptually related to relationships between subsets or even concrete data items. Additionally, datasets often exhibit inhomogeneities, which make separate subset visualizations beneficial or even necessary. Existing approaches addressing these challenges have been presented in Chapter 2. Our work tries to bring it all together: We propose a system for comparative multiform subset visualization of heterogeneous data that follows the divide and conquer paradigm at its core. This system is mainly designed for the analysis of multi-dimensional data, however, it is also applicable on data of different types.

3.1 Preliminaries

As defined in Section 1.3, *homogeneity* and *inhomogeneity* describe how similar or different data is. Inhomogeneity has to be distinguished from *data diversity*. Data diversity refers to the variety of attribute values and how even data items are distributed among these values [PHJ*10]. Therefore high data diversity corresponds to a uniform distribution of data items across all possible values of an attribute.

In context of multi-dimensional data, dimensions and records can be homogeneous with respect to their

- **semantics** - The more related the data is in terms of their meaning, the more homogeneous. For example, in a dataset of persons the dimensions of first and last name are likely to belong together as they encode the name of a person, whereas the dimensions first name and weight are semantically not related. Semantic homogeneity in records can often be found in categorical variables such as profession. For instance, pilots and stewards are more related to one another than to teachers. Note, that external knowledge has to be provided in order to discover semantic homogeneities.

- **characteristics** - The more similar the data types and their value ranges are, the more homogeneous. The type of dimensions can be, for example, numerical or categorical. Thus two dimensions of different types are inhomogeneous with respect to their characteristics. Though, even if they exhibit the same type such as bounded numerical, different value ranges (e.g., $[0 \dots 1]$ and $[10^5 \dots 10^9]$) can make them rather inhomogeneous. In many cases it is very challenging to visualize dimensions with different characteristics together. Characteristics of records causing inhomogeneities include missing or undefined values.
- **statistics** - The more akin the data distribution or behavior is, the more homogeneous. Correlations of dimensions and records can be discovered by methods such as clustering. Records or dimensions that have been assigned to the same cluster can be considered as homogeneous with respect to the similarity measure of the clustering algorithm.

It has to be noted that homogeneity of data always depends on the viewpoint, so that data can be homogeneous with respect to some properties whereas the same data is inhomogeneous with respect to other ones. For example, the profession of two persons can be related, making them semantically homogeneous. Nonetheless, the blood parameters of these persons can be completely different, so that they are inhomogeneous in terms of statistics.

3.2 Dividing Data into Homogeneous Groups

Grouping datasets into homogeneous subsets comes with several advantages: Automatic grouping methods such as clustering often reveal unexpected relationships in the data. The comprehension of the data might be facilitated, if homogeneous subsets are obtained through semantic grouping. If a whole inhomogeneous dataset has to be processed and visualized, very general methods that are capable of handling the very different data have to be applied. However, if the dataset is first divided into homogeneous groups, very specialized and optimized methods can be used for processing and visualizing each group individually. For that reason our approach is based on the divide and conquer paradigm, where the datasets are first divided into homogeneous groups, which are then individually visualized.

3.2.1 Dividing Multi-dimensional Data

As multi-dimensional data is the classical form of data in data analysis [Klö02], our approach for dividing data is mainly targeted on this kind of data. We divide multi-dimensional data in two consecutive steps (see Figure 3.1), similar to the Matchmaker approach [LSP*10].

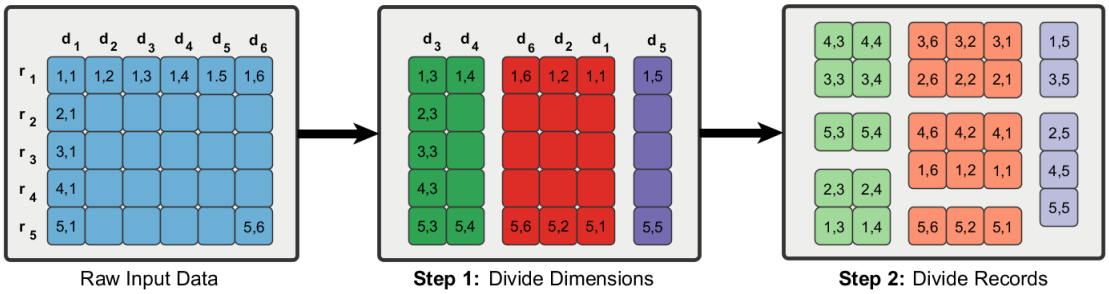


Figure 3.1: Starting from a raw multi-dimensional dataset, it is first divided into homogeneous groups of dimensions, followed by individually dividing each of these groups into homogeneous groups of records. The colors of the cells in the first step are used to emphasize the homogeneity of the groups of dimensions, whereas the colors in the second step indicate the subset relationship of the groups of dimensions and the groups of records they are divided in. Note that this illustration represents a simple case, as records or dimensions can also be assigned to multiple groups.

First, we divide the dataset along the dimensions to form *homogeneous groups of dimensions*. The way these groups are obtained is not defined. They can either be user-specified or the result of an automatic method. However, it might be feasible to group dimensions with respect to their type to facilitate further processing and visualization. In datasets where the types of dimensions do not differ, a semantic grouping of dimensions can often be useful. For example, if the measurements of a set of sensors are acquired under different conditions and at different points in time, the dimensions could be grouped by the conditions or by the points in time.

In the second step we individually divide each of the groups of dimensions obtained in step one further along the records, resulting in *homogeneous groups of records*. Again, it is not defined by what means the groups of dimensions are divided. The division depends on parameters such as the data, its context and analysis goals. In context of numerical groups of dimensions, we mostly apply clustering algorithms to discover unforeseen relationships within the data. On a side note, neither of these subdivisions has to result in disjoint groups. Thus, dimensions and records can be assigned to multiple groups.

In some cases it might be desirable to have access to multiple different groupings of the same dataset. For example, if the results of different clustering algorithms on the data shall be compared, they have to be applied on the same data. Therefore we introduce *perspectives* on dimensions and records. A perspective defines which elements (dimensions or records) are considered, their ordering, and the way they are grouped. Thus, two perspectives, one for dimensions and another for records, define a portion of the dataset and how it is subdivided into groups of dimensions and records.

3.2.2 A Generic Division Approach

In order to integrate general data in our system we have to look at our subdivision process for multi-dimensional data in a more abstract way. The division along dimensions can be considered as assignment of the data to subsets. Similarly, partitioning the groups of dimensions refers to a subdivision of the previous subsets. That way a three-level hierarchy of homogenous subsets is created, where subsets are more homogeneous the deeper they are in the hierarchy. In a hierarchy of multi-dimensional data, the whole dataset is considered to be the first level, the homogeneous groups of dimensions represent the second level, and the homogeneous groups of records the third one (see Figure 3.2).

The concept of a hierarchy of homogeneous subsets is general enough to be applied on most real-world datasets that can somehow be divided into subsets. We will illustrate this concept on a dataset of pathways: Each pathway already represents a homogeneous group of data, since they contain sets of genes that are involved in a chemical process. Therefore a pathway can be considered as a subset on the third level of the hierarchy. Subsets of the second level can be formed by pooling pathways. For example, the amount of shared genes can be regarded as a homogeneity criterion. That way all pathways are grouped together, which contain a minimum number of genes that are also found within the other pathways of that group. Since we do not restrict subsets to be disjoint, a pathway can also be assigned to different pools. The first level of the hierarchy is of course represented by the whole dataset.

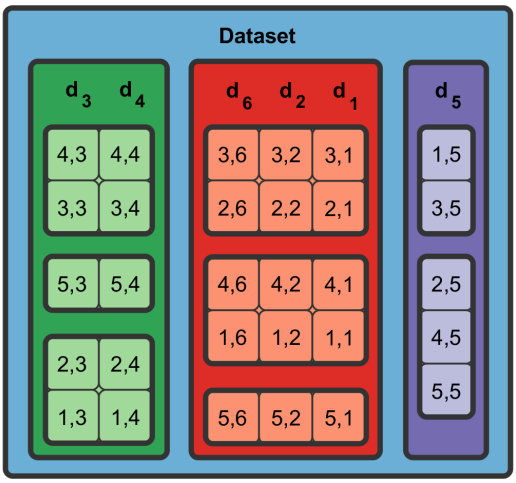


Figure 3.2: Illustration of the hierarchical relationships between subsets of a divided multi-dimensional dataset: The whole dataset containing all data represents the first level of the hierarchy. On the second level are subsets formed by the homogeneous groups of dimensions. These contain the homogeneous groups of records, representing subsets on the third level.

3.3 VisBricks: Conquering Homogeneous Groups of Data

The way datasets can be divided into homogeneous groups has been discussed in Section 3.2. Homogeneous data exhibits two inherent properties: It is much more suitable for abstraction than inhomogeneous data, and it is also much easier to visually encode it. Based on this observation, we introduce *VisBricks*, a multiform visualization for homogeneous groups of data. For the sake of simplicity, we will consider only the presence of one multi-dimensional dataset while illustrating the basic concepts of VisBricks and point out how it can be used in context of general data and multiple datasets in Section 3.3.3.

3.3.1 Layout

The basic building blocks of VisBricks are *bricks*. A brick is a rectangular area within the display that visually represents a homogeneous group of data. There are two basic types of bricks: *summary bricks* and *segment bricks*. A summary brick represents a homogeneous group of dimensions whereas a segment brick indicates a homogeneous group of records. Summary bricks are placed in the *arch* of VisBricks. The arch ranges from the left bottom of the display area over its top side to the right bottom (see Figure 3.3). It is composed of a focus region at its top and two context regions at the sides, the arch legs. If a summary brick is located in the focus region, the segment bricks, whose groups of records have been obtained by dividing the group of dimensions of that summary brick, are positioned un-

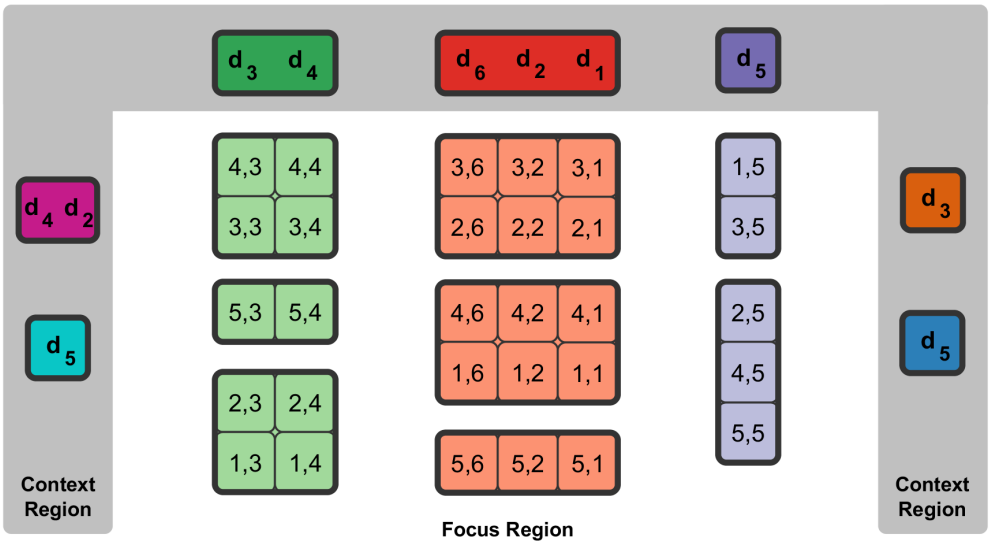
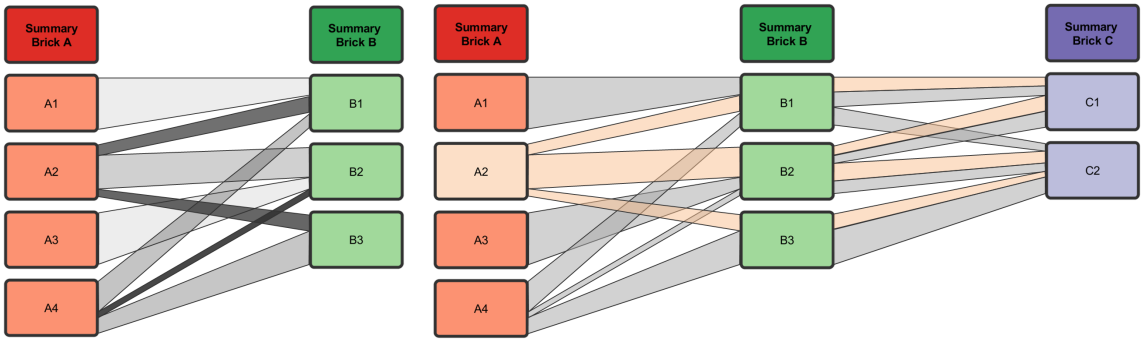


Figure 3.3: The VisBricks Layout: The arch consists of a focus region in the center and two context regions, the arch legs, which only show summary bricks. In the focus region summary bricks are displayed at the top, whereas corresponding segment bricks are added below them.

derneath it, forming a column. Thus, a summary brick shows all the data of the segment bricks below it. The layout resulting from multiple columns in the focus region is reminiscent to that of a table that has been split up by their rows and columns and is similar to that of Parallel Sets [KBH06] or Matchmaker [LSP*10].

Encoding Relationships within a Column of Bricks

Traditional spreadsheets and other visualizations that use rows and columns to represent multi-dimensional data indicate relationships between rows by sorting them according to a certain criterion. The bricks within a column can also be treated like rows, so that they can be sorted to encode relationships between them. For example, if all bricks of the column contain only numerical data, a sorting strategy similar to that of Matchmaker [LSP*10] could be applied: The bricks could be sorted according to the mean values of the bricks, so that bricks with a high mean value are positioned at the top and bricks with a low mean value at the bottom of a column. If every brick in the column represents a nominal category, they could be sorted alphabetically according to the category names. Finding an automatic sorting is difficult if the bricks contain different types of data, such as combinations of numerical and categorical data. In this case, a user-specified ordering could be applied. Note, that the ordering of bricks within a column only affects segment bricks, as the summary brick is always displayed on the top of a column providing an overview.



(a) (b)

Figure 3.4: Visual links connect segment bricks of neighboring columns to indicate the amount of shared data. (a) Illustration of the interactive trends filter: Its focus is on outliers, thus wider links appear increasingly transparent. (b) The selection of segment brick A2 is propagated to all related segment bricks of different columns.

Encoding Relationships between Columns of Bricks

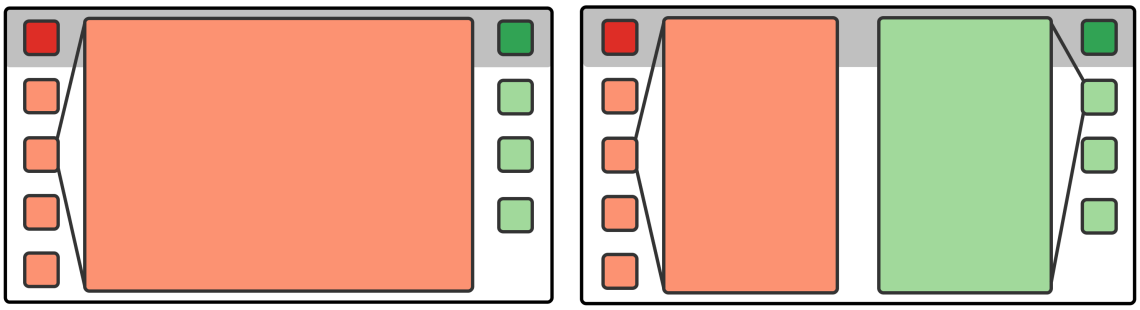
One problem of individually dividing groups of dimensions into homogeneous groups of records is that the ties of the records are broken up, so that the relationships between segment bricks of different columns are not obvious anymore. We tackle this issue in two ways: First, we use linking and brushing for synchronized highlighting. More precisely, if a record within a segment brick is selected, this information is propagated to other segment bricks that also contain data of this record and highlight it.

The second way is to draw visual links between segment bricks of neighboring columns. Two segment bricks are connected by such a link, if both of them contain data of the same record. The width of the links is proportional to the amount of shared records. Hence, major trends are indicated as wide links and outliers as thin ones, so that they can easily be distinguished. As some tasks focus on the identification of outliers and others on the identification of major trends, we developed a method that emphasizes either of them, depending on the current goals of a user. As major trends and outliers are no absolute concepts, we define a dynamically adaptable focus area for the number of shared records. The greater the difference between the number of shared records indicated by a link and the number specified by the focus area is, the more transparent the link is drawn (see Figure 3.4 (a)). In order to reduce crossings, we sort the ends of links at each brick in a manner similar to Parallel Sets [KBH06].

A drawback of any visualization that has a layout similar to that of parallel coordinates is that only neighboring columns can be directly compared. A common solution, which we also incorporated in VisBricks, is to allow an interactive rearrangement of columns. In some cases though, it is desirable to see where the records of a brick reside in all columns at the same time. Therefore, bricks can interactively be brushed, so that all links relating records contained in that brick are highlighted, even across columns that are not direct neighbors of the brushed brick (see Figure 3.4 (a)). Thereby links that also relate other records are only partially highlighted. Similar effects can be achieved by brushing a specific link instead of a whole brick.

Focus Duplicates

On conventional displays, bricks are often too small to show details of the data they represent. However, some tasks require the user to drill down to the level of individual data values. We provide this functionality within *focus duplicates* of bricks, which is conceptually similar to detail heatmaps in Matchmaker [LSP*10]. A focus duplicate can be created for every brick in the focus region of the arch. It is essentially an enlarged version of the original brick that is placed between two columns, taking up all the space between them (see Figure 3.5 (a)). In order to provide the opportunity of detailed comparisons of two bricks of neighboring columns, two focus duplicates are placed next to each other, sharing



(a) (b)

Figure 3.5: Illustration of focus duplicates: (a) The whole area between two brick columns is occupied by one focus duplicate. (b) This area is shared among two focus duplicates for making detailed comparisons.

the space between the columns (see Figure 3.5 (b)). When focus duplicates are shown, additional display space is gained by hiding the legs of the arch and only displaying the source columns.

3.3.2 The Multiform Nature of Bricks

Bricks are used to visualize the groups of data that are associated with them. The defining property of a brick is its multiform nature: The data associated with a brick can be visualized in various ways. The concrete visualization technique used depends on different aspects. As a start, the type of the brick (summary or segment brick) plays an important role. The goal of summary bricks is to summarize the data of all segment bricks of its column, whereas the goal of a segment brick is to show its homogeneous group of records in a comprehensive way. Other than that, the suitability of a visualization method depends primarily on two criteria: The *data characteristics criterion* and the *scalability criterion*.

The primary question behind the data characteristics criterion is, whether a visualization method is suitable for the given data type. For groups that are homogeneous with respect to their characteristics, a set of suitable visualizations can be found quite easily. However, if a group does not exhibit this kind of homogeneity, the least common visualization capable of handling the different data characteristics has to be used. Such a generic visualization often does not provide an optimal representation for the different kinds of data characteristics.

The scalability criterion refers to the suitability of a visualization technique for a given amount of data within an allocated display area. As VisBricks displays multiple bricks simultaneously, the display area for each of them is quite limited. Hence, displaying abstractions of the data plays an important role, since this can save a lot of screen space.

For that reason we define two modes for bricks: a *regular mode*, and a *compact mode*. The meaning of these two modes for summary and segment bricks will now be examined.

- **Regular Summary Brick:** As the intention of a summary brick is to give an overview of the data of its column, only visualization methods that can be accommodated within a fixed display area without losing too much of their expressiveness are suitable. For example, aggregation methods such as histograms or even parallel coordinates, which make use of sub-setting if the amount of records is large, could be used. Heatmaps or other tabular representations are not suited for summary bricks, as they require additional space for each record.
- **Compact Summary Brick:** If a summary brick is placed in a leg of the arch, it is displayed in its compact mode. In this case only a very small area that is completely fixed in height and width can be used for visualization, which makes a very abstract visualization method necessary.
- **Regular Segment Brick:** Most options regarding methods for visualization are available for segment bricks in their regular mode. Virtually any visualization, no matter if it requires scaling depending on the amount of records or not, can be used.
- **Compact Segment Brick:** The main limiting factor in terms of display space within a column is the height. Therefore the height of compact segment bricks is reduced to a minimum, thus only allowing very abstract representations of the data.

3.3.3 Generalizing VisBricks

Up to this point, VisBricks has only been described in the context of a single, multi-dimensional dataset. Nevertheless, its concept can easily be extended to general data and even multiple datasets.

Integrating General Data

In the context of multi-dimensional data, a summary brick presents a homogeneous group of dimensions, and a segment brick shows a homogeneous group of records. With regard to the hierarchy of homogeneous subsets, this means that a summary brick displays the data of a subset on the second level, and a segment brick the data of a subset on the third level. So, any dataset exhibiting such a hierarchical subset organization can be integrated into VisBricks. From now on, we will use the more general terms *segment group* for the portion of data, that is displayed within a segment brick, and *column group* for the data of a whole column, i.e., the data shown by a summary brick.

So far, links between segment bricks of neighboring columns were defined to be drawn, when they show data of records that occur in both of the associated record groups. Looking at this definition in a more abstract way, we can say, that two segment bricks are

visually linked, if their segment groups contain data of shared conceptual entities. For instance, the segment bricks in two neighboring columns of pathway data are linked, if their pathways contain the same genes.

Integrating Multiple Datasets

The basic integration of column groups originating from different datasets is rather straightforward: For each column group, either a column of bricks is displayed in the focus region, or only the associated summary brick is shown in one of the context regions. Since a user has to distinguish between the origins of the column groups, the relationships between datasets and its column groups have to be indicated. We do this by using color coding, where a unique color is assigned to each dataset, and the background of a column appears in the color of the associated dataset.

The concept for visually linking two segment bricks of neighboring columns, each belonging to a different dataset, is basically the same as for general data: A link is drawn if the segment bricks contain data of the same conceptual entities. Though, meaningful connections between bricks from two columns of different datasets cannot be drawn in some cases, even though they show data of the same entities. We will illustrate such a case in an example: Assume that there are two multi-dimensional datasets A and B of patient data. In A, patients are represented as records and dimensions depict personal information about them such as name or age. In B, patients are represented as dimensions, while genes are mapped to records. If A is divided according to our method described in Section 3.2.1, patients serve as the entities, for which segment bricks of different columns are compared and therefore linked. However, if B is divided in that way, genes are used for comparison, so that segment bricks from A and B cannot be compared. Of course, the records of A could be compared to the dimensions of B. But this would result either in no connections between bricks, or in connecting a brick in a column of A to all bricks in a column of B, which does not provide useful information. A solution to this problem is provided by transposing dataset B, so that records and dimensions are swapped. Generally, two segment bricks that belong to different datasets can only be visually linked, if the *entity types* that are used for comparison can be mapped. An entity type refers to a certain kind of entities, such as genes or patients.

Sometimes it might be desirable to see a segmentation of a column group from one dataset in a column group of another dataset. This is especially the case, if the segmentation was produced by an automatic method such as clustering. Therefore, we introduce the concept of *dependent columns*, which are created by dividing a column group of one dataset according to the segmentation of a source column group from a different dataset. Of course, the segmentation can only be performed for shared entities. Note, that the creation of dependent columns can be difficult or even impossible for some types of datasets. For example, a dataset of pathways cannot be arbitrarily subdivided according to a source column group.

However, dependent columns make perfect sense in context of multiple multi-dimensional datasets. In this case, a dependent column can essentially be created by using a record perspective of one dataset in the context of another one.

3.4 The Data-View Integrator

The provision of orientation or guidance within a visualization system during the exploration of multiple data sets can be very beneficial [SSL*11]. In our approach we orient the user in the information landscape using a visualization called *Data-View Integrator* (DVI) that serves two main purposes:

1. Providing an overview of the datasets and their relationships.
2. Offering an interactive way to specify subsets of data and to assign them to views.

3.4.1 Basic Representation

As the datasets and their interdependencies yield a graph structure, the DVI uses a node-link representation for its visualization. We chose this type of graph representation, because it is most suitable for the tasks that shall be accomplished using the DVI. Since we also expect the graph to be rather small, the disadvantages of node-link representations compared with matrix representations on large graphs will not come into effect. In the DVI, each dataset is represented as a node and relationships between datasets are indicated as links between the corresponding nodes. We define two datasets to have a relationship, if they contain data that can be mapped. As the entity types that datasets have in common typically represent important information, it is pointed out to the user by displaying their designations as labels on the corresponding links. The node of a dataset (*data node*) is drawn as a colored rectangle, with the color being unique for each dataset.

Besides adding a node for each dataset, a node (*view node*) is also added for each view within the system. A link is drawn between a view node and a data node, if the view of the view node displays some portion of data from the data node's dataset. In order to make the different types of links easier to distinguish, we display a link between a data node and a view node as a band, indicating a 'data flow', and a link between two data nodes as a simple line. The color used for drawing a band is equal to that of the corresponding data node, so that its data origin is perceived at a glance.

The DVI provides two different node layouts. Their use is dependent on the current task of the user, i.e., if an overview of the data shall be gained, or data shall be assigned to views for their exploration. While the goals of both tasks can be achieved with both layouts, each of them outperforms the other one in one specific task.

3.4.2 Providing an Overview

Gaining an overview of the present data to see how they are related is typically one of the first steps in an analysis process. At such an early stage, usually either no view or only few of them are opened to show data. Thus, mainly data nodes are displayed within the DVI, so that a user can focus on the investigation of dataset relationships without being distracted by view nodes.

A Force-directed Layout

A layout that places directly related nodes close together and others farther apart can be very beneficial when trying to get an overview. For that reason we use the force-directed layout algorithm by Kamada and Kawai [KK89], which directly reflects the graph theoretic distances between nodes in the layout. The high runtime complexity of this algorithm in not an issue, as the number of nodes to layout will generally be quite low. Another disadvantage of this algorithm is that adding new nodes can have a severe impact on the layout. Hence, the aesthetic criterion of preserving the *mental map* [MELS95] is violated. In this context the mental map refers to the abstract structural information formed by a user when looking at the graph [DG02]. New nodes or edges are commonly introduced to the graph if data is assigned to views. This is one of the reasons, why the force-directed layout algorithm is not well-suited for this task. However, when a user just wants to get an overview of the present datasets, typically no new nodes will be added dynamically, so that this disadvantage hardly ever comes into effect.

Edge Routing

In some cases, a user might want to visually group certain data nodes according to some criteria. For that reason, all nodes within the DVI are draggable to enable the user to create a custom node layout. In such layouts links do often overlap with nodes producing visual clutter. Occlusions between nodes and links can also occur in force-directed layouts, especially if the graph contains many edges. We address this issue by routing the links in a similar way as Dokulil and Katreniakova [DK08]. The basic principle of the algorithm is to define rectangular bounding boxes for each node and route the edges along their corners (see Figure 3.6). The algorithm consists of two steps:

1. **Intersection avoidance:** For every edge between two nodes we test, whether the direct connection of its anchor points intersects with the bounding box of another node. There are basically two types of intersections: Either the connection crosses two orthogonal sides of the bounding box, or two parallel ones. In the former case a new anchor point for the edge is created at the corner that connects the sides of the bounding box. In the latter case, a new anchor point is created at the corner that is closest to the intersection points. Thus, the original connection is split into two segments. For each of these segments that process is repeated until no more

intersections are detected. A link that is drawn along all the resulting anchor points does not intersect with any nodes.

2. **Optimization:** As a low number of edge bends has been found to be an important aesthetic criterion [WPCM02], which essentially reflects the Gestalt law of *good continuation* [Wer23], we try to reduce the number of bends in an optimization step. In this step we check, whether the removal of anchor points would introduce new intersections. If not, these points are removed. In order to make the links even more continuous, we interpolate a curve between the anchor points instead of drawing straight lines or bands.

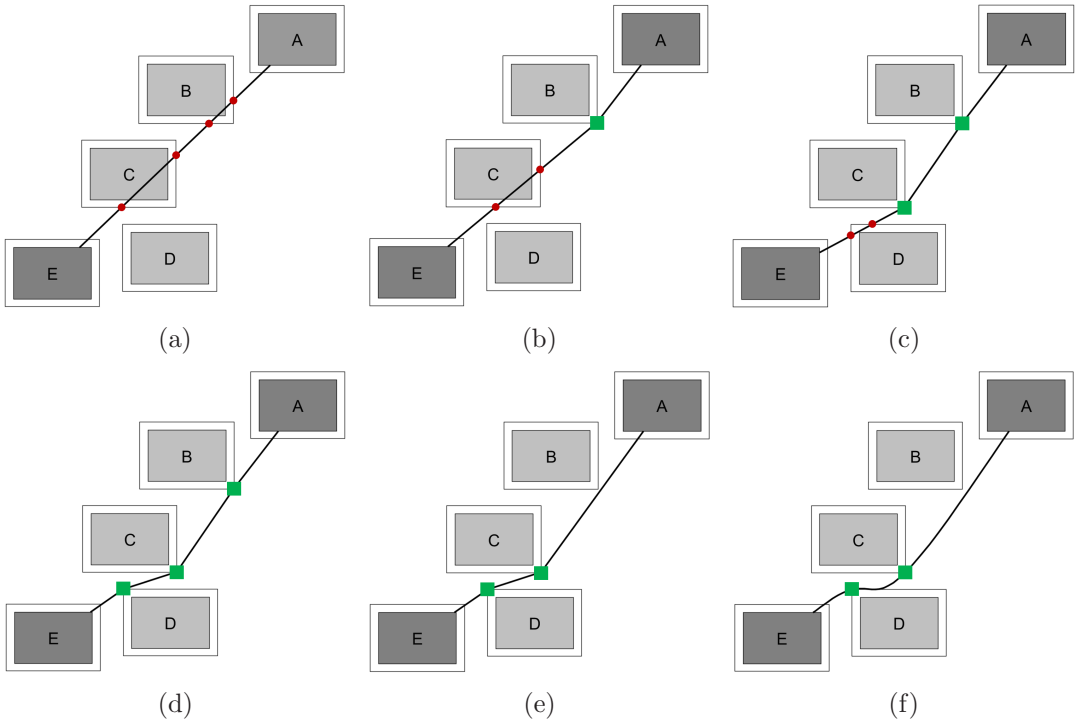


Figure 3.6: Illustration of the edge routing algorithm for a link between the nodes A and E: (a) The direct connection between the nodes intersects with the bounding boxes of the nodes B and C. (b) An anchor node is added to the link at the lower right corner of B's bounding box, thus dividing it into two segments. One segment still intersects with C, making the addition of another anchor node necessary (c). By adding the last anchor node, a new intersection with the bounding box of node D was introduced. (d) No more intersections detected after adding one more anchor node. However, the anchor node at B can be removed without introducing new intersections, so that the link is straightened (e). (f) Finally, the edge is drawn by interpolating a curve between the anchor points.

3.4.3 Assigning Data to Views

When a user has gained an overview of the data, the exploration of its interesting parts is typically the next step. For the actual visualization of data other views have to be consulted, as the DVI is not supposed to visualize the data itself. Instead, the DVI can be regarded as a support view that can be used to assign parts of the data, more specifically column groups, to different views, including VisBricks.

Representation and Specification of Assignable Data

Before data can be assigned to views, the available column groups have to be presented in the DVI. For each column group of a dataset, a bar is drawn within the corresponding data node, resulting in a sequence of bars for multiple column groups. This is the standard representation of column groups for all datasets. In addition to this representation, we define a more sophisticated representation for multi-dimensional datasets, which allows the user to dynamically create column groups based on dimension- and record perspectives. This representation is essentially a matrix, where the record perspectives defined for the dataset are depicted as rows and the dimension perspectives as columns (see Figure 3.7). A column group can be created by selecting a cell within the matrix. The combination of the corresponding dimension- and record perspectives defines the dimensions, the records, and the segmentation along the records for the column group. To indicate the existence of a column group, the associated cell in the matrix is marked. If a dimension perspective defines a grouping for its dimensions, one additional child column is added after the column of that perspective for every group. The same applies to record perspectives, where child rows

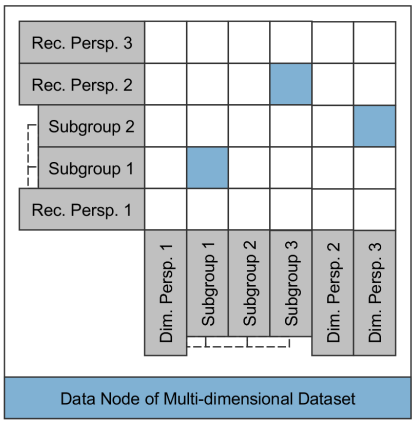


Figure 3.7: The matrix representation of a data node that belongs to a multi-dimensional dataset. Dimension perspectives and their subgroups are depicted as columns, whereas rows indicate record perspectives and their subgroups. Combinations, for which a column group has been created, are shown as blue cells within the matrix.

are added. Thus, column groups can also be created for combinations of these subgroups of records and dimensions. If there are multiple perspectives and subgroups, the matrix can get quite large and therefore requires a lot of screen space. We address this issue by making the perspectives collapsible, so that the rows or columns of their subgroups can be hidden. For the sake of simplicity we refer from now on to bars and cells that represent column groups as *column group indicators*.

Relating Data and Views

The interactive assignment of column groups to views can be carried out using the context menu of the column group indicators to open views, or by dragging them to view nodes. If a column group is displayed within a view, the column group indicator in the data node is directly connected to the view by a band. Since VisBricks can actually visualize multiple column groups, the column groups that have been assigned to it are represented as column group indicators, i.e., a sequence of bars, within the associated view node. Two corresponding column group indicators from a data node and the view node of VisBricks are connected by a band to emphasize their relationship. This would result in multiple bands running in parallel, if VisBricks shows multiple column groups of the same dataset. In order to reduce the complexity of the visualization, we bundle such bands in one wider band a short distance after they leave their column group indicators on each end. Additionally, we draw the column group indicators in the color of the corresponding dataset to facilitate the identification of its origin even further.

A Two-Layered Layout

To reduce the complexity of the data assignment process, we decided to use a two-layered graph layout, which reflects the bipartite nature of the graph. In this layout view nodes are arranged in a sequence at the top of the display area, whereas data nodes are accommodated at the bottom (see Figure 3.8). The advantages of this layout in the context of the view assignment task are that the two node types can be easily distinguished, specific view nodes can be located very fast when trying to find the destination for dropping a column group indicator, and the relationships between data nodes and view nodes are emphasized by granting the bands a lot of space between the sequences of nodes.

As in all layered graph layouts, the number of crossings between bands connecting the two different sets of nodes is dependent on their permutation. We reduce these crossings by ordering the view nodes according to their barycenters, which represents a part of the algorithm by Sugiyama et al. [STT81]. In contrast to the original algorithm, we only reorder one set of nodes for the following reason: In the process of assigning data to views, new view nodes are dynamically added to the DVI, so that only the view node sequence is regularly affected by changes. Hence, in order to comply with the aesthetics criterion of preserving the mental map to some extent, we only reorder the view nodes to reduce

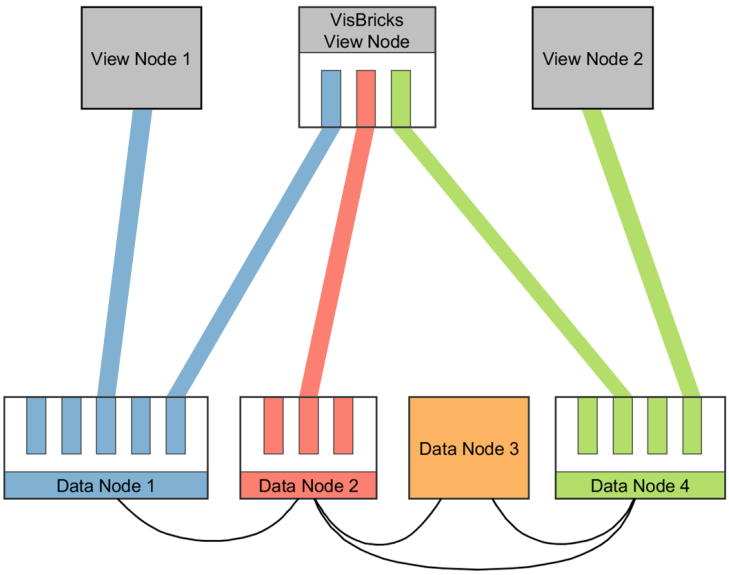


Figure 3.8: A schematic illustration of the two-layered layout of the DVI: Data nodes are arranged at the bottom and arcs between them indicate their relationships. Data Nodes 1, 2, and 4 represent their column groups as bars within the nodes. No column group is defined for Data Node 3. View nodes are arranged at the top, one of them being the view node of VisBricks. The column groups shown within VisBricks are indicated as a sequence of bars, similar to their representation in data nodes. Corresponding bars of the VisBricks node and data nodes are connected. The other view nodes are also linked to bars of data nodes, thus indicating that the corresponding column groups are visualized within the associated views.

crossings. In addition to reordering view nodes, we also order the column group indicators within the view node of VisBricks according to the order of the data nodes they belong to. Therefore crossings between bands connecting to column group indicators in the view node of Visbricks are avoided.

The edges between datasets are also indicated by drawing arcs that connect the bottom sides of data nodes, which is similar to the way projection edges are displayed in the table-based visualization for bipartite graphs by Schulz et al. [SJUS08]. As keeping track of these arcs is much harder than following rather straight lines and data nodes are arranged in a sequence, this layout is only of limited use for overview tasks.

3.5 Summary

The proposed system consisting of VisBricks and the Data-View Integrator yields two main use cases:

1. The detailed analysis of homogeneous groups from a single multi-dimensional dataset.
2. The in-depth examination of multiple general datasets, their subsets and relationships.

Overall, the system follows Schneiderman's information seeking mantra [Shn96]. An *overview* of the information landscape is gained within the Data View Integrator. *Zooming and filtering* is essentially provided by interactions that are used to select the data of interest for a more detailed inspection. For example, the assignment of certain column groups to views or their accommodation in the focus region of VisBricks can be regarded as such interactions. *Details* of the data are provided by switching from an abstract representation of the data within a brick to a more detailed one, or by using focus duplicates.

In the next chapter we present the realization of our concept of the DVI and VisBricks in two prototype visualizations.

Chapter 4

Results

In this chapter we present and demonstrate the Data-View Integrator and VisBricks prototypes. They are discussed in the order they are used in a typical analysis session, where first the DVI is used, followed by VisBricks. We illustrate them in the context of two biological case studies. All figures of this chapter were produced by using datasets from the TCGA project, KEGG and BioCarta pathways, and also a gene expression dataset of our collaborators from the *Institute of Pathology at the Medical University of Graz*, who focus on finding genetic factors of liver cirrhosis.

4.1 The Data-View Integrator

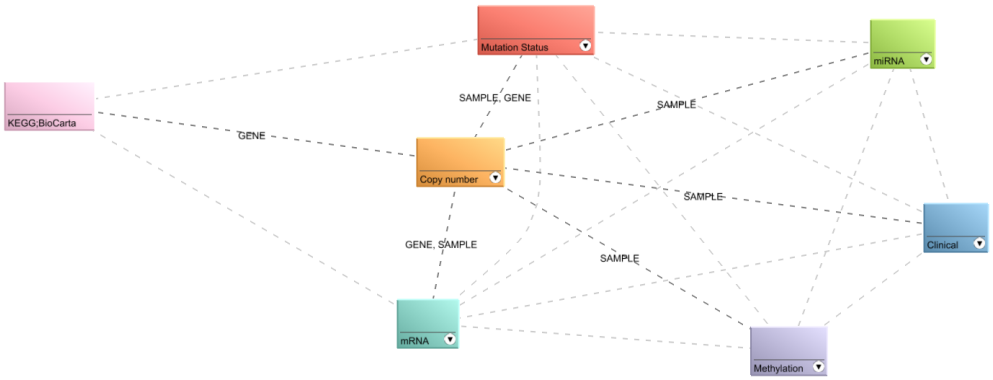


Figure 4.1: The force-directed layout of exclusively data nodes. Edges between data nodes are indicated as dashed lines, whereat highlighted edges also show the names of common entity types as a comma separated list. In this case all datasets are generally highly inter-connected, except for the pathway dataset. The edges of the copy number data node are currently highlighted, which reveals that its dataset has samples in common with the miRNA, methylation, and clinical datasets, genes in common with pathways, and that it shares both of these entity types with the mRNA and mutation status datasets.

The purpose of the DVI is to provide an overview of the datasets and to enable the specification and assignment of portions of data to views for their detailed examination. Figure 4.1 shows the DVI in its force-directed layout, which is used to get an overview. It presents TCGA datasets of mRNA expression, miRNA expression, DNA methylation, gene mutation, copy number, and clinical data. Additionally, pathways from KEGG and BioCarta jointly represent a pathway dataset. A unique color from a qualitative color schema of Colorbrewer [Bre09] is assigned to each data node. As evident in the image, the datasets are generally highly interconnected, except for pathways, which only have relationships to datasets that contain information about genes. The display gets cluttered very easily by labels indicating the entity types that datasets have in common. Therefore we only show the labels for the relationships of a data node that is currently hovered by the mouse. Additionally, we highlight the links (bands and lines) of a node on mouse over to make them easier to follow among the others. The links between data nodes are drawn as dashed lines to make them easy to distinguish from bands, which connect data and view nodes.

A custom node layout is illustrated in Figure 4.2, where also the view node of VisBricks is added. Such a layout can easily be created by dragging nodes to desired positions on the display. The figure shows how a band that connects a data node from the far left to the view node on the right is routed around nodes in between them to avoid occlusions. Custom layouts are also prone to introduce many edge crossings, resulting in a cluttered display, especially in a very dense graph. For that reason connections between data nodes can be hidden, so that only the links of a currently hovered data node are shown.

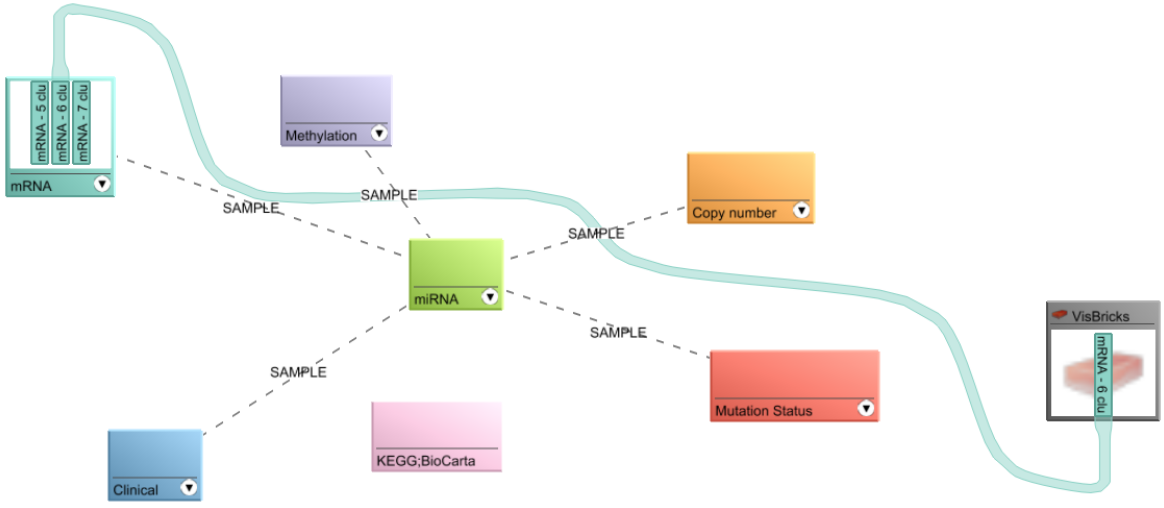


Figure 4.2: Illustration of how a band is routed around the bounding boxes of nodes in a custom layout. As the band can only dock to one side of the nodes it connects without causing occlusions, the bounding boxes of these nodes also have to be taken into consideration while routing.

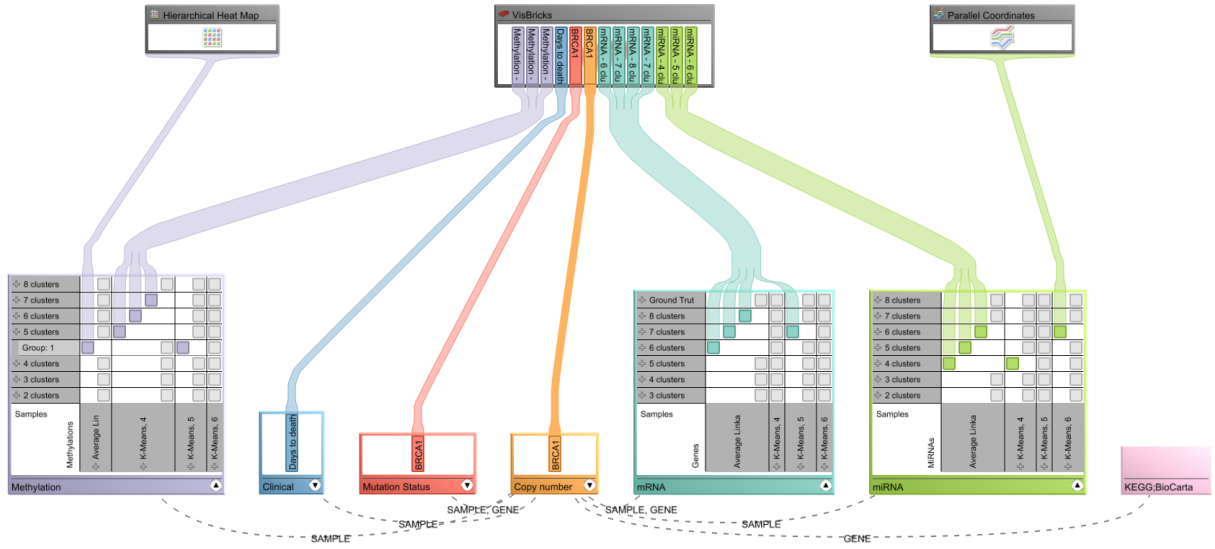
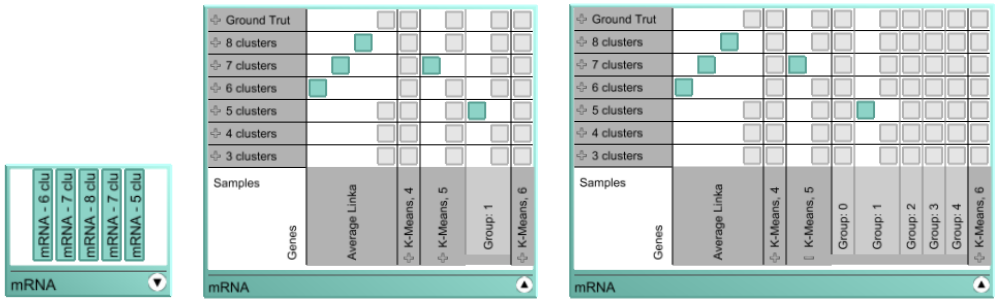


Figure 4.3: The two-layered layout of the DVI. The data nodes at the bottom are connected to view nodes at the top to indicate where their data is displayed. In this case the vast majority of the data is visualized in VisBricks, but some portions of methylation and miRNA data are displayed in different views, i.e., a heatmap, and a parallel coordinates view.

A user can interactively switch between the force-directed layout and the two-layered layout, which is shown in Figure 4.3. This layout is meant to be used for column group specification and their assignment to views. The data nodes are displayed at the bottom, which are connected by dashed arcs representing their relationships. View nodes are arranged at the top. They are designed to look like small windows, thus being reminiscent to the views they belong to. The colored column group indicators also remind of the colored columns within VisBricks (see Section 4.2). As illustrated in the figure, the column group indicators are connected by bands either directly to view nodes, or to corresponding column group indicators within the VisBricks node, thus indicating where they are displayed. All bands from a data node that connect to the same view node are bundled to reduce the complexity of the visualization. The more bands are bundled, the wider the resulting band is displayed. However, there is an upper limit for the width of bands as very wide bands require quite a lot of display space, which is especially a problem when they have to be routed between nodes.

In order to create column groups for investigation, a user can switch from the normal representation of a data node that refers to a multi-dimensional dataset to its matrix representation (see Figure 4.4). As described in Section 3.4.3, columns represent dimension perspectives and their subgroups, and rows depict record perspectives and their subgroups. Grey cells indicate combinations of perspectives and subgroups respectively, for which no column group has been created yet, whereas existing column groups are portrayed by col-



(a) (b) (c)

Figure 4.4: (a) The standard representation of a data node that corresponds to a multi-dimensional dataset. By clicking on the arrow on the lower right corner of the node, the node switches to its matrix representation (b). For each additional colored cell a column is added in order to allow bands to directly connect to these cells without occluding important elements. Note, that one subgroup of the dimension perspective 'K-Means, 5' is shown although that perspective is currently collapsed, because a column group has been created for that subgroup. (c) That perspective is expanded by clicking on its '+' icon.

ored cells. A column group can be created by simply clicking on a grey cell, which opens a dialog. In this dialog the name of the column group has to be specified, which will then be displayed on corresponding bar column group indicators. After the name was confirmed by the user, a colored cell is added for the newly created column group. To create column groups for dependent columns, foreign record perspectives have to be used in the combination. A record perspective can be imported into a dataset by simply dragging its row from one data node to another one.

The assignment of created column groups to views is trivial. To show a column group in VisBricks, the corresponding column group indicator can either be dragged to the VisBrick node, or it can be assigned via the indicator's context menu. The context menu can also be used to assign the column group to a different view, thus creating a new view within Caleydo, which is reflected within the DVI by the adding a corresponding view node. The DVI can also be used to navigate between different views. A view that has already been created can be opened by double-clicking on its view node.

The primary factor that limits the scalability of the DVI is the number of nodes that can be displayed simultaneously. Two main parameters influence this number. The first one is the size of the nodes in relation to the available screen space. This is especially an issue in the two-layered layout, as the nodes are shown in two rows. Depending on the number of defined column groups and the nodes' representations, i.e., standard vs. matrix, up to approx. 15 nodes can be accommodated in one row on conventional displays. The second influential parameter is the number of distinguishable colors. As each dataset is as-

signed to a unique color, only as many datasets can be reasonably integrated as colors can be discriminated. Healey [Hea96] confirmed that seven isoluminant colors can effectively be distinguished, but the limit for distinguishable colors is probably above this number.

4.2 VisBricks

VisBricks can be used for the detailed examination of multiple column groups that have been assigned using the DVI. The layout of VisBricks is illustrated in Figure 4.5. In the focus region, each column group is depicted as a column of bricks with one summary brick

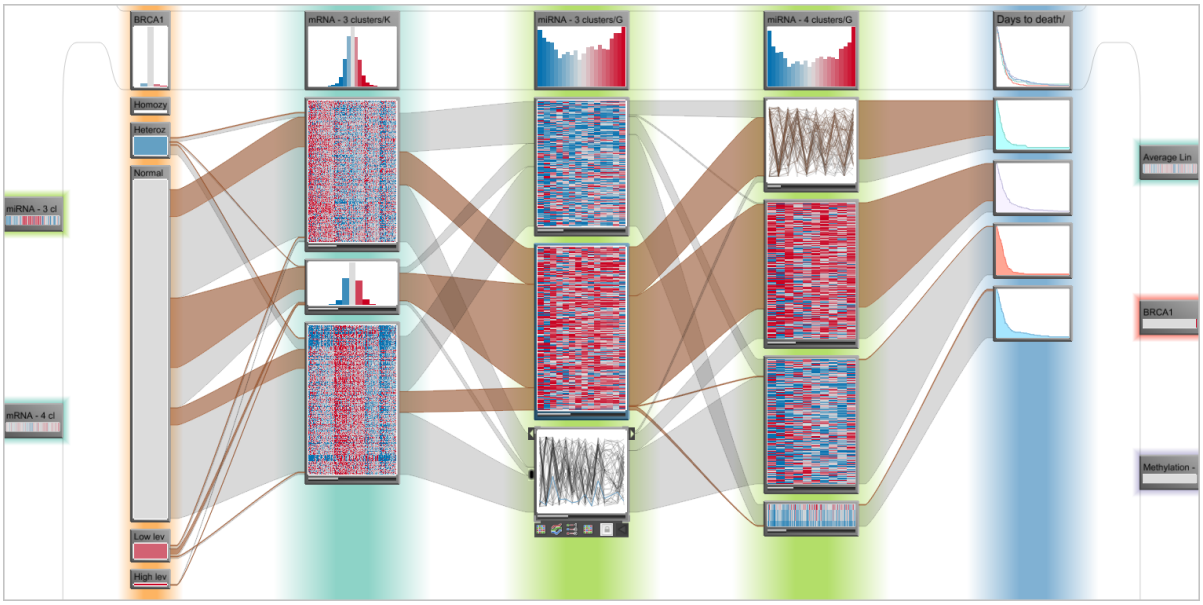


Figure 4.5: VisBricks showing multiple column groups from different TCGA datasets. The association to their datasets is encoded in the colored glow of the columns in the focus region and the summary bricks in the context regions respectively. The links between segment bricks of different columns indicate how many patients are shared between them. Note, that a brick in the center of Column 3 is brushed to highlight all links associated with its data. Visualizations within the bricks either have a fixed size, such as parallel coordinates or histograms, or scale the size of a brick with the amount of data they show. For example, heatmaps scale vertically depending on the number of records. The visualizations of a brick can interactively be switched using an on-demand toolbar, which can be seen in the bottom brick of Column 3. Column 1 shows categorical data: Each brick represents a category of the copy number status of one gene. Column 5 is a dependent column that shows the survival curves for the groups of patients from Column 4.

at the top and multiple segment bricks below. The context regions at the sides depict column groups as compact summary bricks. There is a colored 'glow' in the background of each column that allows to associate the column with its data node in the DVI. This glow is also present at the borders of compact summary bricks in the context region.

As evident in the image, VisBricks can show several different visualizations in bricks. The visualization type of a brick can interactively be switched using its toolbar, which is only shown on-demand to save screen space. VisBricks currently supports parallel coordinates plots, heatmaps, and histograms. Additionally, there are two abstracted heatmap representations available: The first one only consists of a single row, where each cell represents the average value of a dimension. In the second abstracted heatmap representation there are two additional rows added above and below the average row, which represent the average of a dimension plus, respectively minus its standard deviation in their cells. Categorical columns are depicted in a similar way as in Parallel Sets [KBH06], by using the height of the bricks to encode the number of elements of their categories. The category names are used as captions for their bricks. Pathways are represented by their images, and Kaplan-Meier (KM) curves [RNP*10] are used for clinical data. KM curves are essentially line plots that are used to illustrate times-to-event data such as the survival time of cancer patients after their cancer was diagnosed. Therefore the visualizations available for a specific brick depend on the kind of data displayed in the brick. As summary bricks are used to give an overview of the data in their column, and the segment groups can be shown in more detail in segment bricks, the available visualizations also depend on the type of the brick. The current mode of a brick, i.e., whether it is in its compact or regular mode, also has a great impact on the selection of available visualizations. Figure 4.6 illustrates the different combinations of brick types and modes.

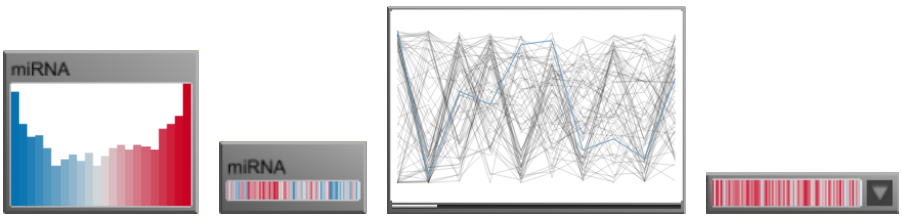


Figure 4.6: Summary and segment bricks in their different modes. (a) A regular summary brick gives an overview of its data with a histogram. (b) An abstracted heatmap representation is used for the compact summary brick of a multi-dimensional dataset of numerical data, as it requires little display space. (c) A regular segment brick shows its data with a parallel coordinates plot. Note the small bar at its bottom, which indicates the amount of data of that brick relative to the total amount of data in the corresponding column group. (d) An abstracted heatmap representation is also used in compact segment bricks.

As described in Section 3.3.1, visual links are drawn to indicate the relationships between segment bricks of neighboring columns. Visual clutter can be introduced by many links crossing each other. This issue can be addressed by using the interactive trends filter (see Figure 4.7 (a)), thus emphasizing major trends or outliers, or by only showing the currently highlighted links (see Figure 4.7 (b)).

Depending on the amount of data shown by a brick, focus duplicates have to be used for a detailed inspection of the data (see Figure 4.8). A focus duplicate can be created on either side of a column to allow detailed comparisons with bricks of both neighboring columns. In some cases even focus duplicates do not provide enough display space to recognize details. For that reason we also implemented zooming and panning within bricks.

VisBricks also supports a re-segmentation of existing column groups for numerical multidimensional data. This can be achieved by applying different clustering algorithms, which can be chosen from a dialog that is accessible via the toolbar of a summary brick. Pathway column groups can also directly be created within VisBricks. Using the context menu of a segment brick that represents data of genes, a special dialog can be accessed, which displays a list of all available pathways and the number of genes they share with the segment brick. All the pathways selected by the user are then used to create a new column group of pathways.

(a) (b)

Figure 4.7: (a) The interactive trends filter is used to emphasize on the links representing major trends. In this case, outliers are completely invisible. (b) Different colors are used to highlight visual links. The complexity of the visualization is reduced by only showing links that are currently highlighted.

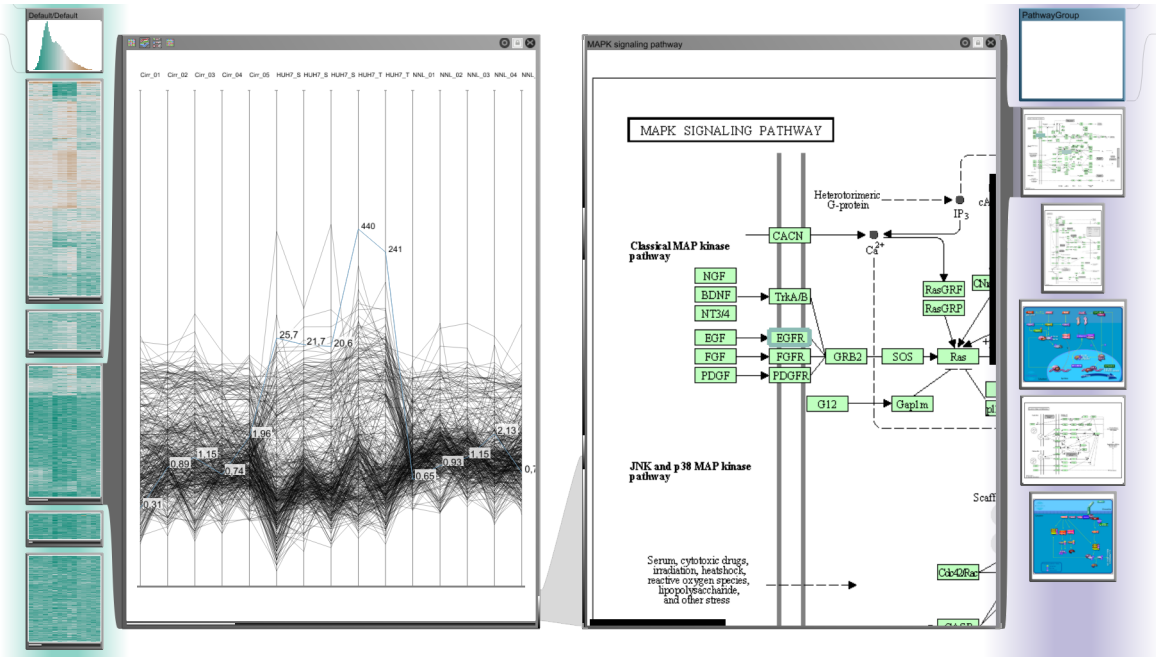


Figure 4.8: Focus duplicates are used for the detailed examination of gene expression data and a pathway. As there is enough space in the focus duplicate, the parallel coordinates plot shows certain details of the data that would not be visible in regular segment bricks. The focus duplicate of the pathway is zoomed in to a certain region in order to increase the size of its labels to make them legible.

VisBricks offers various possibilities to interact with bricks and column groups. Summary bricks can be dragged to move them to the focus or context regions and also to change the order of columns to make direct comparisons of all columns possible. The distance between columns can also be interactively adjusted. Increasing the space between column groups can typically reduce the clutter caused by links. Segment bricks can be moved horizontally to align bricks of different columns, thus making them easier to compare. Additionally, the size of each brick is adjustable and the ordering of segment bricks within a column can interactively be altered via drag and drop. At last, segment bricks can be split into two bricks by a connecting link, which results in one brick containing all elements the link was referring to, whereas the other one contains the rest.

The scalability of VisBricks is generally determined by the number of bricks that can be simultaneously displayed. On a conventional display up to 15 summary bricks can be accommodated in each context area and about 8-10 columns fit the focus area, resulting in a maximum of 40 column groups to be simultaneously integrated. Depending on the abstraction level of their visualizations, up to about 20 segment bricks can be shown within a column.

4.3 Case Studies

The TCGA consortium maintains an analysis pipeline called *Firehose*[1] that is used to perform preprocessing and a variety of biomedical analyses on their collected data. These analyses include clustering algorithms for the quantitative mRNA, miRNA, and methylation data, and the identification of copy number variations and mutated genes for all samples of patients that suffer from a certain kind of cancer. The relevant results of Firehose in terms of cancer subtype characterization are the quantitative data, different clusterings of this data that can be used to stratify patients into disjoint groups, and categorizations of copy number variations and gene mutations for each gene in each patient.

Our collaborators from the *Broad Institute of MIT and Harvard* are domain experts and use TCGA data to analyze different types of cancer. They were interested in exploring datasets for the brain cancer *Glioblastoma Multiforme* (GBM) [The08], which were prepared according to the results of Firehose and augmented by some additional stratifications provided by our collaborators, with our visualizations. The following case studies report observations that were made during the evaluations with our collaborators. Since an in-depth data analysis can only be performed within VisBricks, the evaluation focuses on this visualization. However, the DVI was always used to assign column groups from different datasets to VisBricks for their examination.

4.3.1 Comparison of Clusterings

Verhaak et al. [VHP*10] identified four mRNA gene expression subtypes for GBM: *Mesenchymal*, *proneural*, *neural*, and *classical*. However, the 'best' clustering reported by Firehose exhibited only three clusters. For that reason we wanted to compare the stratification of patients according to the classification by Verhaak et al. to clusterings for of three, four, and five clusters (see Figure 4.9). The first of our observations was made based on the pattern of links revealed by brushing a brick from the three cluster solution. The brick's data was split up mainly into two clusters in the four cluster solution and almost all of the data from these two clusters were united again in a segment brick in the five cluster solution. Though this behavior could be an artifact of the clustering algorithm, our collaborator stated that this could also be a biologically meaningful result due to a second observation: The aforementioned two clusters were actually a mix of the proneural and neural subtypes, whereas the other two clusters from the four cluster solution matched the mesenchymal and classical subtypes very accurately. Since Verhaak et al. also confirmed that tumors from the proneural and neural subtypes exhibit similar expression patterns, the clusterings actually represent reasonable solutions.

[1] gdac.broadinstitute.org/

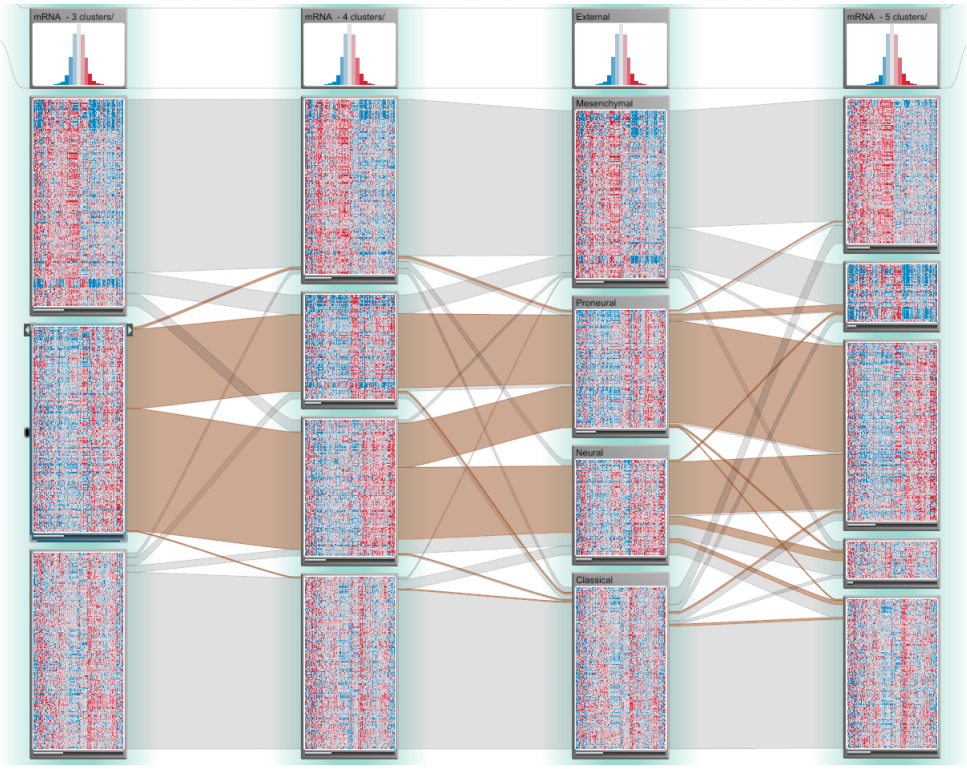


Figure 4.9: The comparison of the patient stratification according to the mRNA subtype classification for GBM by Verhaak et al. [VHP*10] to clusterings from Firehose. Columns 1, 2, and 4 represent the clusterings with three, four, and five clusters respectively. In Column 3, each subtype identified by Verhaak et al. (mesenchymal, proneural, neural, and classical) corresponds to one brick.

4.3.2 Combination of Methylation Data and Gene Mutation Status

Noushmehr et al. [NWD*10] identified three GBM subtypes by analyzing DNA methylation data. One of them is based on hypermethylation of certain regions of the genome, which causes a repression of gene expression in those regions. Additionally this subtype exhibits mutations of the gene IDH1 and falls in the proneural subtype in many cases. When our collaborators analyzed the methylation clustering with three clusters from Firehose they quickly realized that this subtype was not detected by the clustering algorithm since there was no strong association between any of the clusters to the mutation of IDH1. Therefore other clusterings were added using the DVI. One cluster from the clustering with eight clusters exhibited a distinct methylation pattern and also was the only cluster where IDH1 of all patients was mutated, whereas no patients of other clusters from this clustering were affected by a mutation of IDH1. This cluster was used to split one cluster from a two cluster solution, resulting in a total number of three clusters. As hypothesized by our collaborators, the newly created cluster was likely to refer to many patients with the Noushmehr et al. subtype, since all of them had mutations in IDH1 and there was

also a large overlap of this cluster with the proneural subtype. In order to find additional evidence that supported this hypothesis, a dependent column showing the survival curves of the patients that correspond to the three clusters of methylation data was created (see Figure 4.10). Indeed, the patients of the newly created cluster had better survival outcomes than patients from the other two clusters, which was also reported by Noushmehr et al.

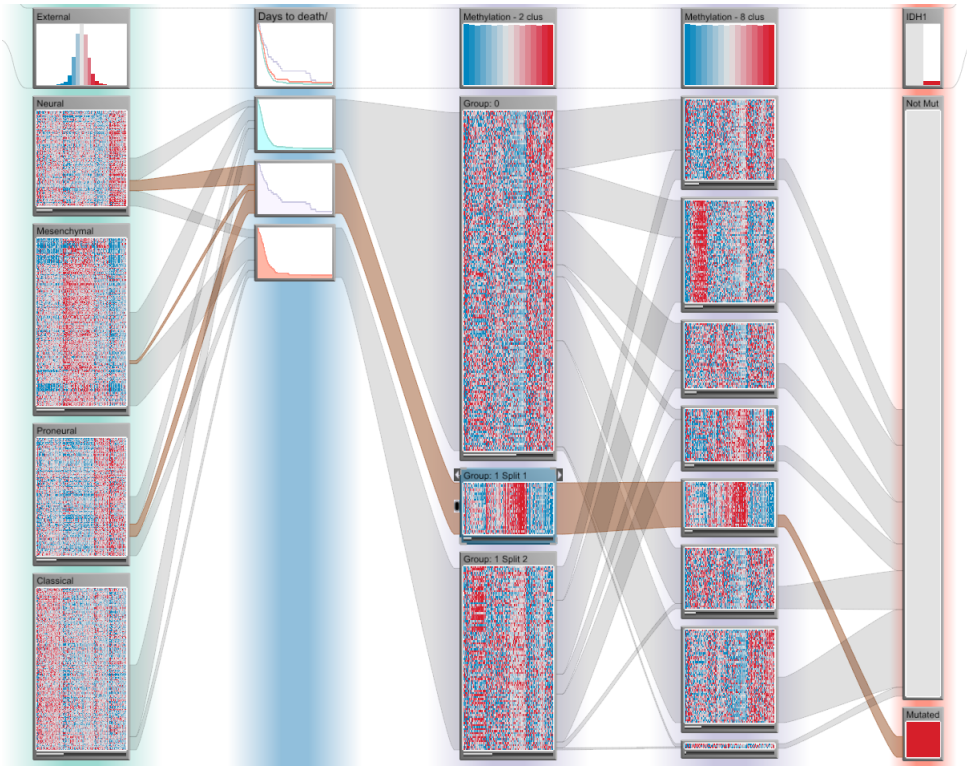


Figure 4.10: Combined analysis of different datasets to identify patients with a GBM subtype characterized by Noushmehr et al. [NWD*10]. Column 1 shows the mRNA subtype classification for GBM according to Verhaak et al. [VHP*10]. Column 2 shows the survival curves for the patients from the clusters of Column 3, which shows methylation data. This column was refined by splitting one of its clusters according to the data from one cluster from column 4. The clustering algorithm used for segmenting this column was the only one that detected a cluster exhibiting a characteristic methylation pattern and that was also strongly associated with gene mutations of IDH1. The mutation status of IDH1 (mutated, not mutated) is shown in Column 5.

4.4 Summary

Our prototypes of the Data-View Integrator and VisBricks can be used for the exploration and analysis of multiple datasets and have been specifically designed to integrate biological data. The force-directed layout of the DVI provides an intuitive overview of the present datasets and their relationships. With the DVI, interesting subsets of data can easily be defined for multi-dimensional datasets in the matrix representation of their data nodes. Subsets can interactively be assigned to VisBricks or other views for a detailed examination. This process is facilitated by using the two-layered layout, which provides a comprehensible overview of where the subsets of different datasets are currently displayed. VisBricks is used to investigate the subsets assigned using the DVI in detail. Each subset is indicated by a column of bricks that visualize the homogeneous groups the subset is segmented in. VisBricks supports various visualizations for bricks, which can be interactively switched. Bricks from neighboring columns are visually linked to indicate relationships between the data they display. Numerous possibilities are offered to interact with bricks in order to facilitate the comparison of data from different subsets or their detailed investigation. We evaluated our prototypes in case studies in the context of TCGA data, which showed that the DVI and VisBricks can be used to easily reproduce known results from the literature and to gain further insights.

Chapter 5

Implementation

The software prototypes of the Data-View Integrator and VisBricks were realized within the Caleydo framework [LSKS10], an information visualization system with the focus on the visual representation of genetical and clinical data. In this chapter we discuss how these prototypes are implemented. Section 5.1 briefly introduces different technologies that are used, and Section 5.2 discusses the key mechanics of the Caleydo framework. The main software components and concepts of the Data-View Integrator view and the VisBricks view are presented in Sections 5.3 and 5.4 respectively. Note, that this chapter only provides an overview of the system and many details are omitted since describing them would go beyond the scope of this document.

5.1 Used Technologies

The Caleydo framework is written in the *Java*[1] programming language and developed using the *Eclipse* [2] IDE under Linux and Windows operating systems. It uses the *Rich Client Platform (RCP)*[3], which is essentially a set of plugins that allows the development of rich client applications. One of the core components of RCP is the *Standard Widget Toolkit (SWT)*[4]. SWT is a portable toolkit that provides user-interface facilities, which are coupled with the underlying operating system, thus giving a native look and feel. As complex visualizations cannot be realized with common widgets, Caleydo uses *JOGL*[5], a library that grants Java programs access to *OpenGL*[6] APIs for hardware accelerated graphics.

[1] `http://www.java.com`
[2] `http://www.eclipse.org/`
[3] `http://wiki.eclipse.org/index.php/Rich\_Client\_Platform`
[4] `http://www.eclipse.org/swt/`
[5] `http://kenai.com/projects/jogl/pages/Home`
[6] `http://www.opengl.org/`

5.2 The Caleydo Framework

Caleydo is an information visualization framework that allows the rapid integration of new visualizations into a multiple view system. The most important aspects and basic mechanics of the framework are discussed in the following.

5.2.1 Views

In order to provide facilities that allow the independent development of different views, each view is realized as an RCP plugin. Basically Caleydo supports two kinds of views: SWT views, and OpenGL views. SWT views are rather simple views where no complex drawing is done. They consist of basic widgets such as buttons, lists, and checkboxes. OpenGL views are used to represent more sophisticated visualizations. The flexibility provided by the use of OpenGL allows to create visualizations such as parallel coordinates or heatmaps, which cannot be realized with the basic building blocks of SWT widgets.

Generally, all OpenGL views are subclasses of `AGLView`. The abstract methods that need to be implemented by its subclasses mainly concern view initialization and rendering. These methods are automatically called by the framework at appropriate times. For example, the rendering method of an OpenGL view is called in every frame after the view has been initialized, but only if it is currently visible. As an `AGLView` is not tightly coupled with the window frame it is displayed in, an OpenGL visualization can also be remotely rendered within another OpenGL view.

5.2.2 Layout in OpenGL Views

One strength of OpenGL is its flexibility to draw elements in a completely user-defined way. However, aligning the sizes and positions of multiple elements appropriately is often tedious. For that reason Caleydo provides a set of classes to facilitate the layouting of elements. The core of these classes is represented by an `ElementLayout`. This class essentially defines a rectangular drawing area within the display. It is associated with a `LayoutRenderer`, whose subclasses can be used to draw elements within the area defined by the `ElementLayout`. A special example of such a subclass is the `ViewLayoutRenderer`, which is used to remotely render an `AGLView`. The `Row` and `Column` classes are derivates from `ElementLayout` and can be used for nesting `ElementLayout` objects to define complex layouts for multiple elements. Once such a nested hierarchy is defined, the `LayoutManager` is used to recursively calculate the sizes and positions of the drawing areas of each `ElementLayout`, thus layouting the elements that are rendered by the associated `LayoutRenderer` objects.

5.2.3 Events

Events in Caleydo serve the communication between different views and also between SWT widgets and OpenGL views. The event system essentially follows the *observer pattern* [GHJV95, pp. 278-286], which is generally used to notify other objects about a change in a certain object. Event listeners register for a certain type of event at a globally accessible event publisher. When an event is triggered using the event publisher, all listeners that registered for that event type receive it. Events can contain virtually any information. For example, sending the IDs of currently selected elements in a view with an event enables the realization of linking and brushing.

5.2.4 Data Management

Caleydo supports the simultaneous analysis of multiple datasets of different types. Each dataset is represented by an instance of a subclass of `ADataDomain`. The subclass thereby specifies the data type. For example, pathways are represented by the `PathwayDataDomain`, and multi-dimensional datasets are reflected by derivates from `ATableBasedDataDomain`. The data of multi-dimensional datasets is stored in a `DataTable`, which is held by the associated data domain class. Each `DataTable` holds a set of `DimensionPerspective` and `RecordPerspective` objects, which are the implementation of the perspectives concept introduced in Section 3.2.1. The rows or columns defined for a perspective object are specified by its `VirtualArray`, which is essentially a list of indices that refer either to columns or rows from the `DataTable`. A `VirtualArray` also holds a `GroupList` consisting of `Group` objects. Each of these objects specifies a start and an end index for the corresponding `VirtualArray`, thus defining a group of columns or rows for the perspective. A `TablePerspective` holds the combination of a `DimensionPerspective` and `RecordPerspective` object to specify a subset from the `DataTable`. Therefore a `TablePerspective` can conceptually refer to a column group, and also to a segment group, when perspectives are created for the elements specified by `Group` objects. `TablePerspective` objects are stored in the corresponding data domain class. The relationships between different datasets are reflected in a `DataDomainGraph`. For each data domain object that corresponds to a dataset, a node is added to the graph. An edge between two nodes is created if the datasets contain elements of the same `IDCategory`.

5.3 The Data-View Integrator

As the DVI is a rather complex visualization, an OpenGL view is used for its realization. `GLDataViewIntegrator` is a subclass of `AGLView` and represents the core of the DVI view. It is responsible for creating and maintaining the graph of views and datasets, visualizing the graph, and handling events.

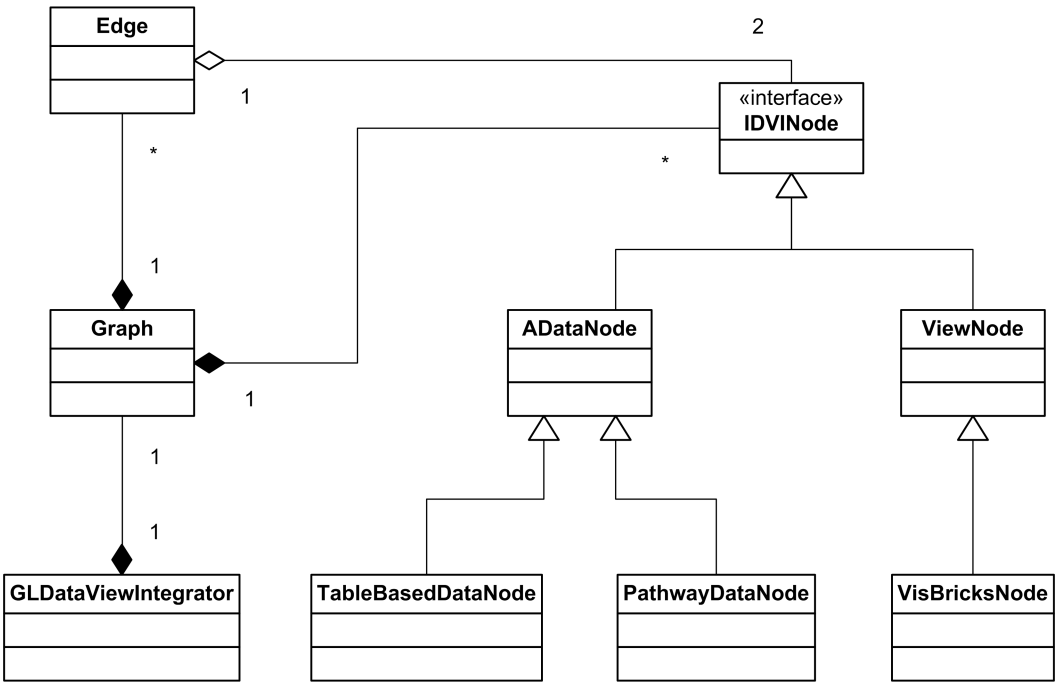


Figure 5.1: Components of the graph data structure and their dependencies in the DVI.

The main idea of the DVI is to bring views and datasets together in one graph visualization. The underlying data structure is represented by a `Graph` class that consists of `IDVINode` objects and `Edge` objects (see Figure 5.1). For each data domain object, a `ADataNode`, which implements `IDVINode`, is added to the `Graph`. The data type of the corresponding dataset is reflected by the concrete subclass of `ADataNode`. A `TableBasedDataNode` refers to a multi-dimensional dataset, whereas a `PathwayDataNode` represents pathways. Similarly to data nodes, a `ViewNode` is added for each view that is currently opened in the system and capable of displaying data. Since multiple column groups can be added to the VisBricks view, its node in the DVI has to be treated in a special way. For that reason a `VisBricksNode`, which is a subclass of `ViewNode`, is used to represent the VisBricks view. `Edge` objects that connect two data nodes are added to the `Graph` in accordance with the `DataDomainGraph`. An `Edge` that connects a data node and a view node is added, if the view that corresponds to the view node shows the data specified by a `TablePerspective` from the data node's data domain. The `Graph` is dynamically adapted to new conditions of the system. New nodes and edges are added when new views are opened, new datasets are loaded, or data is assigned to views. This is realized by listening to the corresponding events.

Generally, all nodes are rendered as rectangles that display elements such as labels or icons. These elements are layouted within a node using the Caleydo layouting classes described in Section 5.2.2. The main difference between the visual representations of the node classes is the way column group indicators are rendered. A `ViewNode` does not represent

them at all, a PathwayDataNode and a VisBricksNode renders them as a list of bars, and a TableBasedDataNode additionally provides the matrix representation. The TablePerspectiveListRenderer and the TablePerspectiveMatrixRenderer, both being a subclass of LayoutRenderer, are responsible for drawing this list and matrix respectively.

The position of the nodes within the view is automatically determined by one of two subclasses of AGraphLayout. The first one is the ForceDirectedGraphLayout, which is essentially an implementation of the algorithm of Kamada and Kawai [KK89] provided by Steffen Hadlak. The second subclass is the TwoLayeredGraphLayout, which positions data nodes at the bottom and view nodes at the top of the view.

The way edges are drawn mainly depends on which types of nodes they connect. Edges between data nodes are drawn as dashed lines and edges that connect data nodes and view nodes are drawn as bands. Each Edge is associated with an AEdgeRenderer, whose subclasses AEdgeLineRenderer and AEdgeBandRenderer determine whether the edge is drawn as a band or as a line. However, the concrete way an edge is drawn is also influenced by the current graph layout. For example, when the TwoLayeredGraphLayout is used, the bands must only connect to the bottom sides of a ViewNode, whereas a band can also connect to different sides in the ForceDirectedGraphLayout. Therefore, depending on the current conditions, one of the classes FreeLayoutEdgeLineRenderer, FreeLayoutEdgeBandRenderer, TwoLayeredEdgeLineRenderer, or TwoLayeredEdgeBandRenderer, which are all subclasses of AEdgeLineRenderer and AEdgeBandRenderer respectively, is used to

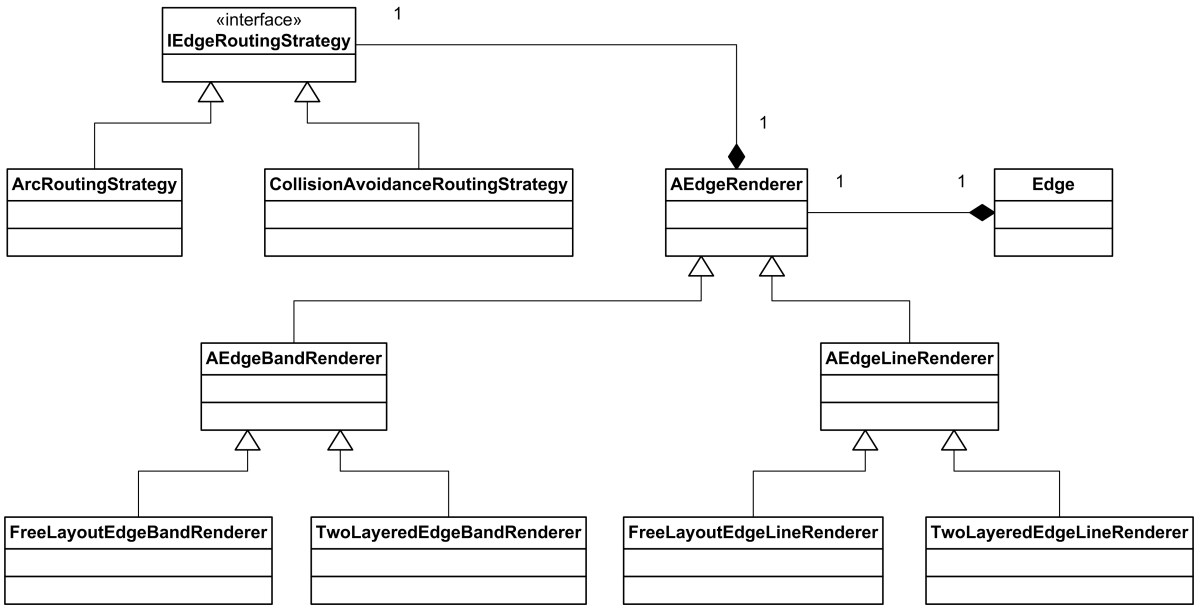


Figure 5.2: The basic components and their relationships responsible for drawing edges in the DVI.

draw the edge. Overall this approach follows the *strategy pattern* [GHJV95, pp. 295-302], which is used to define families of exchangeable algorithms. This pattern is also applied when it comes to edge routing. Each `AEdgeRenderer` is associated with an `IEdgeRoutingStrategy` (see Figure 5.2). This interface is implemented by the `CollisionAvoidanceRoutingStrategy` and the `ArcRoutingStrategy`. The former strategy is responsible for determining the anchor points of an edge at the bounding boxes of nodes to avoid occlusion, whereas the latter routes an edge in the shape of an arc, which is used for the edges between data nodes in the `TwoLayeredGraphLayout`.

There are two basic ways how data can be assigned to views. The first one is the creation of a new view within the system based on the data of a column group, which is specified by a `TablePerspective`. The second possibility is intended for the assignment of column groups to VisBricks by triggering the `AddGroupsToVisBricksEvent`, which contains the `TablePerspective` that refers to a column group.

5.4 VisBricks

Like the DVI, the concept of VisBricks is too complex to be implemented by an SWT view, so that an OpenGL view has to be used for its realization. The main class of VisBricks that is responsible for the whole visualization and handling events is `GLVisBricks`, a derivate of `AGLView`.

Another key class of VisBricks is `GLBrick`. It is also a subclass of `AGLView` and represents one brick in VisBricks. The type of the brick and its current mode are determined by the brick's `ABrickLayoutConfiguration`. Its concrete subclasses are `SummaryBrickLayout`, `CollapsedSummaryBrickLayout`, `SegmentBrickLayout`, and `CollapsedSegmentBrickLayout`. The mode of a brick can easily be switched by applying a different `ABrickLayoutConfiguration`, which is actually determined by the current `ABrickLayoutConfiguration`. For example, in order to switch to the correct compact mode of the brick, the `SegmentBrickLayout` replaces itself with the `CollapsedSegmentBrickLayout`. This behavior is based on the *state pattern* [GHJV95, pp. 287-294], which allows an object to change its behavior according to its internal state.

The type and mode of a brick is reflected by the way it is drawn, which is the responsibility of the brick's `ABrickLayoutConfiguration`. All elements that are shown within a brick are arranged using the layouting system of Caleydo. The most important element of a brick is the visualization of its associated data. Every `ABrickLayoutConfiguration` is associated with an exchangeable `LayoutRenderer`. Therefore visualizations can be integrated by using a `ViewLayoutRenderer`, which remotely renders a view, or any other subclass of `LayoutRenderer`, such as the `OverviewHeatMapRenderer`, which draws the abstracted heatmap representations introduced in Section 4.2.

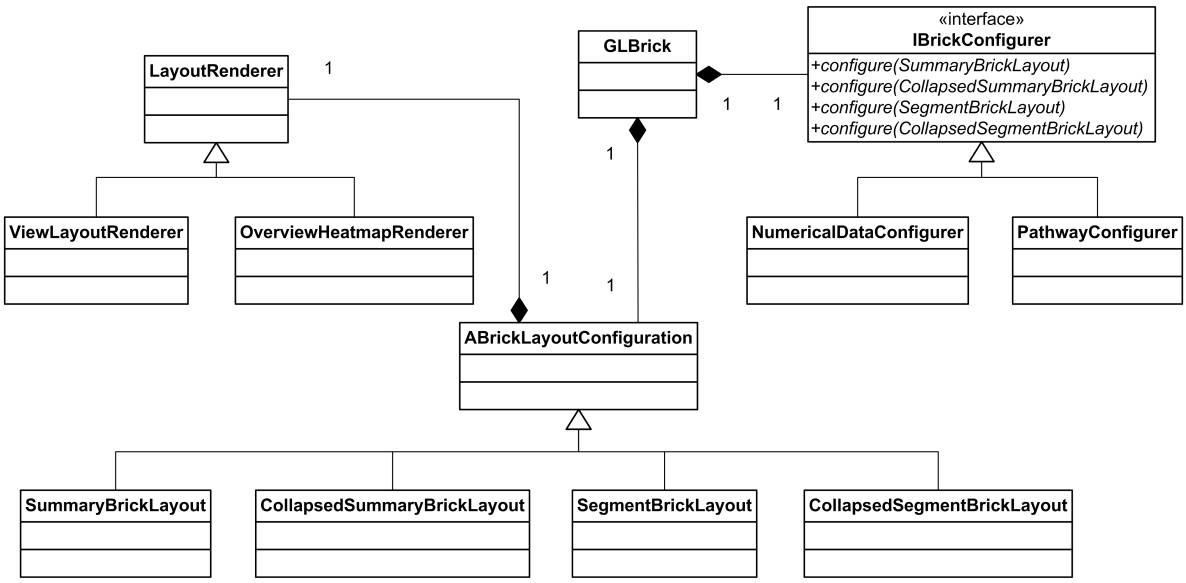


Figure 5.3: The components that determine the way a brick is rendered.

The visualizations that are available for a brick depend on its type and mode on the one hand, and on the data that needs to be presented on the other hand. We resolve this dependency using the *double-dispatch technique* that represents a part of the *visitor pattern* [GHJV95, pp. 308-319]. For each type of data an implementation of `IBrickConfigurer` is provided (see Figure 5.3). For example, the `NumericalDataConfigurer` is used for numerical multi-dimensional data, and the `PathwayDataConfigurer` is used for pathways. Each concrete `IBrickConfigurer` implements one `configure()` method for each `ABrickLayoutConfiguration` that takes the concrete subclass of `ABrickLayoutConfiguration` as an argument and defines the visualizations available for that `ABrickLayoutConfiguration`. The appropriate visualizations for a brick are therefore determined by calling the `configure()` method of the current `IBrickConfigurer` with the current `ABrickLayoutConfiguration` as parameter.

The whole layout of Visbricks is defined using the layout classes of Caleydo and all bricks in VisBricks are remotely rendered with a `ViewLayoutRenderer`. A column of bricks is added to VisBricks when `GLVisBricks` receives an `AddGroupsToVisBricksEvent`, which specifies a `TablePerspective` that refers to a column group. The data specified by such a `TablePerspective` is represented in a summary brick. A `TablePerspective` is also created for every segment group, whose data is visualized within a segment brick. Visual links connect segment bricks of neighboring columns that have common elements in their associated `TablePerspective` objects. The width of the link thereby depends on the amount of common elements. The pairwise similarity between the data of all bricks is calculated and stored in `SimilarityMap` objects, which are managed by the `RelationAnalyzer`.

Chapter 6

Conclusion and Future Work

In this thesis we presented a visualization system for the analysis of multiple datasets that facilitates the discovery of relationships in the data within and across datasets at different levels of granularity. This system consists of two novel visualization techniques: The Data-View Integrator and VisBricks.

The DVI integrates datasets and views as nodes in an abstract graph visualization. It typically represents the starting point in an analysis session by providing an overview of the information landscape formed by the datasets and showing how they are conceptually related. Additionally, the DVI serves as management tool for data and views. Using the DVI, a user can specify subsets for datasets and interactively assign them to VisBricks and other views for their investigation. By intuitively indicating in which views the different subsets are currently displayed, the DVI also provides orientation to the user throughout the analysis session.

VisBricks is an agile multiform visualization that is used to explore the subsets assigned using the DVI in detail. Each subset is individually visualized in a brick, the basic building block of VisBricks. A brick provides a selection of visual representations that suit the characteristics of its associated subset, and that can interactively be switched. Relationships between different subsets are indicated by visual links between bricks and also via linking and brushing. Besides providing an overview of how the different subsets are related and thus enabling their comparison, VisBricks also provides interactive drill-down features that allow the inspection of subsets up to the level of individual data items. The multiform property of bricks empowers VisBricks to describe existing visualization techniques such as Matchmaker [LSP*10] by using a heatmap in each brick to present its subset.

The usefulness of our visualization techniques was evaluated in case studies with domain experts using multiple genomic datasets that are used for cancer subtype characterization. The results of this evaluation are promising, as known results from the literature could easily be reproduced and new insights were gained.

Future Work

The sizes of the nodes and their contained elements in the DVI are optimized for being handy to interact with, so that all elements can easily be selected, or dragged and dropped. Of course, the sizes of these elements also determine a minimum size of the whole visualization by which the DVI is still useable. Currently, the DVI requires a lot of screen space in a typical analysis session with multiple datasets and views. Thus, only little screen space is given to other views that should be the simultaneously shown on the same display. This is especially an issue in single-display environments, when the DVI is used to provide orientation while actually exploring data in a different view. That issue could be addressed by introducing a compact mode for the two-layered layout. As in this layout only the minimum horizontal size of the DVI is affected by the number of displayed nodes, the vertical size of the view could be constantly reduced by minimizing the height of the nodes. Therefore the DVI could be displayed above or below another view without using too much screen space.

Another improvement of our system involves the process of configuring subsets and assigning them to views. In the current state, a user has to select a perspective combination in the matrix mode of a multi-dimensional data node and assign the resulting subset to a view in order to explore it. However, in many cases a user does not know in advance how the subset specified by a perspective combination will look like in a view and therefore cannot judge its significance for the analysis. For that reason a preview of the result of a perspective combination could be shown on demand before a subset is actually created. For example, a small abstract visualization of how the brick column of the resulting subset would look like in VisBricks could be displayed.

Additional support in the analysis process could be provided by pointing out potentially interesting subsets to the user. For instance, the column group indicators of those column groups could be highlighted, whose segment groups have a high correlation to the segment groups of other column groups that were already assigned to VisBricks.

List of Figures

Bibliography

[AA06] Andrienko N., Andrienko G.: *Exploratory analysis of spatial and temporal data: a systematic approach*. Springer-Verlag, 2006.

[AJL*07] Alberts B., Johnson A., Lewis J., Raff M., Roberts K., Walter P.: *Molecular Biology of the Cell*, 5 ed. Garland Science, Nov. 2007.

[AK02] Abello J., Korn J.: MGV: a system for visualizing massive multidigraphs. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (Mar. 2002), 21–38.

[AKK96] Ankerst M., Keim D. A., Kriegel H.: Circle segments: A technique for visually exploring large multidimensional data sets. In *Proceedings of the IEEE Conference on Visualization (Vis '96), Hot Topic Session* (San Francisco, CA, 1996).

[AMA07] Archambault D., Munzner T., Auber D.: TopoLayout: multilevel graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (2007), 305–317.

[And84] Anderson T. W.: *An introduction to multivariate statistical analysis*. Wiley, 1984.

[Ber10] Bertin J.: *Semiology of Graphics: Diagrams, Networks, Maps*, first published in french in 1967 ed. ESRI Press, Dec. 2010.

[BH86] Barnes J., Hut P.: A hierarchical O(N log n) force-calculation algorithm. *Nature 324*, 6096 (Dec. 1986), 446–449.

[BLS99] Brandstädt A., Le V. B., Spinrad J. P.: *Graph classes: a survey*. SIAM Monographs on Discrete Mathematics and Applications, 1999.

[Bre09] Brewer C. A.: Colorbrewer. http://colorbrewer2.org/, 2009. last accessed Mar. 07, 2012.

[CC05] Craft B., Cairns P.: Beyond guidelines: What can we learn from the visual information seeking mantra? In *IV '05: Proceedings on Information Visualisation* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 110–118.

[CC07] Collins C., Carpendale S.: VisLink: revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '07) 13*, 6 (2007), 1192–1199.

[Cha77] Chambers J. M.: *Computational methods for data analysis*. Wiley, 1977.

[Che73] Chernoff H.: The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association 68*, 342 (June 1973), 361–368.

[Che76] Chen P. P. S.: The entity-relationship model toward a unified view of data. *ACM Transactions on Database Systems (TODS '76) 1*, 1 (1976), 9–36.

[Cim06] Cimikowski R.: An analysis of some linear graph layout heuristics. *Journal of Heuristics 12*, 3 (May 2006), 143–153.

[CM88] Cleveland W. C., McGill M. E.: *Dynamic Graphics for Statistics*. CRC Press, Inc., 1988.

[CMS99] Card S. K., Mackinlay J. D., Shneiderman B. (Eds.): *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[Cod70] Codd E. F.: A relational model of data for large shared data banks. *Commun. ACM 13*, 6 (June 1970), 377–387.

[CPC09] Collins C., Penn G., Carpendale S.: Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '09) 15*, 6 (2009), 1009–1016.

[CWRY06] Cui Q., Ward M., Rundensteiner E., Yang J.: Measuring data abstraction quality in multiresolution visualizations. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 709–716.

[CYHY09] Cai Y., Yu X., Hu S., Yu J.: A brief review on the mechanisms of miRNA regulation. *Genomics, Proteomics & Bioinformatics 7*, 4 (Dec. 2009), 147–154.

[DBETT99] Di Battista G., Eades P., Tamassia R., Tollis I.: *Graph drawing : algorithms for the visualization of graphs*. Prentice Hall, Upper Saddle River N.J., 1999.

[DE02] Dix A., Ellis G.: By chance: enhancing interaction with large data sets through statistical sampling. In *Proceedings of the ACM Conference on Advanced Visual Interfaces (AVI '02)* (2002), ACM Press, pp. 167–176.

[DG02] Diehl S., Görg C.: Graphs, they are changing. In *Revised Papers from the 10th International Symposium on Graph Drawing* (London, UK, 2002), GD '02, Springer-Verlag, pp. 23–30.

[DK08] Dokulil J., Katreniakova J.: Edge routing with fixed node positions. In *Information Visualisation, 2008. IV '08. 12th International Conference* (July 2008), IEEE, pp. 626–631.

[DMM97] Dogrusöz U., Madden B., Madden P.: Circular layout in the graph layout toolkit. In *Proceedings of the Symposium on Graph Drawing* (London, UK, 1997), GD '96, Springer-Verlag, pp. 92–100.

[DPS02] Díaz J., Petit J., Serna M.: A survey of graph layout problems. *ACM*

Comput. Surv. 34, 3 (Sept. 2002), 313–356.

[DRK97] Derthick M., Roth S. F., Kolojejchick J.: Coordinating declarative queries with a direct manipulation data exploration environment. In *, IEEE Symposium on Information Visualization, 1997. Proceedings* (Oct. 1997), IEEE, pp. 65–72.

[Ead84] Eades P.: A heuristic for graph drawing. In *Congressus Numerantium* (1984), vol. 42, pp. 149–160.

[ESBB98] Eisen M. B., Spellman P. T., Brown P. O., Botstein D.: Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences USA 95*, 25 (1998), 14863–14868.

[FdOL03] Ferreira de Oliveira M. C., Levkowitz H.: From visual data exploration to visual data mining: A survey. *IEEE Transactions on Visualization and Computer Graphics 9*, 3 (2003), 378–394.

[FPF*06] Freeman J. L., Perry G. H., Feuk L., Redon R., McCarroll S. A., Altshuler D. M., Aburatani H., Jones K. W., Tyler-Smith C., Hurles M. E., Carter N. P., Scherer S. W., Lee C.: Copy number variation: New insights in genome diversity. *Genome Research 16*, 8 (2006), 949–961.

[FR91] Fruchterman T. M. J., Reingold E. M.: Graph drawing by force-directed placement. *Softw. Pract. Exper. 21*, 11 (Nov. 1991), 1129–1164.

[Fur86] Furnas G. W.: Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '86)* (New York, NY, USA, 1986), ACM Press, pp. 16–23.

[FWD*03] Fekete J., Wang D., Dang N., Aris A., Plaisant C.: Interactive poster: Overlaying graph links on treemaps. In *Proceedings of the IEEE Symposium on Information Visualization Conference Compendium (InfoVis '03)* (2003), IEEE Computer Society Press, pp. 82–83.

[GAB07] Goldberg A. D., Allis C. D., Bernstein E.: Epigenetics: A landscape takes shape. *Cell 128*, 4 (2007), 635–638.

[GFC05] Ghoniem M., Fekete J., Castagliola P.: On the readability of graphs using node-link and matrix-based representations: A controlled experiment and statistical analysis. *Information Visualization 4*, 2 (June 2005), 114–135.

[GG99] Goebel M., Gruenwald L.: A survey of data mining and knowledge discovery software tools. *SIGKDD EXPLORATIONS 1* (1999), 20–33.

[GHJV95] Gamma E., Helm R., Johnson R., Vlissides J.: *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman, 1995.

[GK03] Graham M., Kennedy J.: Using curves to enhance parallel coordinate visualisations. In *IV '03: Proceedings of the Seventh International Conference on*

Information Visualization (Washington, DC, USA, 2003), IEEE Computer Society, pp. 10–16.

[GK07] Gansner E. R., Koren Y.: Improved circular layouts. In *Proceedings of the 14th international conference on Graph drawing* (Berlin, Heidelberg, 2007), GD'06, Springer-Verlag, pp. 386–398.

[GS01] Gegenfurtner K. R., Sharpe L. T.: *Color vision: from genes to perception.* Cambridge University Press, May 2001.

[Hea96] Healey C. G.: Choosing effective colours for data visualization. In *Proceedings of the IEEE Conference on Visualization (Vis '96)* (1996), IEEE Computer Society Press, pp. 263–270.

[Hel02] Heller M. J.: DNA microarray technology: Devices, systems, and applications. *Annual Review of Biomedical Engineering 4*, 1 (Aug. 2002), 129–153.

[HFM07] Henry N., Fekete J. D., McGuffin M. J.: NodeTrix: a hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '07) 13*, 6 (2007), 1302–1309.

[Hol06] Holten D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '06) 12*, 5 (2006), 741–748.

[HvW09] Holten D., van Wijk J.: Force-directed edge bundling for graph visualization. *Computer Graphics Forum (EuroVis '09) 28*, 3 (2009), 983–990.

[ID90] Inselberg A., Dimsdale B.: Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *Proceedings of the IEEE Conference on Visualization (Vis '90)* (San Francisco, CA, USA, 1990), pp. 361–378.

[JM97] Jünger M., Mutzel P.: 2-layer straightline crossing minimization: performance of exact and heuristic algorithms. *J. GRAPH ALGORITHMS APPL 1* (1997), 1–25.

[JMF99] Jain A. K., Murty M. N., Flynn P. J.: Data clustering: a review. *ACM Comput. Surv. 31*, 3 (Sept. 1999), 264–323.

[JS91] Johnson B., Shneiderman B.: Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the IEEE Conference on Visualization (Vis '91)* (1991), pp. 284–291.

[KAG*08] Kanehisa M., Araki M., Goto S., Hattori M., Hirakawa M., Itoh M., Katayama T., Kawashima S., Okuda S., Tokimatsu T., Yamanishi Y.: KEGG for linking genomes to life and the environment. *Nucleic Acids Research 36*, Database-Issue (2008), 480–484.

[KBH06] Kosara R., Bendix F., Hauser H.: Parallel sets: Interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (2006), 558–568.

[KEC06] Keller R., Eckert C. M., Clarkson P. J.: Matrices or node-link diagrams: Which visual representation is better for visualising connectivity models? *Information Visualization 5*, 1 (Mar. 2006), 62–76.

[Kei02] Keim D. A.: Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (Mar. 2002), 1–8.

[KHG03] Kosara R., Hauser H., Gresh D. L.: An interaction view on information visualization. In *State-of-the-Art Proceedings of EUROGRAPHICS 2003 (EG 2003)* (2003), pp. 123–137.

[KK89] Kamada T., Kawai S.: An algorithm for drawing general undirected graphs. *Inf. Process. Lett. 31*, 1 (Apr. 1989), 7–15.

[KKEM10] Keim D. A., Kohlhammer J., Ellis G., Mansmann F. (Eds.): *Mastering The Information Age - Solving Problems with Visual Analytics*. Eurographics, 2010.

[Klö02] Klösgen W.: Types and forms of data. In *Handbook of data mining and knowledge discovery*, Klösgen W., Zytkow J. M., (Eds.). Oxford University Press, 2002, pp. 33–44.

[Kob12] Kobourov S. G.: Force-directed drawing algorithms. In *Handbook of Graph Drawing and Visualization*, Tamassia R., (Ed.). CRC Press, 2012. In Press.

[Kos10] Kosara R.: Turning a table into a tree: Growing parallel sets into a purposeful project. In *Beautiful Visualization: Looking at Data through the Eyes of Experts*, Steele J., Iliinsky N., (Eds.). O'Reilly, 2010, pp. 193–204.

[KS02] Kreuseler M., Schumann H.: A flexible approach for visual data mining. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 39–51.

[LPP*06] Lee B., Plaisant C., Parr C. S., Fekete J., Henry N.: Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization* (New York, NY, USA, 2006), BELIV '06, ACM, pp. 1–5.

[LSKS10] Lex A., Streit M., Kruijff E., Schmalstieg D.: Caleydo: Design and evaluation of a visual analysis framework for gene expression data in its biological context. In *Proceeding of the IEEE Symposium on Pacific Visualization (PacificVis '10)* (2010), IEEE Computer Society Press, pp. 57–64.

[LSP*10] Lex A., Streit M., Partl C., Kashofer K., Schmalstieg D.: Comparative analysis of multidimensional, quantitative data. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '10) 16*, 6 (2010), 1027–1035.

[LSS*11] Lex A., Schulz H., Streit M., Partl C., Schmalstieg D.: VisBricks: multiform visualization of large, inhomogeneous data. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '11) 17*, 12 (2011), 2291–2300.

[LSS*12] Lex A., Streit M., Schulz H., Partl C., Schmalstieg D., Park P. J.,

Gehlenborg N.: StratomeX: visual analysis of Large-Scale heterogeneous genomics data for cancer subtype characterization. *Conditionally accepted for: Computer Graphics Forum (EuroVis '12) 31*, 3 (2012), fff–lll.

[LWW90] LeBlanc J., Ward M. O., Wittels N.: Exploring n-dimensional databases. In *Proceedings of the IEEE Conference on Visualization (Vis '90)* (Oct. 1990), IEEE, pp. 230–237.

[MELS95] Misue K., Eades P., Lai W., Sugiyama K.: Layout adjustment and the mental map. *Journal of Visual Languages & Computing 6*, 2 (June 1995), 183–210.

[Mis06] Misue K.: Drawing bipartite graphs as anchored maps. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60* (Darlinghurst, Australia, Australia, 2006), APVis '06, Australian Computer Society, Inc., pp. 169–177.

[MW95] Martin A. R., Ward M. O.: High dimensional brushing for interactive exploration of multivariate data. In *Proceedings of the IEEE Conference on Visualization (Vis '95)* (1995), IEEE Computer Society Press, pp. 271–278.

[MXH*03] MacEachren A., Xiping D., Hardisty F., Guo D., Lengerich G.: Exploring high-d spaces with multiform matrices and small multiples. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '03)* (2003), IEEE Computer Society Press, pp. 31–38.

[NCS02] North C., Conklin N., Saini V.: Visualization schemas for flexible information visualization. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '02)* (2002), IEEE, pp. 15–22.

[NS97] North C., Shneiderman B.: *A Taxonomy of Multiple Window Coordinations.* Tech. Rep. CS-TR-3854, University of Maryland Computer Science Dept., 1997.

[NS00] North C., Shneiderman B.: Snap-Together visualization: A user interface for coordinating visualizations via relational schemata. In *Proceedings of the ACM Conference on Advanced Visual Interfaces (AVI '00)* (2000), ACM, pp. 128–135.

[NUUT07] Naud A., Usui S., Ueda N., Taniguchi T.: Visualization of documents and concepts in neuroinformatics with the 3D-SE viewer. *Frontiers in Neuroinformatics 1* (2007), 1–6.

[NWD*10] Noushmehr H., Weisenberger D. J., Diefes K., Phillips H. S., Pujara K., Berman B. P., Pan F., Pelloski C. E., Sulman E. P., Bhat K. P., Verhaak R. G. W., Hoadley K. A., Hayes D. N., Perou C. M., Schmidt H. K., Ding L., Wilson R. K., Van Den Berg D., Shen H., Bengtsson H., Neuvial P., Cope L. M., Buckley J., Herman J. G., Baylin S. B., Laird P. W., Aldape K.: Identification of a CpG island methylator phenotype that defines a distinct subgroup of glioma. *Cancer Cell 17*, 5 (2010), 510–522. PMID: 20399149.

[PHJ*10] Pham T., Hess R., Ju C., Zhang E., Metoyer R.: Visualization of diversity in large multivariate data sets. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '10) 16*, 6 (2010), 1053–1062.

[Pur97] Purchase H. C.: Which aesthetic has the greatest effect on human understanding? In *Proceedings of the 5th International Symposium on Graph Drawing* (London, UK, 1997), GD '97, Springer-Verlag, pp. 248–261.

[PXY*05] Phan D., Xiao L., Yeh R., Hanrahan P., Winograd T.: Flow map layout. In *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 219–224.

[QE01] Quigley A., Eades P.: FADE: graph drawing, clustering, and visual abstraction. In *Proceedings of the 8th International Symposium on Graph Drawing* (London, UK, 2001), GD '00, Springer-Verlag, pp. 197–210.

[RC94] Rao R., Card S. K.: The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '94)* (New York, NY, USA, 1994), ACM, pp. 318–322.

[RD83] Ramos E., Donoho D.: *The car dataset.* 1983. Available at: http://lib.stat.cmu.edu/datasets/.

[RKS11] Rohn H., Klukas C., Schreiber F.: Creating views on integrated multidomain data. *Bioinformatics 27*, 13 (2011), 1839–1845.

[RMF09] Rufiange S., McGuffin M. J., Fuhrman C.: Visualisation hybride des liens hiérarchiques incorporant des treemaps dans une matrice d'adjacence. In *Proceedings of the Conference on Association Francophone d'Interaction Homme-Machine (IHM '09)* (2009), pp. 51–54.

[RNP*10] Rich J. T., Neely J. G., Paniello R. C., Voelker C. C. J., Nussenbaum B., Wang E. W.: A practical guide to understanding Kaplan-Meier curves. *Otolaryngology–Head and Neck Surgery 143*, 3 (2010), 331–336.

[Rob00] Roberts J. C.: Multiple-view and multiform visualization. In *Proceedings of the SPIE Conference on Visualization and Data Analysis (VDA '02)* (2000), vol. 3960, pp. 176–185.

[RRB*04] Rosario G. E., Rundensteiner E. A., Brown D. C., Ward M. O., Huang S.: Mapping nominal values to numbers for effective visualization. *Information Visualization 3*, 2 (2004), 80–95.

[Sch10] Schulz H.: *Explorative Graph Visualization.* PhD thesis, University of Rostock, 2010.

[Shn96] Shneiderman B.: The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*

(VL '96) (1996), IEEE, pp. 336–343.

[SJ08] Shendure J., Ji H.: Next-generation DNA sequencing. *Nature Biotechnology 26*, 10 (Oct. 2008), 1135–1145.

[SJUS08] Schulz H., John M., Unger A., Schumann H.: Visual analysis of bipartite biological networks. In *Proceedings of the Eurographics Workshop on Visual Computing for Biomedicine (VCBM '08)* (2008), Eurographics, pp. 135–142.

[SR06] Siirtola H., Räihä K.: Interacting with parallel coordinates. *Interacting with Computers 18*, 6 (Dec. 2006), 1278–1309.

[SS06] Schulz H. J., Schumann H.: Visualizing graphs - a generalized view. In *Tenth International Conference on Information Visualization, 2006. IV 2006* (July 2006), IEEE, pp. 166–173.

[SSL*11] Streit M., Schulz H., Lex A., Schmalstieg D., Schumann H.: Model-driven design for the visual analysis of heterogeneous data. *IEEE Transactions on Visualization and Computer Graphics PP*, 99 (2011), 1–1.

[STH02] Stolte C., Tang D., Hanrahan P.: Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 52–65.

[STT81] Sugiyama K., Tagawa S., Toda M.: Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics 11*, 2 (Feb. 1981), 109–125.

[SWS*11] Steinberger M., Waldner M., Streit M., Lex A., Schmalstieg D.: Context-preserving visual links. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '11) 17*, 12 (Dec. 2011), 2249–2258.

[SZ00] Stasko J., Zhang E.: Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Proceedings of the IEEE Symposium on Information Vizualization (InfoVis '00)* (2000), IEEE Computer Society Press, pp. 57–65.

[The08] The Cancer Genome Atlas Research Network: Comprehensive genomic characterization defines human glioblastoma genes and core pathways. *Nature 455*, 7216 (Oct. 2008), 1061–1068.

[TPRH11] Turkay C., Parulek J., Reuter N., Hauser H.: Integrating cluster formation and cluster evaluation in interactive visual analysis. In *Proceedings of the Spring Conference on Computer Graphics (SCCG '11)* (2011).

[vH03] van Ham F.: Using multilevel call matrices in large software projects. In *IEEE Symposium on Information Visualization, 2003. INFOVIS 2003* (Oct. 2003), IEEE, pp. 227–232.

[VHP*10] Verhaak R. G., Hoadley K. A., Purdom E., Wang V., Qi Y., Wilkerson M. D., Miller C. R., Ding L., Golub T., Mesirov J. P., Alexe

G., Lawrence M., O'Kelly M., Tamayo P., Weir B. A., Gabriel S., Winckler W., Gupta S., Jakkula L., Feiler H. S., Hodgson J. G., James C. D., Sarkaria J. N., Brennan C., Kahn A., Spellman P. T., Wilson R. K., Speed T. P., Gray J. W., Meyerson M., Getz G., Perou C. M., Hayes D. N.: Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in PDGFRA, IDH1, EGFR, and NF1. *Cancer Cell 17*, 1 (Jan. 2010), 98–110.

[vHvW02] van Ham F., van Wijk J. J.: Beamtrees: compact visualization of large hierarchies. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002* (2002), IEEE, pp. 93–100.

[Wal01] Walshaw C.: A multilevel algorithm for force-directed graph drawing. In *Graph Drawing*, Marks J., (Ed.), vol. 1984. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 171–182.

[Wat02] Wattenberg M.: Arc diagrams: visualizing structure in strings. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002* (2002), IEEE, pp. 110–116.

[Wer23] Wertheimer M.: Untersuchungen zur Lehre von der Gestalt. II. *Psychologische Forschung 4*, 1 (1923), 301–350.

[WL96] Wegman E. J., Luo Q.: High dimensional clustering using parallel coordinates and the grand tour. *COMPUTING SCIENCE AND STATISTICS 28*, 124 (1996), 361–368.

[WOA*01] Woodruff A., Olston C., Aiken A., Chu M., Ercegovac V., Lin M., Spalding M., Stonebraker M.: DataSplash: a direct manipulation environment for programming semantic zoom visualizations of tabular data. *Journal of Visual Languages & Computing 12*, 5 (2001), 551–571.

[WPCM02] Ware C., Purchase H., Colpoys L., McGill M.: Cognitive measurements of graph aesthetics. *Information Visualization 1*, 2 (June 2002), 103–110.

[WPL*10] Waldner M., Puff W., Lex A., Streit M., Schmalstieg D.: Visual links across applications. In *Proceedings of the Conference on Graphics Interface (GI '10)* (2010), Canadian Human-Computer Communications Society, pp. 129–136.

[ZKG09] Zhou J., Konecni S., Grinstein G.: Visually comparing multiple partitions of data with applications to clustering. In *Proceedings of the SPIE Conference on Visualization and Data Analysis (VDA '09)* (2009), SPIE, pp. 72430J–1–12.

[ZMC05] Zhao S., McGuffin M., Chignell M.: Elastic hierarchies: combining treemaps and node-link diagrams. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '05)* (2005), IEEE Computer Society Press, pp. 57–64.