



Universität Rostock

Fakultät für Informatik und Elektrotechnik

Graph Visualization in Space and Time

Dissertation

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

eingereicht am 24. Juni 2014, Rostock
von **Steffen Hadlak**
geboren am 21. Januar 1985 in Rostock
wohnhaft in Rostock

Principal Advisor:

*Prof. Dr.-Ing. habil. Heidrun Schumann
Universität Rostock, Deutschland*

External Reviewers:

*Prof. Dr.-Ing. Bodo Urban
Fraunhofer-IGD, Universität Rostock, Deutschland*

*Prof. Dr.-Ing. Andreas Kerren
Linnaeus University, Schweden*

Date of Defense:

27. November 2014

Schlagwörter:

Graph Visualisierung, Explizite und Implizite Baumlayouts, Clustering dynamischer Graphen, Degree-of-Interest Funktionen, Kombination von Visualisierungen

Keywords:

Graph visualization, explicit and implicit tree layouts, dynamic graph clustering, Degree-of-Interest functions, combination of visualizations

Klassifikation / Classification (ACM CCS 2012):

Human-centered computing → Information visualization; Networks → Network properties → Network structure, Network dynamics

Kurzfassung

Die visuelle Analyse von Graphen spielt eine große Rolle in vielen Wissenschaftsgebieten. Sie umfasst eine **Vielzahl** verschiedener Aspekte, wie der Graphstruktur und damit verbundener Attribute in ihrem räumlichen und zeitlichen Kontext. Aufgrund der steigenden **Größe** dieser Graphen wird deren Darstellung immer problematischer und bedingt eine Unzahl verschiedener Visualisierungen. Da der Fokus während der Analyse oft wechselt, wird dieses Problem sogar noch schwieriger, so dass dem Anwender ermöglicht werden muss zwischen diesen Techniken **flexibel** wechseln zu können.

Diese drei Herausforderungen werden in dieser Arbeit adressiert. Dafür wird zunächst die Handhabung der verschiedenen Aspekte und dabei auftretender Probleme anhand zweier genereller Ansätze für die Visualisierung von Bäumen exemplarisch diskutiert. Diese basieren auf einer Familie von punktbasierten Layouts und auf einem Design Space, der die grundsätzlichen Design Entscheidungen impliziter Visualisierungen abdeckt.

Anschließend werden neue Methoden für die Reduktion der Größe der Graphen eingeführt. Diese basieren zum einen auf der Abstraktion des strukturellen oder zeitlichen Aspekts und zum anderen auf der Verwendung von Degree-of-Interest Funktionen um interessante Knoten, Kanten und Zeitpunkte zu bestimmen. Mithilfe dieser Techniken kann der Anwender auch bei der visuellen Analyse großer Graphen unterstützt werden.

Abschließend werden neue Ansätze vorgestellt, um die verschiedenen Visualisierungen zusammenzuführen und damit eine flexible Analyse zu ermöglichen. Diese Ansätze umfassen die In Situ Visualisierung, die Portale verwendet um Visualisierungen auch lokal kombinieren zu können, und eine abstrakte Visualisierung aller Aspekte, die eine Synchronisierung verschiedener Visualisierungen erlaubt.

Abstract

The visual analysis of graphs plays an important role in many fields and includes a **diversity** of aspects such as the graphs' structure and associated attributes in their spatial and temporal context. Because of their increasing **size**, their visualization becomes more and more difficult and necessitates a multitude of different visualization techniques. This problem becomes even more severe as with a changing analysis focus on the graph, the analyst needs to **flexibly** switch between different visualizations at any time.

This thesis aims at solving these three challenges. First, the handling of the diversity and emerging problems are discussed exemplary for two approaches each providing a multitude of differently suited tree visualizations. Here, a family of point-based tree layouts is introduced that consists of layouts with a similar layout scheme. And an implicit tree visualization design space is derived by identifying common design decisions.

For a scalable analysis of large graphs, new reduction approaches are introduced. These approaches are based on the one hand on clustering techniques abstracting either the structural or temporal aspect of the graphs. And on the other hand they rely on Degree-of-Interest functions to discern interesting nodes, edges and time points.

Finally, to bring the different visualizations together and thus allow a flexible analysis, novel approaches for their combination and synchronization are introduced. These approaches include the in situ visualization that is based on portals to allow a local combination of visualizations and a novel abstract overview of all aspect allowing the synchronization of multiple visualizations.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mir bei der Erstellung dieser Dissertation mit Tat und Rat zur Seite gestanden haben. Dieser Dank gilt vor allem Professorin Heidrun Schumann, die mir mit ihren stets konstruktiven Gesprächen eine mehr konzeptionelle Sicht auf die Probleme nähergebracht und ermöglicht hat. Auch gebührt mein Dank den Professoren Bodo Urban und Andreas Kerren, mit denen ich sehr interessante Gespräche führen durfte, die mich in der Abhandlung dieser Arbeit sehr unterstützten.

Des Weiteren danke ich meinem langjährigen zunächst Mentor und jetzt geschätztem Kollegen Hans-Jörg Schulz, der mich seit meinem Studium mit interessanten Themen, Diskussionen und Gesprächen unterstützt hat. Zudem danke ich meinen Kollegen vom Graduiertenkolleg, dem Computergrafiklehrstuhl und des Fraunhofer mit ihren vielen Ideen und Diskussionsrunden. Außerdem danke ich Youwei Zheng, Ron Henkel, Till Wollenberg und Georg Fuchs, deren praktische Probleme und Feedback mir geholfen haben, die Anwendbarkeit und Nützlichkeit meiner neuen Ansätze auszuprobieren und zu demonstrieren.

Natürlich wäre ich nie so weit gekommen, wenn mich meine geschätzte Familie, allen voran meine Eltern und meine Schwester, und meine Freunde nicht all die Zeit moralisch als auch mit Speis, Trank und manchmal notwendiger Ablenkung unterstützt hätten. Gerade aber meiner Verlobten Johanna muss ich meinen größten Dank aussprechen, die mir in allen Lebenslagen – auch wenn die Abende mal wieder länger wurden – mit Verständnis und einem Lächeln stets zur Seite stand.

Zu guter Letzt möchte ich auch allen interessierten Lesern dieser Dissertation danken. Last but not least, I wish to thank all interested readers of this dissertation.

Contents

1	Introduction	1
1.1	Motivation and Problem Statement	1
1.2	Research Questions	2
1.3	Outline and Contribution	3
2	Basics and Related Work	7
2.1	Terms and Definition	7
2.1.1	Graph classes	8
2.1.2	Attributes	8
2.1.3	Dynamic graphs	9
2.2	Visualization of Graphs	10
2.2.1	Visualization of Structure	10
2.2.1.1	Matrix Representations	10
2.2.1.2	Node-Link Representations	11
2.2.1.3	Implicit Representations	12
2.2.1.4	Hybrid Representations	13
2.2.2	Visualization of Multiple Attributes	14
2.2.2.1	Mapping to Visual Variables	14
2.2.2.2	Integration of Glyphs	17
2.2.2.3	Tabular Representations	18
2.2.3	Visualization of the Spatial Context	19
2.2.4	Visualization of the Temporal Context	22
2.2.4.1	Focus on Structure	22
2.2.4.2	Focus on Time	26
2.2.4.3	Balancing the Focus between Structure & Time	29
2.2.5	Visualization of Multiple Aspects	32
2.3	Scalability	33
2.3.1	Abstraction	34
2.3.1.1	Abstraction of the Structure	34
2.3.1.2	Abstraction of the Temporal Aspect	36
2.3.1.3	Abstraction of Structure and Time	37
2.3.2	Selection	39
2.4	Summary and Challenges	41
3	Novel Techniques to Visualize Graphs	45
3.1	Introduction	45

3.2	A Point-Based Layout for Large Hierarchies	45
3.2.1	Basic Point-Based Layout	46
3.2.2	A Generalization to Accommodate the Tree Structure	52
3.2.2.1	Selection of presets	53
3.2.2.2	Assignment of nodes	59
3.2.2.3	Combination of presets	60
3.2.3	Visualizing Attributes for Large Structures	62
3.2.4	Extension to Spatial Context	66
3.2.4.1	Layout to Use Space Efficiently	67
3.2.4.2	Layout to Facilitate Visual Comparison	69
3.2.5	Extension to Temporal Context	70
3.2.6	Use Case	74
3.2.6.1	Visualizing the Hierarchical Structure of Time	75
3.2.6.2	Embedding Into the Map Display	77
3.2.7	Conclusion	80
3.3	A Design Space for Implicit Tree Visualization	81
3.3.1	Design Space Definition	81
3.3.1.1	Dimension	82
3.3.1.2	Node Representation	83
3.3.1.3	Edge Representation	84
3.3.1.4	Layout	85
3.3.1.5	Mixing Design Choices	86
3.3.2	Implementation of the Design Space	86
3.3.2.1	General Architecture	86
3.3.2.2	Interactive Mapping Mode	89
3.3.2.3	Batch Mapping Mode	91
3.3.3	Using the Design Space to Adapt to different Aspects	93
3.3.3.1	Structure	93
3.3.3.2	Attributes	98
3.3.3.3	Spatial Context	100
3.3.3.4	Temporal Context	100
3.3.4	Conclusion	107
3.4	Summary	107
4	Scalable Visualization of Dynamic Graphs	109
4.1	Introduction	109
4.2	Clustering based Abstraction of Structure and Time	109
4.2.1	Extraction of the Graph for a given Granularity	111
4.2.2	Clustering of the Structural Aspect	114
4.2.2.1	Clustering associated Temporal Attributes of the Supergraph	114
4.2.2.2	Visualization of the Clustered Supergraph	118
4.2.3	Clustering of the Temporal Aspect	125
4.2.3.1	State Transition Graph Generation	125
4.2.3.2	Visualization of the State Transition Graph	128

4.2.4	Use Case	133
4.2.4.1	Data Description	134
4.2.4.2	Analysis Goals	136
4.2.4.3	Analysis on a Global Level	136
4.2.4.4	Analysis of Local Problems	139
4.2.4.5	Forecasting Recurring Problems	142
4.2.5	Conclusion	144
4.3	A Modular Degree-of-Interest based Selection for Dynamic Graphs	146
4.3.1	Existing DoI Specifications for Graphs	147
4.3.2	A Modular DoI Definition for Dynamic Graphs	149
4.3.2.1	Specification Components	150
4.3.2.2	Transformation Components	152
4.3.2.3	Combination Components	153
4.3.2.4	Propagation Components	153
4.3.2.5	Example: Realizing Furnas' DoI Function	155
4.3.3	An Interactive Visual Analysis Setup based on the DoI Values	156
4.3.3.1	The DoI View	158
4.3.3.2	The Graph View	160
4.3.3.3	Supporting DoI Definition and Adjustment with Presets and Linking	163
4.3.4	Use Case	164
4.3.4.1	Data Description	164
4.3.4.2	Analysis Goals and First Steps	165
4.3.4.3	Further Refinement through DoI Modification	165
4.3.5	Conclusion	168
4.4	Summary	168
5	Flexible Visualization of Graphs in Space and Time	169
5.1	Introduction	169
5.2	An In Situ Visualization for Switching and Combining Dynamic Graph Visualizations	169
5.2.1	Classification	170
5.2.2	In Situ - A new Approach for Large Dynamic Graphs	173
5.2.2.1	Selection of Data Subsets	174
5.2.2.2	Choice of Visualization Techniques	175
5.2.2.3	Embedding of the Visualization	177
5.2.2.4	Practical Considerations for the In Situ Visualization	179
5.2.3	Use Cases and User Feedback	183
5.2.3.1	Model Versioning	183
5.2.3.2	Mesh Networks	186
5.2.3.3	User Feedback	189
5.2.4	Conclusion	191
5.3	Steering the Analysis of Multiple Aspects with a Synchronized Approach	193
5.3.1	Transformation into a Multipartite Graph	193

Contents

5.3.2	Visual design	194
5.3.2.1	Synchronization view	194
5.3.2.2	Data views	196
5.3.3	Interactive exploration	197
5.3.3.1	Modes of exploratory analysis	198
5.3.3.2	Level-based exploration	198
5.3.3.3	Tuple-based exploration	199
5.3.4	Implementation & case studies	200
5.3.4.1	Implementation based on user feedback	200
5.3.4.2	Data description	202
5.3.4.3	Case studies	202
5.3.5	Conclusion	206
5.4	Summary	206
6	Conclusion	207
6.1	Summary	207
6.1.1	Diversity	207
6.1.2	Scalability	208
6.1.3	Flexibility	209
6.2	Discussion and Future Work	210
6.2.1	Technical Limitations	210
6.2.2	Application	212
6.2.3	Generalization and Specialization	214
	Bibliography	217
	Own Publications	237
	Thesis Statements	239

List of Figures

2.2.1	Different matrix visualizations.	10
2.2.2	Different node-link visualizations.	12
2.2.3	Different implicit visualizations.	13
2.2.4	Different hybrid visualizations.	14
2.2.5	Different mappings of attributes to visual variables.	15
2.2.6	Different mappings of attributes to the third dimension.	17
2.2.7	Different integrations of glyphs.	18
2.2.8	Different tabular representations.	19
2.2.9	Direct mapping of spatial references on node positions.	20
2.2.10	Visualization of graphs over a map.	21
2.2.11	Adaptation of the map for integrating graph visualizations.	21
2.2.12	Adaptation of the graph layout to reflect the geographical context.	21
2.2.13	Different visualizations based on the supergraph.	23
2.2.14	Embedding of time plots into supergraph based visualizations.	24
2.2.15	Animated visualization of dynamic graphs.	26
2.2.16	Statistical plots of graphs over time.	27
2.2.17	1.5D visualization of a dynamic graph.	28
2.2.18	Visualization of a dynamic graph with a branching time axis.	29
2.2.19	Different 2D layered visualizations of dynamic graphs.	30
2.2.20	Different 3D layered visualizations of dynamic graphs.	31
2.2.21	Combination of visualizations showing different aspects.	33
2.3.22	Different coarsening strategies for graphs.	35
2.3.23	Visualizations for clustered graphs.	36
2.3.24	Different visual aggregations of the structure.	37
2.3.25	Visual abstraction of structure and time.	38
2.3.26	Overview and detail visualization of a large graph.	39
2.3.27	Different selection methods for attributes, nodes and edges.	40
3.2.1	Computing a layout in a point-based manner.	47
3.2.2	Assigning predefined positions and corresponding spaces to the nodes of the hierarchy according to a spiral.	48
3.2.3	Lightness adaptation to better reflect the visualized structure.	49
3.2.4	Different representations of the parent-child relations.	50
3.2.5	A GraphSplatting visualization communicating regions with hidden information.	51
3.2.6	Rotation based interaction.	52

List of Figures

3.2.7	Three point-based variations based on three different fractals and their extension to wider hierarchies by an additional ring.	55
3.2.8	Three point-based variations using similar fractals with different results.	57
3.2.9	Three strategies to adapt point-based layouts to different settings and hierarchies.	61
3.2.10	Mapping of attribute values on the color of nodes, edges and paths.	63
3.2.11	Mapping of attribute values on the third dimension.	65
3.2.12	Subdivision of an irregular area along its skeleton.	68
3.2.13	A hierarchy layout embedded into an irregular region.	69
3.2.14	Point-based visualization for 2 consecutive time points.	71
3.2.15	Layered visualization of 8 time points of the DMOZ hierarchy.	72
3.2.16	Visualization of hierarchies for selected regions and three time steps.	73
3.2.17	Pencil and helix glyphs visualize data attributes associated with hierarchy nodes.	74
3.2.18	Visualization according to the hierarchical structure of time.	76
3.2.19	Different visual abstractions of the temporal hierarchy layout.	77
3.2.20	Visualization of influenza in Mecklenburg-Vorpommern for the year 2000.	78
3.2.21	Different scopes of comparison of multiple diseases in Mecklenburg-Vorpommern for the year 2000.	79
3.3.22	Different ways for changing a 2D Treemap into 3-dimensional counterparts.	82
3.3.23	Examples for the usage of different node representations.	83
3.3.24	Examples for different edge representations	84
3.3.25	Examples for the usage of different layouts.	85
3.3.26	A snapshot of the Implicit Tree Visualization Toolkit.	88
3.3.27	Transforming a 2D Sunburst into a 3D Circular Treemap using the interactive mapping mode.	90
3.3.28	The transformation of a 2D Sunburst into a 3D Circular Treemap captured in the batch mapping mode.	92
3.3.29	2D and 3D variants of the Sunburst.	95
3.3.30	Different strategies to reduce the visual space assigned to deep narrow parts of the hierarchy.	97
3.3.31	Embedding of charts to visualize multiple attributes.	99
3.3.32	Layered Icicle Plots to convey changes in the structure using a space filling or a comparable layout.	102
3.3.33	A filtered visualization based on layered Icicle Plots to convey changes in the structure.	104
3.3.34	Unstable parts of the hierarchy are shown for multiple time points as layered Icicle plots embedded into a squarified Steptree	105
3.3.35	Embedding of charts to visualize time-varying attributes and adapting the space distribution according to the temporal variance.	106
4.2.1	The effect of the two possible orders for structural coarsening and temporal aggregation exemplified.	112
4.2.2	Handling of missing values during the similarity calculation.	115
4.2.3	Examples of different similarity measures for the clustering.	116

4.2.4	Defining connected subgraphs.	117
4.2.5	The clustered graph view of the dynamic graph.	119
4.2.6	The time plot view of the dynamic network.	120
4.2.7	The construction of a multi-scale time plot.	121
4.2.8	The similarity view for comparing clustering settings	123
4.2.9	Adaptation of the glyphs for representing clusters to accommodate two clustering settings	124
4.2.10	Overview of the state transition graph generation.	129
4.2.11	The temporal overview of the dynamic graph.	130
4.2.12	The structural overview of the dynamic graph.	131
4.2.13	The state transition graph view of the dynamic graph.	132
4.2.14	Structural overview of the OpenNet network.	134
4.2.15	The temporal trends for three different clustering settings based on k-means.	137
4.2.16	A view setup for the analysis of a structural clustering according to a similar link quality.	138
4.2.17	A zoomed-in view of three time intervals showing drop-offs in the quality of clusters with a generally good quality otherwise.	140
4.2.18	The supergraph is clustered according to the time interval as selected in Figure 4.2.17c.	141
4.2.19	A view setup for the analysis of a temporal clustering for exploring branching patterns of the dynamic network.	143
4.2.20	The state transition graph generated for the finest structural scale showing the individual routers at an half hourly temporal scale.	144
4.3.21	The overall visual setup with its two main views: the DoI view and the graph view.	157
4.3.22	Different DoI components and their visual counterparts.	159
4.3.23	Different appearances of the glyph encoding contracted subgraphs.	161
4.3.24	A view of the stacked graph along the timeline showing top authors with low clustering coefficients.	164
4.3.25	A full view of the DBLP data set for 2007, capturing all top authors with low clustering coefficients and a high increase of node degree over time.	166
4.3.26	The same view as in Figure 4.3.21, but this time with the found namesakes having been subtracted from it.	167
5.2.1	Different menus for the selection of a visualization to be embedded.	176
5.2.2	Example showing the in situ visualization.	178
5.2.3	Different adjustment strategies for embedding rectangular visualizations in arbitrary shaped selections of a node-link visualization.	180
5.2.4	Possible base visualizations of the versioning history of a biological model	184
5.2.5	Visualization of the revision graph of a biological model.	185
5.2.6	Visualization of the general structure of a biological model.	186
5.2.7	Possible base visualizations of the OpenNet mesh network.	187
5.2.8	Visualization of the general structure of the Opennet in its spatial context.	188
5.2.9	Linear representation of the temporal course of the Opennet.	190

List of Figures

5.2.10	Non-linear representation of the temporal course of the Opennet.	190
5.3.11	The synchronization view showing multiple levels of space, time, and different attributes.	195
5.3.12	Investigating the largest increase in voter popularity for each poll in the synchronization view.	203
5.3.13	Effect of the March 11, 2011 Fukushima Daiichi nuclear disaster on German voter behavior.	204
5.3.14	Effect of the controversial governmental development project Stuttgart21 on local voter behavior in the German state Baden Württemberg.	205
6.2.1	State transition graph based visualization of multiple simulation runs of a power grid.	211
6.2.2	State transition graph visualization of temporal patterns for a high-speed train connection network in Germany.	213
6.2.3	Smooth blending of tree visualizations.	215

List of Tables

4.3.1	Categorization of basic sources for specifying interest.	150
5.2.1	A classification of visualization approaches for large dynamic graphs. . .	171

1 Introduction

1.1 Motivation and Problem Statement

Graphs are a versatile tool utilized in many fields. They are used for instance for the classification of websites (see the Open Directory Project at <http://www.dmoz.org/>), to capture social ties between persons such as co-authorships [New04], to model the communication between households [Wol12], to organize configurable systems [MPG⁺04], and even for the segmentation of images into homogeneous regions [FH04] or foreground and background [GOV⁺11]. Because of their importance, the visual analysis of graphs is an important endeavor. Yet, as diverse as the application areas are the aspects constituting these graphs.

Considering their structure, graphs can range from simple tree like structures, to more general networks, up to multipartite graphs. In case of classifications or categorizations, trees are usually used to describe their hierarchical structure. Contrary, networks are often used to describe the more general connections between persons such as family ties or friendship relations in social networks. For modeling bio-chemical reaction networks commonly hyper-graphs or bipartite graphs are used to distinguish between different species and the reactions they are involved in.

Besides the structure, a multitude of different attributes can be associated with both nodes and edges. These can contain on the one hand simple attributes such as names of persons in social networks or numerical values such as reaction rates or concentrations in reaction networks. On the other hand, nodes and edges can be linked with spatial information such as bio-chemical reactions occurring only in specific compartments or routers featuring exact geographical locations in communication networks.

Furthermore, these graphs are subjects to change over time influencing the structure as well as the associated attributes. In case of real world recorded graphs such as social or communication networks, this necessitates the analysis of graphs along a linear time axis. When analyzing versioning data, these changes can also be the result of manual manipulations of these graphs resulting not only in a linear but a complex branching time axis. And even if the dynamic graphs are recorded along a linear time axis, they may contain similar graph structures occurring at multiple time points, thus indicating some recurring temporal patterns such as cycles.

Besides the different aspects like structure and time, the analysis of these graphs becomes increasingly difficult considering their size. Classifications as well as social networks can easily contain millions of nodes and edges. Whereas a logging of communication networks can result in hundreds of thousands time points. Thus, to support a thorough analysis their visualization has to be as versatile as the graphs coping with both the different aspects as well as the size of the graphs. This thesis aims at providing

novel visualization strategies for handling both: the different aspects and the size of these graphs.

1.2 Research Questions

The described problems hampering the analysis of graphs can be summarized into two major challenges:

Diversity concerning the different aspects – *structure, attributes, spatial and temporal context* – of the graphs.

Scalability regarding the size – *number of nodes and edges* as well as the *amount of time steps* and *number of attributes* – of the graphs.

Typically, there are two approaches to solve these challenges. First, developing only a single but complex visualization to cope with the two challenges diversity and scalability at the same time. Such a visualization most often tries to communicate all information of the data with a single image. And second, combining multiple but simpler visualization techniques each addressing a specific subset of the graph at one time point. In this way, multiple images are presented to the user each providing a different view on a subset of the data.

As it is in general difficult to resolve all aspects equally when the graph becomes larger such solutions come with specific limitations. For instance, animations on the one hand are often used to convey the dynamics of graphs showing the whole graph structure separately for each time point. Such an approach proves useful in analyzing larger changes in the graph structure over time. Yet, when it comes to analyze the temporal development of particular nodes this becomes a tedious endeavor. It involves switching back and forth in time to find and compare these nodes at all time points. On the other hand, techniques focusing directly on the development of particular nodes lack to support the analysis of larger changes in the structure. This would then involve to switch between all nodes and edges to identify such structural changes. In both cases, it is nevertheless possible to visualize every information captured in the graph. However, the efforts to analyze the data strongly depends on the congruence of the user's interest and the focus provided by the visualization.

This problem becomes even more severe if confronted with unknown data for which it is not known in beforehand which aspect may be important and should thus be focused by the visualization. The user's interest may then shift during the analysis as more information is revealed. In this case, a more flexible approach allowing to switch the focus of the visualization at any time becomes very helpful. This opens up another challenge:

Flexibility regarding the different foci on the graphs as a result of the interplay of diversity and scalability.

Such a flexible approach can be achieved by combining visualizations featuring different foci. Yet, to fulfill this combination a couple of problems have to be solved. The first two arise directly from the two challenges concerning the visualization techniques to be combined:

- To provide different foci on the graph data thus primarily addressing its diversity appropriate visualization techniques have to be available.
- To handle the size of the graphs and thus assure the scalability of the visualization appropriate techniques for their reduction are necessary.

Most of the existing techniques address these problems, but they are often restricted to a couple of these aspects or focusing on a reduced subset of nodes, edges and time points. In this way, they do not allow a free switch between different views on the data. Furthermore, not all combinations of aspects and their reduction have been sufficiently addressed yet.

The last problem concerns the way the combination is performed. This poses questions such as:

- How many visualizations to show at the same time?
- How to arrange the different visualization techniques?
- Is there additional information to maintain between the visualizations?

The most prominent way of combining multiple visualizations is to use multiple linked views showing different aspects side by side. Yet, often the visible visualizations are fixed and thus do not necessarily allow a free choice of focus on the graph. Furthermore, nodes share relations in the form of edges that are of importance for the analysis, yet are not necessarily maintained when visualized in separated views.

These three challenges are targeted by this thesis. Therefore, new visualization and reduction techniques for graphs are proposed to provide scalable solutions for different subsets of aspects that are not yet sufficiently addressed by existing solutions. Finally, requirements and strategies for combining graph visualizations are discussed allowing a more flexible visual analysis of graphs than currently available.

1.3 Outline and Contribution

First, Chapter 2 provides necessary definitions and gives an overview of the existing solutions for graph visualization according to the three challenges.

In Chapter 3 the challenge of handling the diversity is discussed exemplary for two new visualization approaches:

The first approach described in Section 3.2 is a new point-based layout for hierarchies that scales well to multiple hundred thousand items while maintaining an overview of the structure. It does so by utilizing the smallest graphical primitive – the point primitive – and a fixed layout scheme that also enables a direct embedding into a spatial and temporal context. The point-based layout has been published as “*Point-Based Visualization for Large Hierarchies*” 2011 in the IEEE Transactions on Visualization and Computer Graphics [SHS11b], and its embedding into the spatial and temporal context as “*Visualization*

1 Introduction

of *Attributed Hierarchical Structures in a Spatio-Temporal Context*” 2010 in the International Journal of Geographical Information Science [HTSS10].

The second approach is a generalization of implicit visualization techniques for static and dynamic trees that is described in Section 3.3. Therefore, a design space and a rapid prototyping toolkit have been developed. These can be used to flexibly handle an arbitrary number of attributes as well as to derive different new visualization techniques for static as well as dynamic hierarchies. Preliminary work concerning visualizations for static hierarchies has been published as *“The Design Space of Implicit Hierarchy Visualization : A Survey”* 2011 in the IEEE Transactions on Visualization and Computer Graphics [SHS11a].

Two new strategies to handle the size of the graphs and thus the scalability are introduced in Chapter 4. They are based on clustering techniques as well as the definition of a Degree-of-Interest by the user.

At the example of the aspects structure and time, two contrary clustering techniques to approach and reduce the overall size are discussed in Section 4.2. The first technique clusters nodes and edges of the graph that show a similar trend over time. In contrast, the second technique groups time points if they show a similar graph structure. Both methods have been applied to a communication network that has been observed over multiple thousand time points. Preliminary work on these clustering techniques have been published as *“Supporting the Visual Analysis of Dynamic Networks by Clustering associated Temporal Attributes”* 2013 in the IEEE Transactions on Visualization and Computer Graphics [HSCW13].

The second strategy is based on a generalization of the DoI specification for dynamic graphs that is described in Section 4.3. Therefore, the different terms of existing DoI functions were extracted as modular components. In this way, the user can more flexibly define which parts of the graph are of interest for him by composing and adapting his own DoI function. Based on this DoI function the interesting parts are then extracted and visualized. Besides discussing possible applications of the DoI within the visualization, this approach is applied to the DBLP data set which contains nearly a million nodes across 22 time points. This modular DoI specification is described in the publication *“A Modular Degree-of-Interest Specification for the Visual Analysis of Large Dynamic Networks”* 2014 in the IEEE Transactions on Visualization and Computer Graphics [AHSS14].

Chapter 5 then discusses two novel approaches to combine both, diversity and scalability, in a flexible manner.

The first approach is based on a direct embedding of visualizations and described in Section 5.2. This allows the user to locally adapt and change the focus of a base visualization to his needs. This approach has been evaluated by a qualitative user study and its application to two different use cases. The results appeared 2011 under the title *“In Situ exploration of large dynamic networks”* in the IEEE Transactions on Visualization and Computer Graphics [HSS11].

The second approach primarily focuses on providing an abstract overview of all aspects of the data that is described in Section 5.3. Additional views then provide more details according to specific aspects. In this way, the overview serves as a synchroniza-

tion mechanism for the different views and allows to steer the analysis. German election data has served as a basis for an evaluation of this approach. This work has been published as “A Visualization Approach for Cross-level Exploration of Spatiotemporal Data” 2013 at the International Conference on Knowledge Management and Knowledge Technologies [SHS13].

A summary of the results as well as possible directions for future work are finally discussed in Chapter 6.

Special note about the organization of Chapters 3 to 5: In large parts this thesis represents a cumulative dissertation generally citing the aforementioned papers in accordance with the corresponding coauthors.

The original point-based layout as well as the design space for implicit tree visualizations were developed in strong collaboration with Hans-Jörg Schulz during his dissertation. While he has focused primarily on the theoretical foundations of these two approaches, the focus of this thesis’ author laid on their practical realization. Hence, Chapter 3 only presents a brief summary of the general concepts¹ as published in the corresponding papers (“Point-Based Visualization for Large Hierarchies” and “The Design Space of Implicit Hierarchy Visualization : A Survey”) that are relevant for this thesis. Afterwards, novel extensions of these general concepts to accommodate the different aspects – structure, associated attributes and the spatial and temporal context – are introduced that were developed in the scope of this thesis and go beyond the original publications. In case of the embedding of the point-based layout in the spatial and temporal context, these extensions resulted in an individual publication (“Visualization of Attributed Hierarchical Structures in a Spatio-Temporal Context”). This paper is partially adopted in the Sections 3.2.4 to 3.2.6.

Furthermore, parts² of the Chapters 4 and 5 are adopting the aforementioned papers in most cases with some small adjustments in the wording to adapt the description to the notation used throughout this thesis and some additional details. An exception concerns the use case described in Section 5.2.3.2 that was updated to a larger variant. Furthermore, the use case showing the application of the clustering of the structural aspect of dynamic graphs³ originates from a joint work with Till Wollenberg. The clustering of the temporal aspect of dynamic graphs in Section 4.2.3 describes a not yet published approach. The synchronization view was also designed in strong cooperation with Hans-Jörg Schulz whose main focus here laid especially on the different modes of exploration⁴.

¹The basic point-based layout is described in Section 3.2.1 and the implicit tree visualization design space in Section 3.3.1.

²This concerns especially the Sections 4.2.2, 4.2.4, 4.3, 5.2 and 5.3.

³This use case is described in the Sections 4.2.4.1 to 4.2.4.4.

⁴These modes are discussed in Section 5.3.3.

2 Basics and Related Work

This chapter starts with basic definitions of graphs formally describing their structure and characteristics such as attributes and temporal changes. On this basis existing visualization techniques are summarized according to their handling of the different aspects – structure, attributes, spatial and temporal context – and of the size of these graphs. A final summary discusses open problems and challenges resulting from this overview.

2.1 Terms and Definition

Graphs are a common principle to model relationships between objects. As an important field of research within mathematics and informatics the *graph theory* provides extensive definitions and algorithms to describe and handle graphs and their properties. In the following, the notation of graphs as introduced in [Bra94] is adapted.

In its simplest form a *graph* G can be defined as $G = (V, E)$. Here, V describes the set of objects often called *nodes*. E describes the set of connections or relations between the nodes also called *edges* and is generally defined as $E \subseteq V \times V$. Every edge e can then be defined as a pair $\{u, v\}$ with $u, v \in V$. Here, e is considered *incident* with u and v , whereas u and v are considered as *adjacent*. If no differentiation between nodes and edges is necessary, they are referred to as *graph elements* $x \in V \cup E$.

If considering *directed graphs* each edge has an orientation. In this way, two edges can exist between two nodes u and v : (u, v) and (v, u) . The edge (u, v) then describes that it is leading from the start node u to the end node v .

For a set of nodes $U \subseteq V$ the induced *subgraph* $G(U)$ is defined as $G(U) = (U, E')$ with $E' = E \cap (U \times U)$. In case of $E' = U \times U$ describing the existence of all possible edges between the nodes in the graph $G(U)$ this set of nodes U is called a *clique*.

A *path* P between two nodes a and b is a series of edges e_1, \dots, e_k with $\forall i \in \{2, \dots, k\} : e_i$ shares a node with both e_{i-1} and e_{i+1} as well as $e_1 = \{a, v_1\}$ and $e_k = \{v_{k-1}, b\}$. The *length* $|P|$ of this path equals the number of its edges. The *graph-theoretical* or *geodesic distance* between two nodes equals the length of the shortest path between them. In this context, a graph contains a *cycle* if it contains a path $P = (v_0, \dots, v_k)$ with $v_0 = v_k$. A graph containing no cycles is called *acyclic*. A graph is considered *connected* if a path exist between each pair of nodes.

For the comparison of n graphs $G_i = (V_i, E_i)$ with $1 \leq i \leq n$ often the *supergraph* or union graph $SG = (V_{SG}, E_{SG})$ is calculated (similar to [DGK00]). Here, $V_{SG} = \bigcup_{i=1}^n V_i$ and $E_{SG} = \bigcup_{i=1}^n E_i$ summarize the structure over all graphs. In this way, the supergraph contains all nodes and edges of all graphs providing an overview of the overall structure. Another important structure is the *maximum common subgraph* [BY08] $MCS = (V_{MCS}, E_{MCS})$ with $V_{MCS} = \bigcap_{i=1}^n V_i$ and $E_{MCS} = \bigcap_{i=1}^n E_i$. In contrast to the supergraph the maximum

common subgraph only contains those nodes and edges that exist in all graphs and thus provides an overview of what all graphs have in common.

2.1.1 Graph classes

Depending on the structure denoted by the edges different graph classes can be defined. The most important classes for this thesis are networks and trees. From these classes, *networks* represent the most general graph class posing no restrictions on which edges are allowed whereas the trees are the most restricted class.

Trees contain those graphs that are connected and acyclic. In this way, trees pose the most restrictions to the edge set. In a tree exactly one path exists between each pair of nodes u and v . As a result of this requirement a tree contains no cycles. Nodes with a single incident edge are also called *leaves*.

Trees with a designated node $r \in V$ also referred to as *root node* are called *rooted trees*. In rooted trees edges are often directed away from the root node and describe a *parent-child-relation*. The start node of an edge is then called *parent node* and its end node is called *child node*. In this sense, nodes having no children are referred to as *leaf nodes* or *leaves*. The *depth* of a node is defined as its path length to the root node. A specific *depth level* then contains all nodes featuring this depth. The *subtree* rooted at the node v is the subgraph induced by the node v and its children. Rooted trees are most commonly used to model hierarchies.

Trees are also the basis for hierarchically organizing graphs often called *clustered graphs*. According to [EF97] a clustered graph $C = (G, T)$ consists of a graph G and a rooted tree T . Here, the nodes of G corresponds to the leaves of T . Whereas each node v of the tree represents a cluster of nodes in G that are leaves of the subtree rooted at v .

Besides these classes there exists a wide range of different classes posing other freedoms and restrictions. For instance, *hypergraphs* describe the most general class of graphs also allowing connections between more than two nodes. In *bipartite graphs* the set of nodes can be split into two disjoint partitions X, Y so that $E \subseteq (X \times Y) \cup (Y \times X)$. An edge $e \in E$ then connects one node of each set whereas connections within the same node set are not allowed. More information on graph classes can be found in [BLS99, dR13] containing more than 1400 different classes.

2.1.2 Attributes

For both nodes and edges a multitude of attributes may be defined. These can be distinguished in structural and associated attributes [BCD⁺10]. Such graphs containing multiple attributes are often referred to as multivariate graphs.

Structural attributes are implicitly contained in the structure describing graph-theoretical properties and thus may be computed of the structure. A common attribute is the *node degree* that equals the number of incident edges. More complex attributes contain betweenness centrality and clustering coefficient. The *betweenness centrality* of a node or edge sums up the number of paths going through it [Bra01a, GN02]. The *clustering coefficient* is only defined for nodes describing the fraction of the existing edges to all possible edges between the neighbors of each node [WS98].

Associated attributes are given explicitly with the graph such as names, reaction rates or geo-coordinates. They are often modeled as functions in the form: $a : V \rightarrow \mathbb{R}$ or $w : E \rightarrow \mathbb{R}$ for numeric attributes. In case of edges, associated attributes are referred to as *weights*. Weights are often used to describe costs or capacities of the connections between nodes. In this way, the path length changes to a weighted sum of the associated edge weights.

Altogether, a multivariate graph G can be described by $G = (V, E, A, W)$. Here, A describes the set of node attributes defined for the nodes and W contains the weights defined for the edges. In the following, both, node attributes and edge weights are assumed to be *attributes* defined for all graph elements. Hence, node and edge attributes both map a given graph element to a numerical attribute $attr(x) : V \cup E \mapsto \mathbb{R}$ with $attr(x) \in A \cup W$. In case that an attribute is only partially defined, i.e., only for nodes or only for edges, it maps all other elements to *undefined*.

2.1.3 Dynamic graphs

The temporal context is considered as a totally ordered set T of time points t_i with $1 \leq i \leq |T|$. These time points do not have to be equidistant, i.e., the duration between two time points $t_{i+1} - t_i$ may vary. On this basis a *dynamic graph* $DG = (G_0, \dots, G_{|T|})$ can be defined as a sequence of graphs $G_i = (V_i, E_i, A_i, W_i, t_i)$ (similar to [DGK00]). Each graph G_i forms a 5-tuple of a node set V_i , an edge set E_i , a set of node attributes A_i , and a set of edge weights W_i at the time point $t_i \in T$.

The dynamics of the graph can be distinguished in structural and attribute related changes [vLKS⁺11]. *Structural changes* are modifications of the node and edge sets such as additions or deletions. Depending on the graph class larger modifications based on additions and deletions may have different semantics. For instance, in trees a deletion of an edge leads to a disconnection or deletion of a whole subtree. A sequence of deletions and additions connecting a subtree to a different part of the tree is often perceived as a move within the modeled hierarchy. In general networks on the other hand, this describes solely breaking up old ties and establishing new ones.

Attribute changes on the other hand concern changes of node attribute values or edge weights. As nodes and edges may not exist at all time steps their attribute values are marked as missing by a particular value. In this way, structural changes can also be perceived as attribute changes. The other way around, attribute changes can lead to structural changes if they are describing the existence of nodes and edges. This allows to model relations in a fuzzy way discriminating between strong ties and loose contacts.

Besides such a rather linear temporal domain, some existing use cases also utilize more complex temporal structures such as cyclic time or branching time, to capture for instance recurring trends or to describe the versioning history of a graph. Here, an additional graph structure is used to model the relations between different time points. More information on these three classes of time can be found in [AMM⁺07].

2.2 Visualization of Graphs

The definitions of the underlying data as provided above underline the diversity one is confronted with when visualizing graphs. Here, every aspect – structure, attributes, spatial and temporal context – poses different requirements to the visualization that are in part corresponding and contradicting. Therefore, the visual representations for all aspects are summarized individually beginning with different representations of the structure. Then follows a general description of visual mappings of attributes. As spatial attributes often includes the usage of maps the visualization of the spatial context is discussed separately. At last, strategies to convey the dynamics of graphs are described. Finally, some general strategies are presented to visualize multiple aspects at the same time.

2.2.1 Visualization of Structure

According to the State-of-the-Art report [vLKS⁺11] the majority of graph representations can be divided into four groups: matrix representations, explicit node-link representations, implicit representations as well as their hybrids. As the aim of this section is to provide a general overview of the representations more information on this subject can be found in this report.

2.2.1.1 Matrix Representations

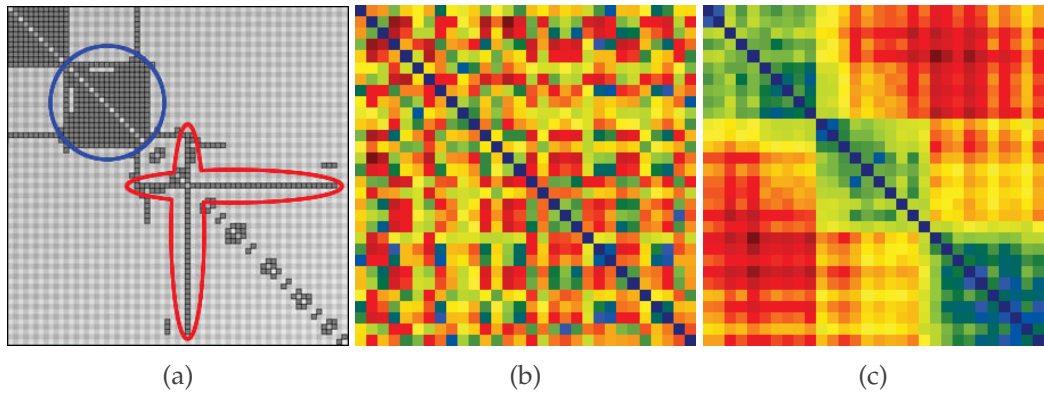


Figure 2.2.1: Different matrix visualizations: (a) shows a simple adjacency matrix [EDG⁺08]. A clique (a subgraph whose nodes are completely connected) has been highlighted in blue and a star (a subgraph where all nodes are connected to a single node only) in red. (b) and (c) show a matrix with edge weights mapped onto the color [WTC08]. In (c) the matrix was ordered to form homogenous blocks of similar edge weights.

Matrix representations are a direct visualization of the adjacency matrix of a graph. The nodes are drawn as rectangles on the left as well as on the top of the matrix. Each directed edge is represented as a rectangle in the row of its starting node and in the column of its end node. For an undirected graph, the matrix representation becomes symmetrical. They are good suited to visualize dense graphs containing many edges as well as to identify specific substructures such as cliques.

The effectiveness of the matrix representation is based on the order of the nodes. In [WTC08] different ordering strategies are discussed to form homogenous blocks within the matrix representation (compare Figure 2.2.1b and 2.2.1c). Each homogenous block then represents a clique which is a group of nodes that is nearly completely connected, see for instance the subgraph highlighted blue in Figure 2.2.1a. Such an ordering can also reveal other patterns such as stars or hubs showing nodes connected to many other nodes that have no connections among themselves. This pattern will emerge as a cross in the visualization as highlighted in red in Figure 2.2.1a. Spammers in a communication network show a similar pattern. These are nodes having a high number of outgoing connections (visible as a horizontal line in the matrix) but no incoming connections.

2.2.1.2 Node-Link Representations

In node-link representations the nodes are often rendered as circles or rectangles. The edges are then drawn as lines or curves linking their start and end nodes. In this way, node-link representations *explicitly* show the relations between nodes. Therefore, they can especially support the users in analyzing paths in the graph as it is only necessary to follow the links from one node to the next.

The major problem of this representations is the layout of the nodes and edges as a bad layout can affect the readability of the structure. There exists a vast amount of layout techniques in the field of Graph Drawing providing different strategies for positioning the nodes and routing the edges as exemplified in Figure 2.2.2. Examples for such layout strategies are force-directed, constraint-based, multi-scale or layered layouts for positioning the nodes as well as orthogonal or edge bundling approaches for routing the edges (see [Bra94, BETT99, HvW09, GFV12, LLCM12] for a more detailed description). Force-directed and constraint-based layouts are especially interesting in the context of this thesis as they also play an important role in the visualization of dynamic graphs.

Force-directed layouts are often based on the simulation of forces resulting from electrical charges or springs. Here, repulsion forces are used to evenly push nodes across the display area while attraction forces try to pull incident nodes together. The layout is then calculated iteratively evaluating the forces in every step until an equilibrium of the forces is reached. In Figure 2.2.2a the result of such a layout is shown using the example of the Kamada-Kawai algorithm [KK89]. It shows cliques that are tightly connected in the upper left region of the visualization and more sparsely connected in the lower left region. In contrast to a matrix visualization, the fact that the right subgraph is only connected by a single path with the remaining graph can be perceived much easier in a node-link representation.

Constraint-based layouts introduce additional constraints to the positioning of nodes. Besides the structure, they try to communicate additional information such as closeness to similar nodes and thus restricting the formerly free layout. The example given in Figure 2.2.2b is based on a graph in which the nodes are assigned to different compartments that are highlighted in yellow and green in the background. As these compartments are defined in an abstract way they do not provide spatial regions in which the nodes could be embedded. Yet, they pose a constraint to the layout to group nodes contained in the same compartment even if they have no edges connecting them. Especially the last point

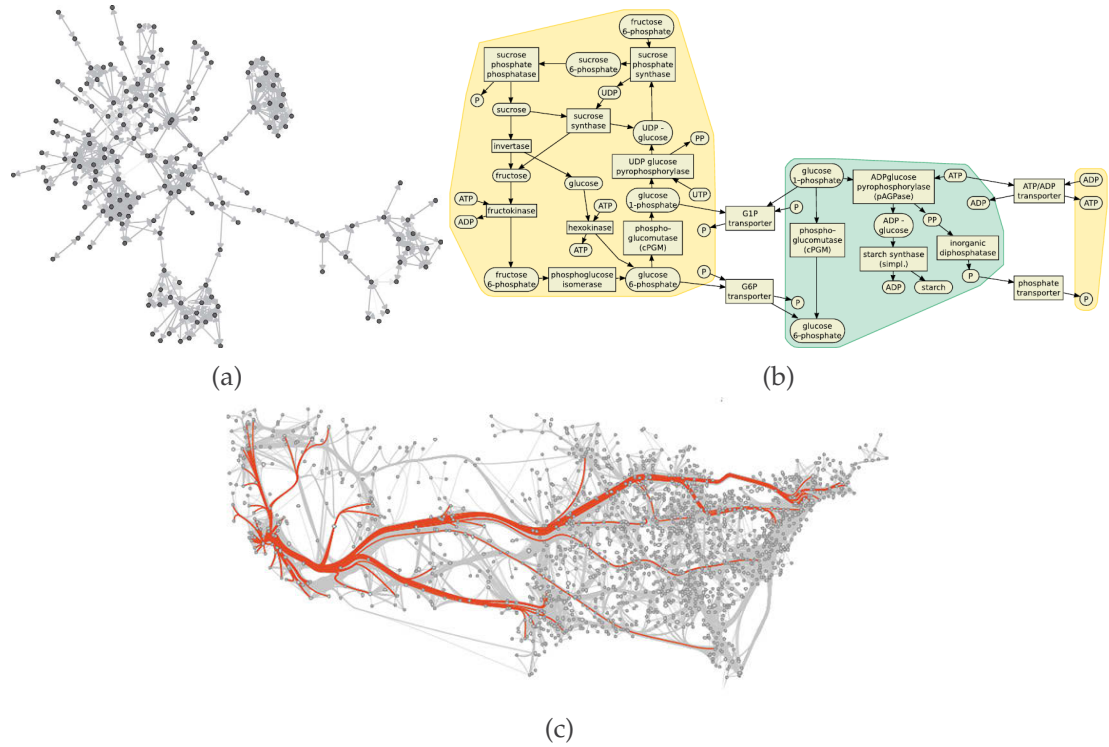


Figure 2.2.2: Different node-link visualizations: (a) shows a graph laid out with the Kamada-Kawai algorithm [KK89] (a force-directed layout). (b) exemplifies a constraint-based layout in which nodes that are contained in the same spatial compartment are also visually grouped [SDMW09]. (c) shows a visualization using edge bundling to better reflect the overall structure [HvW09].

is visible on the right side of Figure 2.2.2b where 3 unconnected nodes are laid out in close proximity.

2.2.1.3 Implicit Representations

In contrast to node-link representations *implicit* visualizations do not explicitly render edges but rely on spatial relations such as adjacency, containment or overlap to convey the connections between nodes. On the one hand, this allows to use the whole display space for the layout of the nodes. For this reason, these techniques are also referred to as space filling techniques. On the other hand, the dependency on spatial relations also limits the classes of graphs that can be represented. They are usually used to visualize tree like graphs providing a very compact overview.

The layout of the nodes also poses the major challenge for this kind of representations. A thorough overview of (implicit) representations for trees is given in [Sch11] containing more than 130 different implicit visualizations. Some examples for such techniques are Treemaps, Icicle Plots or Sunbursts:

Treemaps are using containment as the spatial relation and a rectangular subdivision strategy to distribute the space to the nodes. These strategies may be rather simple such as slice and dice [JS91] eventually creating narrow rectangles as visible in Figure 2.2.3a up to more complex algorithms such as squarified [BHvW00] to create rectangles with

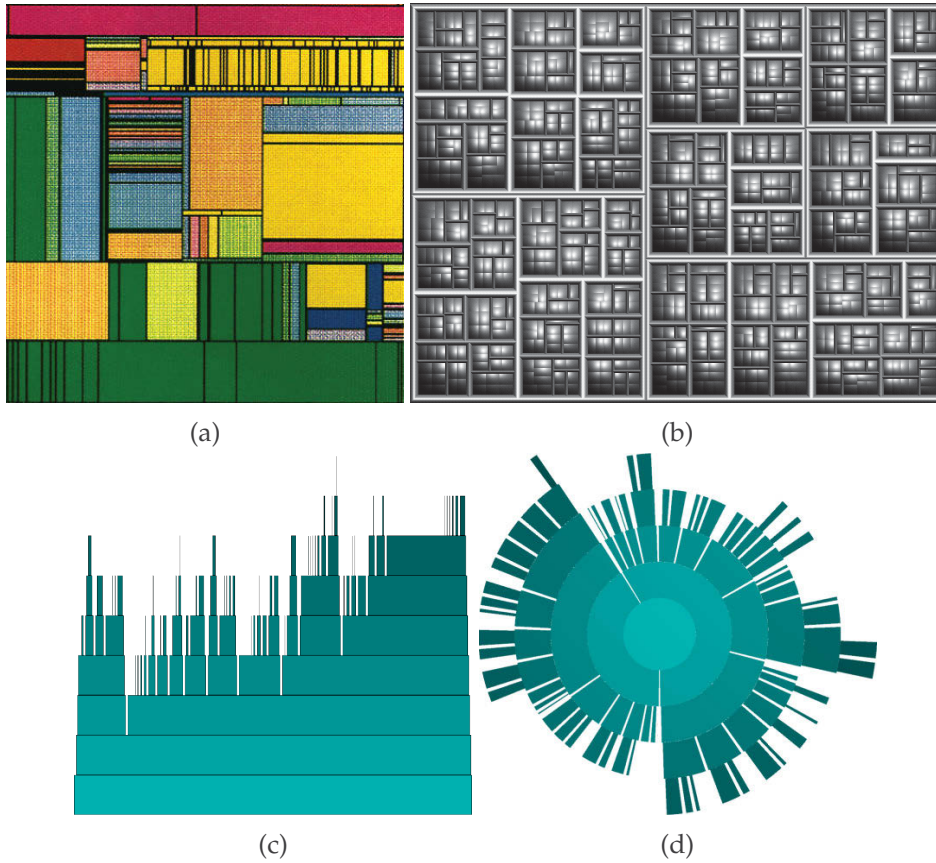


Figure 2.2.3: Different implicit visualizations: (a) shows a Treemap using a slice and dice layout [JS91]. (b) shows a squarified Treemap [BHvW00] with applied cushion shading [vWvdW99]. (c) exemplifies an Icicle Plot (generated according to [KL83]). (d) represents a Sunburst visualization (generated according to [SZ00]).

an aspect ratio of ~ 1 . While these techniques can utilize the whole display space this is often on the cost of the readability of the structure. Thus additional cues have to be integrated such as spending space for borders or use shading to better reflect the structural relations [vWvdW99] as applied in Figure 2.2.3b.

Icicle Plots [KL83] and *Sunbursts* [SZ00] on the other hand both utilize adjacency. Yet, Icicle Plots rely on an axis-parallel subdivision whereas Sunbursts rely on a radial subdivision. In this way, they are better suited to reflect the structure, yet they do so by leaving space empty.

2.2.1.4 Hybrid Representations

Hybrid visualizations are based on the combination of different kinds of representations to better reflect specific subsets of the graph. In this way, they try to combine the advantages of the different representations. Existing hybrid representations contain the combination of matrix and node-link [HFM07], matrix and implicit [RMF09] as well as node-link and implicit representations [ZMC05].

Their combination opens up new ways to approach the graph visualization. In Figure 2.2.4a for instance, cliques or clique like subgraphs are visualized using small matri-

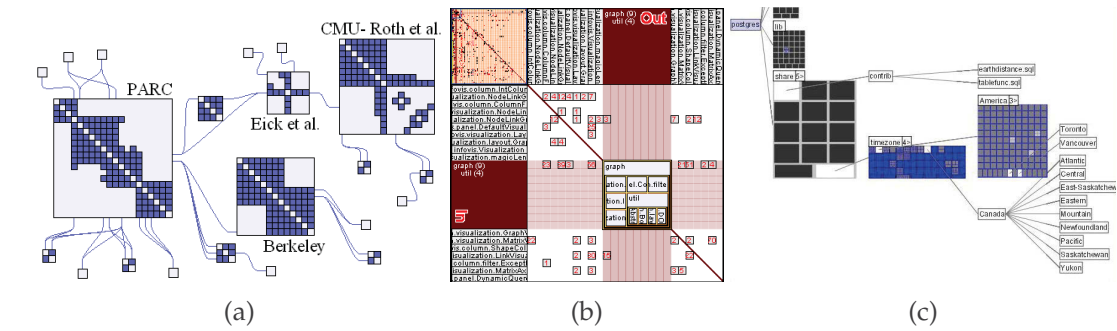


Figure 2.2.4: Different hybrid visualizations: (a) shows a combination of a matrix and a node-link representation [HFM07]. (b) represents a hybrid of a matrix and an implicit visualization [RMF09]. (c) is an example for the combination of a node-link and an implicit technique [ZMC05].

cases whereas the sparse connections between the subgraphs are shown by explicit links. This combination provides on the one hand a much clearer view of the overall structure with less clutter. And on the other hand, missing edges within cliques can be grasped much easier in a matrix visualization than in an overplotted node-link view which is usually the case for cliques. The other way around, by embedding Treemaps etc. into a matrix visualization as it is done in Figure 2.2.4b allows a better perception of substructures such as trees that are hard to read in a matrix. Yet, such a hybrid visualization can also facilitate a focus+context based approach. In Figure 2.2.4c such a switch between techniques is used to enhance the screen space for specific subtrees. Therefore, these subtrees are drawn side by side with their parents and connected by explicit links.

While the combination in general may provide many benefits it also poses the challenge of determining which subgraphs shall be visualized by which representation. In [AMA07] on the one hand the graph is automatically divided into different parts of similar topology in a preprocess. For each part then an adequate representation can be used in the visualization. On the other hand, other approaches leave the choice to the user and provide interactions to group nodes and edges and change their representation [HFM07, MJ09].

2.2.2 Visualization of Multiple Attributes

Techniques for visualizing the associated attributes are often based on the representations described in the previous section. These techniques are adapting the visual variables of nodes and edges or integrating small glyphs. Other techniques neglect the structure to focus on the attributes by utilizing a table based approach.

2.2.2.1 Mapping to Visual Variables

For the visual mapping of data values in general a couple of different visual variables were identified [Ber83, Mac86] such as position, size, orientation, shape, color, texture etc. Which of these variables can be used strongly depends on the graph representation.

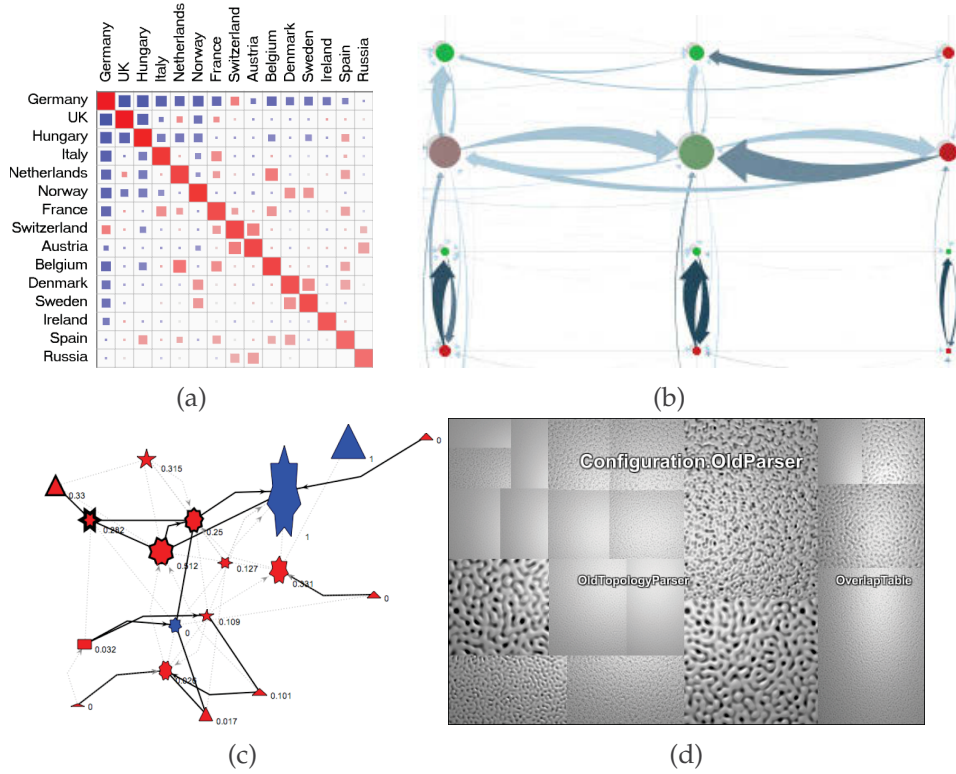


Figure 2.2.5: Different mappings of attributes to visual variables: (a) depicts a mapping to the color and size of the rectangles representing the edges in a matrix visualization [vHSD09]. (b) shows a mapping to the x-y position, the color and size of the nodes as well as on the color and width of the edges in a node-link visualization [Wat06]. (c) illustrates the mapping to the shape of the nodes [OFS⁺05]. (d) depicts the mapping to the brightness and texture of the node representation in an implicit visualization [HVvW05].

Matrix Representations: Because of the regular matrix grid the shape of both nodes and edges is relatively fixed solely depending on the number of nodes. This limits the possibility to map attributes on the position, form, orientation or size. Yet, as described above the order of the nodes is important for the analysis of the graph [EDG⁺08, WTC08]. Besides the structural ordering the nodes can also be ordered according to attribute values as shown in Figure 2.2.1. Thus, matrix representations allow a limited mapping of attributes to the position of nodes also affecting the positions of the edges.

Nevertheless, there are some approaches using the shape of the edges to communicate attributes [EDG⁺08, vHSD09]. An example for such a mapping is given in Figure 2.2.5a where the rectangles are scaled down according to an attribute. This is only feasible if the rectangles have an adequate size allowing such a scaling. Yet, further deformations of the regular grid and thus of the rectangles as proposed in [HSS10] have not been reviewed for matrix visualizations so far.

The usage of textures is also problematic because of the necessary size to distinguish these textures. Therefore, most approaches rely on color and brightness to convey associated information. Such a mapping is also exemplified in Figure 2.2.5a.

Node-Link Representations: In contrast, explicit representations have only a few restrictions for the mapping of attributes. On the one hand, it is often desired to have an overlap free visualization posing some limits especially to the size of the nodes. And on the other hand, the edges are designed to visually link the nodes and thus at least their start- and endpoint is restricted. Besides that, nodes and edges can be configured independently of each other.

While graph layouts are often used to communicate the structure some techniques perform a direct mapping of node attributes to their positions [Wat06, WT06, BCD⁺10, RMFS⁺11]. Hereby, each dimension can be assigned for a different attribute. Figure 2.2.5b shows a mapping in 2D space and Figure 2.2.6a in 3D space. Such a mapping can obstruct the readability of the structure, yet it can help to determine clusters of nodes sharing similar attributes.

To support the differentiation of multiple node types, each type is often mapped on a distinct node shape [OFS⁺05, SGL08, DS13]. An example using different shapes is shown in Figure 2.2.5c. The size (may be assigned independently for each dimension), color and brightness are also commonly used for quantitative attributes as depicted in Figure 2.2.5b. Similar to matrix representations the usage of textures is also bound by a minimum node size.

Commonly, the edge shape is used to convey its direction [Wat06, HivWF11] allowing to distinguish edges connecting the same nodes yet going into different directions. Analogous to nodes, different attributes can be assigned to the edge width, color and brightness as shown in Figure 2.2.5b. As edges occupy the area between their nodes they occupy a larger area by design. This allows the usage of textures to further map attributes. In [HivWF11] textures such as line stipples are used to further enhance the readability of the edge directions.

Implicit Representations: In contrast to the previous representations, edge attributes cannot be visualized with implicit visualization techniques as they do not provide an edge representation at all. Furthermore, positions and shapes of nodes are strictly defined by the layout and its space distribution posing similar restrictions to the mapping of attributes as in matrix representations.

Yet, most layouts allow the selection of weights to steer the space distribution. In this way, the ordering and thus the positions of the nodes as well as their sizes can be influenced by attribute values. Already the first Treemap shown in Figure 2.2.3a was designed to visualize the space consumption of files in a directory tree on the hard disk [JS91]. Here, the size of each node equals the size of the corresponding file.

Because implicit visualizations have a high space utilization nodes are often represented by larger shapes. Hence, textures can be used besides color and brightness to encode associated attribute values. In [vWvdW99] a special shading is used to emphasize the parent-child relation as shown in Figure 2.2.3b. Whereas [HVvW05] uses textures for the mapping of attributes. An example for this visualization is given in Figure 2.2.5d.

As many graph representation rely only on two dimension to convey the structure. This allows to map attribute values directly on the third dimension. Such a mapping can be a direct mapping on the z-coordinate [BCD⁺10] as shown in Figure 2.2.6a or on

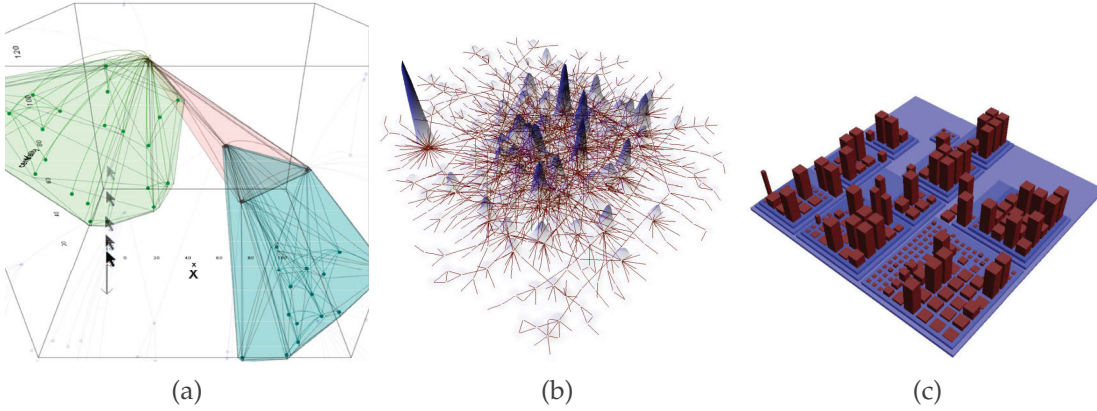


Figure 2.2.6: Different mappings of attributes to the third dimension: (a) shows a direct mapping of three attributes to x-y-z coordinates of nodes in a node-link visualization [BCD⁺10]. (b) features a 3D overlay of an attribute as a height map over a 2D node-link representation [XCHT07]. (c) shows a similar approach referred to as cityscape metaphor to an implicit visualization [WL07].

the height [WL07, XCHT07] creating a visualization similar to a height map as shown in Figures 2.2.6b and 2.2.6c. However, the usage of the third dimension always include occlusions and thus the need for additional interactions such as rotation.

In summary, visual variables such as color and brightness posing no demands to the shape of nodes or edges can be used with any representation. The use of textures demands at least for a minimum shape size. Whereas shape related variables such as its form, size or orientation are except for node-link representation often only usable in a very limited way.

Furthermore, as described in [Mac86] every visual variable differently supports the kind of attribute to encode. For instance, the size is very good to encode quantitative attribute values whereas its usability for ordinal or nominal attributes is limited. Besides that, the effectiveness of visual variables is influenced by themselves. So, to read texture and form of a shape its size should not become too small.

All in all, a mapping to visual variables is a good choice when visualizing only a couple of attributes simultaneously. When dealing with a larger number of attributes other approaches should be preferred.

2.2.2.2 Integration of Glyphs

One alternative to the mapping of attributes to the basic visual variables is the integration of glyphs into these representations. As glyphs can be considered as a subdivision of a shape into smaller shapes, each resulting shape provides a new set of visual variables for the mapping of different attributes. Their integration is often only limited by the available size of the node or edge shape.

Observing this constraint, the utilization of different glyphs have been proposed in combination with all representations. These glyphs can be simple barcharts in different variants (stacked, radial) [EDG⁺08, ME09, ZJK12, KKZ12] as shown in Figure 2.2.7a, 2.2.7c and 2.2.7d for all representations. Examples for more complex glyph types are

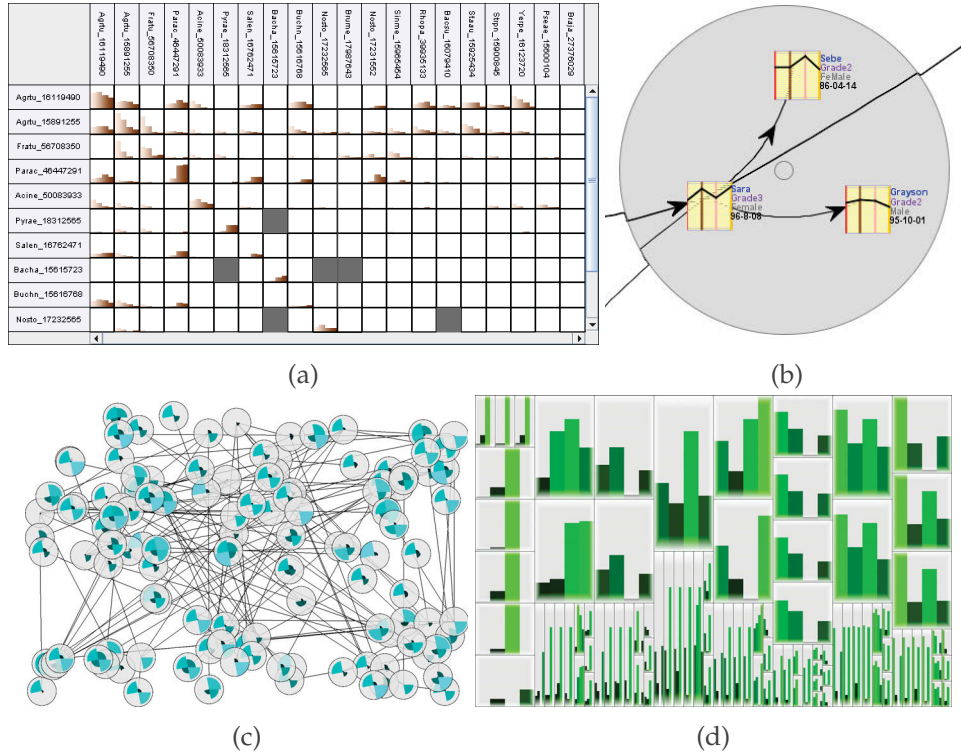


Figure 2.2.7: Different integrations of glyphs: (a) depicts the integration of a barchart into a matrix representation [EDG+08]. (b) shows the integration of a parallel coordinates glyph into a node-link representation [JDK10]. (c) illustrates the integration of a polar barchart into a node link-representation [ME09]. (d) depicts the integration of a barchart into an implicit visualization [ME09].

parallel coordinate plots (shown in Figure 2.2.7b) or star plots [JDK10].

Besides the available size affecting all representations, problems with the readability of the glyphs occurs especially with node-link and implicit representations:

In case of node-link visualizations, dense subgraphs can affect the suitability of glyphs. While in general enough display space may be available for the glyphs, dense parts can lead to much overlap and thus information can be occluded as exemplified in Figure 2.2.7c. Here, a solution may be to show such additional information only on demand by using for example a network lens [JDK10] as shown in Figure 2.2.7b.

In case of implicit representations, the comparability of the glyphs between different nodes strongly depends on the aspect ratio of the node representations. As shown in Figure 2.2.7d the layout of the Treemap resulted in a large variety of rectangles ranging from very narrow to square like. In this way, even small differences in attribute values can look much larger in narrow rectangles as well as when comparing a narrow rectangle with a square like rectangle.

2.2.2.3 Tabular Representations

In opposition to the previous approaches focusing primarily on the structure of a multi-variate graph, tabular representations are better suited for the direct visualization of the associated attributes. They are based on representing the node and edge set as large ta-

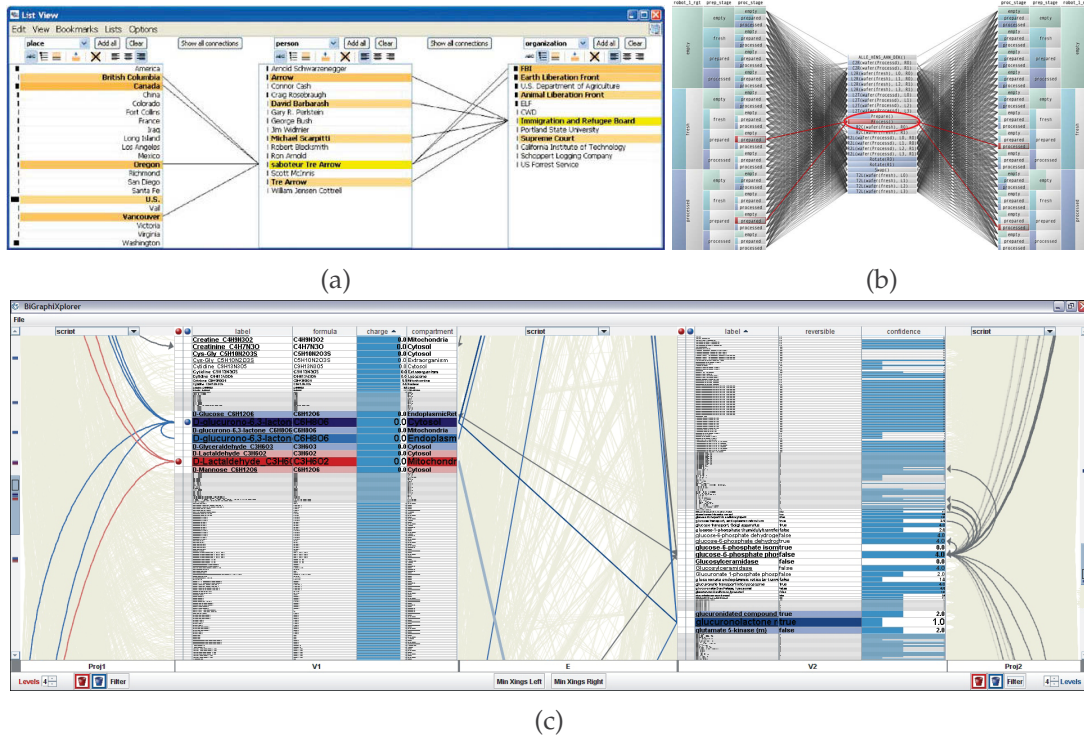


Figure 2.2.8: Different tabular representations using (a) color [SGL08], (b) filling level [PvW08] and (c) multiple columns [SJUS08] to communicate attribute values.

bles as shown in Figure 2.2.8. To communicate the structure, the rows of the tables are connected by links.

Attributes may then be visualized by using color [SGL08] (see Figure 2.2.8a), a filling level [PvW08] (see Figure 2.2.8b) or by introducing multiple columns for the attributes [SJUS08] (see Figure 2.2.8c). Especially the third approach allows the representation of the correct attribute values for each node and edge as labels in the corresponding column. Furthermore, these tables can then be sorted and filtered according to the attributes.

In this way, tabular representations facilitate the visualization and exploration of multiple attributes even for larger graphs. Yet, they do so on the cost of the readability of the structure.

2.2.3 Visualization of the Spatial Context

Attributes describing a spatial reference often pose higher demands to the visualization than discussed in the previous section. Besides reflecting the structural relations between nodes they also demand for communicating their locations in a spatial frame of reference such as a map. In the same way, a spatial reference can be given for the edges in the form of line strips, for instance when dealing with transportation networks [IYT⁺13]. The most common approach for their visualization is a direct mapping of the positions on top of a map display as shown in Figure 2.2.9a. If no spatial reference is given for the edges, this reduces the layout afford to the edge routing as it is done for flow maps using

edge bundling techniques [PXYH05, LBA10]. Two examples for such results are given in Figure 2.2.9b and 2.2.9c.

These solutions can be categorized into 3 classes based on how they maintain the spatial reference in the visualization:

- first, by an overlay of the subgraphs over a map,
- by adapting the map to facilitate an integration into the map, or
- by adapting the graph layout so that the nodes better reflect their spatial coordinates.

Adaptation of the map: An alternative is based on adapting the map by using cartograms or RecMaps [HKPS04]. They transform the irregular shapes of geographical regions into circular or rectangular shapes as shown in Figure 2.2.11. These transformed shapes are better suited for a direct embedding of the subgraphs. Therefore, the subgraphs may then be assigned to regions rather than specific geographic positions.

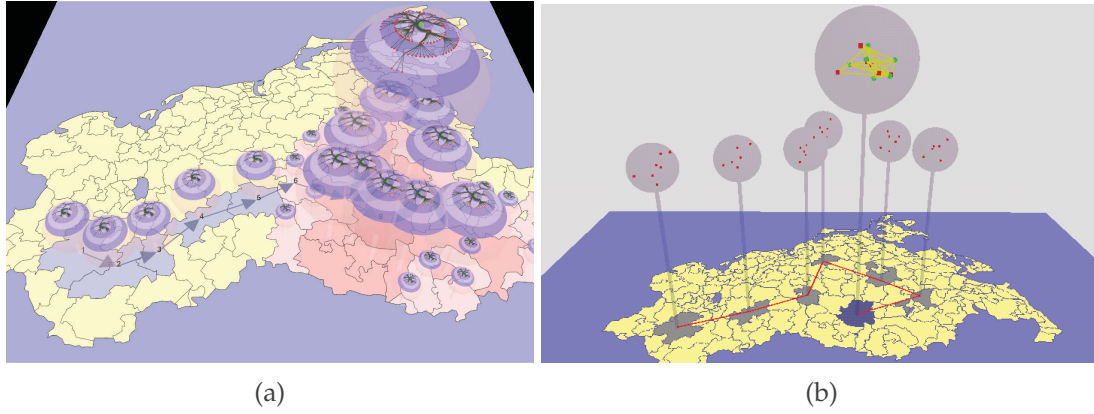


Figure 2.2.10: Visualization of graphs over a map (a) by a direct overlay of the subgraph representation [ST11] or (b) by an eccentric positioning of the subgraph representation [SK03].

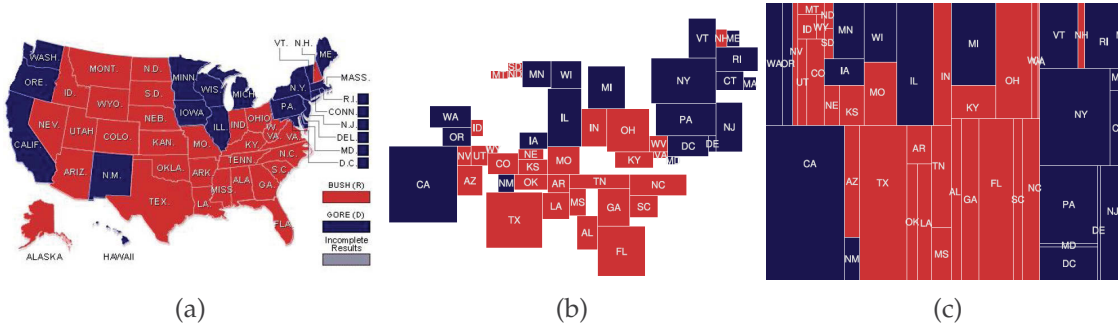


Figure 2.2.11: Adaptation of the map display for a simpler integration of graph visualization techniques by transforming the irregular shapes into rectangles [HKPS04]. (a) the original map, (b) an adapted variant of the map, and (c) a space-filling adaptation of the map.

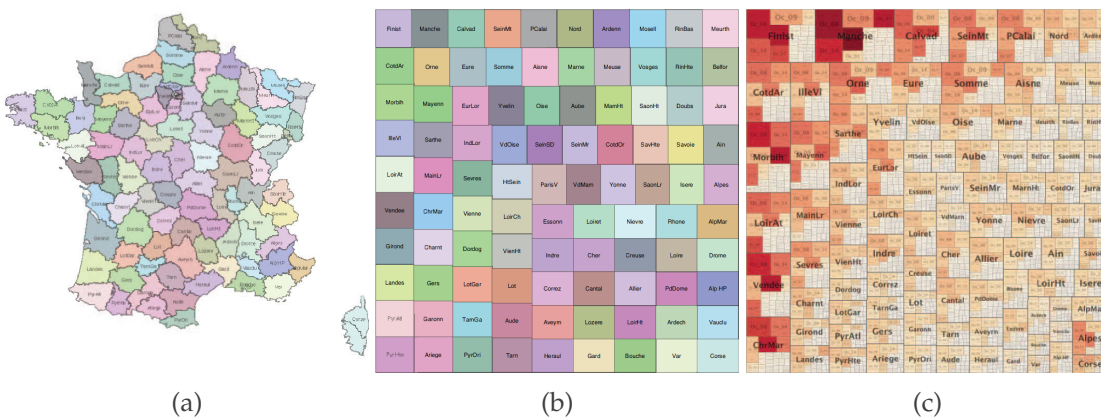


Figure 2.2.12: Adaptation of the graph layout to reflect the geographical context on the basis of a Treemap visualization [WD08]. The display space is distributed to the children in a way that the overall geographical displacement is minimized. (a) the original map, (b) an adapted space distribution minimizing the overall error, and (c) the final visualization of the graph.

Adaptation of the graph layout: This also holds true for the last class of techniques that follows a similar idea. Here the layout is adapted so that the positions of the nodes and edges better reflect their spatial coordinates. Therefore, these approaches try to minimize the error between node position and geographical coordinate during the layout calculation [SDW08, WD08]. In Figure 2.2.12 an example for such an adaptation of a Treemap visualization is shown.

The first solution maintains the direct relation of subgraphs to their spatial context but possibly with additional occlusions. Whereas the last two solutions allow a much better space utilization at the cost of spatial error. Yet, a direct embedding into the original irregular regions has not been addressed so far.

2.2.4 Visualization of the Temporal Context

As for the spatial context the representation of the temporal context of dynamic graphs poses additional demands to the visualization. Their visualization has mainly evolved from the techniques used for static graphs. Thus, most visualizations tend to focus on conveying the structure over time adapting the existing techniques for instance by using animations. Yet, as the number of collected time points has increased over the last years new visualization strategies to approach these dynamic graphs have been developed. These strategies focus more on the temporal aspect thus providing a quite different view on the data. It is noteworthy for the importance of this topic, that this increase of visualizations has also led to a recent publication of a State-of-the-Art report for dynamic graphs [BBDW14] which however focuses mostly on the graph structure and a design space of temporal graph visualizations [KKC14a] whose dimensions are based on a structural (graph representation) and a temporal encoding (time shown explicitly as multiple time points or embedded into the graph representation).

The first class of techniques focusing on the structure are well perceived in the literature (see for instance the State-of-the-Art reports [vLKS⁺11, BBDW14]). Yet, the second class focusing on time are often neglected when deriving new visualizations for dynamic graphs, even the design space [KKC14a] captures only some of them under a very general “other” term. Therefore, this section tries to provide a more complete and broader view on the problem of visualizing dynamic graphs. Here, the different visualization techniques are grouped based on their focus into three different classes: those focusing primarily on the structure, those focusing on the temporal aspect and finally those trying to find a balance between structure and time.

2.2.4.1 Focus on Structure

Techniques of this class tend to utilize most of the screen real estate to communicate the structure of the dynamic graph. Therefore, they often build on the representations described in Section 2.2.1 for the visualization of the structure. They mainly differ in the way they convey the dynamic nature of these graphs. On the one hand, there are approaches based on the supergraph providing a structural overview of the whole dynamic graph in a single image. And on the other hand, there are approaches based on

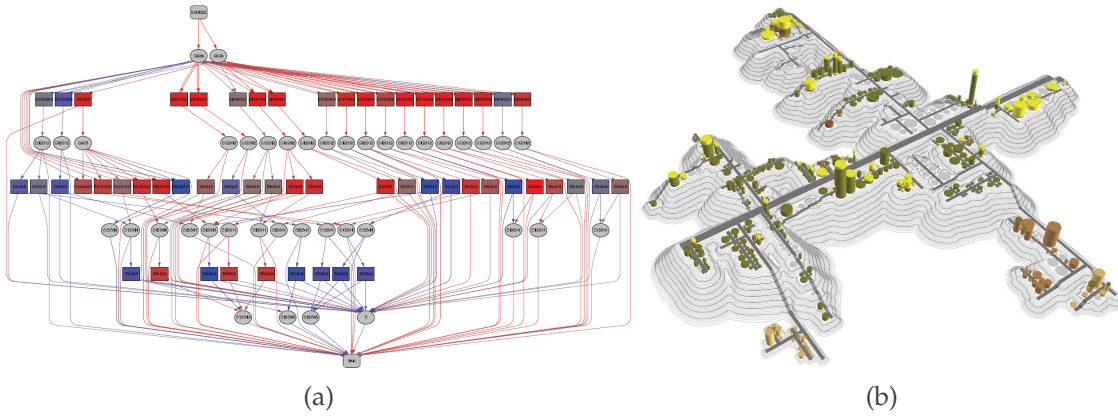


Figure 2.2.13: Different visualizations based on the supergraph of the dynamic graph: (a) color is used to communicate the first appearance of nodes and edges (red: earlier, blue: later) in a node-link representation [RUK⁺10]. (b) z-position, height and color of the cylinders encode the first appearance of nodes, the number and the source of changes [SL10].

providing multiple drawings each showing the structure of the graph at a different time point in more detail.

Supergraph based approaches: These techniques first calculate the supergraph SG of the dynamic graph DG by summarizing the graphs of all time points $G_i \in DG$ as described in 2.1. This supergraph allows them to show an abstract and cumulative view of the graph structure over all time steps. Hence, they can give an overview of the graph with all nodes and edges that were present at least for one time step. Such a view can provide insights into the overall connectivity of nodes and edges allowing for instance the determination of structurally interesting nodes (e.g. nodes with high node degree or connecting cliques).

Yet, it can also lead to false conclusions. For example, a spammer in a communication network having only a few connections yet to different sets of nodes at each time point will result in a node having a much higher node degree in the supergraph. Thus, new attributes used for the visualization are often calculated along with the construction of the supergraph such as the time point a node was created or last modified. These attributes then allow on the one hand the discrimination of such a spamming node from nodes having a high node degree over all time points. And on the other hand, they provide additional feedback of the development of the graph over time in general.

Furthermore, the supergraph defines only the process of summarizing the structure into a single graph, yet the handling of the attributes is left open. Here, two different approaches are commonly employed in the literature:

- First, the attribute values for each node and edge are reduced into a single value that can be directly incorporated into the visualization as described in Section 2.2.2.1 for the mapping on visual variables.
- Or a subsampling is used to reduce the number of values to capture the major trend of the evolution of each node and edge that is then embedded into the structural

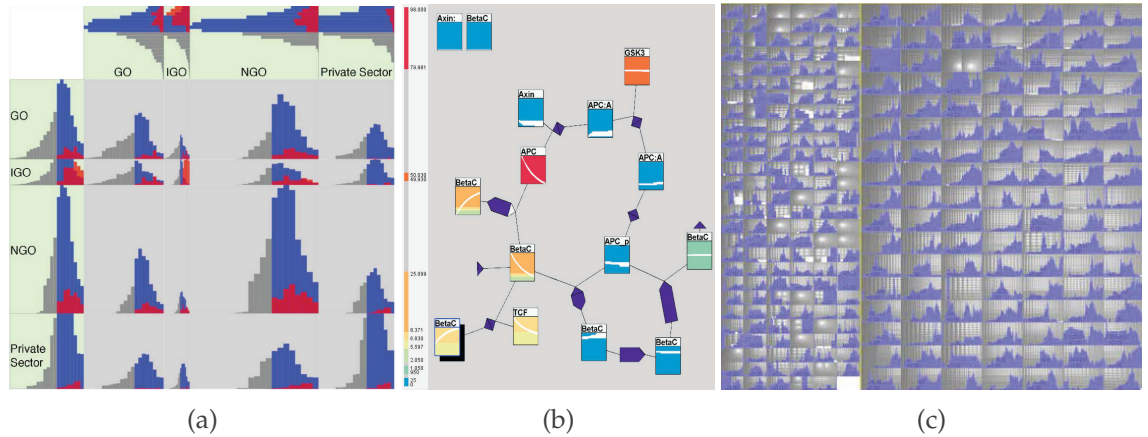


Figure 2.2.14: Embedding of time plots into supergraph based visualizations: (a) matrix representation showing temporal course of attributes for nodes and edges [YEL10]. (b) node-link representation with small time plots representing the nodes [US09] and (c) implicit representation [Tel06] showing development only for nodes.

representation similar to the usage of glyphs described in Section 2.2.2.2.

In Figure 2.2.13 two examples are shown following the first approach. The node-link visualization in Figure 2.2.13a utilizes color to map the creation date or the existence of nodes and edges at specific time points thus supporting the analysis of the graph's temporal development [KNC⁺11, RUK⁺10]. Whereas for the implicit visualization in Figure 2.2.13b the creation date is mapped on the z-coordinate of the cylinder representing the nodes [SL10]. Furthermore, the number of changes over all time points was summarized into a single value and mapped onto the height of each cylinder.

For the visualization of major trends often small time plots are embedded into different kinds of representations: matrix views [SWS10, YEL10], node-link visualizations [BBD08, US09] and implicit techniques [Tel06, TS07]. Some examples are exemplified in Figure 2.2.14. In contrast to the mapping of single values, these plots can only be embedded into the visualization if enough space is available.

Overall, these techniques can provide a good first overview on the structure and also on the major trends of attributes. Yet, especially structural changes are hard to grasp with a supergraph based approach as these have been summarized into a single static image.

Approaches based on multiple drawings: Here, structural changes are better reflected because the graph structure is shown separately for each time point allowing the comparison of multiple time points. In this way, additions, deletions or movements of nodes and edges can be better perceived as their visual counterparts also exhibit these characteristics. Larger changes of attribute values may be visible if the mapping on the visual variables also result in distinct changes of the counterparts. To convey these separated views and thus the dynamics of the graph typically two approaches are used:

- Mapping time on space and thus represent the dynamic graph by multiple small drawings [PS12, SLN05].

- Mapping time on time and thus visualize the dynamic graph using animations [FE01, BPF14a].

Showing all time points as small multiples at the same time allows their direct comparison. Yet, this also limits the space available for each drawing and thus affects the readability of the graph at each time point. Animations on the other hand allow the utilization of the whole display solely for the graph of one time point. However, for comparing different time points the user has to jump back and forth between them. Thus, which approach to use depends on the goal of the user which has been confirmed in a recent study [APP11]. Most approaches tend to use animations as they are perceived more natural by the user.

Independent of how time is represented the most important goal is the maintenance of the mental map of the graph structure as good as possible [PHG07]. Originally, graph layouts try to fulfill specific constraints to provide a single readable image of the graph structure for one time point. Thus, static graph layouts do not consider the layout of other time points. This can lead to movements of nodes and edges that are not even related to any changes within the data. As a result, it becomes difficult to track nodes and edges over time but also real changes can be drowned out by this additional movement.

Solving this problem is the common goal of the classical dynamic graph drawing approaches [Bra01b]. Therefore, a wide variety of different concepts exist to stabilize the graph layout. In [BIM12] these have been grouped into the following three classes:

Aggregation: A layout is calculated on the basis of the supergraph and is then used to lay out the graph of each time point.

Linking: The graphs of all time points are linked by additional edges between the appearances of the same node in graphs of consecutive time points. A layout is then calculated for the resulting combined graph and thus for all graphs.

Anchoring: Only the layout of a single time point is calculated. Yet, additional edges are used to anchor nodes to specific positions such as their previous location according to the layout of another time point.

Layout techniques based on aggregation provide the highest stability as the nodes have a fixed position over all time points. But they can also result in bad layouts considering the readability or space utilization for an individual time point. In the lower row of Figure 2.2.15 the graphs of three different time points laid out by such an approach are shown. These layouts tend to leave open space where nodes are added later on.

Layout techniques based on linking have a higher adaptability to the structure of each time point. Yet, the calculation effort increases with each new time point as the number of nodes and edges to be considered in the layout is multiplied by the number of time points.

The last class of layout techniques based on anchoring the nodes features the worst stability of all as nearly no information of the dynamic graph is used. However, this is also the advantage of these techniques making them the most versatile. Especially, when the complete dynamic graph is not known in before but arriving in a stream (also referred as online dynamic graph drawing [DG02, FT08]) this is the only class of layout techniques

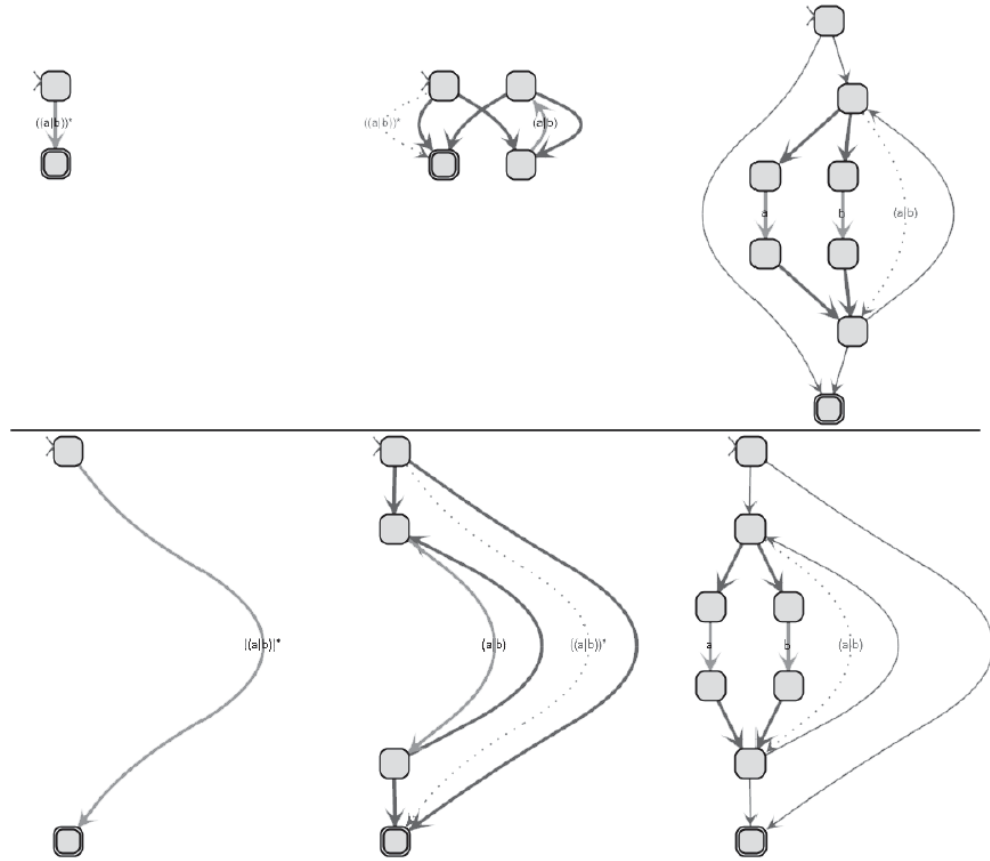


Figure 2.2.15: Animated visualization of dynamic graphs. The upper row shows an ad-hoc layout resulting in much node movement over time. For the lower row a foresighted layout was used that is based on a supergraph to minimize the node movement [DGK01].

that is usable without additional problems.

While both approaches, those based on supergraphs and those based on multiple drawings, are very different in the way they handle the temporal domain there are approaches allowing their combination. For example, the Gephi graph visualization platform [BHJ09] allows the user to specify a time window and thus an interval for which the supergraph is calculated and visualized. This window can then be moved across the time axis resulting in a new supergraph laid out according to the previous one and the transition is then animated. Contrary, in [RM13] an approach was introduced that allows the user to subdivide the temporal axis into smaller time intervals and to select different representations for each of these intervals.

2.2.4.2 Focus on Time

The previous approaches have largely focused on the structure of dynamic graphs. This is feasible as long as a rough overview of the structure and attributes is of interest or only a couple of time points have to be analyzed. The analysis becomes increasingly difficult and time consuming with a rising number of time points. Finding interesting

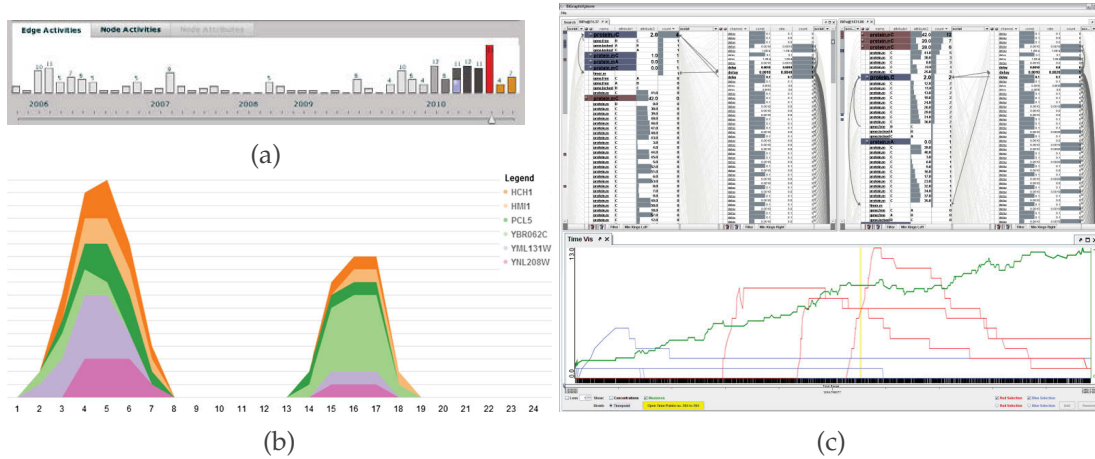


Figure 2.2.16: Statistical plots of graphs over time summarizing activities of (a) the edges [ATMS⁺11] and (b) the nodes [CXP⁺12] or (c) providing an overview of the structural complexity (green) and attribute values for a couple of selected nodes (red and blue) [JSS⁺13]. These views provide the basis for determining interesting time points for further inspection.

time points according to large structural changes or specific changes of attribute values can be a tedious endeavor using animations. Here, techniques focusing on the temporal aspect of the dynamic graphs may provide a better overview of the dynamic nature.

At the moment, there are only a few techniques with a focus on time. Yet, they already range from abstract statistical plots summarizing the overall dynamics of the graph to 1.5D visualizations providing a detailed visualization of attributes and structure for a specific node over time. Besides that, one technique also addresses a more complex branching time axis.

Statistical plots: These visualization techniques are based on abstracting the graph capturing its structural and attribute related characteristics by a collection of a few values and plotting them over time. Structural characteristics can be described by graph complexity measures [BB05] such as the overall number of nodes and edges. Such a complexity measure is plotted over time [JSS⁺09] in green in Figure 2.2.16c. Attribute related properties can be visualized by aggregating node and edge attributes over time. This is shown for edges [ATMS⁺11] in Figure 2.2.16a and nodes [CXP⁺12] in Figure 2.2.16a. Furthermore, these characteristics may be calculated for the whole graph or subgraphs such as clusters [FBS06] or even specific elements as shown in Figure 2.2.16b and 2.2.16c.

This gives a very basic overview of the development of the graph without showing any details of the structure. Yet, it allows the determination and selection of time points for which the structure changes and hence should be investigated in more detail. Therefore, most techniques following this idea provide additional detail views such as shown in the upper part of Figure 2.2.16c to analyze interesting time points. In this way, this class of techniques represent an important addition to the structure focusing approaches since they provide a quite different view on dynamic graphs.

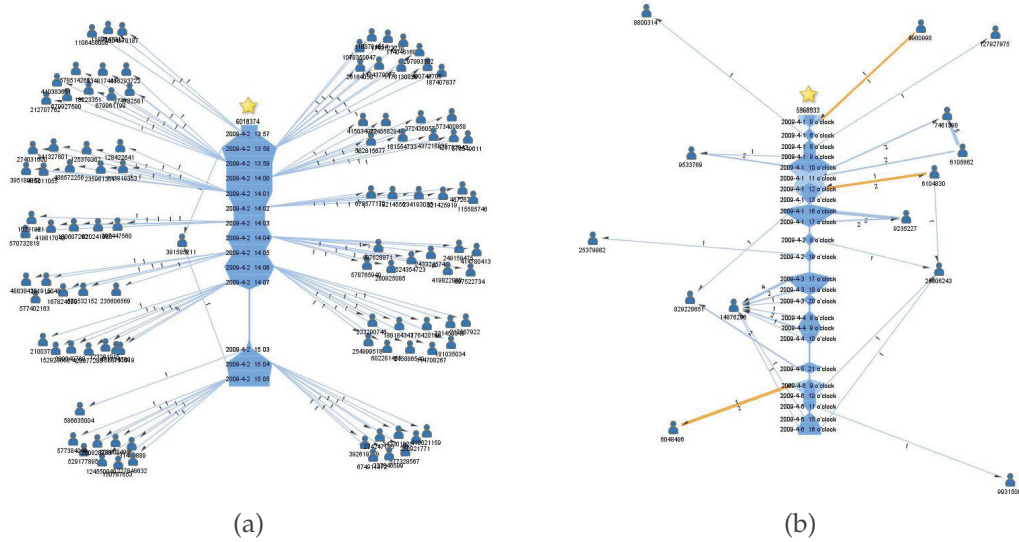


Figure 2.2.17: 1.5D visualization of a dynamic graph showing the temporal development of a selected focus node along the y-axis. Nodes sharing connections with the focus node are linked to those time points where the connections appear [SWW11]. In this way different kinds of nodes can be distinguished: (a) spammer pattern showing connections with different sets of nodes over time and (b) normal behavior showing connections with the same nodes over time.

1.5D visualization: This visualization combines the two previous approaches as it provides an overview of an attribute such as node degree and also the structural information over time but only for a single node. Therefore, all adjacent nodes are extracted and visualized at both sides of the plot representing the time varying node attribute. These nodes are then connected with lines to all time points of the plot iff they share a connection at that time.

This provides a very detailed view for a single node as shown in Figure 2.2.17 for the visualization of a communication network [SWW11]. It shows two different patterns found in the dynamic graph. On the left, a typical spammer pattern showing connections of the node to different groups of nodes at different time points. And on the right, normal behavior showing only a few connections which are maintained over multiple time points.

Yet, similar to animations, an analysis of the whole dynamic graph can become time consuming with this approach, since every node has to be analyzed separately. But it can provide an interesting counterpart to supergraph based approaches where an overview of the structure is given in which interesting nodes may be determined more easily for a closer inspection with a 1.5D visualization.

Visualization of branching time: Almost all techniques discussed so far are based on a linear time axis. Yet, especially in the domain of modeling and simulation more complex temporal domains can be found. In such cases, the temporal domain itself can be described by a graph (see Section 2.1.3). While a supergraph could still be calculated, utilizing animation approaches or statistical plots becomes difficult as it is not clear in which order to traverse the time points. Here, it is important to first convey the nature of

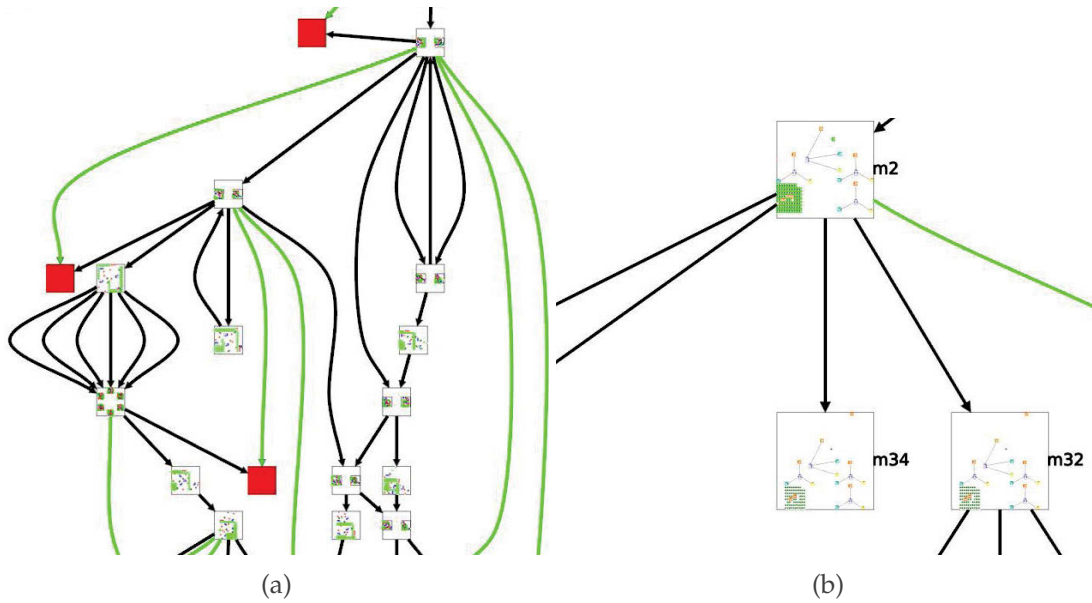


Figure 2.2.18: Visualization of a dynamic graph with a branching time axis. (a) The complex time axis itself is visualized as a graph with a node-link representation. The time points are drawn as rectangles in which small node-link views of the dynamic graph are embedded (see (b) for a larger view). Transitions between time points are rendered as links with edges representing the direction of the transition [PMD12].

the temporal domain.

In [PMD12] a visualization was proposed that represents the branching time axis by a node-link representation. Here, rectangular nodes represent the time points, and the transitions between them are shown by links. If sufficient space is available small graph views are embedded into the representation of the time points to show the graph structure at that time. This visualization is exemplified in Figure 2.2.18. On the basis of this visualization they provide a variety of interaction techniques such as the selection of a path to be used for an animated visualization of the structure.

While this is a very specific visualization it nicely demonstrates the gain of an interplay of time and structure focusing visualizations.

2.2.4.3 Balancing the Focus between Structure & Time

Both classes of approaches, those focusing on time and those focusing on structure, provide a rather restricted view on dynamic graphs showing either structure or time and try to embed the respectively other. Now, techniques are considered that try to find a compromise between both. Therefore, most techniques of this class rely on a layered approach to show the development of the graph structure. Here, each layer represents the structure of the graph for a specific time point. By showing multiple layers simultaneously arranged along a time axis these techniques provide a similar representation of the dynamic graph as small multiples. Yet, they often connect the occurrences of nodes between subsequent layers by additional links thus making the temporal relations between the corresponding time points explicit.

These techniques differ mainly by:

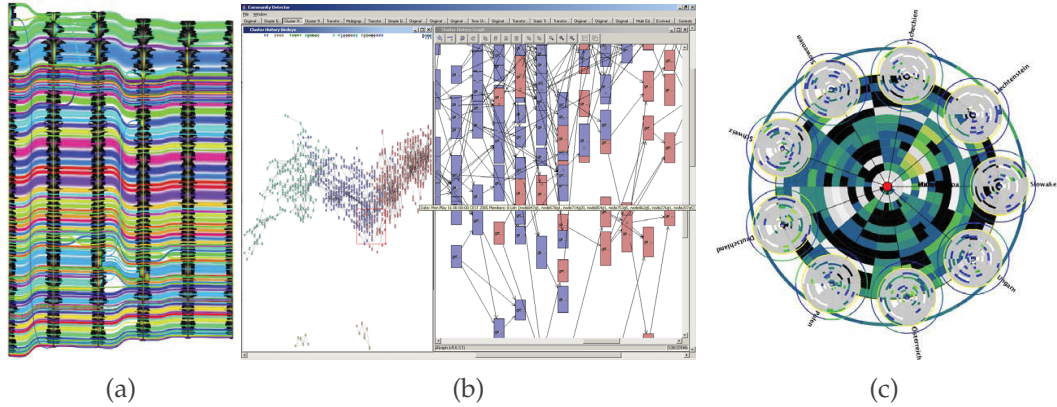


Figure 2.2.19: Different 2D layered visualizations of dynamic graphs: (a) implicit visualization based on Icicle Plots [TA08]. Nodes appearing at different time points are connected via colored links providing feedback of creations, deletions and movements in the graph. (b) explicit visualization of a clustered graph [FBS06]. Multiple links leading to and from nodes represents merging and split events of clusters. (c) implicit visualization based on a radial layout of nodes [BD08].

- the dimension they use to represent time (2D or 3D),
- the alignment of the layering (axis-parallel or radial), and
- the links they draw between the layers.

Dimensionality: As the layering of graph representations introduces a new dimension, the time axis, the dimensionality of the layered visualization depends on the dimensionality of the basic graph representation.

Hence, 2D visualizations such as those shown in Figure 2.2.19 are based on a 1D graph representation which leaves the second dimension for the time axis. However, such 1D graph representations rely on a linear ordering [BVB⁺11, vdEHBvW13] of the nodes which is often only feasible for simpler graphs. For instance, Figure 2.2.19a is a visualization for dynamic hierarchies using Icicle Plots which are basically a hierarchical 1D subdivision. Similar approaches were introduced in [FBS06, RTJ⁺11, RB10] for the visualization of clustered dynamic networks as depicted in Figure 2.2.19b. While they visualize dynamic networks they focus mainly on the evolution of clusters and thus neglect the interconnections of clusters existing at the same time. A more general visualization shown in Figure 2.2.19c represents nodes (large inner circle) along their connections (smaller outer circles) as circle sections [BD08]. Especially as the size of the outer circles depends on the number of nodes this techniques is only suited for smaller graphs.

Yet, most commonly 2D representations are used as a basis and thus a layering results in a 3D visualization [BPF14b, BC03, EHK⁺04, GW06] as shown in Figure 2.2.20. Compared to the special 1D layouts 2D representations provide in general a better readability of the structure. Yet, 3D visualizations are always accompanied by overlap and occlusion requiring additional interactions. Furthermore, determining the layer on which nodes lie may become more difficult because of the 2D projection and thus the missing depth perception. For this problem, visual cues such as semi-transparent rectangles to repre-

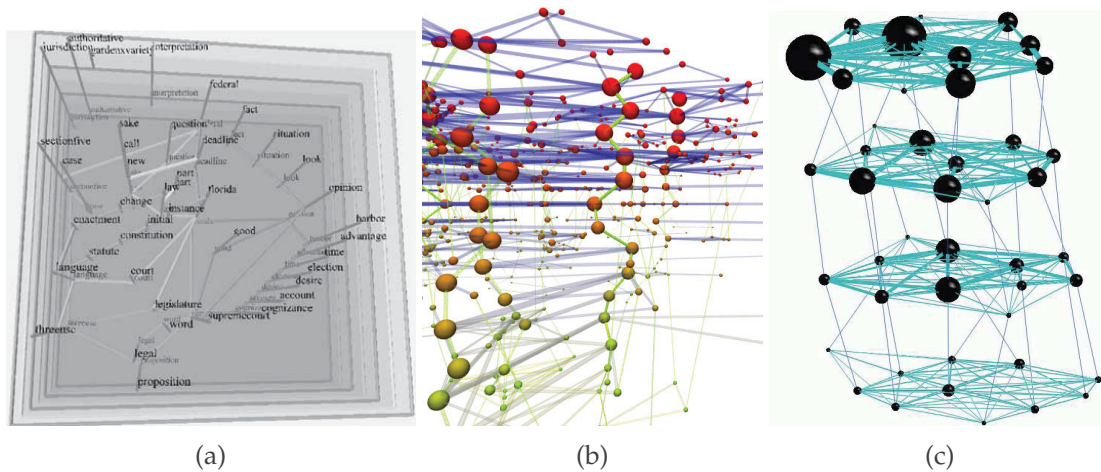


Figure 2.2.20: Different 3D layered visualizations of dynamic graphs. The third dimension represents the time. Each link between two layers connect the appearance of the same node at those time points. (a) semi-transparent layers are used to de-emphasize older time points [BC03]. (b) color is used to encode the time points [GW06]. (c) node size is used to convey attribute values allowing the analysis of attributes over time [EHK⁺04].

sent the layers (see Figure 2.2.20a) or a layer dependent color mapping of nodes (see Figure 2.2.20b) are used to increase the readability.

Alignment: Most layering approaches rely on an axis-parallel alignment as most of the depicted visualizations show to convey the temporal axis in both 2D and 3D. One exception is exemplified in Figure 2.2.19c using a radial layering in 2D [BD08] similar to the tree ring metaphor. Its unfamiliar design is both a burden and an advantage of the visualization. As the authors stated it needs some training to get used to. But therefore it allows an overlap free visualization of the dynamic graph.

Inter layer linking: The most important information is nevertheless contained in the links visually representing the temporal development of the graph. Except for the radial technique shown in Figure 2.2.19c which uses adjacency, almost all layering techniques explicitly link all occurrences of nodes by introducing additional lines between the layers. Thus, each node is represented by a sequence of lines allowing its observation across multiple time points. In this way, changes of attributes that are visually mapped can be analyzed by just following these sequences. These explicit connections also allow the identification of additions and deletions in the graph structure as these are represented as starts and endings of such sequences. Especially interesting for hierarchical graphs, movements in the hierarchy may be perceived as overlapping lines as exemplified in Figure 2.2.19a. When visualizing clustered graphs these links are often used to communicate the evolution of clusters. As it is unlikely that exactly the same cluster is found in two different time points multiple links may emerge from or meet at the same cluster. These links then describe merge and split events of clusters [FBS06]. An example for such a visualization is given in Figure 2.2.19b.

The major advantage of the layering techniques is their explicit visualization of the temporal development. This relieves the user of finding nodes in consecutive time points himself and thus supports him in analyzing structural as well as attribute related changes. Yet, the available screen real estate limits the number of nodes and edges as well as time points that can be shown simultaneously. Furthermore, showing all links between the different layers of larger graphs will result in massive clutter especially regarding 3-dimensional layering approaches.

2.2.5 Visualization of Multiple Aspects

The previous sections have shown the difficulty of visualizing the different aspects of graphs: structure, attributes, spatial and temporal context. Their size often neglects to show every information in a single visualization. This problem resulted in a wide variety of techniques addressing specific subsets of these graphs based on different combinations of aspects. This variety poses the problem for the user to select the right visualization to achieve his goals. However, as each visualization with its specific focus on the data may provide useful insights not achievable with a single visualization alone this variety also bears much potential for a more flexible and thorough analysis.

Because of this reason many visualization frameworks try to integrate different visualization techniques and thus to combine their advantages for the analysis. Generally, there are two ways for their combination:

- Embedding a different visualization into a base visualization.
- Showing two or more visualizations side by side.

The advantage of the first approach is the direct linking of what is shown in the embedded visualization to the content of the base visualization. One example is the local adaptation of a part of a node-link representation by embedding a matrix view as discussed in Section 2.2.1.4. Such a hybrid visualization may better reflect the structure of the graph. Other embedding examples include the glyphs discussed in 2.2.2.2 to show multiple attributes and small time plots in 2.2.4.1 to show the temporal aspect besides the structure. However, such an embedding may not be possible at any time as the base visualization can already be very dense and thus does not provide enough space for a direct embedding.

This problem can be avoided by showing these visualizations in separated views. Yet, this separation often comes at the cost of losing the direct linking making it necessary to introduce additional linking or coordination mechanisms such as similar color mappings or highlighting and linking corresponding data items on demand. Therefore, this kind of approach is also referred to as *multiple coordinated views* and is the most common way for combining a multitude of visualizations.

Two examples are shown in Figure 2.2.21. The first views setup (Figure 2.2.21a) concerns the visualization of a high number of attributes. Here a parallel coordinates visualization is used to visualize the attributes whereas a node link view represents the structure of the multivariate graph [SHQ08]. A couple of nodes have been selected resulting in their highlighting in both views. The second example (Figure 2.2.21b) shows

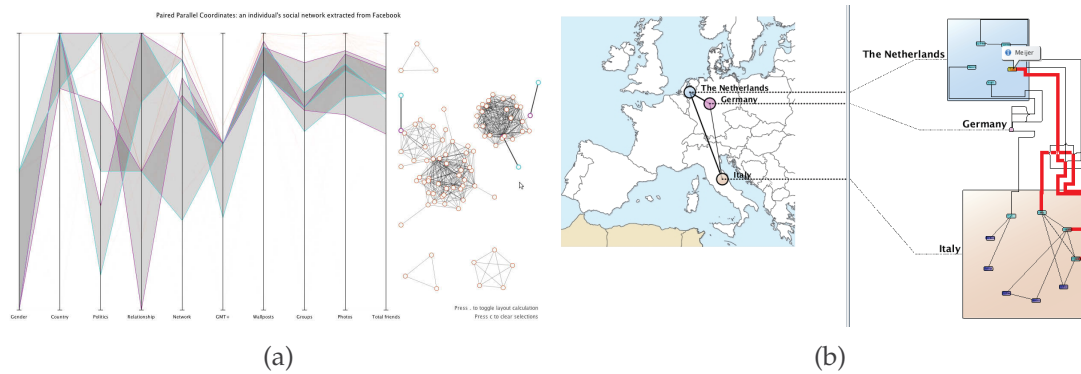


Figure 2.2.21: Combination of visualizations showing different aspects: (a) a parallel coordinates plot shows a multitude of attribute associated to the graph shown in the second view [SHQ08]. (b) a map display visualizes the spatial context of a graph [GDLP09]. Explicit links connect subgraphs with their geographical location.

a graph and its spatial context visualized in a separated map view [GDLP09]. Here, beside a similar color coding the subgraphs are also connected by explicit links with their associated geographical positions.

2.3 Scalability

The problem of visualizing large graphs is caused by the different aspects that must be considered: structure, attributes and time. Already one of these aspects, e.g. a large node set, can pose a challenge for the visualization. The combination of multiple aspects even aggravates this situation. This problem is captured best by the concept of the *visual entity budget* as introduced in [EF10]. The visual entity budget represents an upper boundary for the number of visual entities that can be displayed simultaneously or perceived by the user. This budget can be the result of limited screen space, limited processing capabilities, or even the limits of the human cognition. Thus any visualization has to reduce the overall number of displayed items to fit this budget. For this reason, it does not permit to show every aspect or in special cases even a single aspect in full detail.

Therefore, current visualization techniques try to find compromises regarding the level of detail on which each of these aspects is shown. In principal, there are two ways to reduce the amount of data and thus to stay below a given visual entity budget [LA93]:

- Selecting a subset of the graph which is of particular interest with regard to a given task at hand for example selecting subgraphs, time intervals or specific attributes, or
- Abstracting the graph so that multiple data items are mapped onto a smaller amount of visual entities such as clustering nodes or aggregating time points.

Whereas a selection may preserve individual details it also leads to a loss of the general overview of the whole graph. On the other hand, an abstraction may preserve an overview of the overall characteristics but is also drowning out the details.

Most visualization approaches follow a two-fold strategy first providing an abstract overview facilitating the determination of interesting subsets in the data. These subsets are consequently selected in this overview to show in more detail. Thus in the following, abstraction techniques for the different aspects are described before presenting ways to specify selections in large graphs.

2.3.1 Abstraction

Abstractions may be performed individually on any aspect, e.g. abstracting cliques into clusters, or on combinations of aspects, such as grouping nodes with similar attribute values. However, most of the existing techniques focus on structural or temporal abstractions. Only a few consider both at the same time. And nearly none are based on abstracting and thus reducing the number of attributes. One example for a more general approach in this direction is the principal component analysis [VMCJ10] removing redundancies in an n -dimensional data space corresponding to the attributes of the graph. This is done by calculating orthogonal axes within this nD space eliminating any correlated axes.

The remainder of the discussion focuses on the aspects structure and time.

2.3.1.1 Abstraction of the Structure

For abstracting the structure a multitude of techniques exist. They are either based on performing the abstraction in data space or visual space. Abstractions in data space are based on reducing the amount of data prior to the visualization. Whereas abstractions in visual space are based on mapping multiple data elements onto a few visual entities. Most approaches have been developed with static graphs in mind. For most of them extensions exist allowing their application to dynamic graphs.

Abstractions in data space can be grouped into three classes:

- By using measurements to describe the general characteristics of the graph.
- By coarsening the graph thus removing unimportant nodes and edges.
- By clustering groups of similar nodes and edges into clusters or metanodes.

The first class is a strong abstraction of the structure merely reducing it to a number of values. These values may describe complexity measures [BB05] such as the number of nodes and edges, or aggregated node and edge attributes [ATMS⁺11, CXP⁺12]. These measures can also be applied to abstract subgraphs [FBS06]. Examples based on such measures are the statistical plots described in 2.2.4.2 providing a temporal focus on dynamic graphs.

The second class tries to reduce the structure by local adaptations. These adaptations may be simple rewriting rules replacing specific patterns in the graph by small glyphs [Arc09, DS13]. Two patterns are shown in Figure 2.3.22a and 2.3.22b. The first pattern concerns groups of nodes sharing their only adjacent node and which are then replaced by fan glyphs. Whereas the second pattern captures equivalent parallel paths connecting

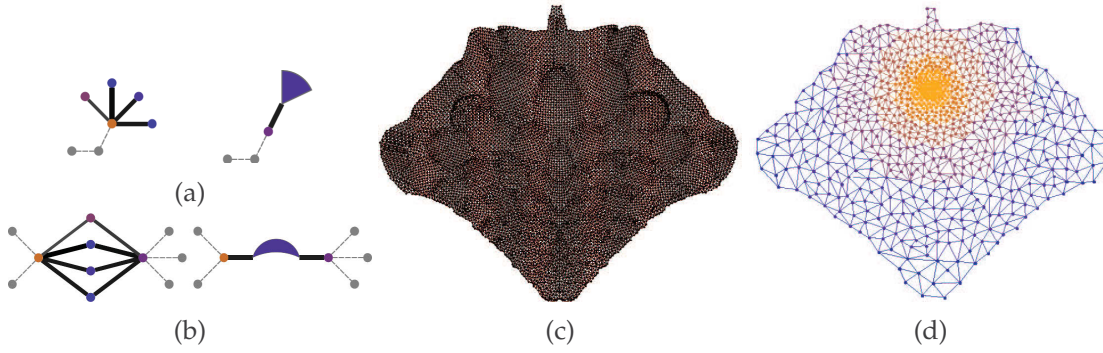


Figure 2.3.22: Different coarsening strategies for graphs: (a) a fan-glyph abstracting a group of nodes connected only by a single head node to the rest of the graph [DS13]. (b) a parallel glyph abstracting a group of functionally equivalent span nodes connecting two anchor nodes [DS13]. (c) a large graph and (d) its coarsened version [GKN05]. Colors represent different levels of detail.

the same pair of nodes. In general, coarsening or stratification approaches iteratively merge groups of nodes while maintaining the topology of the graph to gain a rough approximation of the overall structure [GKN05, KG06]. These coarsening approaches are often based on fast heuristics for determining important nodes to maintain and merging them with their adjacent nodes to create a coarsened version of the graph. In this way, an overview of the fine graph can be given by a visualization of the approximation and then details can be embedded on demand as shown in Figure 2.3.22c and 2.3.22d by using a topological fisheye.

Clustering techniques are similar in the sense that they also merge groups of nodes. The difference between clustering and coarsening is that clustering approaches do not try to create an approximation of the graph but to find subgraphs sharing similar properties. An example is the separation of the graph into large cliques and thus into subgraphs consisting of well-connected nodes [FBS06]. Generally, graphs are clustered according to structural properties [BC01, BDL⁺10] or to similarities in the associated attribute values [PvW08].

For the visualization of coarsened or clustered graphs often additional interactions are included [HE98]. Therefore, the graph is first drawn on an abstract level. The user can then interactively steer the level of detail to be shown by expanding or folding clusters. Expanded clusters are then embedded in detail showing the contained nodes and edges. Rectangles are often surrounding these nodes and edges to represent their cluster affiliation. An example for such an embedding is shown in Figure 2.3.23a where the subgraph of the expanded cluster is drawn using a radial layout [BDL⁺10].

In case of dynamic graphs, these clustering approaches are often applied separately for each time point which can lead to sudden changes in the clustering. To lessen such effects and to increase the calculation time extensions for dynamic graphs were developed. These extension range from averaging attribute values considered during the clustering over multiple time points [KG06] to adapting previous clustering results to the changed graph [GMSW10]. Based on these clustering results for the different time points the history of each cluster is extracted identifying events such as birth, death, merge and split

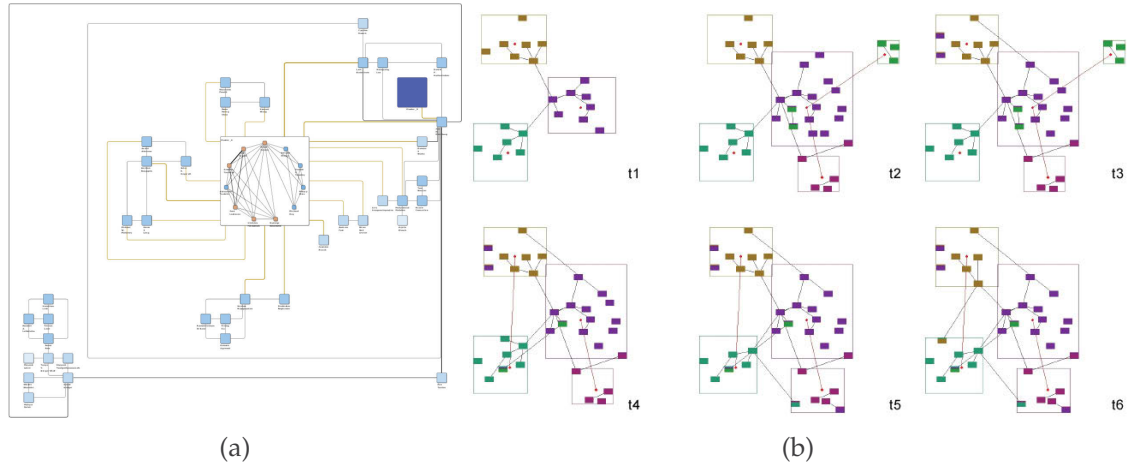


Figure 2.3.23: Visualizations for clustered graphs. (a) node-link representation of the clustered graph. The subgraph of an expanded cluster is drawn using a radial layout [BDL⁺10]. (b) animation of a clustered graph over 6 time points [FT04].

of clusters [GDC10]. This information can then be used in the visualization of the graph. This can be explicitly shown by a layering approach as already discussed in Section 2.2.4 using links between the layers to represent these events. Or by using animations as exemplified in Figure 2.3.23b. Here, clusters as well as their elements are shown. The growth of clusters as well as the movement of nodes from one cluster to another etc. are animated [FT04].

Abstractions in visual space are based on first calculating a layout of the graph and visually aggregating nodes and edges that are located close to each other. An example for such a class of approaches is edge bundling for aggregating groups of parallel edges to reduce visual clutter as discussed in Section 2.2.1.2. Just recently, extensions for dynamic graphs have been published taking multiple time points into account [HEF⁺13, NEH13] to create more coherent edge bundles over time. Other approaches are density based techniques aggregating the number of nodes and edges falling on the same pixel. Then instead of visualizing the graph directly its density field is visualized. This can be done while maintaining the overall graph structure as shown in Figure 2.3.24a and 2.3.24b rendering two separated density fields for nodes and edges [ZBDS12]. A more implicit visualization is shown in Figure 2.3.24c and 2.3.24d using a single density field [vLdL03]. Here red regions identify very dense regions in the graph leading to much overplotting and thus are interesting to investigate.

2.3.1.2 Abstraction of the Temporal Aspect

Almost all visualization approaches based on the abstraction of time points are using aggregation. Examples for such visualizations were already discussed in Section 2.2.4.1. To summarize, they first calculate the supergraph. Then depending on the level of aggregation for each attribute either one value is calculated or a subsampling is performed reducing the overall amount to only a few values. Furthermore, structural properties such as creation or last modification dates etc. are extracted.

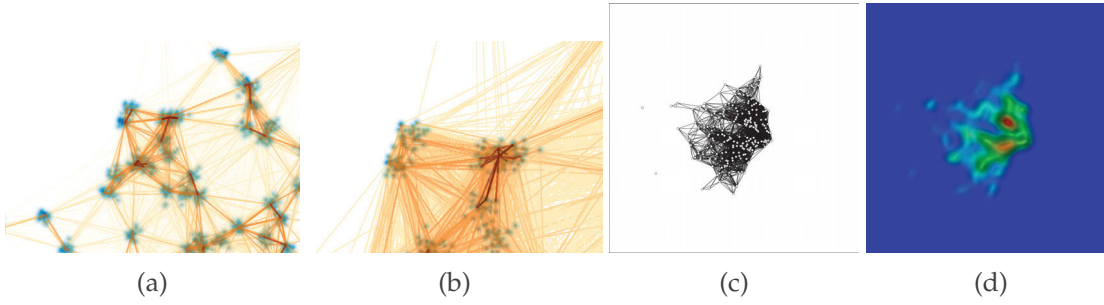


Figure 2.3.24: Different visual aggregations of the structure. (a) and (b) showing different zooming levels on a large graph providing different visual levels of detail [ZBDS12]. Blue color represents density-based node aggregation and orange/red color edge aggregation. (c) node-link representation of a graph and (d) its splat field representation showing high density of nodes and edges in red [vLdL03].

In this way, only subsequent time points are grouped and abstracted. In Section 2.2.4.2 one approach [PMD12] was described that relaxes this requirement by grouping specific time points anywhere on the time axis. This approach is designed for a rule-based modeling and simulation framework making it necessary to consider branching time. Therefore, they use a graph based description of the time axis for the visualization. During the simulation, they can automatically merge time points featuring exactly the same graph structure. This allows them to capture and describe cyclic phenomena in the dynamic graph.

However in general, it is unlikely that real world graphs will feature time points with exactly the same structure and attribute values. Especially for the analysis of dynamic graphs more sophisticated clustering techniques are proposed to group time points exhibiting a similar graph structure [BY08]. Therefore, different similarity measures and methods for calculating cluster centers are introduced that work together with common clustering methods such as k-means or agglomerative hierarchical methods. Examples for similarity measures are based for instance on *graph edit distances* [GXTL10] or on the size of the maximum common subgraph (as defined in Section 2.1). A center can be determined for instance by using the maximum common subgraph or selecting the *median graph* of a cluster which is a graph of the cluster that has the least distance to all other graphs in the cluster [BY08]. However, such a clustering of the temporal aspect according to the graph structure has not been applied for the visual analysis of dynamic graphs so far.

2.3.1.3 Abstraction of Structure and Time

The abstraction of structure and time simultaneously has only been addressed by two approaches. The first approach is only applicable for the analysis of two time points as it is based on clustering the *difference graph* [Arc09] which is basically the supergraph for two time points. In this difference graph, each node and edge receives a label describing its affiliation to these time points. Possible labels are: existence at the first, existence at the second and existence at both time points. Based on these labels the graph is clustered in a topology preserving or *path-preserving* way. That means, each clusters contains

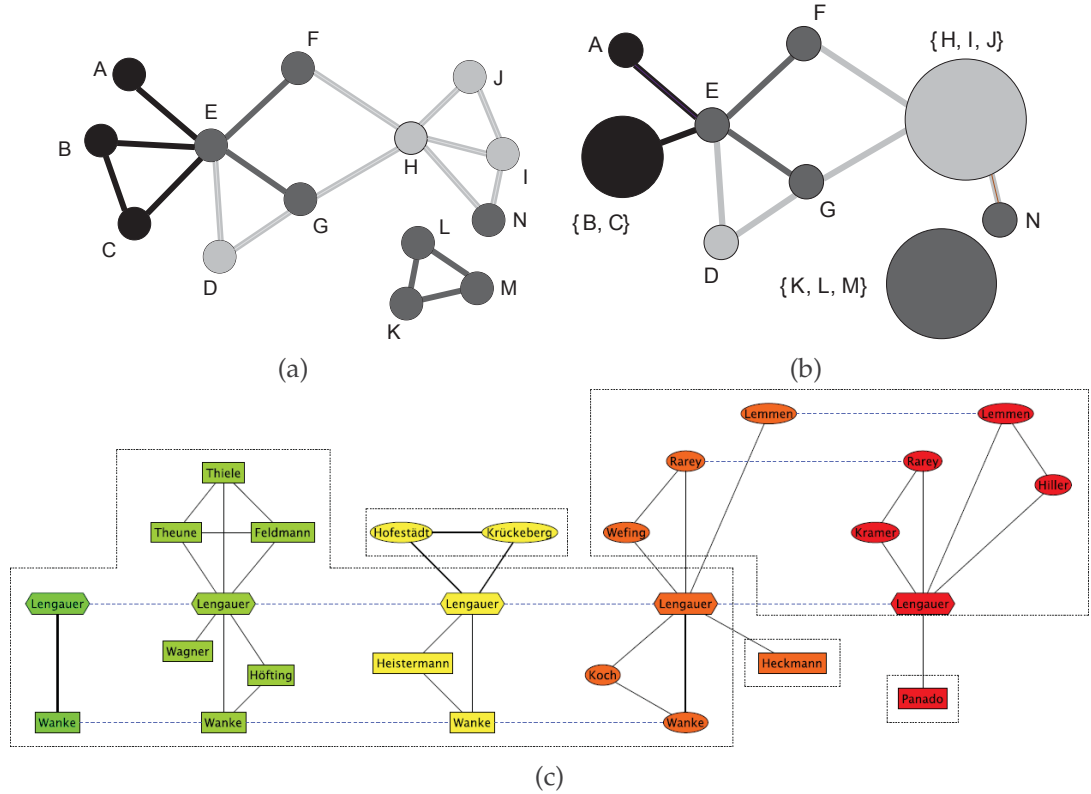


Figure 2.3.25: Visual abstraction of structure and time. (a) shows the difference graph and (b) its clustered variant [Arc09] for 2 time points. Nodes and edges in black exist only at the first time point, in dark gray at both time points and in light gray only at the second time point. (c) clustering of the time-expanded graph [GGWW06] over multiple time points. Here, clusters may be defined over multiple time points containing a varying set of nodes and edges at different time points.

a connected subgraph and iff a connection between two subgraphs exists a connection between the containing clusters exists. In this way, nodes and edges of a cluster share the same label. This approach is exemplified in Figure 2.3.25a and 2.3.25b showing the separation of the graph into stable, deleted and created parts.

The second approach [GGWW06] is based on linking the time points which is very similar to the idea described in Section 2.2.4.1 for the stabilization of a graph animation. Here, the graphs of all time points are linked by additional edges between the appearances of the same node. This super structure is referred to as *time expanded graph*. On this time expanded graph static graph clustering algorithms are applied resulting in a time dependent clustering as shown in Figure 2.3.25c. It is noteworthy, that these clusters are defined over multiple time points but for each time point it may contain different sets of nodes and edges. That means, if a node at a specific time point is part of a cluster it is not necessarily part of other time points covered in this cluster. This provides a high generality for capturing and describing temporal patterns. However, this generality has its price. On the one hand, the weights for the inter-time edges have to be configured correctly to gain practical results [Gla08]. And on the other hand, the clustering itself can become increasingly time consuming with each additional time step.

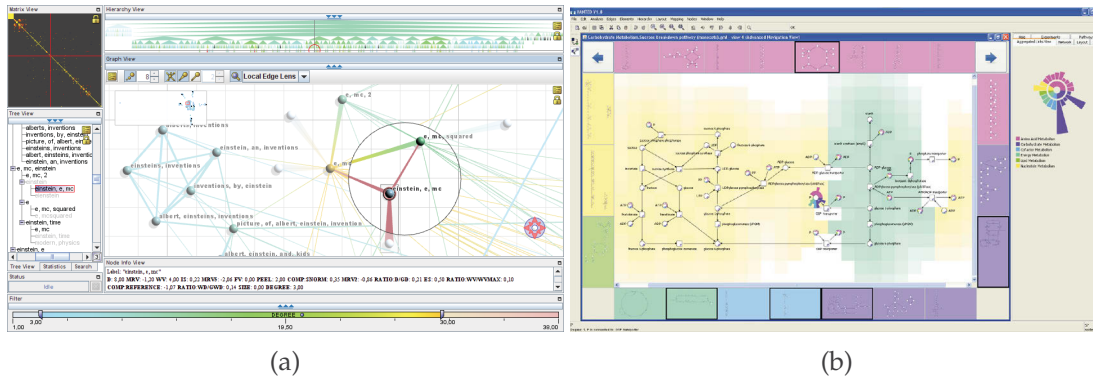


Figure 2.3.26: Overview and detail visualization of a large graph. (a) the clustering hierarchy of a clustered graph is visualized as an overview on the top while the detail view below shows a selected subset of the graph structure [ASST07]. (b) the detail view in the center shows a subgraph (in this case a metabolic pathway) and small graph icons provide an overview of connected pathways [JKKS12].

2.3.2 Selection

By selections the visualization can be restricted to subsets of nodes, edges, time points and attributes and thus the visible level of detail can be steered. According to [vLKS⁺11] interactions in general and selections in particular may be performed at any stage of the Information Visualization reference model affecting the view, the visual display or the data. Yet, interactions at any stage may also lead to changes at other stages, e.g. data filtering may affect the visual encoding such as the graph layout. In the following, common interactions for selecting subsets are briefly discussed for each stage.

View stage: These interactions affect the visible region of a visualization, for instance influencing the visible nodes and edges in a graph representation or the time interval of a complexity plot. Here generally, zooming and panning are used to enlarge and navigate to interesting parts of a visualization, for instance to look into dense parts of a graph or at peaks or valleys of a time plot. As such interactions cut off parts of the visualization they reduce the number of visual entities that are visible simultaneously. In this way, additional details may be included, for instance by automatically expanding and folding clusters in an abstracted overview depending on the zoom level. This interaction is also referred to as semantic zooming [AvH04] and represents a combination of interactions on the view and data stage.

To maintain the orientation in graphs, such detail views are often linked to an overview as shown in Figure 2.3.26a. The visualization is based on a clustered graph [TAS09]. The tree describing the clustering hierarchy (see definition of clustered graph in Section 2.1.1) is then visualized to provide an abstract overview on the graph. The detail view provides a node-link representation of a portion of the graph structure on a higher level of detail. A slightly different approach is exemplified in Figure 2.3.26b showing multiple small views that are placed around a detail view [JKKS12]. These small views provide a preview of subgraphs that are connected to the subgraph currently visualized in the detail view. In this way, they serve as navigation guides to other not visible parts of the graph.

2 Basics and Related Work

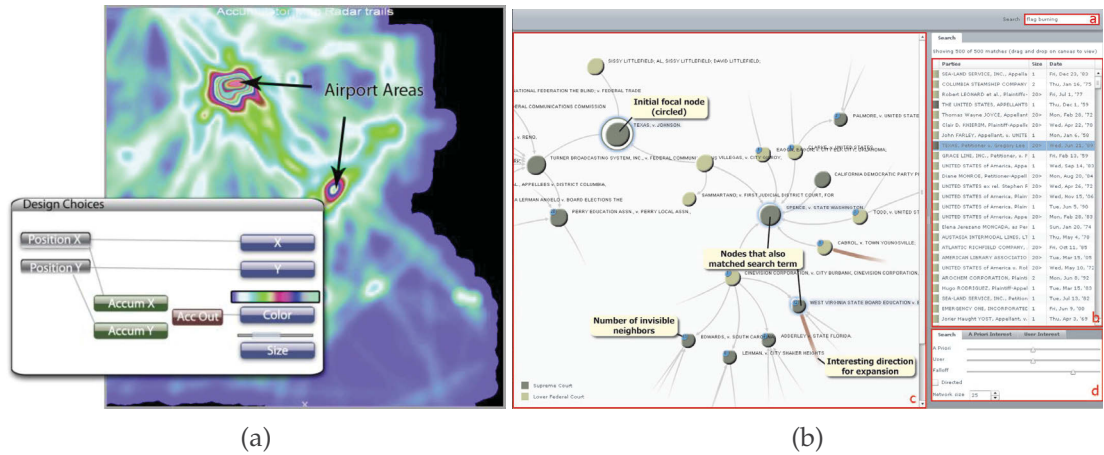


Figure 2.3.27: Different selection methods for attributes, nodes and edges. (a) shows an intuitive mapping interface to select attributes and their visual encoding for a visualization [HTC10]. (b) shows a filtered view on a large graph using a DoI function [vHP09].

In contrast to zooming which affects the whole visualization, Fisheye lenses can be used to locally increase the level of detail. Here, the visualization is graphically distorted around a focal point providing more display space for the focus while maintaining the context. Similar to zooming, such lenses may be combined with other modifications of the visualization at any stage [TFS08a], such as changing the visual encoding or embedding glyphs to represent additional attributes within the area of the lens [JDK10].

Visual display stage: Interactions on this stage affect the visual mapping of the data, especially the visual encoding of attribute values. Because of the size of graphs, often not all attributes can be visualized at the same time. Hence, visualization tools provide different methods to select attributes and specify their mapping to visual variables, such as using color or glyphs. For example, in [HTC10] an intuitive mapping interface was introduced to allow the user an explicit way to adapt the visual encoding of attributes as depicted in Figure 2.3.27a. Here, the user just needs to drag the attribute of interest to a visual variable to define the visual mapping. The current mapping is then represented by links between the attributes and the assigned visual variables.

Data stage: Interactions on this stage affect the set of nodes, edges or time points to be visualized. Such selections may be performed *directly* in an abstract overview, for instance by drawing a selection shape (often rectangles but also lasso tools, see [MJ09] for a more diverse set of selection methods) around specific nodes, edges or clusters. All following stages are consequently updated, e.g. a new graph layout is calculated for the selected subset. Navigation across a time line, for instance by clicking on a time point in a time slider, with a subsequent animation of the graph visualization is also a typical selection on this stage for dynamic graphs [BHJ09].

Contrary, such interactions may also be performed *indirectly* by selecting nodes, edges or time points according to their characteristics, such as specifying ranges of attribute values or graph complexity measures. These specifications may be performed numeri-

cally by using sliders or input fields, or graphically by using data centric views such as histograms, scatter plots or parallel coordinates similar to brushing and linking [HP14, SHQ08]. A general approach for such a specification for large graphs is based on the utilization of so called *degree-of-interest* (DoI) functions [Fur86, vHP09, RPD09]. These functions are mainly used to quantify the interestingness of nodes and edges. Based on such a specification the most interesting parts of the graph can then be filtered and visualized as exemplified in Figure 2.3.27b. Here, nodes are filtered according to the similarity of their label to a search key and a predefined attribute (referred to as a priori interest). In contrast to the previous selection methods, these indirect specifications do not require an abstract visualization of the graph that may be time consuming to calculate. DoI functions will be described in more detail in Section 4.3 introducing a flexible DoI function definition for large dynamic graphs.

2.4 Summary and Challenges

In this chapter, the essential terms and definitions for the visualization of graphs were introduced. A graph visualization has to cope with both the different aspects of these graphs and their size. The most important aspects as described in Section 2.1 are their structure, their associated attributes, their spatial and their temporal context. Based on these aspects the size depends on the one hand on the number of nodes and edges but on the other hand also on the number of associated attributes and time points.

For handling the different aspects, a wide variety of visualization techniques have been invented as presented in Section 2.2. The majority of these techniques often only considers a subset of these aspects at the same time such as showing the structure together with some associated attributes, in its spatial context or over time. And even for the same set of aspects a multitude of different visualizations exists each providing a different view and focus on the data. Hence, depending on the task at hand different techniques may be suitable. Yet, there is no approach that is able to tackle all aspects simultaneously.

This is partially due to the size of these graphs and different reduction strategies have been proposed for both the structural and the temporal aspect. However, as discussed in Section 2.3 most of these strategies target only the structural complexity by reducing the number of nodes and edges for instance by clustering or performing selections. For reducing the temporal aspect, only very simple aggregation methods are applied for the visualization of large dynamic graphs. In this way, the temporal complexity has not been sufficiently addressed so far. And similar to the different aspects, depending on the utilized reduction method different foci or goals may be supported.

Based on these observations three important challenges can be devised as they were outlined in Section 1.2:

Diversity concerning the different aspects – *structure, attributes, spatial and temporal context* – of the graphs.

Scalability regarding the size – *number of nodes and edges* as well as the *amount of time steps* and *number of attributes* – of the graphs.

Flexibility regarding the different foci on the graphs as a result of the interplay of diversity and scalability.

Each of these challenges addresses a specific problem encountered during the visual analysis of dynamic graphs.

Diversity concerns the existence of suitable visualization techniques for a given combination of aspects to fulfill for instance a certain task at hand. This demands on the one hand to provide visualizations for any combinations of aspects that have not been considered sufficiently yet. For instance, in case of graphs featuring only vague spatial references, visualizations may be developed that take the irregular shapes of geographic maps into account during the calculation of the graph layout. In case of dynamic graphs, visualizations are often only focusing on the structural aspect providing only a restricted view on the given data. Hence, alternative visualizations with a focus on the temporal aspect may yield further insights into the graphs' dynamics.

Yet, on the other hand, it is also interesting to determine how many aspects may be incorporated into a single visualization and with which limitations. In this sense, identifying similarities between the different visualizations may help on the one hand to handle the variety of different visualizations and thus to more easily provide a large number of them. But on the other hand, extracting similar design decisions may also support the adaptation of a given visualization regarding different aspects.

Scalability concerns the processing or reduction of graphs so that existing visualization techniques may be applied for their representation. Similar to the diversity, most existing strategies for both abstracting the graph or selecting subsets are merely focusing on the structural aspect. Hence, new strategies are necessary that also take the temporal aspect into account. For instance, abstraction strategies may be developed to summarize the structure according to the temporal development of the nodes in contrast to abstracting each time step individually. Alternatively, the temporal aspect may also be abstracted according to the structure yet not by simply aggregating subsequent time points which may not be similar at all.

Flexibility concerns the focus of the analysis depending on the current goal or task. While each visualization provides a specific and often restricted view on the graph as described above, the analysis goal may shift as new information is revealed. Thus the user may need to switch between different visualizations. And similar to maintaining the mental map when animating a graph over time, it is as important to support the user when switching visualizations. Hence, it may be interesting to investigate new ways for integrating multiple visualization in a seamless manner in contrast to just using multiple linked but also separated views.

In this sense, determining how many and which views to use is an important problem to solve. Especially for unknown graphs, where nothing is known in before it may become difficult to choose a suitable initial visualization. Here, an abstract overview just informing the user of all aspects of a given graph may be a good basis to decide which

views to use. In the same way, such an overview may also serve as a mechanism to synchronize the different views and to support the mental map of the user.

Each of these three challenges is addressed individually in the following three chapters providing first steps to close the aforementioned gaps. Here, Chapter 3 is primary focusing on novel approaches for handling the diversity of graphs by introducing a family of point-based tree layouts and a design space for implicit tree visualizations. Novel reduction techniques for a scalable visualization of dynamic graphs are introduced in Chapter 4 introducing new clustering strategies to abstract the structure as well as the temporal aspect and a flexible selection method based on a more general DoI function definition. Finally, the combination of general visualization strategies and their requirements for a flexible analysis is discussed in Chapter 5 by introducing a new way for a seamless integration of graph visualizations and an abstract overview for synchronizing different views.

3 Novel Techniques to Visualize Graphs

3.1 Introduction

As described in the previous section, for conveying the structure, the attributes as well as the spatial and temporal context of graphs often different visualizations are used. In this way, the user has to be familiarized with multiple visualization techniques to be able to analyze this diversity of graphs. This problem becomes even more severe when the size of these graphs is increasing. In this chapter, two approaches are introduced each providing a different strategy to resolve this problem with a single visualization paradigm:

1. A family of space-filling layouts for large graphs based on representing nodes by the smallest visual primitive – points – and assigning them to fixed predefined positions. In this way, this fixed layout poses restrictions as well as provides benefits for its applicability to visualize the different graph aspects.
2. A design space generalizing implicit visualization techniques. This generalization provides a much higher flexibility to adapt the visualization to the different aspects than any single visualization.

However, such a compact space-filling layout as well as such a design space can only be created by taking additional information according to the visualized graph class into account. In this case, both approaches are strongly utilizing the hierarchical structure of tree-like graphs and are thus only applicable to the class of trees. In the following, both approaches are described and discussed in more detail.

3.2 A Point-Based Layout for Large Hierarchies

The visualization of large hierarchies is an important endeavor in many fields. One example is the DMOZ¹ hierarchy, a manually maintained classification of the internet, which contains 778'854 categories. Many visualization techniques are therefore trying to efficiently use the available display space for representing the structure of the hierarchy. These space-efficient visualization can be, for instance, explicit node-link layouts such as the RINGS [TM02] or the Space-Optimized Tree [NH03] layout, or implicit techniques such as Treemaps or Sunbursts.

In either case, the layouts have to make a compromise between showing a high amount of nodes and the readability of the structure. For example, the RINGS layout recursively

¹Snapshot taken from <http://www.dmoz.org/> at 02/06/2013.

puts the children of a node on multiple rings around it and assigns them non-overlapping circular region into which they can lay out their children. As this leads to empty spaces between the subgraphs, the structure can be well perceived but on the cost of a not so efficient space usage. Contrary, the Space-Optimized Tree layout is based on a polygonal subdivision of the display space dividing it into unregularly shaped regions for each subgraph. In this way, this layout can utilize the available space more efficiently, yet on the cost of the readability of the structure. This compromise is similar for Treemaps having a better space utilization and Sunbursts better reflecting the overall structure.

The new *point-based layout* introduced in this section² tries to accomplish both, a good readability of the structure while more efficiently using the available display space. On the one hand, the layout therefore uses points, the smallest visual primitive, for the node representation allowing an overlap free placement of a high number of nodes. On the other hand, it assigns nodes to fixed predefined positions that capture each node's location in the hierarchy while also allowing a space-filling placement. This fixed layout especially poses restrictions to the visualization of attributes as it provides very little display space for encoding additional information for each node. Yet, its fixed and stable nature also provides advantages to convey the structure as well as the spatial and temporal context.

In the following, first the basics of the point-based layout concerning the representation of nodes and edges are introduced. Based on this basic algorithm its utilization for the different aspects – structure, attributes, spatial and temporal relation – is discussed. Therefore, a family of adapted variants of the basic layout is described to cope with hierarchies different in width and depth. Based on this compact representation of the structure, limitations and necessary extensions for communicating associated attributes are discussed. The layout then poses as the basis for a direct embedding of hierarchies into their spatial context. It is then used to showcase the problems confronted concerning the visualization of large structures that are varying over time. Finally, a use case concerning health care data brings all these aspect together into a single context and exemplifies the utilization of this new layout.

3.2.1 Basic Point-Based Layout

The point-based layout is inspired by the point-based computer graphics. Here, points are used to more efficiently represent large scenes by replacing the huge amount of small triangles. They therefore apply recursive sampling patterns to render surfaces in a space-filling manner. One such pattern is the $\sqrt{5}$ sampling introduced in [GP07] that recursively replaces one sample by 5 more narrow positioned samples.

Node representation: Based on this pattern the novel point-based layout allocates a unique pixel for each hierarchy node. It can be described as follows:

²Parts of this section have been published as “Point-Based Visualization for Large Hierarchies” 2011 in the IEEE Transactions on Visualization and Computer Graphics [SHS11b] and as “Visualization of Attributed Hierarchical Structures in a Spatio-Temporal Context” 2010 in the International Journal of Geographical Information Science [HTSS10]. Especially the latter publication has focused on extensions for space and time concerning the Sections 3.2.4 to 3.2.6.

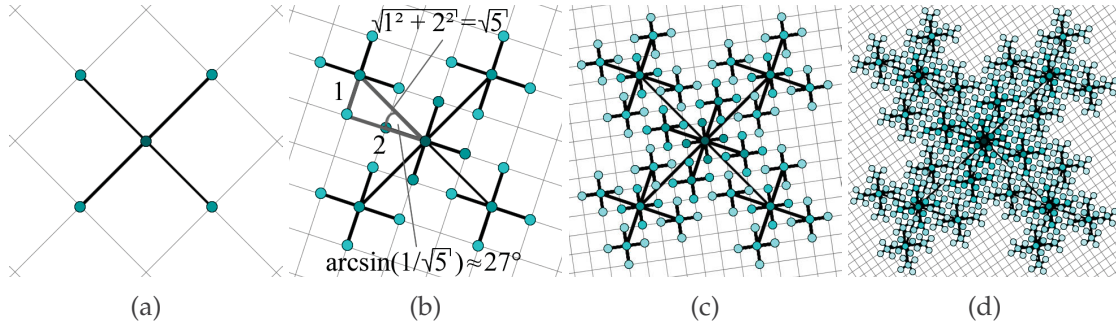


Figure 3.2.1: Computing a layout in a point-based manner.

- (a) The root node is placed in the center and its first 4 children are arranged around it forming a regular grid (see Figure 3.2.1a).
- (b) The positions of the next 4 children of the root are obtained by rotating the grid by $\sim 27^\circ$ and scaling it by a factor of $1/\sqrt{5}$. The first 4 children of each of the previously laid out nodes are positioned with the very same rotation and scaling scheme (see Figure 3.2.1b).
- (c) This procedure is repeated to layout the next 4 children of the root node, the next 4 children of the nodes laid out in step (a), and the first 4 children of the nodes that have been added in step (b) (see Figure 3.2.1c).
- (d) The procedure continues until all children have been associated with a unique layout position (see Figure 3.2.1d).

The two factors for rotation and scaling describe the relation between the different regular grids shown in Figure 3.2.1 and guarantee a space filling placement. The space filling property has been proven in multiple ways by describing the layout according to four different established space-filling approaches such as the embedding or fractal approach [Sch10]. The rotation and scaling scheme generates a fixed layout that can be pre-computed independently. To visualize a concrete hierarchy, the only necessary step is to assign nodes to the pre-computed node positions. This is done in a well-defined way by assigning the nodes according to the spiral shown in Figure 3.2.2a. For a higher space utilization the nodes are ordered according to their subtree size providing the most space to the largest subtrees first. Due to precomputation and mapping of nodes to unique pixels, larger hierarchies can be visualized very compactly and effectively.

In case the number of children is not divisible by 4, positions are left unoccupied. In this way, both, occupied as well as unoccupied positions communicate information according to which nodes of the hierarchy are existing or missing. This additional information is an important advantage to other visualization techniques which can only communicate the existing nodes as they are adapting the layout to minimize the empty space.

Furthermore, the rather fixed nature of the layout enables a comparison not only between different hierarchies but also between subgraphs within the same hierarchy. This high comparability results from the fractal shape that is filled by the layout for the complete hierarchy but also for each subgraph as can be seen in Figure 3.2.2b. These shapes

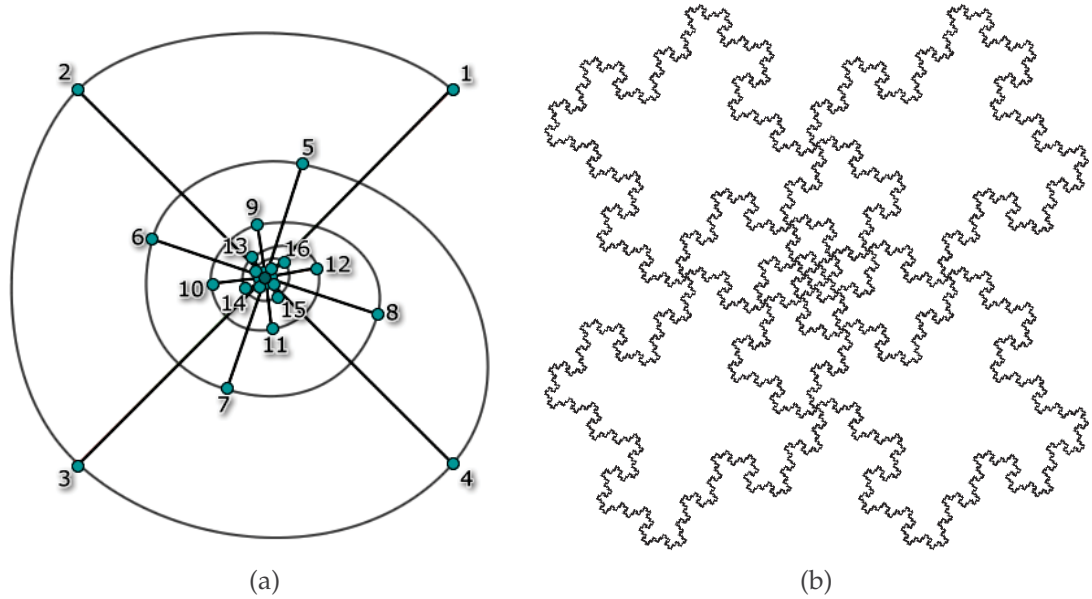


Figure 3.2.2: Assigning predefined positions (a) and corresponding spaces (b) to the nodes of the hierarchy according to a spiral.

just differ in the applied rotation and scaling making the visualizations of subgraphs comparable.

Lightness adaptation for an increased comparability: One problem affecting this comparability however results from the different size of these fractal shapes. As less nodes are necessary to completely fill smaller shapes, subtrees visualized within these shapes may look as dense as subtrees rendered in larger shapes while containing fewer nodes. This problem is exemplified in Figure 3.2.3a. Here, the areas filled by the two highlighted subtrees (orange and red boundaries) look equally filled leading to the assumption that both subtrees may also be equal. Yet, visualizing solely one subtree at a time at the same display resolution as depicted in Figures 3.2.3b and 3.2.3c shows that the orange subtree appears darker than the red one thus containing more nodes. This separated visualization reveals that the subtrees are very different considering the number of nodes.

One solution for this problem is to adapt the lightness of all nodes so that the overall lightness within a subtree better reflects its real structure. The main idea of this approach is to lay out each subtree in the largest space available and then scaling it down. During this scaling, possibly empty pixels (white) and filled pixels are aggregated resulting in brighter colors. Smaller subtrees will then appear brighter whereas larger and denser subtrees will keep their original color allowing their differentiation. In this way, the visualization of subtrees will appear similar only if they are really similar.

The adapted lightness of each node can be calculated by a single bottom up run of an adapted variant of the basic layout algorithm. During this bottom up run, it iteratively gathers the number of pixels that would be used by the subtree when laid out in the largest space. The factor for adapting the lightness is then the fraction of the gathered number of filled pixels by the maximum number of pixels. The result of this adaptation is shown in Figure 3.2.3d, now clearly reflecting the difference between the selected sub-

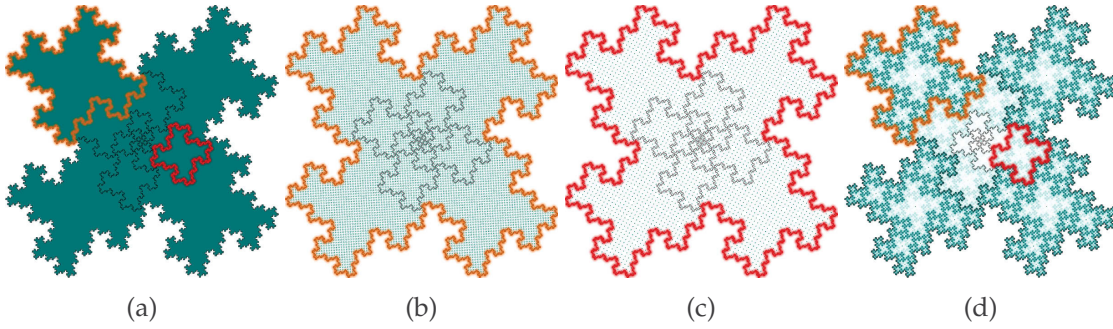


Figure 3.2.3: Lightness adaptation to better reflect the visualized structure. (a) a normal visualization of a hierarchy showing an equal utilization of the display space. (b) and (c) reveal that the two highlighted subtrees differ in their size yet still looking similar in the original visualization. (d) By adapting the lightness of each subtree according to its structure this ambiguity can be solved increasing the comparability.

trees in their lightness. In this way, the lightness of a node is also a direct measurement for the space utilization of its subtree according to the layout.

Edge representation: The point-based layout exhibits characteristics from both, explicit techniques by using small primitives such as points leaving space between nodes and implicit techniques by encoding the relation between nodes into their positions. In the same manner, it is feasible to represent the edges and thus the parent-child relation in an explicit way by rendering them as lines or in an implicit way by visualizing the space distribution.

Because of the compact representation of the hierarchy rendering all lines as exemplified in Figure 3.2.4a may result in much visual clutter and thus hides information. However, edges may only be needed in sparse regions in which the relation between the nodes becomes unclear. Thus, by rendering edges only in sparse regions the visual clutter can be minimized. Here, to determine these sparse regions and thus which edges shall be drawn the bottom up approach described for the lightness adaptation can also be applied. Beginning from the root, edges are drawn only to those nodes whose lightness falls below a user defined threshold. An example for this adapted edge rendering is given in Figure 3.2.4b.

The visual clutter may be further reduced by neglecting all edges and resorting to the idea of implicit techniques. Here, borders are rendered to visually separate the subtrees partially transforming the point-based layout into an implicit visualization. These borders are depicted in Figures 3.2.4c for the first level only. Similar to the rendering of edges, not for all subtrees borders may be necessary. They are especially useful in dense regions to support the visual separation of the different subtrees. Again, the bottom up approach for the lightness adaptation can be applied but now those nodes are chosen whose lightness excels a user defined threshold. This approach is depicted in Figure 3.2.4d.

Both ways describe a compromise between using the display space mainly for the nodes and possible overplotting to enhance the representation of the relations between them. Yet, overplotting may already been caused by the node placement as which each recursion step of the layout the distance between parent and child is reduced.

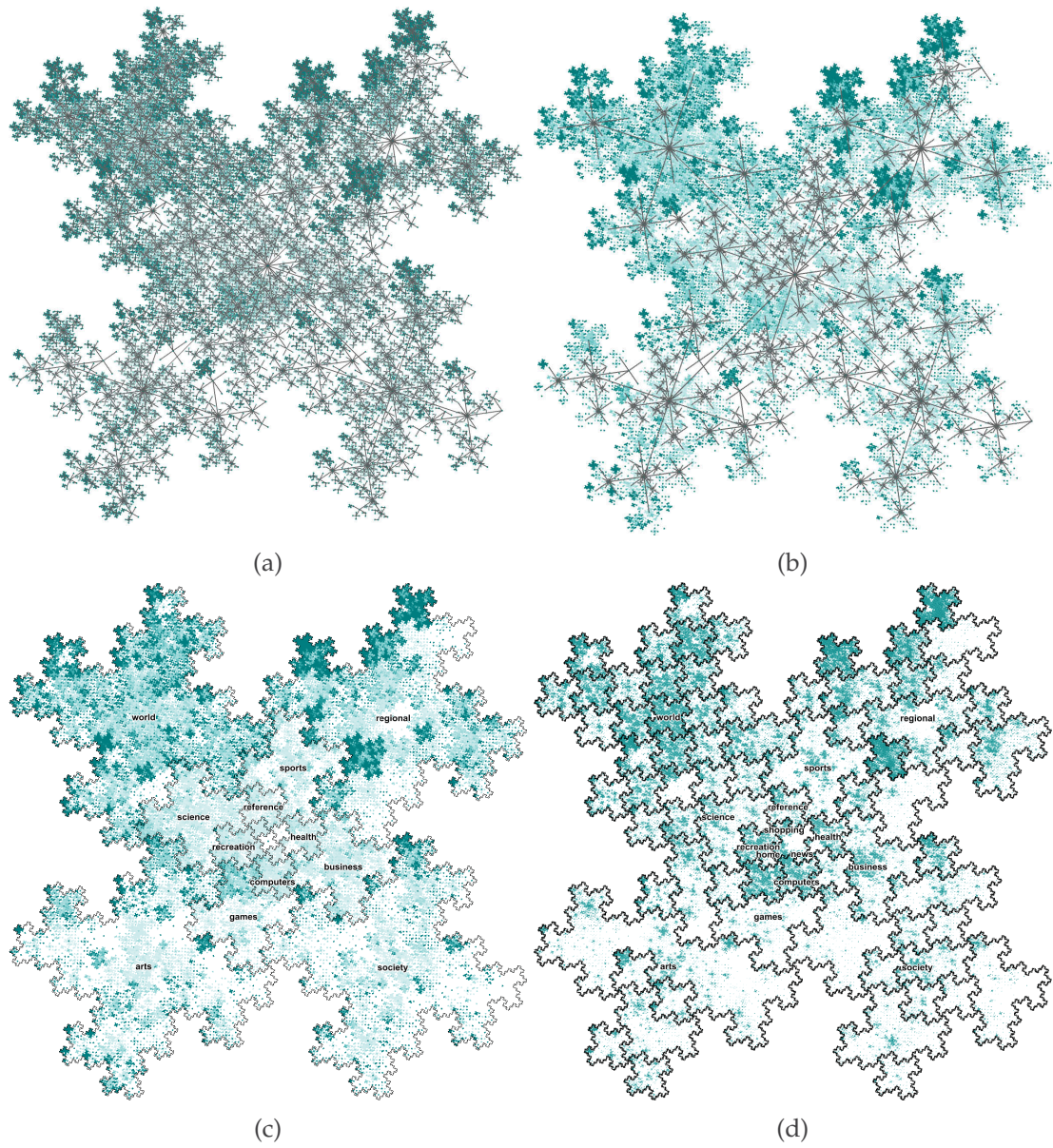


Figure 3.2.4: Different representations of the parent-child relations. Explicit representation showing (a) all edges between nodes and (b) adaptively determined edges connecting nodes only in space regions. Implicit representation by including subtree borders for (c) the first level and (d) dense subtrees.

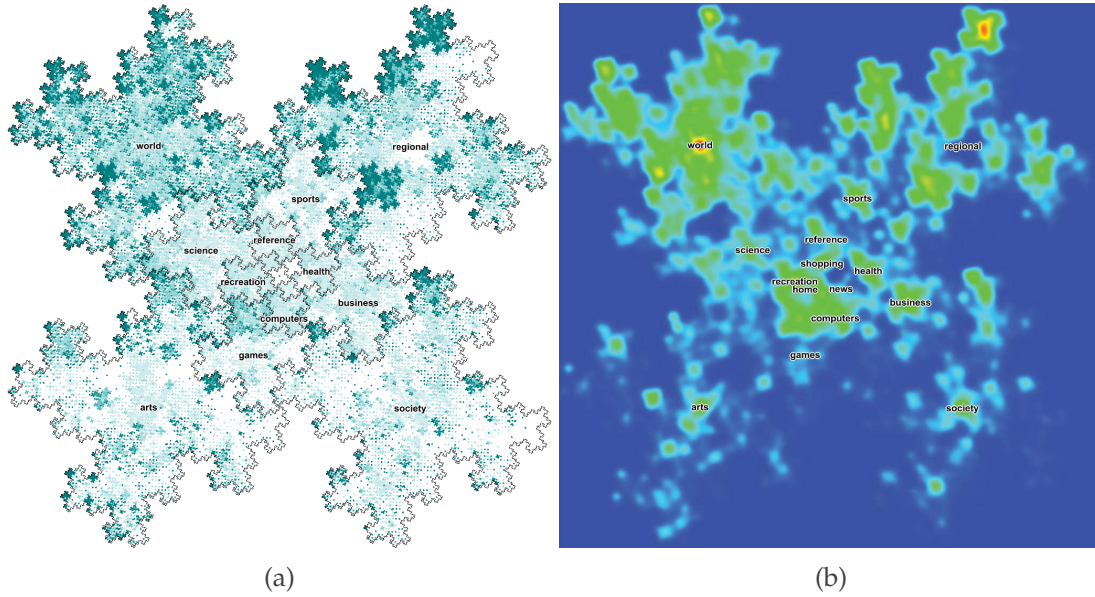


Figure 3.2.5: A GraphSplatting visualization communicating regions with hidden information. (a) a node-link representation and (b) its corresponding GraphSplatting visualization. Red regions point to dense parts of the hierarchy that were hidden otherwise.

Alternative visualization to communicate hidden information: For dense graphs it has proven useful to provide visual abstractions as discussed in Section 2.3.1.1 to provide an overview of the node distribution and thus to point out to regions of high overplotting. One example is the GraphSplatting technique [vLdL03] which calculates and visualizes a density field. In this field, red regions identify very dense parts of the network that may be interesting to analyze further. For this reason, besides the node-link representation discussed so far a GraphSplatting view is provided as an alternative visualization. As can be seen from Figure 3.2.5 a lot of overplotting appears in the center of the world subtree (upper left part of Figure 3.2.5b) and in one of the upper subtrees of the regional subtree (upper right part of Figure 3.2.5b). In this case, the overplotting is mainly caused by a too wide subtree (the world subtree) and a too deep subtree (the regional subtree).

Interactions: All enhancement, the adapted lightness, the representation of the parent-child relation and the graph splats, improve the comparison and readability of the hierarchy. Yet, to be able to analyze specific details in the hierarchy such as regions of high overplotting additional methods are necessary. On the one hand, common interaction techniques are provided, such as filtering and zooming. Filter mechanisms allow a fast localization of nodes by specifying desired properties such as a minimum number of children or an interval of associated attribute values. Zooming mechanisms enable the user to cope with the unequal distribution of display space to nodes and their subtrees, and to access and analyze dense regions.

On the other hand, the layout itself facilitates an additional interaction paradigm: By changing the order of nodes on a level, subtrees may be assigned to larger shapes for their visualization. Besides showing specific subtrees in focus and thus providing more

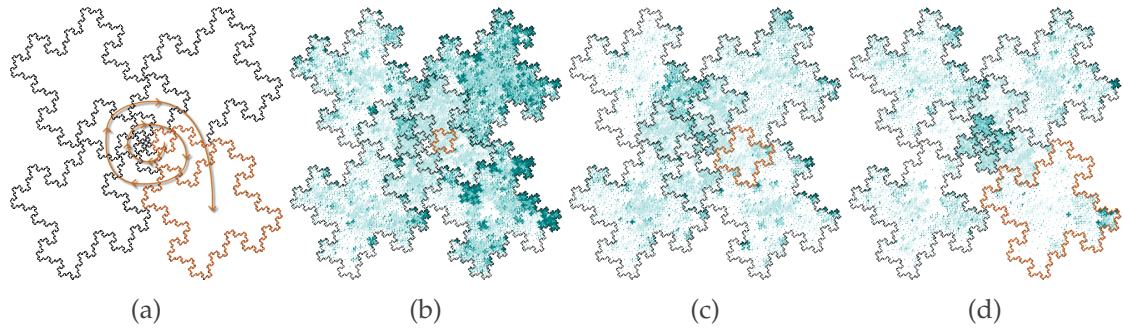


Figure 3.2.6: Rotation based interaction: (a) a subtree of interest can be analyzed in more detail by rotating it to an outer position providing it more display space. (b-d) showing the result for rotating the health subtree of the DMOZ hierarchy.

details, this approach has the advantage of still showing the other subtrees allowing their comparison and thus maintaining the context. This change of the order can be realized by a rotation of the nodes as shown in Figure 3.2.6. In this way, the user is able to scroll through the subtrees and thus to analyze each subtree in more detail.

For a more numerical evaluation of the layout concerning for instance space utilization and overplotting, as well as a comparison with other layouts such as RINGS based on a user study, the interested reader is referred to [Sch10]. This basic point-based layout actually describes the foundation of a whole family of different hierarchy layouts as introduced next.

3.2.2 A Generalization to Accommodate the Tree Structure

While a tree itself describes a very specific graph class, its representatives may still be very different considering their structure. These trees can be differentiated according to their widths – the maximum number of children found in the tree – and their depths – the maximum distance of a node to the root. Bases on these two features, width and depth, trees can be categorized into:

- narrow trees having a low overall width,
- wide trees having a high overall width,
- flat trees having a low depth,
- deep trees having a high depth, and
- balanced trees showing none of these characteristics.

Here, wide deep trees are the most complex kind of trees that cannot be visualized in full detail. Most visualizations are either specialized to deep or wide trees.

For instance, Sunbursts on the one hand perfectly reflect the depth of a tree by mapping the levels on specific layers. Hence, it scales well to any depth as long as the layers can be differentiated within a given display resolution. Yet, analyzing the width of the tree is difficult because its visualization may result in many very small angles.

The RINGS layout on the other hand supports the analysis of wide trees as it tries to distribute the visual space evenly to all subtrees. Thus, it scales well to any width. Yet, for wide trees, each subtree will be assigned only a very small space limiting the number of levels that can be visualized simultaneously. In this way, the visible depth of the tree is directly affected by its width. In the worst case, the circular area assigned to the root node will be completely filled and each child node only receives some pixels for their subtrees resulting in a single visible level.

As a result of the fixed nature of the point-based layout, it does not adapt itself to reflect either one of those features in more detail as existing layouts do. Instead, it guarantees a visualization of the hierarchy up to a specific depth and width independently of the tree characteristics. In this way, it provides a compromise between visualizing depth and width. For a resolution of 1000×1000 pixels for instance up to 8 depth levels and 32 children resulting in a total number of 390'625 nodes can be shown simultaneously without overlap. Every further level as well as each additional child however will be mapped into a subpixel region and thus will not be visible. Yet, as for the other layouts these additional information may only be explored by more interactive methods.

The balance between the number of levels and children that can be shown simultaneously is a direct result of the fixed nature of the point-based layout. Actually, the point-based layout algorithm in general does not only define a single layout but captures a whole family of layouts with similar characteristics that differ only in some parameters. Depending on these parameters the point-based layout differently supports the visualization of deep and wide trees. Hence, by selecting a specific parameter set, in the following referred to as *preset*, the balance and thus the focus on a given tree can be steered. While the preset itself generally only defines possible node positions the question how the nodes are assigned to these positions is basically still left open. Thus, this assignment provides another degree of freedom for steering the view on the tree. Finally, as the resulting visualizations are based on the same algorithm for calculating the layout they may also be combined to create new visualizations. These three points – selecting a preset, assigning nodes to the predefined positions, and combining different presets – are discussed in detail below.

3.2.2.1 Selection of presets

The point-based layout generally consists of 3 steps that are performed recursively:

- Place an arbitrary number n of nodes per step,
- perform a rotation around a specific angle α , and
- scale down the distance according to a distinct factor d .

Hence, the final visualization depends on three parameters: the number of nodes n , the rotation angle α and the distance scaling factor d .

Based on these parameters, the maximum number of recursions k until nodes are laid out in a subpixel region can be determined³. A recursion occurs every time after placing n children as well as for each depth level. Hence, the maximum number of children to be laid out for the root is $k \times n$ while the maximum visible depth level is k . That means, that the balance between visible depth and width is mainly influenced by the number of nodes n . If one would increase the number of nodes per step one could steer the balance to favor the visualization of wide trees. Whereas a reduction of this number may steer the balance to favor deep trees.

However, finding an appropriate positioning pattern for an arbitrary number of nodes may be tricky. A basis for finding a suitable positioning is the resemblance of the area filled by the point-based layout to a fractal shape. Actually, the fractal defined by the basic point-based layout is known in the literature as the *Quadratic Snowflake*. Hence, a first step to derive a new point-based layout preset for a specific number of nodes n is to use another existing fractal and utilize its parameters. On this basis, additional presets can then be derived by extending those existing fractals altering any of its parameters where applicable. Finally, to select a good preset and thus a good layout for a given hierarchy all presets can be evaluated according to their space utilization.

As an exhaustive search for all possible presets is beyond the scope of this thesis the following discussion presents some general strategies and examples.

Utilizing existing fractals as presets: Besides the Quadratic Snowflake the literature bears a multitude of different fractals⁴. Yet, from these fractals only those may be used which share a similar creation algorithm consisting of the aforementioned three steps: place an arbitrary number of points, perform a rotation and scaling, and then starting all over again.

A very generic fractal is the n -flake which places n points on a circle around the center. Each point is then the center of another yet smaller circle which is used to perform the same procedure recursively. Hence, the scaling factor of the preset corresponds to the relation between the radii of the larger and the smaller circles. Whereas, the rotation angle can be chosen to minimize edge crossings. In this way, it may allow a definition of a preset for any number of nodes. However, as it is very similar to a Balloon Drawing [LY07] it is generally not a space-filling layout and leaves much space unused.

The reason for this low space utilization lies in the use of regular polygons with n edges. A space-filling tiling however is only possible with triangles, squares, and hexagons or by at least mixing multiple shapes. Hence, there are special cases for n -flakes that are truly space-filling such as the *Sierpinski Triangle* (left side of Figure 3.2.7) with 3 nodes per step, the *Quadratic Snowflake* (right side of Figure 3.2.7) with 4 nodes per step, and a special version of the *Hexaflake* also referred to as the *Gosper Flowsnake* (center of Figure 3.2.8) with 6 nodes per step. The *Sierpinski Carpet* (center of Figure 3.2.7) may also be considered as special case for an 8-flake with the only difference that nodes are not distributed on a circle but on a square. These figures all show the different parameters,

³Neglecting the influence of the rotation, the maximum recursion may be roughly approximated by solving the equation: $(1/d)^k = \text{resolution}$.

⁴See for instance http://en.wikipedia.org/wiki/List_of_fractals_by_Hausdorff_dimension for a listing of potential preset candidates.

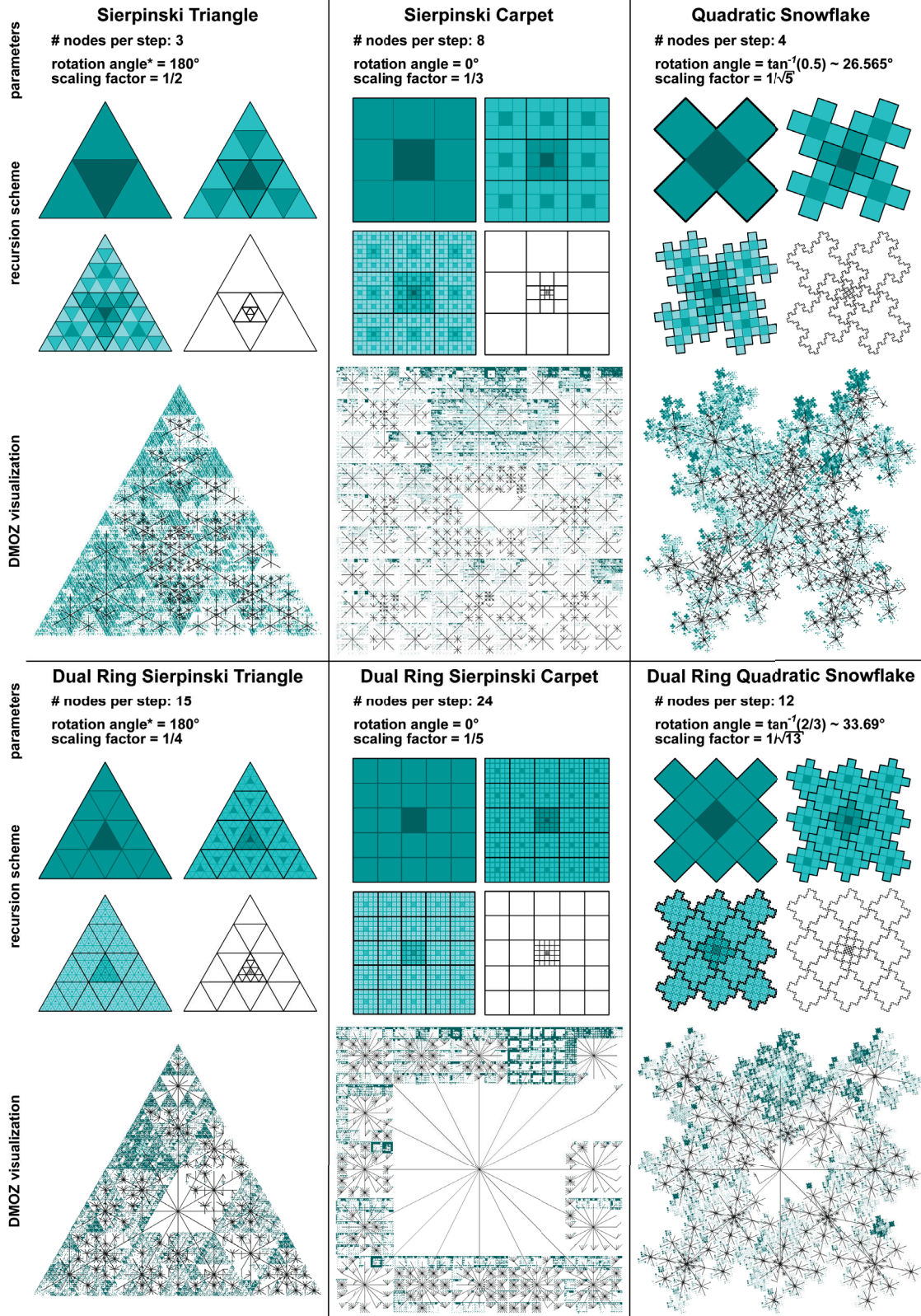


Figure 3.2.7: Three point-based variations based on three different fractals placing 3 (left), 8 (middle) and 4 (right) nodes per step. The lower row shows their adaptation to wider hierarchies by introducing a second ring of node positions around the original ones. (*) during the calculation of the Sierpinski Triangle layouts the rotation is not applied for each child node.

the basic recursion scheme for creating the fractal and the layout as well as an exemplary visualization of the DMOZ hierarchy. Especially from the recursion scheme, the space-filling property can be easily determined. If the used tiling is space-filling it is most likely that the resulting layout will also be space filling (compare for instance the two variants of the Hexaflake in the center of Figure 3.2.8).

It has to be noted, that the fractals may follow a similar algorithm but also slightly differ which is already visible from these examples. In case of the *Sierpinski curve* (lower left of Figure 3.2.8), the number of nodes placed per step differs for the first level (4 nodes) and all following levels (3 nodes). In its restricted case (upper left of Figure 3.2.8), the number of nodes is strictly limited to a maximum of 4 children per node. A rotation may not be needed for every fractal (Sierpinski Carpet or Hexaflake) or may not be applied for every recursion (in case of the Sierpinski Triangle a rotation occurs only after placing 3 nodes). Hence, when designing new layout presets one may not only change the different parameters but also their use during the algorithm.

Altering presets: Having found some presets, additional presets can be derived from them by altering its parameters. Generally, each parameter may be changed, yet often with different results. Changing the number of nodes placed per step steers the balance between showing width and depth as discussed above. Whereas, changes to the rotation and especially to the scaling factors mainly affect if it will be a space filling technique and its overall space-utilization.

Steering the balance between width and depth: For increasing the number of nodes to be placed per step while maintaining its general properties each fractal can be extended by adding multiple additional rings around the original positions. If a fractal is space-filling, adding an additional ring will most likely also result in a new space-filling fractal. The new positions can be easily determined from the adapted recursion scheme. This addition is demonstrated in the lower row of Figure 3.2.7 for the Sierpinski Triangle, the *Sierpinski Carpet* (8 nodes per step) and the Quadratic Snowflake resulting in layouts that may be better suited for wider hierarchies. As can be seen from these examples, adding additional rings also influences the rotation and scaling factors as for instance with more nodes to be placed simultaneously less space is available for each node.

Achieving the space-filling property: Based on the scaling factor d , the *fractal* or *Hausdorff dimension* of a fractal can be calculated by

$$fractal\ dimension = \frac{\log(number\ of\ points)}{\log(d^{-1})} = \frac{\log(n + 1)}{\log(d^{-1})}$$

This dimension describes the space a fractal is filling. If it is 2 in a two dimensional space it means that the fractal is space-filling. In this way, the space-filling property of a layout can be determined by calculating the fractal dimension of the corresponding fractal. This correspondence also allows to calculate a scaling factor for a given number of nodes and a desired fractal by solving that equation.

For instance, except for the Hexaflake (upper center of Figure 3.2.8) all fractals and thus the corresponding layouts in Figures 3.2.7 and 3.2.8 have a fractal dimension of 2 and are thus truly space-filling. In general, n -flakes have a fractal dimension lower than 2. The dimension of the Hexaflake is $\log(7) / \log(3) \approx 1.77$. For creating a space-filling

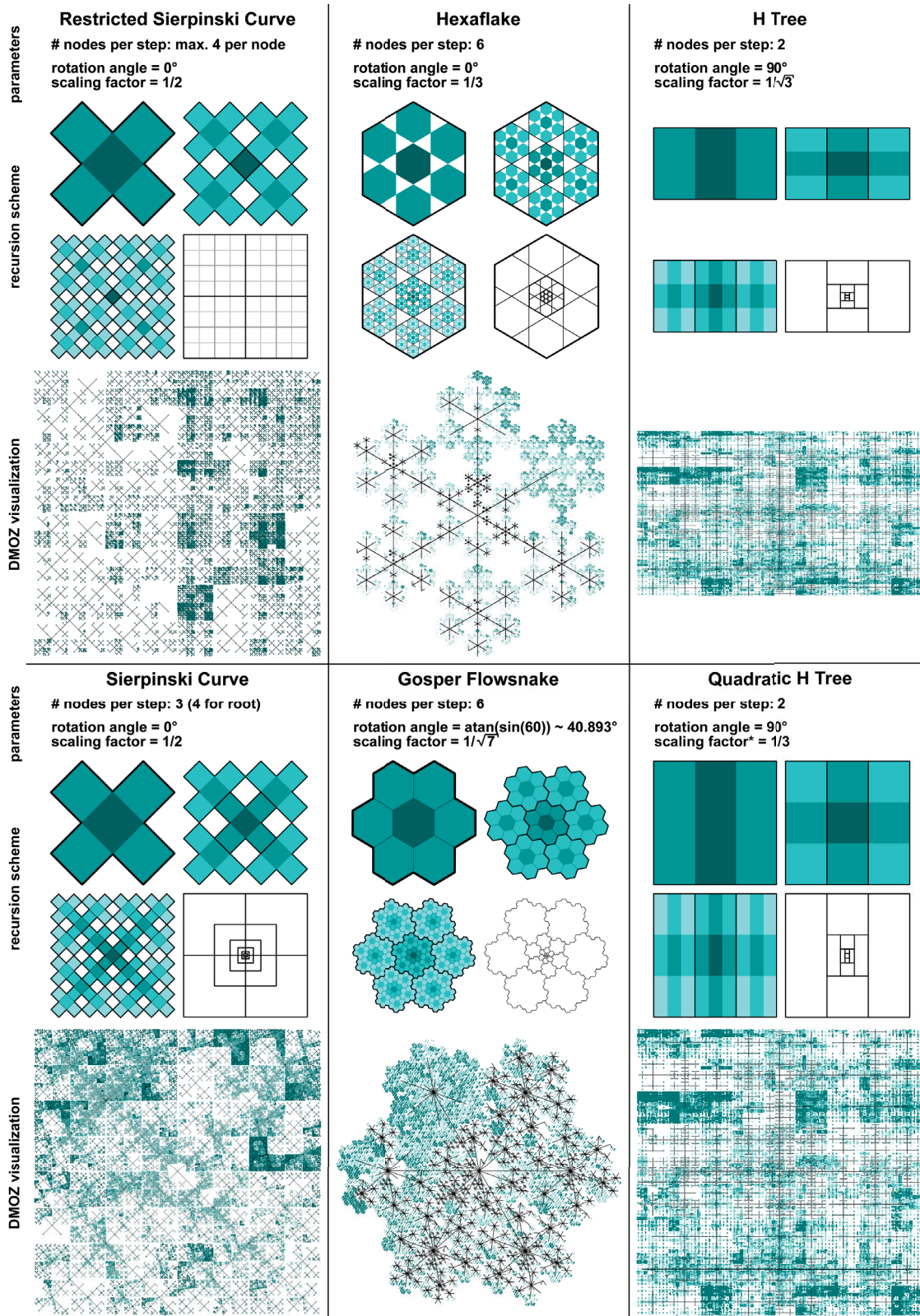


Figure 3.2.8: Three point-based variations using similar fractals with different results. Left: both layouts assign the same coordinates but are suited for different kinds of trees: trees with a maximum width of 4 (upper), general trees (lower). Middle: both layouts place 6 children per step, yet different factors transform the second variant into a space-filling layout. Right: both layouts differ only in the scaling (*) allowing the second variant to fill a quadratic space.

variant of the Hexaflake, a suitable scaling factor can be calculated by converting the fractal dimension resulting in:

$$\log(d) = \frac{\log(n+1)}{\text{fractal dimension}} = \frac{\log(7)}{2}$$

In this way, a space-filling variant of the Hexaflake must have a scaling factor of $\sqrt{7}$. The only problem left is finding a suitable rotation factor that allows an overlap free tiling of the fractal. In the lower center of Figure 3.2.8 a space-filling variant based on the Gosper Flowsnake is shown featuring a rotation factor of $\sim 40.893^\circ$. While this gives a general guideline for determining the scaling factor finding a suitable rotation factor may not be possible in every case. This problem is mostly a cause of a space-filling tiling only working with triangles, squares and hexagons.

Increasing the space utilization: Each fractal has a specific shape that may be filled completely. Yet, if the fractal shape is not matching the available drawing area, space is left unused. This is a problem especially for the irregular layouts such as the Quadratic Snowflake or the Gosper Flowsnake. Hence, besides determining factors to create a space-filling layout, they may be altered for an overall increased space utilization.

The main reason for the irregular shape of most fractals is the rotation. Hence, by neglecting the rotation as it is the case for the Sierpinski Carpet most often a quadratic space may be completely used. Yet, not all fractals have a rotation free counterpart. A working example is the Triple Ring Quadratic Snowflake (a Quadratic Snowflake with two additional rings) placing 24 nodes per step as does the Dual Ring Sierpinski Carpet providing two alternatives. Because of its axis parallel layout the Sierpinski Carpet shows a high regularity leading to much overlap when rendering edges on the one hand and in unwanted patterns that may not even be contained in the hierarchy on the other hand. The slight distortion based on the rotation of the Quadratic Snowflake may reduce both of these concerns. Yet, the rotation also complicates the estimation of boundaries between subtrees which is much easier in the more regular Sierpinski Carpet. Here, a rotation may be chosen to steer the tradeoff between space utilization and readability.

Contrary, the scaling factor is generally fixed according to the fractal dimension. However, it only captures the “average” scaling to be performed per step. Hence, an increase of the space utilization may be reached by distributing the scaling across multiple steps. On the right side of Figure 3.2.8 an example for this alteration of the scaling is shown. The standard *H Tree* fills a rectangular space and fulfills a scaling of $1/\sqrt{3}$ every step. By delaying the scaling to every second step and decreasing it to $1/\sqrt{3} * 1/\sqrt{3} = 1/3$ the *H Tree* is altered to fill a quadratic space. In average, the scaling performed per step is still the same. As a side effect of this alteration, subtrees of different levels are more difficult to compare as the shapes filled by the subtrees switch between squares and rectangles each step. In this case, based on the scaling the tradeoff between space utilization and comparability can be steered.

Selecting an appropriate preset: It has to be noted, that for any n nodes to be placed multiple fractals may exist as was shown for the example of the Triple Ring Quadratic Snowflake and the Dual Ring Sierpinski Carpet. Hence, the generalization does not only provide to steer the balance between depth and width by selecting an appropriate num-

ber of nodes, but also provides alternatives that may be better suited for certain tasks. Providing such a variety of different layout presets bears the problem of selecting an appropriate preset for a concrete hierarchy. For its selection different properties can be considered such as the average number of children of the given hierarchy or the overall space utilization of the layout according to the preset. Here, the same algorithm as for the lightness adaptation can be used providing a numerical measure to rate and recommend the different presets generally based on the number of filled pixels. From this listing the user can then select a layout preset that is sufficient for his tasks.

Furthermore, such a listing of layout presets and their space utilization already holds interesting information concerning the structure of the hierarchy to be visualized. As each preset captures a specific balance between visible depth and width, interpreting this listing may reveal if the hierarchy is rather deep in case of a high utilization for an H Tree layout, or wide in case of a high utilization for a Sierpinski Carpet with multiple rings. If both layouts show a high space utilization at the same time the hierarchy features both characteristics.

3.2.2.2 Assignment of nodes

In the basic layout, nodes were assigned to the positions calculated by the recursion pattern according to a spiral. In this way, equal space is assigned to n child nodes and their corresponding subtrees per step while preserving the same amount of space for the remaining children around the root node. This assignment allows a general use of the layout as it is independently defined of a specific hierarchy with a given width and depth.

Yet, such an assignment may not be beneficial in any case especially concerning hierarchies with a low overall width. For instance, when visualizing a hierarchy with a maximum number of 5 children using the basic point-based layout preset (Quadratic Snowflake) the first 4 children will be assigned an equally sized area whereas the 5th child is assigned a smaller area and the remaining space is furthermore left empty. Hence, a different strategy is to utilize the complete remaining space for the 5th child which would increase the overall space utilization of the layout and allows the comparison of all five subtrees. In general, every point-based layout divides the given display area in $n + 1$ subareas for the first n children and one for the remaining children (as is visible in the recursion scheme of Figures 3.2.7 - 3.2.9). Thus any layout placing n nodes per step may be used to visualize hierarchies with a width of $n + 1$ in a similar way. One problem concerning this strategy is that the root node and the $n + 1$ th child are assigned to the same position in the center of the drawing space resulting in overplotting either hiding the root or the child node.

Besides assigning the remaining space to a single node, it may also be distributed across the already positioned children where applicable. For instance, the Sierpinski Curve (left side of Figure 3.2.8) provides each node a corner-like shape in the general case. Yet, for a hierarchy with a fixed width of 4 the empty space can also be equally distributed to all children resulting in assigning each of the 4 children one quadrant of the display space. In this way, a better assignment can be facilitated without the ambiguity of overplotting nodes.

In this sense, each fractal may serve as the basis for laying out trees with a fixed width or general trees by adapting the node-to-position assignment.

3.2.2.3 Combination of presets

The previously discussed visualizations use a single preset for the whole hierarchy. Yet, many real world hierarchies feature parts that are wider or deeper than others. In this sense, it may prove useful to locally switch the used preset and thus combine different presets to better adapt to these parts. Furthermore, such a combination can support a more equal distribution of the space to all children or increase the space utilization. While all presets follow a similar algorithm they often differ in the final fractal shape. Hence, possible ways for combining or exchanging presets are:

- a step wise switch of presets, and
- a switch to a compatible preset.

Switch between steps: Most presets have an incompatible yet puzzle like shape that fits together without leaving gaps. In this sense, subtrees may be arranged rather freely. Yet except for the first recursion step, such an arrangement has to resemble a fractal to guarantee a space-filling and overlap free tiling of the display space. Thus in the general case, a switch has to be defined depending on the recursion step.

For instance, when combining a Sierpinski Carpet at the first recursion step with a Quadratic Snowflake on all following steps, the first 8 children of the root are laid out using the Sierpinski Carpet. The following 4 children of the root as well as the first 4 children of the previously laid out 8 children are then placed according to the Quadratic Snowflake. The result is shown in the upper left part of Figure 3.2.9. This combination does not only allow an easier comparison of the first 8 subtrees, it also increases the overall space utilization (from around 60% to 80%) while maintaining the general characteristics of the basic Quadratic Snowflake.

The second example (lower left part of Figure 3.2.9) shows a rather free arrangement of Quadratic Snowflakes in the first recursion step to better fill a rectangular display area by placing additional fractals on the left and right side of the previous visualization. This idea is also the basis for the direct embedding of hierarchies into maps which is discussed in Section 3.2.4.

It may also be useful to remove some fractals during the first step by switching to fractal placing less nodes per step. Looking at the visualization of the DMOZ hierarchy based on the Dual Ring Sierpinski Carpet (lower center of Figure 3.2.7), it is visible that the hierarchy is not wide enough to utilize all available positions in the first step. By neglecting some of the available positions, for instance by using the basic Sierpinski Carpet, the space utilization may also be enhanced.

Switching between compatible presets: Some of the presets presented so far share the same shape such as the Sierpinski Triangle with its additional rings as well as the Sierpinski Carpet with its rings (see Figure 3.2.7). During the visualization of a hierarchy, these presets can be exchanged for each other without any problems. Examples for these

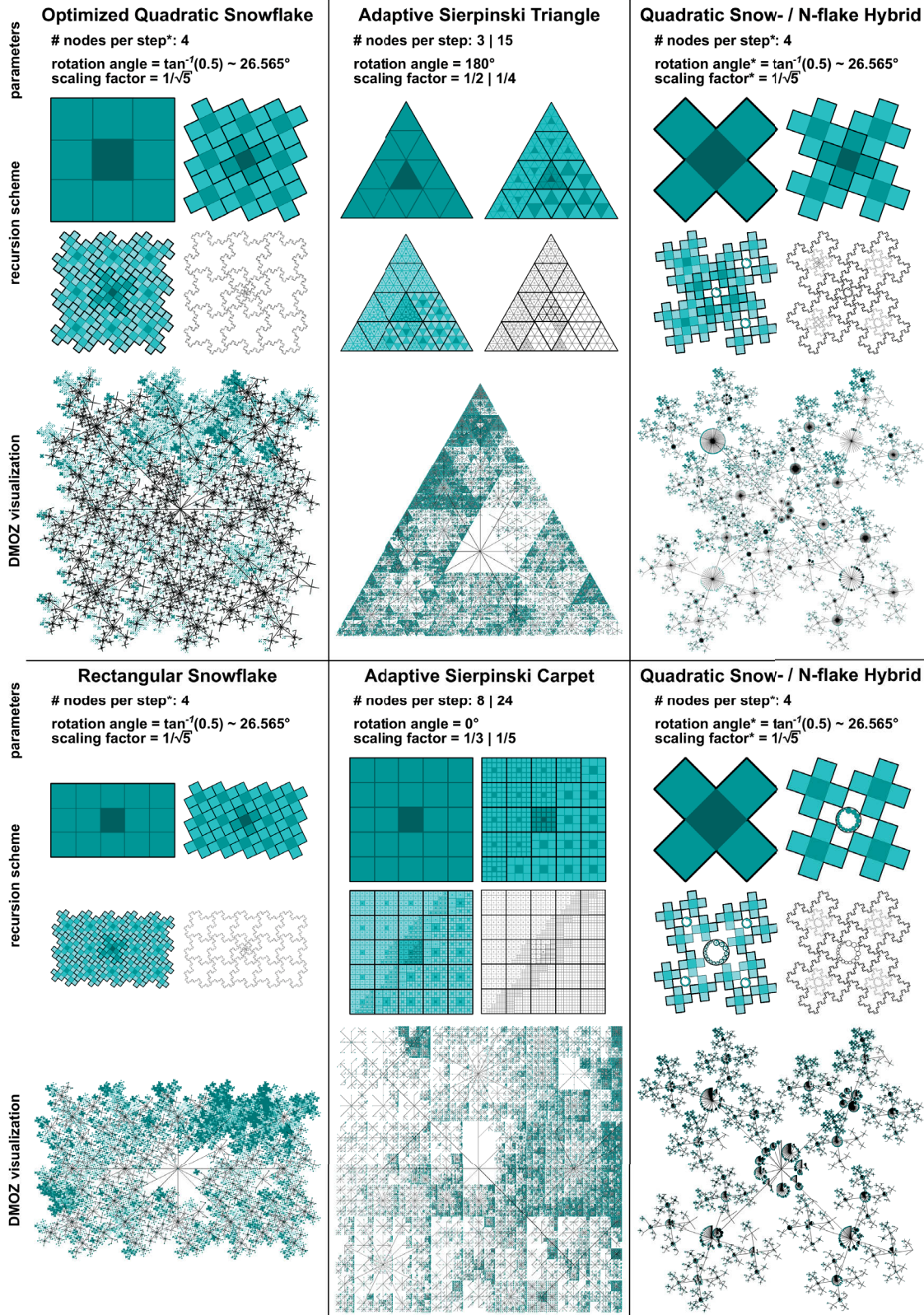


Figure 3.2.9: Three strategies to adapt point-based layouts to different settings and hierarchies. Left: increasing the number of placed children of the root in the first step to adapt to rectangular visual spaces or wider hierarchies. Middle: switching to a compatible layout on any step during the layout to accommodate the subtree's width. Right: embedding different radial layouts to communicate the remaining nodes to be placed.

adaptive layouts are demonstrated in the center column of Figure 3.2.9. Comparing the results for the DMOZ hierarchy with the original presets shows a better space utilization. However, the visualization also becomes harder to read as it is not clear anymore which layout is used at which step. In this way, the relation between a node's position on screen and its location in the hierarchy is not that distinct anymore.

For the combination of other presets, overlaps between different subtrees have to be avoided to maintain the readability of the layout. One such combination is shown on the right side of Figure 3.2.9. Here, the Quadratic Snowflake is combined with an n-flake to lessen the inability of the basic point-based layout to represent wide trees. The n-flake is used to capture all remaining subtrees that would not be visible otherwise and thus better reflects wide parts of the tree, on the cost of the space utilization. In this way, the switch is used to pinpoint wide parts while maintaining a general balance between visualizing the width and the depth of a tree.

3.2.3 Visualizing Attributes for Large Structures

When visualizing large graphs with the main focus lying on the structure, possible ways for communicating attributes are very limited and partially restricted. Especially in case of the point-based layout, its rather fixed nature poses additional restrictions to the mapping of attributes on the node positions. The small primitives used to represent nodes only allow the mapping of attribute values on the color, the layout order of nodes and by utilizing the third dimension (see Section 2.2.2 for a discussion of visualizations of attributes). Furthermore, the number of nodes and edges may exceed the available display space and not all nodes or edges may be visible and thus their attribute values. Hence, in the following strategies are described for mapping attributes on:

- the color of individual nodes or edges,
- the color of paths from nodes to the root,
- the order of the nodes,
- the third dimension, and
- the rendering order.

The second, third and fifth strategy are in particular taking the visibility of individual nodes, edges and their attribute values into account. All these strategies are exemplified for the DMOZ hierarchy. For each category (node) besides structural attributes such as its depth, width or subtree size, the number of associated websites is captured as an additional attribute.

Simple color coding: One of the most common ways to communicate attributes while showing the structure is the utilization of colors to depict certain attribute values. For the mapping of values to color different strategies and goals are feasible. For instance, different color scales are used for identifying general or locating specific attribute values [TFS08b] as well as for representing different types of attributes such as nominal or

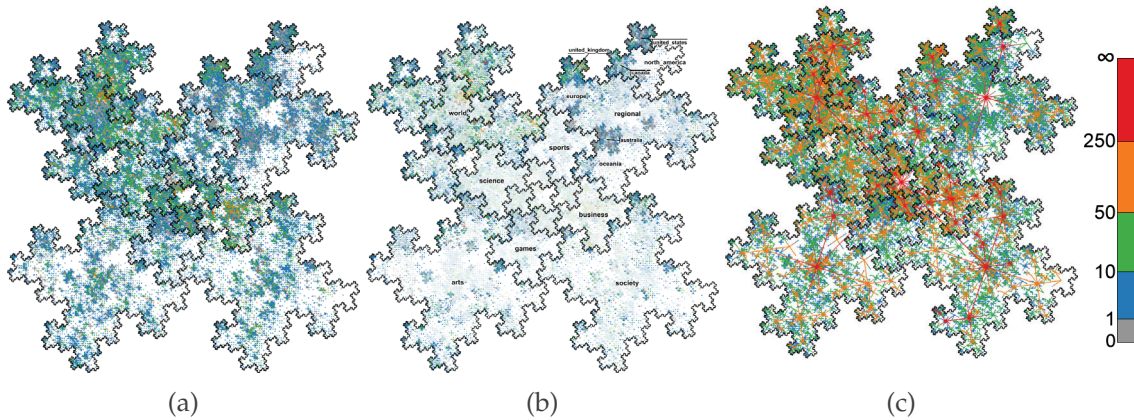


Figure 3.2.10: Mapping of attribute values (number of associated websites) on the color of the nodes without (a) and with (b) lightness adaptation, as well as on the paths to the root (c).

ordinal values [SSSM11]. For the simultaneous visualization of multiple attributes bivariate (showing 2 attributes) and trivariate (showing 3 attributes) color scales may be used [RO86, Pha90] but are also more difficult to read.

For the point-based layout a lightness adaptation is performed to better reflect the structure of the hierarchy. If both, the attribute-color mapping and the lightness adaptation, are to be used at the same time, the adaptation poses limitations on how to use color. In this way, the color mapping results in a bivariate color scale with lightness utilized for the structure leaving hue and saturation for the attribute. Here, the lightness should not exceed a maximum value, so that colors describing particular attribute values are not converted to a complete white color and thus prevent the differentiation of attribute values. In this sense, it is also preferable to use a segmented color scale with a small number of different colors to allow the identification and comparison of specific value ranges even if the lightness is heavily altered.

In Figures 3.2.10a and 3.2.10b this color coding is exemplified without and with lightness adaptation, respectively, showing the distribution of websites in the DMOZ hierarchy. Here, the distribution is best visible in the first figure showing that most of the visible nodes exhibit a low number of websites (depicted by blue and green color). Only smaller parts of the hierarchy show higher number of associated websites (orange and red color). As shown in the second figure, the lightness adaptation severely affects the readability of the associated attributes and the range of the values is only assessable as very different colors have been chosen. For this reason, the lightness adaptation should only be used when the structure is clearly in focus of the analysis.

Furthermore, edges are only drawn where necessary to reduce visual clutter. Yet, to visualize edge attributes this adaptive edge rendering may be turned off to show all edges and their attributes on the cost of overlapping nodes.

Path-based color coding: A major problem when visualizing large structure is the high overlap of nodes and edges. For hierarchies, visualization techniques tend to run into subpixel regions during the layout. Hence, multiple nodes may be mapped on the same

pixel and occluded by their parents as it is also the case for the point-based layout. When analyzing attributes this also means that only those attribute values are visible of the nodes and edges drawn on top of overlapping pixels. In this way, specific attribute values such as extrema may be missed.

Yet especially in hierarchies, each node is not only characterized by its position on the screen and its local neighborhood in the graph, but also by its unique path to the root. By mapping an attribute value not only on the color of the node but also on all nodes and edges lying on the path to the root, the problem of invisible values may at least be lessened for extrema. Here, the color of each node may then be determined by either calculating the minimum, the maximum or even the average of all values existing in its subtree. Every edge may then be assigned the color of the child node. This approach correspond to an aggregation of attribute values from the leaves up to the root.

This mapping is depicted in Figure 3.2.10c for the maximum number of associated websites. By this mapping, edges are guiding to high attribute values that can be followed even into regions that are not yet visible but after a zoom operation. However, this mapping overemphasizes extrema and also introduce some ambiguity as the node color does not only reflect its own attribute value any more. Thus, if a node is found that seems to exhibit a high attribute value, it has to be confirmed in a second step if its color is based on its own high value or on the value of a child. This confirmation may be facilitate for instance by a tooltip showing the concrete information of the node.

Adapting the ordering of the nodes: Another way to increase the visibility of certain attribute values such as extrema is to influence the positioning of the nodes during the layout. While the available positions are fixed, the nodes are assigned to them in a specific order. For a good representation of the structure this order was based on the size of the corresponding subtrees. By determining this order according to an attribute, nodes with either a high or low value (inverse order) may be assigned to the first positions and thus made explicitly visible. Yet, this may reduce the overall space utilization and thus the visible attribute values.

3D extension: Besides color and 2D position in a limited way, the third dimension which is not used by the point-based layout may also be utilized for representing attributes. The two most commonly applied mappings are on the z-coordinate (translation along z-axis) as well as on the height.

Both mappings are shown in Figures 3.2.11a-3.2.11c for the mapping of the depth on the z-coordinate and the number of links on the height, respectively. The utilization of the third dimension always includes occlusion and thus demands for additional interaction techniques such as rotating the scene. Especially in case of a mapping on the z-coordinate, there is a high occlusion caused by the dense layout as can be seen in Figure 3.2.11b. In this way, much more than getting a rough overview of the general depth is not possible with this approach.

Contrary, the mapping on the height facilitates a better identification and comparison of high or low attribute values in opposition to using color. On the one hand, peaks are easier to perceive and visible even if the corresponding nodes may be occluded by their parents. On the other hand, differences in the length can be more accurately compared

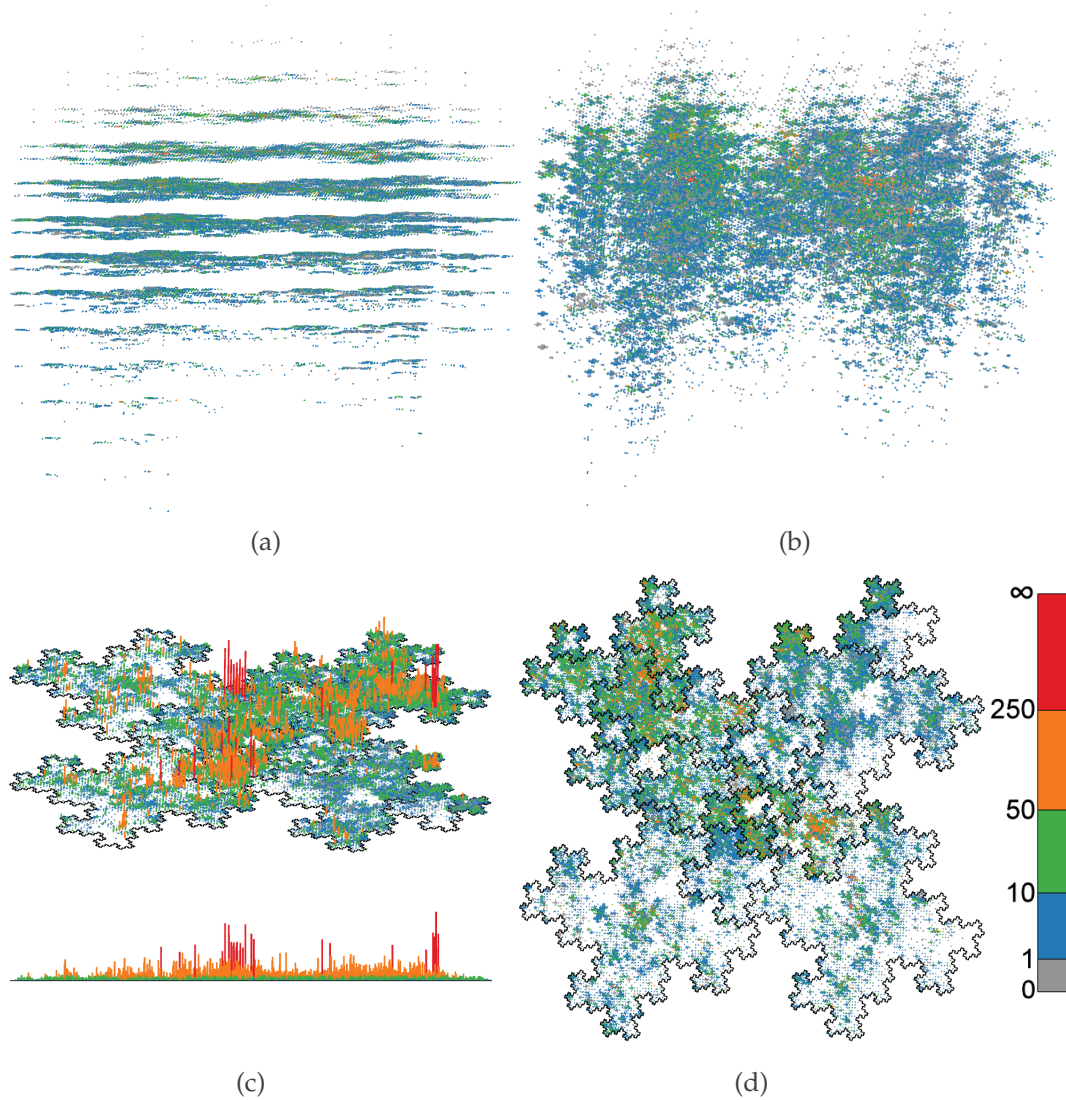


Figure 3.2.11: Mapping of attribute values on the third dimension. (a) the depth is mapped on the z-coordinate allowing the differentiation of the levels and explicitly shows the overall depth of the hierarchy. (b) the same visualization as in (a) with a rotated perspective reflecting the high overlap of this mapping. (c) the number of websites is mapped on the height allowing the localization of high values as well as their comparison. (d) adapted rendering order of the nodes to allow the localization of specific attribute values (in this case, a target number of websites of 200) throughout the hierarchy.

as stated in [Mac86]. However, this approach also only emphasizes extrema. Finding specific values lying anywhere in the domain of the attribute is yet still not possible with a simple mapping on the third calculation.

Adapting the rendering: Instead of mapping the attribute directly on the z-coordinate or height, a two-step approach combining color and z-coordinate mapping may be facilitated targeting the rendering order of the nodes and edges. So far, nodes are visually ordered according to their location in the hierarchy by rendering parents on top of their children. This leads to the problem that nodes with possibly more interesting attribute values are occluded. A simple inversion of this order also did not solve this problem as then children and especially the leaves are occluding their parents.

One solution is similar to the use of Ghost Views [LS08a] by first letting the user define a Degree-of-Interest (DoI) for each attribute value. Based on this DoI definition, a DoI value can be calculated for each node according to its attribute value. By mapping the DoI value on the z-coordinate of each node the rendering order of the nodes is changed in a way that emphasizes nodes with specific attribute values. Finally, the attribute values may be communicated by a color mapping. This adaptation results in rendering specific children on top of their parents and may lead to an ambiguity similar to the path-based color coding. Each position usually reflects a specific node, however now it may also represent one of its children which was mapped to a subpixel region around that position. In contrast to the path-based color coding, this ambiguity is automatically resolved by zooming in resulting in the nodes placed side by side instead of overlapping each other.

Figure 3.2.11d exemplifies this mapping strategy to highlight nodes with a number of associated websites around 200. In contrast to Figure 3.2.10a which has shown mostly nodes exhibiting a low number of websites (blue and green color), now much more orange colored nodes are rendered on top. Especially in the world subtree (upper left corner) this increase can be observed. In this way, this mapping reveals that on lower levels of the hierarchy nodes with more associated websites may be found.

3.2.4 Extension to Spatial Context

As discussed in Section 2.2.3, most available techniques for conveying the spatial context of a graph are either defined for graphs where each node has an exact spatial reference or provide a detached solution showing map and graph visualization side by side. In this section, a new approach is introduced facing the challenge of an embedded visualization of a hierarchy within its associated geographical region. Here, the hierarchies are not related to a hierarchical subdivision of space as commonly seen in GIS, but may be arbitrary hierarchical structures (e.g., structure of time, clustering hierarchies, or organizations). One problem concerning this embedding on the one hand are the usually irregular shapes of geographic regions. Most hierarchy layouts, however, assume specific shapes, for example rectangles or circles, and cannot be adapted easily to handle irregular shapes. On the other hand, an algorithm that adapts the layout of a given hierarchy to irregular shapes can lead to very different visual results. In other words, layouts could vary a lot for the regions of a map, impeding comparison of data associated with the hierarchy.

To tackle these problems, two different approaches based on the point-based layout are proposed for embedding a hierarchy into an irregular region:

- The first approach is more data driven and aims at using the space efficiently showing a maximum number of nodes and therefore most of the associated data.
- The second approach sacrifices adaptiveness to irregular shapes in favor of better readability of the visual representation. It is less space-efficient, but facilitates comparison tasks.

In the following, both approaches are described in more detail.

3.2.4.1 Layout to Use Space Efficiently

Thanks to the fixed and puzzle like nature of the point-based layout, the final shape of the hierarchy representation can be quite easily be adapted by neglecting certain locations during the assignment phase. To achieve a better adaptation of the hierarchy layout to the underlying geographical region, the layout operates on *areas* subdivided into *subareas*. Here, the term *area* is used to denote the display space covered by a region. In this sense, subareas are just a geometrical means for computing the layout of subtrees; they are not related to any organizational structure of geographical space which would be the case for a spatial hierarchy. Subtrees will be assigned to subareas depending on their sizes. For finding a good placement of subtrees the following procedure is applied recursively, eventually resulting in a layout of the hierarchy:

1. Compute center of area.
2. Subdivide area into subareas.
3. Map subtrees onto subareas.
4. Apply procedure recursively.

To embed a hierarchy T into the area A corresponding to its associated spatial region $R \subset \mathbb{R}^2$ one starts with the computation of the center $c \in A$. Secondly, A is subdivided into k subareas A^1, \dots, A^k , where k is the number of subtrees T^j ($1 \leq j \leq k$) below the root r . Then, subareas and subtrees are sorted by size and are assigned to each other accordingly. This procedure continues recursively for each subarea and its associated subtree. While steps 3. and 4. are straight-forward, steps 1. and 2. should be explained in more detail.

Step 1: Compared to regular and convex shapes, it is not a simple task to determine a central position c of an irregular area A , which is necessary to place the root node of a subtree. Obviously, a center has to be inside the area and it should have a maximum distance to the area boundary. Several possibilities exist to choose a center point, but none is without problems. For example, the barycenter of a concave area may be outside of that area, while the center of the largest inscribed circle may turn out to be far from being central. The layout utilizes the barycenter, the center of the largest inscribed circle, and

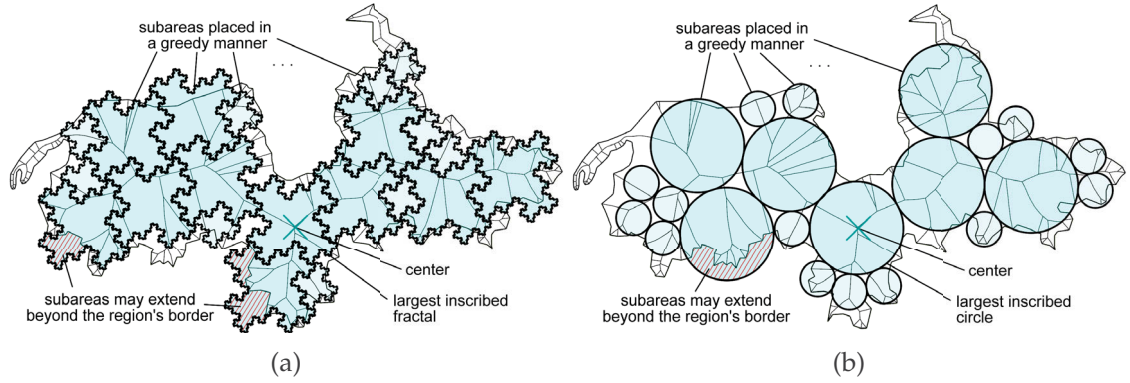


Figure 3.2.12: Subdivision of an irregular area along its skeleton embedding fractal (a) or circular (b) shapes.

the skeleton point with the highest importance as described by [TvW02] and evaluates the results of the different methods. For each center candidate, the visibility polygon is calculated, namely that part of the area that can be seen from the candidate. Since one can interpret this as a measurement of centrality, the candidate with the largest visibility polygon is chosen as the center.

Step 2: The chosen center is the starting point for the area subdivision. The first subarea A^1 is defined according to the largest inscribed circle around the area's center $c \in A$. Here, either the fractal shape of the point-based layout scaled to fit into the circle or the circle itself may be used as the first subarea (compare Figures 3.2.12a and 3.2.12b). Utilizing the puzzle like properties of the fractal shape allows a higher space utilization whereas the usage of circles naturally introduces borders increasing the readability of subtrees. Depending on application needs, other shapes may also be used as long as they fit entirely inside the inscribed circle of A^1 .

In a greedy manner, further subareas of the same size are placed as A^1 along the area's skeleton. In this way, it follows the idea of adapting the first level as outlined in Section 3.2.2 to adapt the layout to different display areas. If no further area of that size can be placed the greedy procedure continues with a search for smaller subareas. The scaling factor may be chosen according to the utilized preset. In case of the basic point-based layout, searching for subareas of size $\frac{1}{5}A^1, \dots, \frac{1}{5^4}A^1$ has proven practical. This procedure stops when the size of subareas is getting too small (ca. 1% of the size of A^1) or a user chosen number m of subareas has been created. Both termination conditions are independent of the hierarchy, which allows the precomputation of the subdivision and its reuse for different hierarchies. Furthermore, subareas do not necessarily have to lie completely inside the irregular area's shape, but may slightly extend beyond it.

After the subdivision has been computed, the k subtrees of a hierarchy are assigned to the m subareas. If $m < k$, the largest A^i are split into further subareas to generate enough subareas. If $m > k$ some A^i can be reunited to get closer to k , and thus, to use space more efficiently. Then, the procedure is applied recursively.

In summary, this approach is capable of embedding a hierarchy layout directly into a 2-dimensional geographic region of irregular shape, while being computationally and space efficient. A result of this layout is illustrated in Figure 3.2.13 for the subtrees of

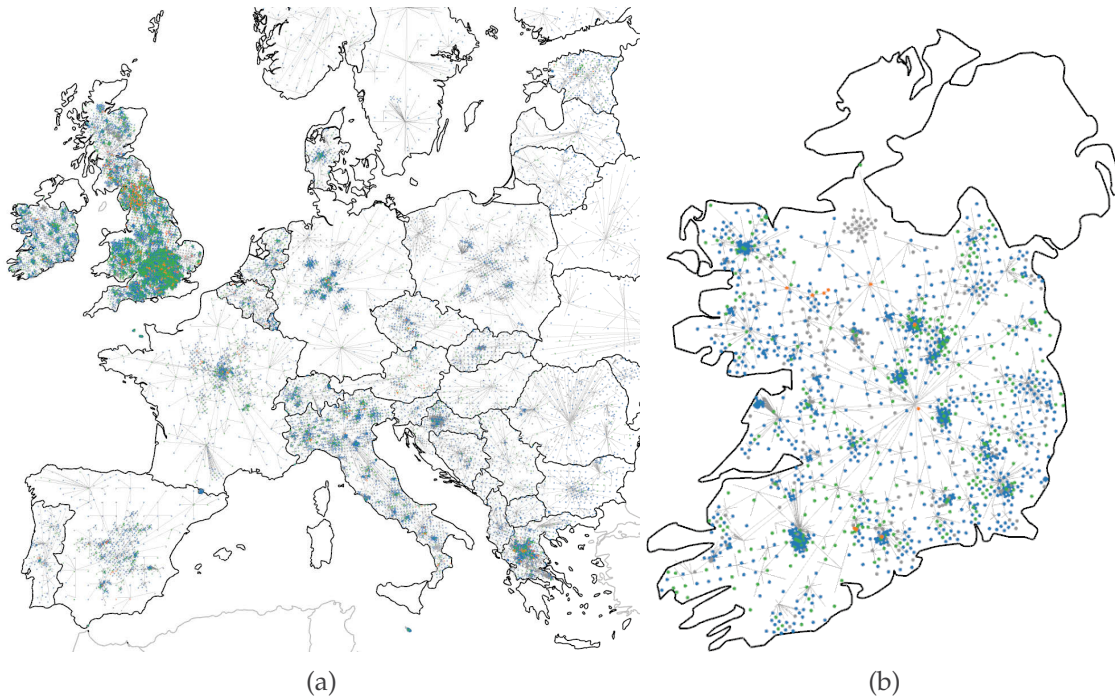


Figure 3.2.13: A hierarchy layout embedded into an irregular region: (a) Europe's regional subtrees of the DMOZ hierarchy laid out on a map of Europe. (b) zoomed view showing only the subtree for Ireland.

Europe's regional branch of the DMOZ hierarchy. Here, each subtree is associated to a specific country and exhibits a different degree of detail visible by the varying space utilization in the different countries. Based on the very dense visualization within the United Kingdoms, its subtree seems to contain the most nodes compared to the remaining country. This is true, even if the comparability may be affected by the different country sizes. Zooming into one of the countries such as Ireland (Figure 3.2.13b) shows the adaptability of the layout to the irregular shape allowing the utilization of almost the whole area.

3.2.4.2 Layout to Facilitate Visual Comparison

The previously described algorithm adapts the layout of a hierarchy quite well to the shape of a region. However, this comes at the price of losing comparability between regions. The simplest solution to ensure comparability is to restrict the layout to the largest inscribed circle of the corresponding area only. Then all layouts would be easily comparable varying only in size. However, space would not be used efficiently as space outside the inscribed circle would be left unused. It is therefore necessary to find a compromise between being adaptive for efficient use of space and not being adaptive to cater for comparability of layouts.

To this end, the space-efficient adaptation is only used down to a certain level of the hierarchy, and a fixed layout is used for the lower levels. This means that at higher levels of the hierarchy a coarse adaptation to the underlying shape is achieved, while lower levels are kept comparable between regions by resorting to a non-adaptive layout. The

level at which to switch the layout strategies is dependent on the data and may also be set interactively by the user.

This concept is implemented by combining a modified version of the space-efficient layout with a regular non-adaptive layout (e.g., the point-based layout or any of the layout presets mentioned in Section 2.2.1). To avoid too small subtrees at lower levels only around three hierarchy levels may be laid out in the first step. For their layout a similar approach is applied as described in the previous section, but with the following modifications:

- There will be no merging of subareas if $m > k$.
- The subareas placed on the chosen depth level must not extend beyond the irregular area's shape.

These restrictions are necessary to ensure that the same initial shape, varying only in size, is available for the fixed layout of lower subtrees. Additionally, these subareas are ordered (e.g., by their x-coordinate) to generate a replicable visual mapping for each region and thus to simplify the search for similar subtrees across different regions.

Then, in the second step, the layout for the remaining subtrees is computed using standard algorithms that produce comparable visual representations. An example for this combined layout strategy will be showcased later in Section 3.2.6. In case that the sizes of the regions differ too much or have to correctly reflect the statistical data associated with the hierarchy (e.g., the population of the region), a combination of this approach with well established transformations of the map display (e.g., cartograms) is possible.

3.2.5 Extension to Temporal Context

As described in Section 2.1.3 the dynamics for graphs can be differentiated into structural and attribute changes. Especially for hierarchies, the structural changes may occur as additions or deletions of nodes, edges or subtrees as well as movements of nodes and their appending subtrees. Besides structural changes the associated attributes are also subject to change. Often the analysis of the evolution of attributes over time is the primary goal of visualization.

As summarized in Section 2.2.4 common ways to communicate the temporal aspect with the structure being in focus are animations, layering, and the combination of different views. For an initial overview, a simple animation of the visualization over time may be useful. When it comes to discerning details in the course of time, visualizations that show a sequence of time steps are better suited. In any case, the large size of the structure and the small representatives for the nodes pose restrictions and requirement to these strategies that are detailed below.

Animation approach: Animations are based on minimizing alterations to the visualization such as node movement to communicate the dynamics in the graph. In this way, changes in the visualization are directly reflecting changes in the data. Because of its fixed nature the point-based layout is a suitable candidate for visualizing structural changes of dynamic hierarchies. Smaller changes will thus only locally affect the layout whereas the remaining layout remains stable.

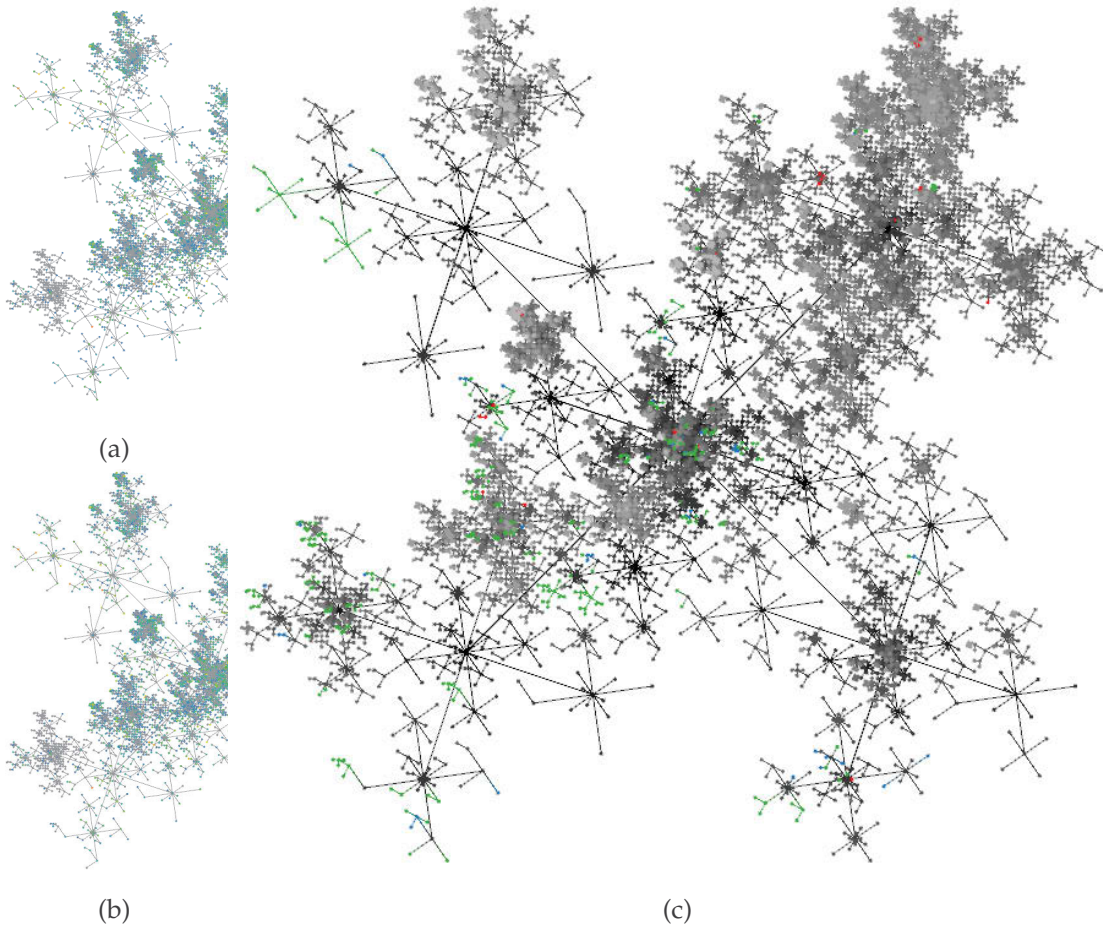


Figure 3.2.14: Point-based visualization of Europe's regional branch of the DMOZ hierarchy for 2 consecutive time points (snapshots taken at 06/01/2013 and 03/02/2013). (a) shows the first time point, (b) shows the second time point, (c) shows also the second time point but highlights changes in the graph structure. Nodes and edges colored in blue have been added in the second time point, in green have moved in the hierarchy from the first to the second time point, and in red have been deleted after the first time point.

However, when visualizing hundreds of thousands of nodes it can become difficult to find all changes, structural as well as attribute related, even with such a stable layout. On the one hand, this problem is due to the small representatives for the nodes whose deletion or addition or even a slight change in its color may be easily missed. And on the other hand, there can be too many changes at the same time to be able to grasp all within a single transition between two time points. To alleviate this problem, changes in the data should be highlighted in the visualization. For enhancing an animation, this highlight may be implemented by coloring additions, deletions and movements of subtrees by discernible colors similar to the difference graph introduced in [Arc09]. Changes of attributes may be encoded similar using specific colors to describe the increase and decrease of attribute values.

In Figure 3.2.14 two time points of a dynamic hierarchy are shown without and with highlighting structural changes. By comparing the normal visualizations (Figures 3.2.14a

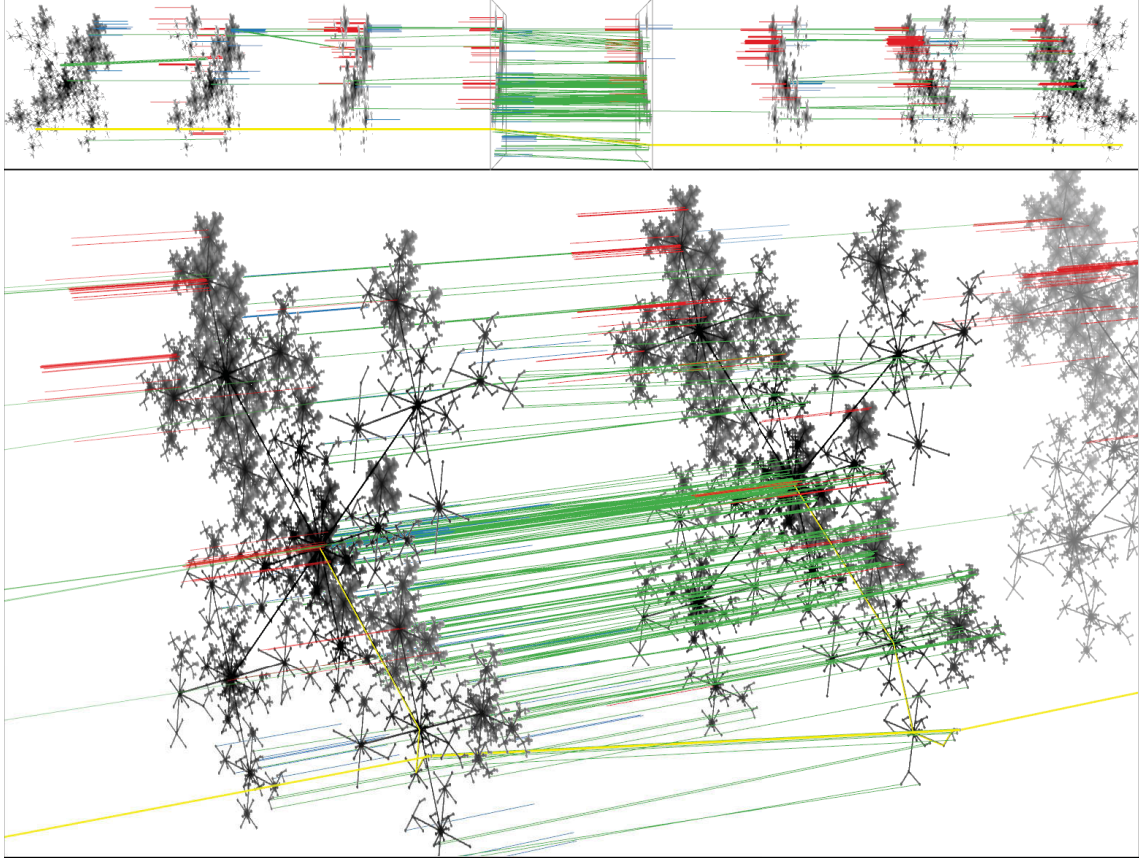


Figure 3.2.15: Layered visualization of 8 time points of Europe's regional branch of the DMOZ hierarchy (snapshots taken each month from 2008/10/07 to 2009/06/01) An overview showing all 8 time points is linked to an detail view showing the changes between two time points (framed by a rectangle in the overview) – Green links between layers indicate movements of nodes and subtrees. Addition and deletion of nodes is shown via red and blue spikes, respectively. The movement highlighted in yellow concerns the maritime transportation system in Greece which is merged with the general transportation branch.

and 3.2.14b) it is difficult to capture all changes between those time points. The highlighting (Figure 3.2.14c) explicitly visualizes the changes making it easier to estimate the amount and location of changes between time points. Yet, here the same problem occurs as with the visualization of attributes. As not all nodes may be visible changes affecting them may be missed. Similar solutions such as using the third dimension may also be applied as outlined below.

Layering approach: As described in Section 2.2.4.3 in a layering approach, the third dimension of the presentation space represents the time axis (analog to the space-time-cube approach, see [Kra03]). To this end, most approaches consider a successive series of time steps (t_i, \dots, t_{i+s}) with $1 \leq i < i+s \leq n$. Yet it may be beneficial for comparing time steps anywhere on the time axis to let the user override this default series with any ordered set of time points. For each time step then a separate layer L_i is rendered representing the hierarchy layout at that time step. Because of the dense visualization on

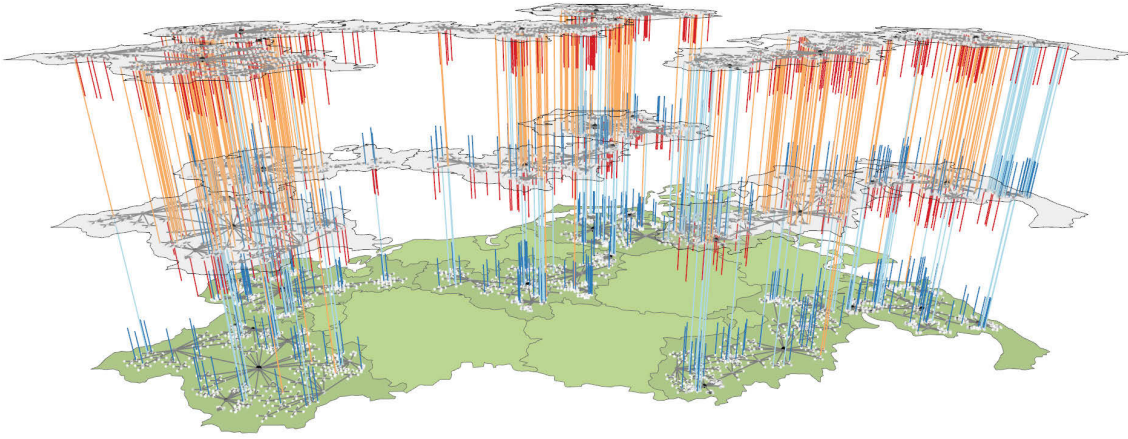


Figure 3.2.16: Visualization of hierarchies for selected regions and three time steps. – Colored links between layers indicate significant increase (red) or decrease (blue) of node attributes. Addition and deletion of nodes is shown via red and blue spikes, respectively.

each layer, rendering all links between subsequent layers as done by existing techniques will result in severe occlusion and is thus not a viable option. In general for layering large graphs, a different strategy based on communicating only the changes should be applied. To facilitate identification of the aforementioned changes in between two layers, besides links additional visual cues are added:

- Differently color-coded *links* connect subsequent layers to indicate nodes that have moved or whose attribute values have changed significantly. Significance is determined by a user-selectable threshold. According to a thermometer scale, positive attribute changes are visualized as red links, negative changes are shown in blue. Links representing node movements are colored with green.
- Additions or deletions of nodes and edges are visualized by *spikes*. Blue spikes point (not connect) from a layer L_i to the subsequent one L_{i+1} to indicate objects that exist at t_i but not at t_{i+1} (deletion). Whereas red spikes point from L_i to L_{i-1} to indicate addition of objects at t_i .

The layering approach in combination with the described visual cues allows the user to compare successive time steps more closely. In Figures 3.2.15 and 3.2.16 all visual cues are shown highlighting structural and attribute changes, respectively. Again, this explicit visualization of changes helps to find even small changes in the data that could barely be seen otherwise. Yet, even changes that would be occluded by other nodes are visible. Furthermore, the spikes and links provide handles to select and thus pinpoint specific changes. The second figure also demonstrates the applicability of these cues in combination with the spatial embedding discussed in the previous section.

However, to avoid problems that are caused by overplotting and visual cluttering, only a smaller subsequence of time steps (e.g., $s < 10$) can be represented in a single visualization.

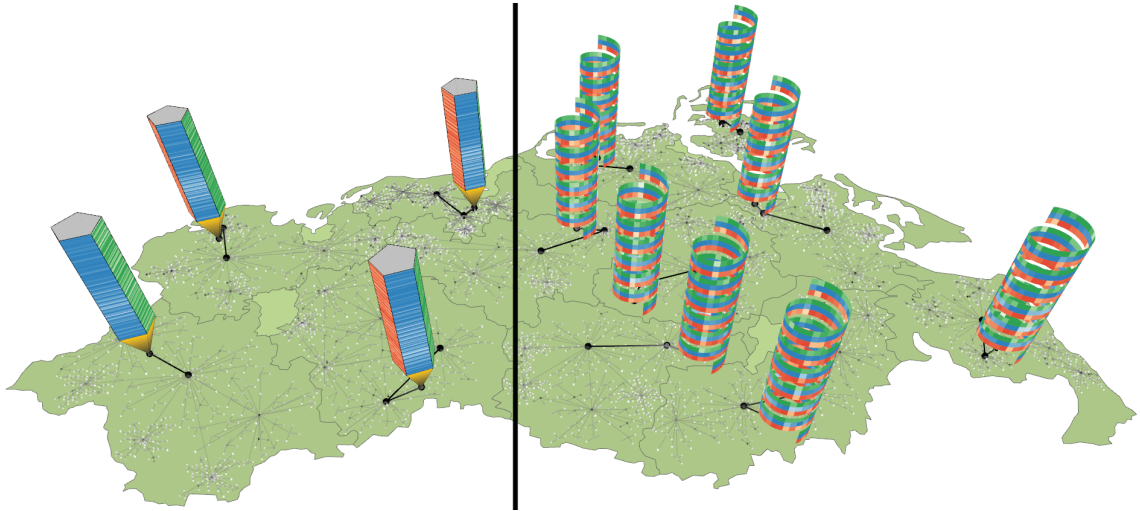


Figure 3.2.17: Pencil and helix glyphs visualize data attributes associated with hierarchy nodes.

Multiple-views approach: To alleviate this restriction and to allow for comparison of time steps that may be located anywhere on the time axis, multiple views are commonly used. Each view may be configured independently to show separate portions of the temporal domain, either a single time step using a 2D representation of the structure or a subsequence of time steps using the layering method. In this sense, it may be useful to provide the user with the functionality to arrange views according to their needs. In this way, it is possible to align views in a row to resemble a temporal sequence, or to set up larger focus views and smaller views for those parts of the time axis the user deems to be contextual.

If multiple attributes should be visualized, the available display space allows their visualization only for a selected subset of nodes or edges. Here, the supergraph of the hierarchy is visualized as a context in which to pick specific nodes or edges to analyze in more detail. The hierarchy layout is therefore dimmed and combined with additional visual representations of data attributes. Pencil and helix glyphs as described by [TSWS05] are two examples which furthermore fit well into the utilized space-time-cube visualization. These glyphs are applied to show the development of multiple data attributes over time for a rather small amount of nodes. Figure 3.2.17 shows pencil and helix glyphs side by side. Pencil glyphs are suited to visualize linear development, whereas helix glyphs are useful for exploring cyclic patterns. The glyphs are positioned with respect to nodes that users may select in the hierarchy visualization, and each glyph represents the data attributes of its associated node via color-coding.

3.2.6 Use Case

The techniques presented in the previous sections provides the general means to visualize attributed hierarchies in a spatio-temporal setting. In the following, these techniques will be applied to a concrete visualization problem, namely the visual exploration of human health data. These data provide information about how many people suffered from a particular disease, in a specific geographic region, during a certain temporal period.

As a matter of fact, diseases, geographic regions, and time, all are organized in a hierarchical structure: ICD10 classification of diseases, administrative districts (state, district, postal code area), and calendar system (year, quarter, month, week, day), respectively. The latter structure is of particular interest as most other approaches consider the temporal domain on a single granularity and across a linear axis. The consideration of a hierarchical representation of time provides a complement to their linear representation of temporal aspects.

3.2.6.1 Visualizing the Hierarchical Structure of Time

Health data may contain various interesting temporal patterns to be addressed by a visualization such as the following three questions:

1. How are diseases distributed during the week? (weekly pattern)
2. Is the number of cases dependent on the season? (annual pattern)
3. Are cases uniformly or non-uniformly distributed? (overall pattern)

The weekly pattern describes the behavior of sick people seeking medical care. Most people hope to recover naturally over the weekend, but are more likely to consult a doctor early in a week to get at least a sickness certificate. A deviation from this pattern might indicate a critical emergency.

The annual pattern describes if and how a disease depends on the season. For instance, most cases of influenza generally occur during fall and winter. A larger number of influenza cases in summer might hint at an import of the virus from somewhere else.

The last pattern considers the overall temporal distribution of cases: is there a uniform distribution of diagnoses over a larger period or are they highly accumulated at certain points in time (e.g., many injuries after a sport festival or a car accident)?

There already exist a number of powerful techniques such as spiral displays or time line plots for visualizing one of the mentioned patterns or finding peaks in the time line. But in order to assist users in answering all three questions, a temporal hierarchy may be constructed posing as the foundation of a visualization reflecting these patterns. As the data is given for years, quarters, months, weeks and days, this data is used directly as the basis of the temporal hierarchy, but other hierarchical decompositions of the time (e.g., by seasons) are also possible.

In this way, a fixed hierarchy is constructed for each year with 4 children for the root (quarters), each of them having 3 children (months) and so on. In its construction the hierarchy resembles the Sierpinski Curve layout described in Section 3.2.2 which provides in the first step 4 equally sized areas and in each following step 3 areas. As this layout is defined for general trees, an adaptation to the fixed hierarchy may be useful in terms of readability and space utilization, especially in small drawing areas. The adapted layout works as follows:

- The year is placed as the root node in the center.
- The four quarters are positioned clockwise around the year.

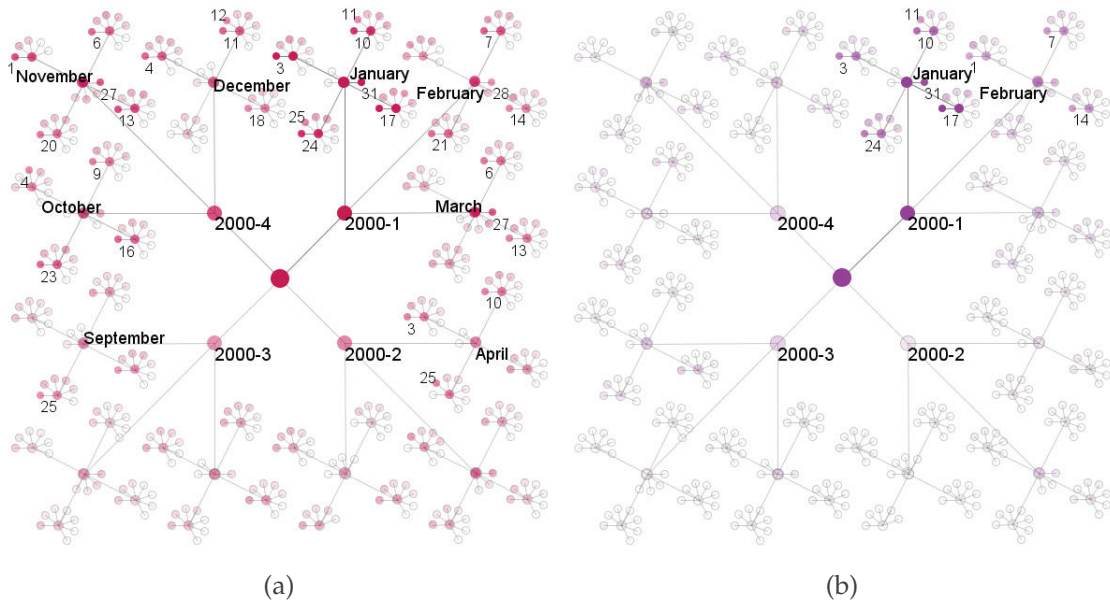


Figure 3.2.18: Visualization according to the hierarchical structure of time. – Left: number of people with problems with the upper respiratory tract; right: number of people suffering from influenza.

- The months are placed clockwise on another *ring* around the corresponding quarter.
- Around each month, its weeks are placed.
- Finally, days will be positioned on smaller rings around the week nodes. A small gap between Mondays and Sundays is inserted to indicate beginning and orientation of a week.
- Days that do not fit into the four week pattern, usually the first and last days of a month, are located directly around the month nodes: the month's first days to the left and its last days to the right.

In this way, the layout represents a combination of the Sierpinski Curve layout, the basic point-based layout (Quadratic Snowflake) and a 7-flake (an n -flake with an n of 7).

In this hierarchy, the leaves contain data measured on a daily basis and the internal nodes (week, month, quarter, year) contain data aggregated by summing up data values along the hierarchy. A simple color coding (compare Section 3.2.3 for suitable mapping strategies of attributes) can be used to visually encode data values. Using white for no cases and a fully saturated color for the maximum number of cases accentuates the nodes with higher number of cases and improves the interpretation of the data value distribution. Additionally, labels are shown for nodes whose values exceed a user-chosen threshold, which further accentuates important nodes.

Figure 3.2.18 shows layout and color coding for two diagnoses, influenza and diseases related to the upper respiratory tract. The adapted layout allows for an easy interpretation of the underlying data especially emphasizing the interesting patterns and for comparison with different diseases or years. With regard to the assumption that people get sick certificates more often in the beginning of a week, one can clearly see that the most

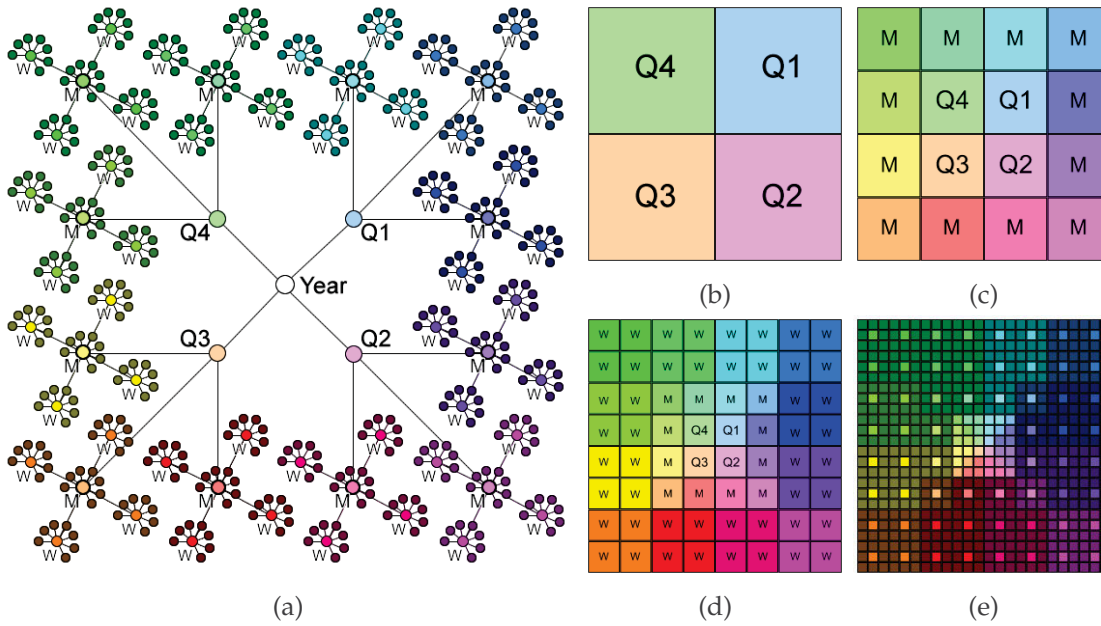


Figure 3.2.19: Different visual abstractions of the temporal hierarchy layout. High-resolution node-link representation (a) and iconic representations on 4 different levels showing the hierarchy up to quarters (b), months (c), weeks (d) and days (e).

saturated colors (high number of cases) appear for the first days of a week. The clockwise positioning of the quarter and month nodes provides a self-contained overview of the annual pattern. For instance, influenza on the one hand is seasonal and occurs mainly in the cold winter months, as it can be seen from the dark colored nodes in the upper right corner of the visualization, pointing to the first three months of the year. On the other hand, diagnoses generally related to the upper respiratory tract are more evenly distributed over the year. Furthermore, the visualization reveals a relation between both diagnoses: at days with larger numbers of people suffering from influenza, there are also more people that have problems with the upper respiratory tract. This is not surprising as influenza has a direct impact on the upper respiratory tract, but it demonstrates quite well the effectiveness and comparability of the visualization. The fixed layout also increases the users' awareness of missing nodes or nodes with only few cases, as such nodes result in unused or white space in the visualization.

3.2.6.2 Embedding Into the Map Display

The goal is to finally compare the temporal trends of different diseases for the regions of a map. For this purpose, layouts (for the time hierarchy as described before) are calculated for all selected regions of a map and for a selected diagnosis. Each layout is placed in the largest inscribed circle of its corresponding region.

Depending on the available display space different visual abstractions of the layout may be rendered (see Figure 3.2.19). The higher resolution node-link representation (Figure 3.2.19a) is then used if plenty of display space is available (e.g., when the user has zoomed into the map). Whereas if display space is limited (e.g., when dealing with small

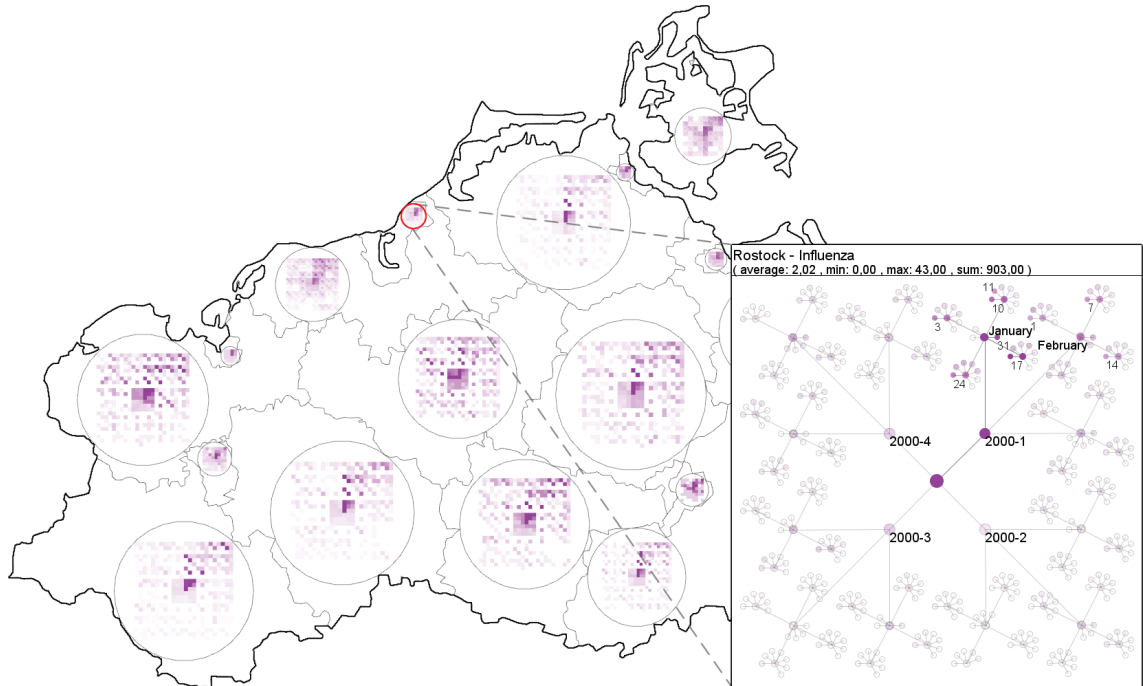


Figure 3.2.20: Visualization of influenza in Mecklenburg-Vorpommern for the year 2000.

regions), the application of an iconic representation of the layout (similar to [SDW08]) may be more suitable. With decreasing display space the number of shown hierarchy levels and therefore the number of nodes inside the iconic representation is reduced (Figures 3.2.19b - 3.2.19e) showing then just a clear overview of the higher levels.

The iconic rendering in combination with a high resolution detail view is illustrated in Figure 3.2.20. One can see that many regions share the same temporal trend as the highlighted Rostock region, showing high occurrences of influenza in the first two months of the year. Some regions show high numbers of affected people also at the end of the year, which might be an indication that the inhabitants of these regions are less vaccinated against influenza.

In order to visualize more than one disease at a time, multiple layouts have to be placed per region. Here, the comparable layout described in Section 3.2.4.2 poses as a suitable basis to embed the layouts into the inscribed circles of the subregions. To enable users to identify the same disease in different regions, each disease is encoded with a unique hue, where similar diseases are allowed to share similar hues (e.g., “influenza” and “upper respiratory tract” use hues of red). The iconic representation on the map allows for an initial comparison of diseases between different regions. Different scopes of comparison are supported by offering different ways of mapping data to color (see Figure 3.2.21):

1. global comparison – data values are normalized between 0 and the overall maximum number of cases per day
2. region-local comparison – data values are normalized between 0 and the maximum number of cases per day for all diseases *on a per region basis*

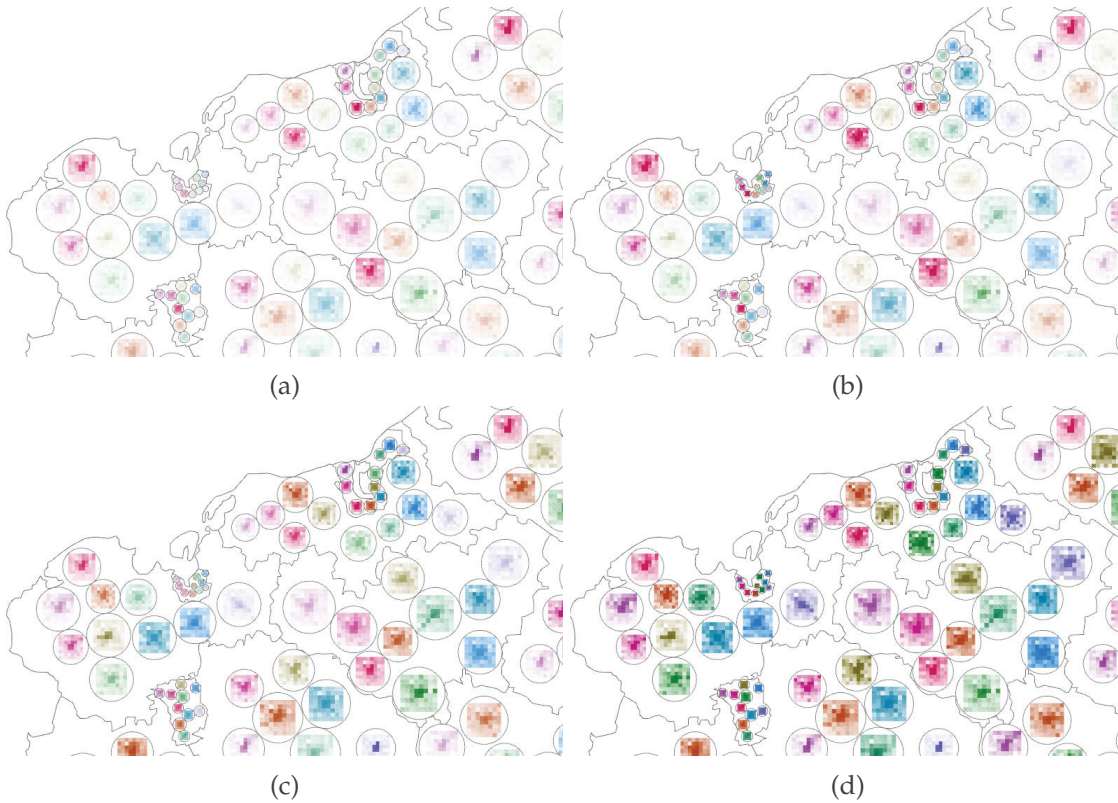


Figure 3.2.21: Different scopes of comparison of multiple diseases (denoted by different hues) in Mecklenburg-Vorpommern for the year 2000. (a) global comparison, (b) region-local comparison, (c) disease-local comparison, (d) local comparison.

3. disease-local comparison – data values are normalized between 0 and the maximum number of cases per day for all regions *on a per disease basis*
4. local comparison – data values are normalized between 0 the maximum number of cases per day *on a per disease and per region basis*

Global comparison can be used, for example, to find the disease and/or region with the highest number of cases at all. Region-local comparison can be used for comparing just the different diseases of each area to find similar trends between them. Disease-local comparison supports finding deviations in the temporal patterns, which might indicate local dependencies of a disease. Local comparison restricts the scope of interest to the analysis of the temporal evolution of each disease for each region.

The visualization described so far shows multiple diseases and multiple regions, but the hierarchy layout shows a single year only. To display multiple years, the layered approach described in Section 3.2.5 can be used. In the special case that just one disease needs to be shown, multiple layouts, each of which representing a different year, may be alternatively embedded into subregions.

Eventually, the discussion has arrived at a visualization that allows users to analyze temporal patterns of multivariate human health data in their spatial distribution on a map display. However, the focus of the visualization lies on communicating as much of the structure as possible. In this way, large time series are not handled sufficiently yet.

3.2.7 Conclusion

This section has introduced a new point-based layout paradigm specifically designed for the visualization of large hierarchies. Therefore, nodes are represented by points providing a very minimalistic view on the structure. Edges are blended in automatically in sparser regions of the visualization to support the readability of the structure. Based on the general layout scheme a family of point-based layouts was introduced that provides a variety of differently suited layout presets concerning the overall structure of a given hierarchy. For the visualization of attributes, different strategies have been discussed mainly based on color coding and using the third dimension.

Because of its fixed nature, even empty regions of the visualization are communicating information about the tree structure. In this sense, different presets were introduced supporting different kinds of hierarchies depending on their characteristics such as wide or deep trees. The fixed placement of the point-based layout has also proven useful for its adaption to irregular shapes allowing a direct embedding of a hierarchy into its spatial context. Furthermore, this fixed layout supports the comparison of different hierarchies which is an important advantage for the analysis of temporal changes in a hierarchy.

However, the visualization of multiple attributes is still difficult and mostly supported by showing them side by side using multiple views. This problem is mainly due to the utilization of points restricting the choices for visualizing attributes. In contrast, the following section focus on implicit techniques which are using area primitives and furthermore completely neglecting the rendering of edges.

3.3 A Design Space for Implicit Tree Visualization

There exists a wide variety of implicit visualizations to cope with the diversity of different hierarchies. Hence, it is important to provide them to the user so he can choose an appropriate technique for his task at hand. Yet, their provision often includes high implementation efforts. In this context, instead of implementing each of these techniques individually it may be more practical to provide a design space in which these design principles are made explicit as independent design axes. Then the implementation efforts are reduced to providing only the different choices of each design axis. Once such a design space is established, it may be used to:

- rebuild existing techniques by plugging appropriate design choices together,
- create new visualizations by identifying formerly unknown combinations of different design choices, and
- adapt visualizations to the specifics of a given data set by altering only some of the design choices.

In this way, a design space may not only be used to access the different visualizations but also as a means to steer the analysis. On this basis, a design space for implicit tree visualizations is described in this section⁵.

In the following, first the design space is introduced by extracting the most common design axes. Then an implementation of this design space is proposed allowing not only the exploration of the design space but also a rapid visualization prototyping. Finally, the design space is put in exemplary use by introducing new and altered visualizations specifically tailored according to the different aspects.

3.3.1 Design Space Definition

By surveying the existing techniques, the following 4 independent design axes are identified and used to span the design space:

- **Dimensionality:** 2D or 3D
- **Node Representation:** graphics primitives like rectangles, circles or spheres
- **Edge Representation:** inclusion, overlap, adjacency
- **Layout:** subdivision, packing

These 4 axes capture the major design decisions one has to make to derive an implicit tree visualization technique. For the practical realization of this design space each axis may have additional parameters capturing a more fine grained subspace of the overall design space. Such parameters are for instance surface properties such as color or texture (node representation parameters), or the amount of overlap (edge representation parameter).

⁵Parts of this section have been published as "The Design Space of Implicit Hierarchy Visualization : A Survey" 2011 in the IEEE Transactions on Visualization and Computer Graphics [SHS11a].

In this sense, a design space should fulfill some important properties: completeness, consistency and practicability. The first two properties demand for the design space to contain all existing techniques while not allowing to create techniques violating basic design principles such as disconnected visualizations. Whereas the third property demands for a more practical use besides the mere need for classification such as interactively deriving new visualizations. The proposed design space is general enough to capture the majority of existing implicit tree visualizations while being concrete enough to allow a practical implementation as will be discussed in Section 3.3.2. For a detailed discussion according to these properties as well as for a classification of existing techniques, the interested reader is referred to the paper and to [Sch10].

The individual axes of the design space and their parameters are discussed in the following sections.

3.3.1.1 Dimension

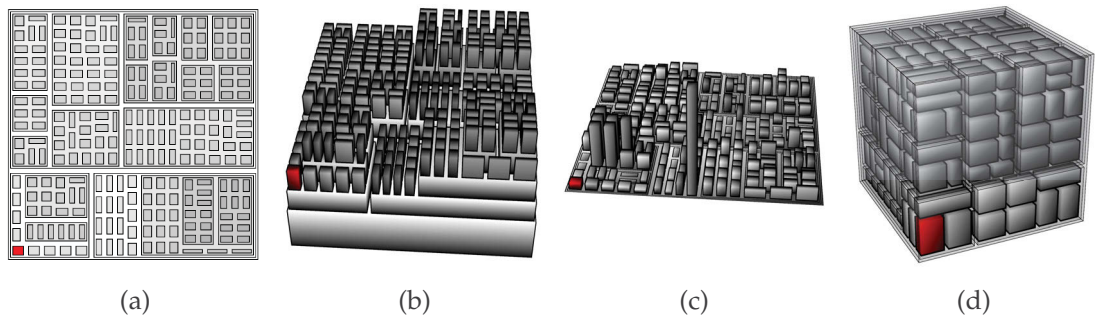


Figure 3.3.22: Different ways for changing a 2D Treemap into 3-dimensional counterparts: a) a standard 2D nested Treemap, b) an extruded Treemap assigning each node of the same level an equal depth, c) an extruded Treemap assigning each leaf a depth proportional to an attribute, d) a real 3-dimensional extension resulting in a subdivision of 3D cuboids. The same node has been highlighted in all visualizations.

2D representations are the most common techniques found in the literature as compared to 3D techniques they contain no occlusion and thus do not need additional interaction efforts to analyze the hierarchy. Yet, 3-dimensional techniques offer additional visual variables (z-coordinate and height) to use for the visualization. For the utilization of these additional variables generally two different approaches are followed: extruding a 2D technique or using a real 3-dimensional layout.

There are basically two different reason for extruding a 2D technique. Foremost adding a height to all primitives may support a better readability of the layout and thus of the structure as levels of the hierarchy are easier to distinguish. Especially for Treemaps the containment relation is transformed to an adjacency relation simplifying the estimation of the nesting level. Some examples for this extrusion are Steptrees (similar to Figure 3.3.22b) [BCS04] or Information Pyramids [AWP97]. The second reason is the possibility to integrate additional node attributes in the visualization. One example is the CodeCity (similar to Figure 3.3.22c) [WL07] mapping a numerical attribute directly on the height of the primitives. While these extruded techniques may contain additional information accessible by a simple rotation an orthogonal view from the top still reveals

the basic 2D layout. In this sense, these techniques are also referred to as 2 1/2D visualizations [TJ92].

The second class are real 3D visualizations that are based on inclusion to communicate the parent-child relation. Therefore, they rely on extensions of subdivision or packing layouts for volumes instead of areas. There are only some examples following this idea such as Information Cubes [RG93] or TreeCubes [TON03]. As can be seen from Figure 3.3.22d these techniques suffer from extensive occlusion and enforce the usage of transparent node primitives.

3.3.1.2 Node Representation

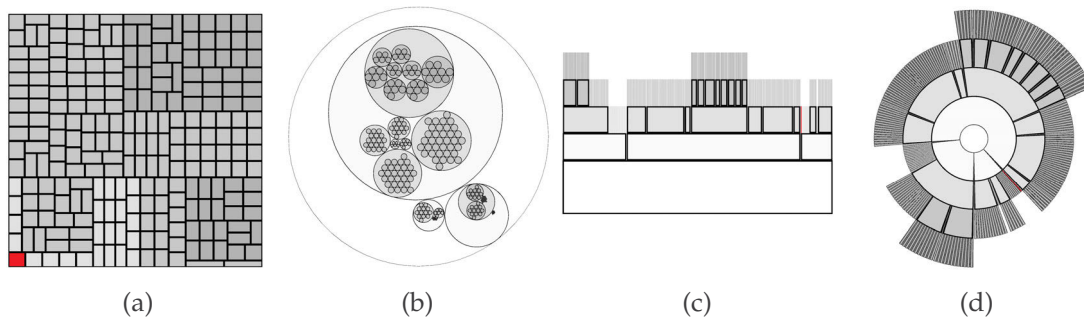


Figure 3.3.23: Examples for the usage of different node representations: a) a Treemap using inclusion and rectangles, b) a Circular Treemap using inclusion and circles, c) an Icicle Plot using adjacency and rectangles, d) a Sunburst using adjacency and circle sections. The same node has been highlighted in all visualizations.

The most common shapes used are rectangles or cuboids because of their simple geometry allowing easy layout calculations and interactive rendering frame rates. Using other shapes has again basically two different reasons concerning the readability of structure and attribute values.

Regarding the structure, tightly nested rectangles as in the case of Treemaps do not leave any space except for the leaves. Thus the underlying structure is hard to perceive as exemplified in Figure 3.3.23a. By switching to other shapes such as circles which cannot be packed as tightly (visible in Figure 3.3.23b) empty space and borders emerges between them. In this way, the structure becomes better visible yet at the cost of the space utilization. Another example concerns the Icicle Plot. As hierarchies tend to grow in the width with increasing depth more space is required for deeper levels. Yet, the Icicle Plot assigns equal space for each level. A switch to a circle reduces this problem as with each additional ring/level the available space increases turning the Icicle Plot into a Sunburst (compare Figures 3.3.23c and 3.3.23d).

Concerning the readability of attribute values, it has to be noted that often values are mapped to the area of the used primitives. Yet, many layouts result in rectangles with awkward aspect ratios making a comparison of their areas very difficult. Again, a switch to circles which have a fixed aspect ratio of 1 may reduce this problem. Besides a direct mapping of attribute values (especially categorical) on the shape, another reason for different shapes is to give each individual node a recognizable shape.

Another facet concerning the node representation is the ambiguity of a 3D extension of 2D shapes. Circles may be extended to spheres or cylinders whereas rectangles may be extended to cuboids or pyramids. To make the different design axes truly independent these ambiguities have to be considered by providing different representations such as rectangle/cuboid and rectangle/pyramid.

Furthermore, a node representation is not only described by its shape but by additional visual variables (surface properties) such as color and texture. These may be used to encode attributes or to increase the perception of the structure by mapping structural properties of nodes. Furthermore, an adaptation of surface properties may be used to guarantee the visibility of important nodes by using Ghost Views [LS08a] increasing the transparency of occluding nodes.

3.3.1.3 Edge Representation

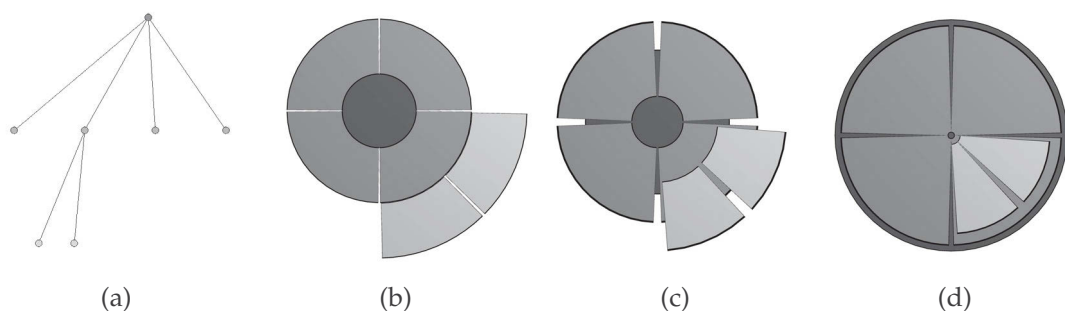


Figure 3.3.24: Examples for different edge representations: a) an explicit representation using straight lines, b) an implicit representation using adjacency, c) an implicit representation using overlap, d) an implicit representation using inclusion.

In implicit visualizations edges are not drawn explicitly but reflected by geometrical relations between the node shapes. The most common relations found in the literature are inclusion, overlap and adjacency. Inclusion (Figure 3.3.24d) has the advantage of being confined to the space assigned to the root node. In contrast, adjacency (Figure 3.3.24b) grows outwards with each additional level. Yet, as each level is explicitly visible when using adjacency more of the hierarchical structure may be perceived compared to inclusion where most of the space is assigned solely to the leaves. Besides these two choices overlap (Figure 3.3.24c) tries to find a compromise between both by restricting the outwards growth yet still revealing more of the structure.

In this way, these relations are also differently suited for instance for deep trees. When using adjacency the available space for each level shrinks drastically with increasing depth. However, for inclusion the depth of the tree only slightly affects the visualization.

In the same manner, these relations differently support the visualization of attributes. As inclusion mainly shows the leaves only their associated attributes may be reflected. Instead with adjacency the attribute values of all nodes can be visualized. Furthermore, adjacency (as well as overlap) introduces another visual variable for mapping a node's attribute value: the *distance* between a node and its children which directly influences the

height (2D) or depth (3D) of the node's primitive. Actually, it is this property that was exploited for mapping attributes to the third dimensions as discussed in Section 3.3.1.1 and exemplified in Figure 3.3.22c.

3.3.1.4 Layout

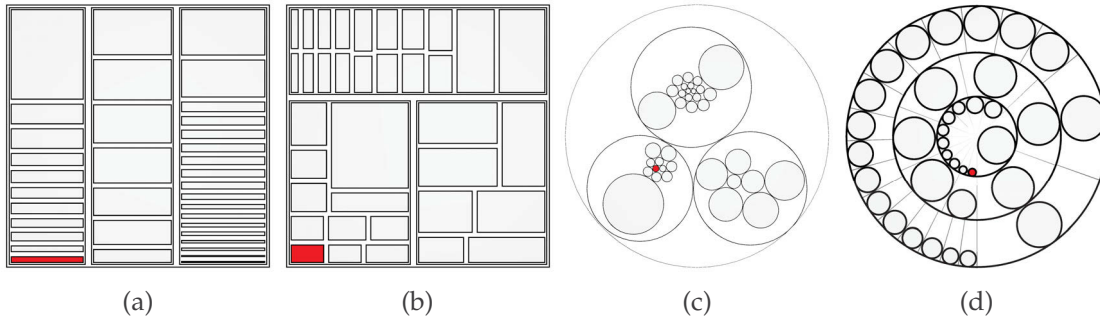


Figure 3.3.25: Examples for the usage of different layouts: a) a Treemap using a slice-and-dice layout, b) a Treemap using a squarified layout, c) a Circular Treemap using a circle packing layout, d) a Circular Treemap using a radial slice-and-dice layout and embedding circles in subdivided circle sections. The same node has been highlighted in all visualizations.

Generally, all layouts can be classified as either subdivision or packing algorithms. Subdivision techniques are typically applied recursively top-down starting with the complete display space for the root. They subdivide the given space into regions and assign them to the children of the current node. In this way, they are able to completely utilize the given display space. Most implicit techniques are using a subdivision layout.

Contrary, packing algorithms are often working bottom-up starting with the leaves at the deepest level in the hierarchy. They then recursively, tightly pack siblings defining the space of their parents. During this packing often empty spaces remain providing a better view on the structure below similar to using non rectangular shapes. While the packing problem in general is NP-complete [GJ79] there are some fast heuristics such as the packing of rectangular shapes used in the Data Jewelry Box [IKIY02, IYIK04] or the circle packing used in Pebble Maps [Wet03] (similar to Figure 3.3.22c).

Besides assigning display space to all nodes of the hierarchy the layout often has to fulfill additional constraints. The most important constraint concerns the size of the assigned space which often has to be proportional to a specific node attribute. In this context, maintaining an aspect ratio of around 1 is also an important aspect to allow a simple comparison of the mapped attribute values (compare for instance Figures 3.3.25a and 3.3.25b). In case of nodes having a spatial reference, an additional constraint may put restrictions on the positions of each node's shape which is done for instance for the Spatially Ordered Treemaps [WD08]. Further constraints for a "perfect" layout were introduced in [Wat05] concerning the temporal stability of layouts for time-varying hierarchies.

While the different design axes should be independent the layout often already defines the shape of the nodes. This especially applies to the subdivision algorithms whereas

packing algorithms in principle may handle any shape. Thus, to guarantee an independent choice of layout and node representation the chosen shape has to be scaled down to fit into the calculated space as depicted in Figure 3.3.25d.

3.3.1.5 Mixing Design Choices

The majority of the existing implicit visualization techniques rely on a single choice per design axis. Yet, such a restriction is not a necessity and would just limit the expressiveness of the design space. Especially, as there are already some approaches mixing the design choice to accentuate specific nodes, often the leaves, a local way to specify such design decisions is a useful property of the design space. Examples for such mixed visualization can actually be found for all design axes. 2 1/2D Treemaps [TJ92] are representing inner nodes with 2D rectangles while accentuating leaves by 3D primitives similar to Figure 3.3.22c. For the 3D Circular Treemap (see Figure 3.3.27d) [WWDW06] different node representations are used: cylinders for the inner nodes and hemispheres for the leaves. Beamtrees [vHvW02] in both variants, 2D and 3D, mix different edge representations. In the 2D case it uses overlap for the inner nodes and inclusion for the leaves. Quantum Treemaps [Bed01, BSW02] utilize different layouts. Inner nodes are laid out by subdivision algorithms whereas leaves are packed together.

3.3.2 Implementation of the Design Space

Up to this point, the proposed design space solely allows a classification of existing techniques according to the choices they make. In this way, it already supports the determination of possibly unknown combinations of different design choices. Yet, a practical implementation of the design space does not only allow the user to explore and access these unknown visualizations but also to adapt existing techniques to specific aspects of their given hierarchies. Therefore, a rapid visualization prototyping software was implemented allowing the definition and alteration of new and existing visualizations. The developed prototype is available online as a java applet⁶. For the 3D rendering and scripting support it relies on JOGL and Groovy, respectively.

In the following, first a general overview of the architecture with its basic components is given. Then, the actual interactive mapping of design decisions to the nodes of the hierarchy is described. Therefore, two different ways are discussed allowing the user to interactively change and specify this mapping.

3.3.2.1 General Architecture

For the rapid prototyping tool to be able to cope with the design space as well as multiple attributes and dynamic hierarchies it consists of six major components:

A modular and expandable software architecture that abstract the different design axes as interfaces. For all axes, each design choice can be defined by a specific implementation. This enclosure allows new primitives, layouts and other design choices to be

⁶<http://vcg.informatik.uni-rostock.de/~hs162/itvtk/start.html>

implemented on demand as Java classes or Groovy scripts.

Multiple data importers and exporters allowing the user to load his hierarchies and thus to design visualizations for his needs. Important supported formats are for instance TreeML [FP03] and the CSV-format used by ManyEyes [VWvH⁺07]. As it was a first use case for Treemaps, any directory tree of the hard drive together with attributes concerning the space consumption can be loaded and also exported as a TreeML for saving and sharing. While there is no standard format for dynamic hierarchies such a data set may be loaded either as a sequence of separate TreeML files or by attaching the different time points and associated values as consecutively numbered attributes (e.g., size0, size1, ...). Internally, dynamic hierarchies are represented by a supergraph over all time points in which the occurrences of each node are captured in associated attributes.

A flexible scripting capability to calculate and derive node attributes such as its depth, its number of children, or the size of its subtree as well as the minimum, maximum and average of associated numerical attributes. Further attributes such as a node's Strahler number [ADD⁺04] or its frequency [AERD10] may be calculated by executing Groovy scripts on the loaded hierarchy. These attributes can be used for instance to calculate the size of a node primitive or to restrict certain design decision to specific nodes.

Multiple data operators to modify the structure of the hierarchy prior to the visualization. Therefore, two kinds of operators are provided:

- **Filtering operators** allow the removal of nodes according to their attributes, for instance to filter out uninteresting nodes or unusually large or small subtrees as discussed in [HDM98].
- **Cloning operators** allow the duplication of nodes to simultaneously visualize multiple attributes or different time points in the sense of glyphs (see Section 3.3.3.2) or small multiples (see Section 3.3.3.4). Therefore, cloning may be performed anywhere in the hierarchy for single nodes or whole subtrees.

A tuple-based visualization design realizing a certain visualization as a sequence of tuples. Each tuple (*axis, choice, feature, function, condition*) represents a single design decision according to a specific design axis. Hence, to specify a certain visualization at least one tuple has to be defined for each design axis. The individual elements of a tuple describe the basic design (axis, choice), the parametrization (feature, function) and the scope (condition) of the design decision:

- The **basic design** selects a concrete **choice** for one of the 4 design axes – dimensionality, node representation, edge representation and layout – as discussed in Section 3.3.1. For instance, using rectangles or spheres for the node representation axis, or slice-and-dice for the layout axis.
- The **parametrization** specifies more fine grained details for the chosen choice by selecting a concrete **feature** and a **function** for calculating its value. Such a parametrization can be an exponential mapping of a node's attribute (function) on the size

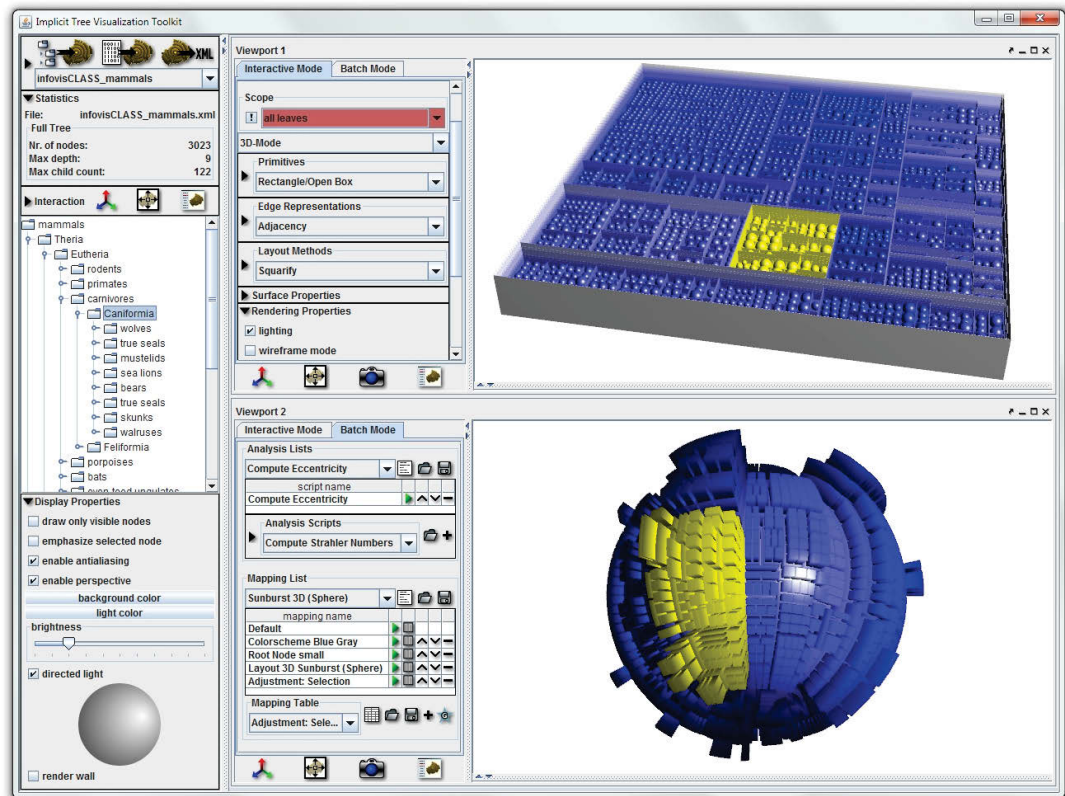


Figure 3.3.26: A snapshot of the Implicit Tree Visualization Toolkit showing the mammals subtree of the InfoVis 2003 contest classification hierarchy "A". The left side shows statistics of the hierarchy, a regular tree view and controls for global rendering properties. The right side shows two viewports allowing the simultaneous and independent creation of new implicit visualization prototypes. Each viewport contains its own set of interaction elements to alter and adjust the design properties either by a point-and-click interface (top viewport) or by a script and table-based interface (lower viewport). The top view shows a mixed visualization consisting of open boxes for inner nodes and spheres for leaves. This mixed design grants the visualization an ordered look while slightly accentuating the leaves. The bottom view shows a spherical Sunburst which is discussed in Section 3.3.3. In all views the "caniforms" subtree is highlighted.

(feature) of its primitive or the utilization of a node's depth (function) on its color (feature).

- The **scope** restricts the nodes for which to apply this design decision by specifying a **condition**. Examples using such a condition are the switch of the representation to hemispheres for leaves or accentuating nodes with high attribute values by only adapting their color.

A visualization is finally realized by consecutively evaluating its sequence of tuples.

Multiple coordinated views holding on the one hand statistical information of the hierarchy and node attributes (minimum, mean, maximum) essential for the mapping as well as global properties such as the projection and light direction for 3D rendering. On the other hand, they allow to compare different visualizations side by side as it shown in Figure 3.3.26. Each visualization has its own set of interaction elements to adapt its design properties independently according to two different modes: an interactive mode for a mere exploration of the design space and a batch mode to support a more descriptive design of new visualization prototypes. Both modes are discussed in more detail in the following sections. At any time each visualization may be cloned providing another view that may be rotated, zoomed and panned in sync. Furthermore, the cloned view may be used to branch the design process while maintaining a copy of the current visualization as a reference.

3.3.2.2 Interactive Mapping Mode

The interactive mapping mode provides direct access to the design space in a point-and-click fashion wrapping the tuple-based mapping in a seamless manner. This wrapping allows the user to manipulate a given visualization and see the immediate effects of his manipulations. In this way, the user is able to explore and get familiar with the design space before switching to the more complex but also more descriptive batch mapping mode.

Therefore, the GUI provides a multitude of different controls to:

- choose a predefined visualization (preset) as a starting point of the exploration of the design space,
- select nodes for inspection and alteration in the tree view as well as directly in the current visualization,
- restrict the scope of manipulations of the visualization design, for instance to all nodes, to all leaves, to the selected node only or its leaves etc., and
- change the design choices and possible additional parameters.

For the general layout of the visualization, surface properties only play a minor role compared to a node's shape. Hence, the node representation axis is split into three axes: the node primitive (rectangle, circle etc.), the surface properties (color or texture) and the rendering properties (lighting etc.). An example featuring this interactive mapping

3 Novel Techniques to Visualize Graphs

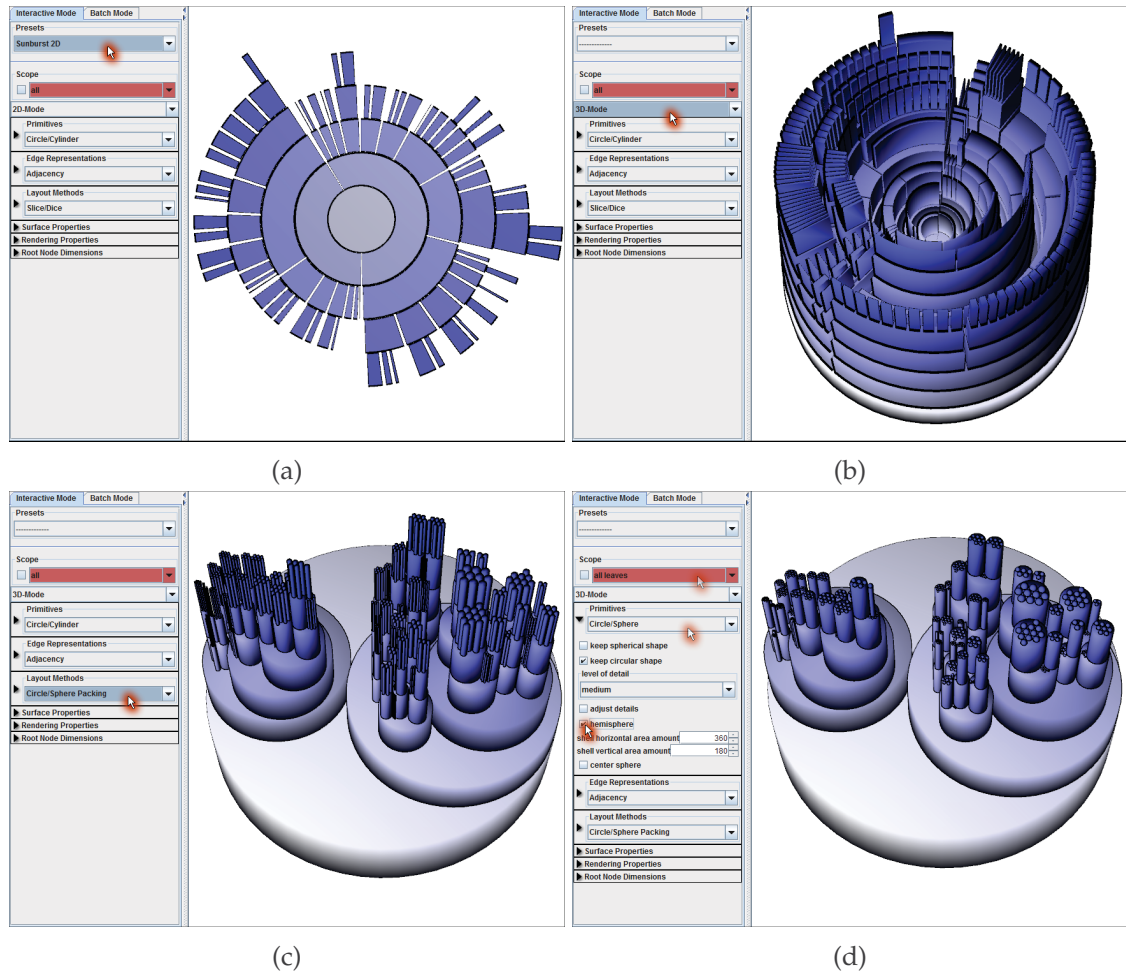


Figure 3.3.27: Transforming a 2D Sunburst into a 3D Circular Treemap using the interactive mapping mode. Step 1: a 2D Sunburst is selected as a preset. Step 2: the dimension is changed to 3D resulting in a 3-dimensional extruded Polar Treemap. Step 3: the layout is changed from a radial slice and dice subdivision to a circle packing resulting in stacked cylinders. Step 4: all leaves are exchanged by hemispheres finally turning the visualization into a 3D Circular Treemap.

mode is given in Figure 3.3.27 showing the interactive switch from a 2D Sunburst to a 3D Circular Treemap in 4 steps:

1. From the available presets the 2D Sunburst is selected as an initial visualization to modify (Figure 3.3.27a).
2. Then the dimensionality axis is changed from 2D to 3D resulting in an extruded Polar Treemap [Joh93] (Figure 3.3.27b). This change has two effects on the visualization. First, by switching the dimension the node representation changes from a 2D circle to a 3D cylinder. Therefore, to maintain the adjacency relation the children are now stacked on at least one side of the cylinder. The default side is the top of the cylinder whereas the coat of the cylinder is also a viable option, as used for instance by Beamtrees. And second, the layout switches from a 1-dimensional slice layout for the Sunburst to a 2-dimensional slice-and-dice layout subdividing the circular top side of the cylinder resulting in a Polar Treemap. As this subdivision creates circle sections the final node representations result in cylinder sections.
3. By switching the layout in the third step to a packing now full cylinders are stacked on top of each other (Figure 3.3.27c).
4. To finally rebuild the 3D Circular Treemap the leaves have to be changed into hemispheres (Figure 3.3.27d). Therefore, first the scope is restricted to all leaves (signalized by the upper arrow). Then their node representation is switched to spheres (center arrow). Handling spheres as hemispheres is captured as an additional parameter of this primitive and checked at last (lower arrow).

Each of these interactions is transformed into a tuple on the fly and captured in a list. This list does not only provide a selective history of one's exploration of the design space allowing the user to access and alter all decisions. It also poses the basis for transforming the exploration results into a first exchangeable visualization prototype as discussed next.

3.3.2.3 Batch Mapping Mode

The batch mapping mode offers a more detached way to utilize the design space by providing explicit access to the tuple-based mapping. This explicit access to the tuples allows a more exact specification of visualization prototypes instead of a pure exploration compared to the interactive mode taking both structure and associated attributes into account.

Therefore, this mode provides control elements to:

- load and execute scripts for deriving new attributes (e.g., Strahler numbers or attribute derivatives) to be used in the mapping,
- create and adapt mapping tuple lists to derive high level building blocks on top of the basic design choices offered by the design space,
- combine building blocks into sequences and execute them step-by-step allowing the debugging and modular definition of visualization prototypes, and

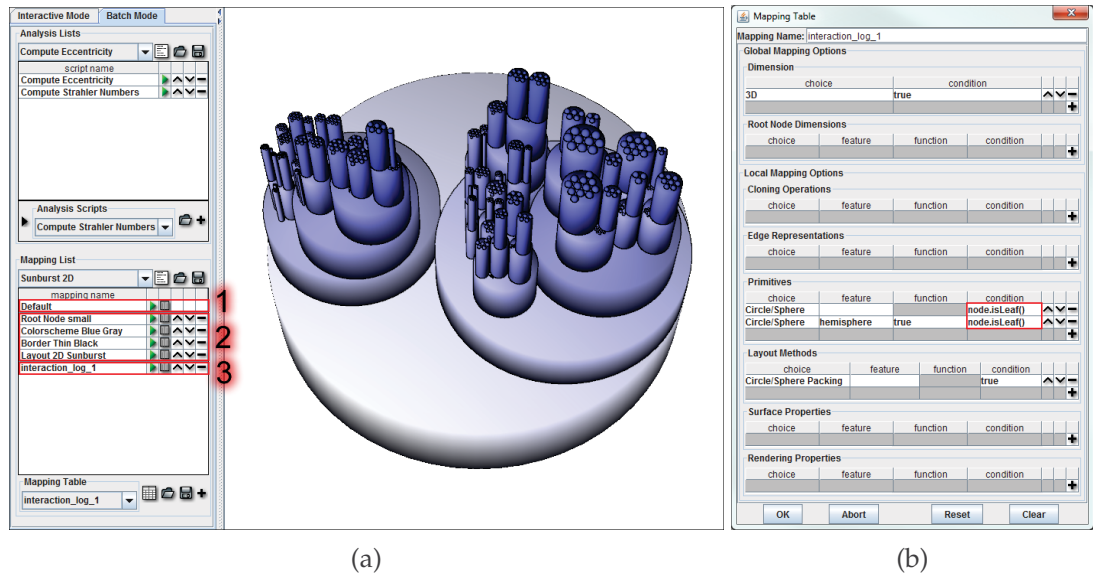


Figure 3.3.28: The transformation of a 2D Sunburst into a 3D Circular Treemap captured in the batch mapping mode. a) the batch mode allows the step by step execution of scripts and mapping listings. Scripts allow the calculation of attributes to use for the mapping. Each mapping listing captures specific modular designs. b) An example listing that captures the modifications as performed interactively in Figure 3.3.27 as tuples visualized in multiple tables for all design axes.

- load and store individual building blocks as well as complete sequences.

By splitting up the visualization design in a sequence of higher level building blocks the user is able to capture often recurring patterns in visualization designs. For instance, most Treemap variants are 2D visualizations using rectangles as node representation and inclusion as edge representation while only differing in the used layout. The color coding is just another example as it only influences the appearance of the visualization. A separation into multiple building blocks then allows the switch of the color coding without the need to adapt the whole visualization design. Furthermore, different color codings may be created to improve the readability of the structure, to communicate attribute values, or even to highlight specific nodes.

An example for such a sequence of building blocks is given in Figure 3.3.28a based on the interactive transformation of a 2D Sunburst into a 3D Circular Treemap as performed in the previous section. Here, the default building block (highlighted with 1 in Figure 3.3.28a) ensures an equal initial state for the execution of any mapping. As the user selects the preset for the 2D Sunburst its sequence of building blocks is loaded which in this case consists of 3 blocks a color coding, an adaptation of the area assigned to the root node and the actual Sunburst layout (highlighted with 2 in Figure 3.3.28a). All changes to the visualization performed in the interactive mode are captured within the last building block providing a log of all his actions (highlighted with 3 in Figure 3.3.28a).

When the user has finished his exploration he can then look into this log to derive a visualization design. The interaction log for the transformation is exemplified in Figure 3.3.28b. Each list of mapping tuples is presented by individual tables for all design decisions providing access to all elements of the different tuples. For both, decision and

function, a structure and attribute dependent formula may be inserted. For instance, the restriction of the scope to leaves for changing the node representation to hemispheres is reflected by using the structural property *isLeaf* (highlighted in red in Figure 3.3.28b). But more complex formula are also possible and supported by a dialog providing information of available variables and specific build-in functions depending on the expected return type.

Together with the powerful scripting functionality this implementation provides much freedom for the exploration of the design space and the (re-)creation of new visualization techniques. Actually, all examples of this section have been created using this tool and further examples for using it to create and adapt visualizations to cope with different aspects such as deep or dynamic trees are discussed in the next section.

3.3.3 Using the Design Space to Adapt to different Aspects

As discussed in Section 2.2 each aspect of a hierarchy – its structure, associated attributes as well as its spatial and temporal reference – poses different requirements to the visualization. Often a single visualization is not able to handle all of these requirements equally well. Having a design space not only allows access to existing techniques but also the combination, adaptation and creation of whole new visualizations that may be better suited for the different aspects.

In the following, some examples are given showing how to put the design space in use. As the design space provides limitless possibilities for combinations and mixing of design choices the following discussion⁷ is far from being exhaustive but shows some first ideas.

3.3.3.1 Structure

Important structural features of a hierarchy are the width (number of children) and the depth (distance to the root) of the nodes.

Treemaps and most inclusion based visualizations in general tend to adjust quite well to either wide or deep hierarchies as they focus mainly on visualizing the leaves. Most of the display space is therefore reserved only for their display while minimizing the space used for inner nodes. And in case no borders are included, the number of ancestors of a node have thus nearly no influence on the space distribution. In this way, they mostly depend on the number of leaves independently of the inner structure. Yet, by neglecting the inner nodes the readability of the overall structure is severely affected. Hence, these techniques are best suited for a compact representation of leaves and their attributes.

Contrary, adjacency based visualizations such as Icicle Plots showing all nodes better support the visibility and thus the readability of the structure. Yet, as they are drawing all nodes their scalability tends to be more affected by both wider and deeper trees. With each additional depth level the available space for all other levels is reduced. Whereas the number of nodes per level is bound by the space assigned to each level, in case of Icicle Plots by the maximum of width or height of the display space.

⁷Most parts of this discussion are beyond the scope of the originally published design space and show some novel visualization prototypes.

In either case, adaptations to the chosen visualization are necessary to better reflect the structure or to cope with the width and depth of the hierarchy. Yet, especially for Treemaps multiple variations are already known, such as Cascaded Treemaps [LF08] which exchange a strict inclusion relation by overlap or Ellimaps [ONG⁺07] that switch the node representation from rectangles to ellipses to provide more space for the inner nodes. Hence, the following discussion focuses on adjacency based techniques which poses a good starting point for designing a structure focusing visualization. In this case, adaptations along the 4 design axes should target an improvement concerning the width and the depth of the hierarchy. As nodes are placed on different depth levels according to the adjacency relation, changes on the node representation, dimensionality and layout axes mostly influence the handling of the width. Whereas changes on the edge representation axis may yield better results concerning the depth.

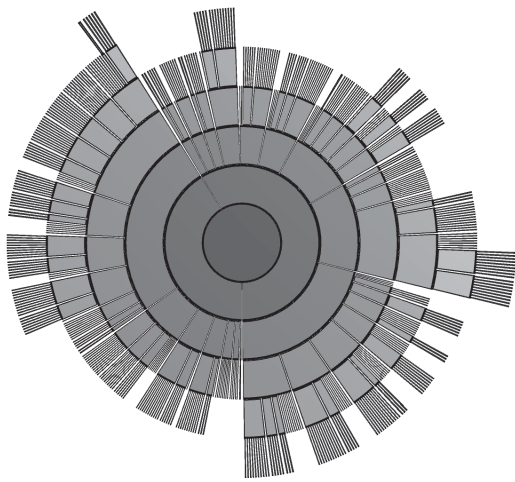
Node representation: One has to note that the Sunburst actually represents a first adaptation of the Icicle Plot to increase the available space with increasing depth by switching the node representation to circles. Basically, this switch results in an increase of the layout space from a single side of a rectangle to the complete circumference of a circle. In this way, the Sunburst already better supports wider hierarchies than the Icicle Plot by providing more space on deeper levels.

Dimensionality: Another step to increase the layout space of the root node is to switch the visualization from 2D to 3D at the price of additional interactions to navigate and explore the visualization. As discussed for the node representation there are multiple ways to extend a 2D primitive to 3D. In [SKW⁺07] a 3-dimensional Sunburst (similar to Figure 3.3.29b) was introduced on the basis of a cylinder already providing more space. However, using a cylinder only allows the visualization to grow horizontal.

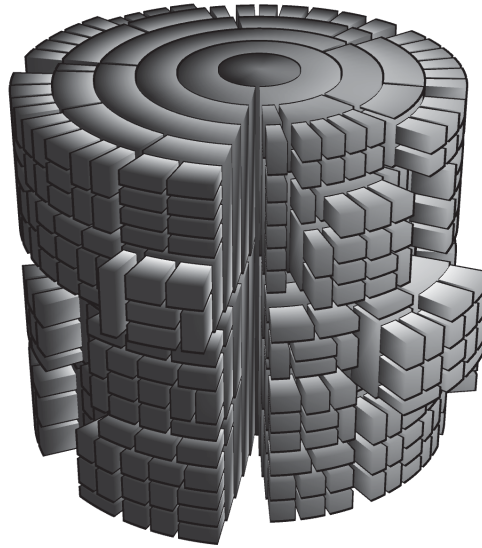
By switching to a sphere the visualization may use the whole 3-dimensional space and thus provides more space to deal with wider hierarchies. This novel visualization is exemplified in Figure 3.3.29c. Similar to Treemaps this visualization accentuates the leaves while covering most of the inner nodes. Yet, it still allows to estimate the depth of the hierarchy.

A compromise between the visibility of the inner structure and the scalability of the visualization can be achieved by restricting the layout area on the surface of the sphere to a small, horizontal belt as shown in Figure 3.3.29d. This restrictions enables the user to peek into the visualization from above and below.

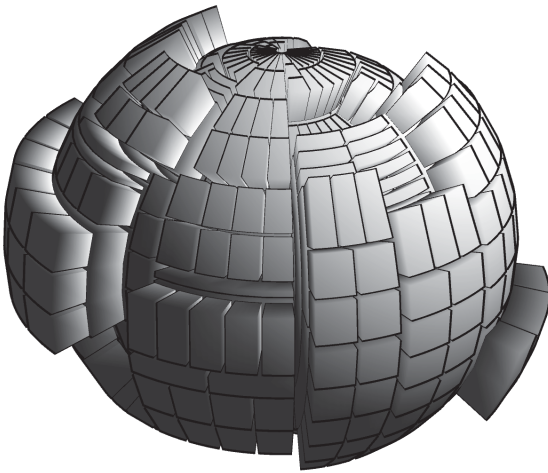
Layout: Besides providing more space for laying out the hierarchy, another problem concerns extremely wide parts of the hierarchy such as nodes having hundreds of children. These wide parts affect the overall readability of the structure as the remaining nodes are pushed close together and thus will appear very dense and undiscernible. One way to solve this problem is to use the filtering operator to remove such nodes similar to using the automatic folding introduced in [HDM98]. This approach has the advantage that it does not need an adaptation of the visualization and thus is compatible with any visualization. Yet, it is based on completely removing information that may become important during the analysis. Contrary, these negative effects can also be lessened by



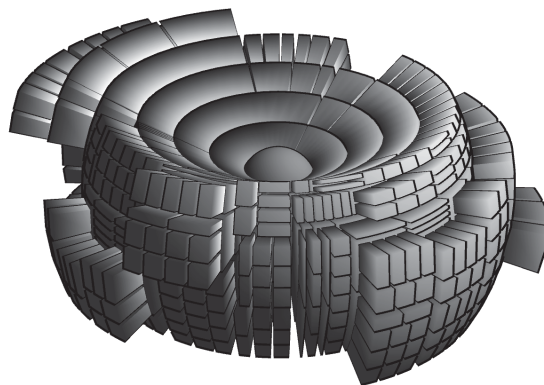
(a)



(b)



(c)



(d)

Figure 3.3.29: 2D and 3D variants of the Sunburst. a) A standard 2D Sunburst. b) A 3D Sunburst based on cylinders (variant "Cylinder"). c) A 3D Sunburst based on spheres (variant "Sphere"). d) A 3D Sunburst based on a sphere but restricting the used space to a narrow, horizontal belt (variant "Wheel").

adapting the layout, for instance by reducing the layout weights for these subtrees while maintaining every information.

Edge representation: Deep and narrow parts of the hierarchy can severely reduce the available space of every depth level. Again, such parts of the hierarchy may again be removed using the filtering operator. Yet, to maintain most of the information, it may be desirable to combine the high compactness of a Treemap with an adjacency based visualization. Therefore, the design space offers multiple ways on the basis of the edge representation:

- adapt properties of the adjacency relation,
- exchange adjacency by inclusion, and
- exchange adjacency by overlap.

All three strategies are exemplified for a 2D and 3D ("Wheel") Sunburst in Figure 3.3.30 by reducing the display space of nodes with only a single child (highlighted in red).

Adapting the adjacency relation: As discussed in Section 3.3.1.3 for the adjacency relation the distance between each node and his parent can be defined independently. The first strategy builds upon this variable distance to reduce the influence of narrow parts on the overall depth. Nodes having only a single child are therefore assigned a low distance pulling their children closer to them. This reduction of the distance results in a more compressed visualization (compare for instance Figure 3.3.30a and 3.3.30b) while still maintaining the visibility of all nodes. However, without using color it may become difficult to discriminate modified from normal nodes. Furthermore, this strategy neglects the possibility of a further mapping of attributes on the distance.

Utilizing the inclusion relation: Replacing the adjacency relation by inclusion for these narrow nodes as shown in Figure 3.3.30c provides the clear distinction between both types of nodes. Yet, either the number of possible inclusions or the border width is limited by the available space as with each inclusion the space for the child node gets smaller. So either the children and their children are assigned much smaller spaces because of the borders or their parent nodes become invisible because they are overplotted by their children.

Utilizing the overlap relation: As overlap represents a compromise between adjacency and inclusion it may be used to combine the advantages of the previous two strategies. A result showing its use is depicted in Figure 3.3.30d. The overlap of node primitives clearly depicts the difference between modified and normal nodes. Together with the outwards growth of the Sunburst it is possible by using overlap to maintain the size of the node shape with each overlap step solving the problem of the inclusion based strategy. Furthermore, overlap allows the simultaneous mapping of other attributes to the variable distance.

Because of the occlusion in 3D the results appear different compared to the 2D case especially for overlap. Whereas in 2D child shapes are drawn on top of their parents' shapes, in 3D the children are drawn within their parents. This rendering necessitates the

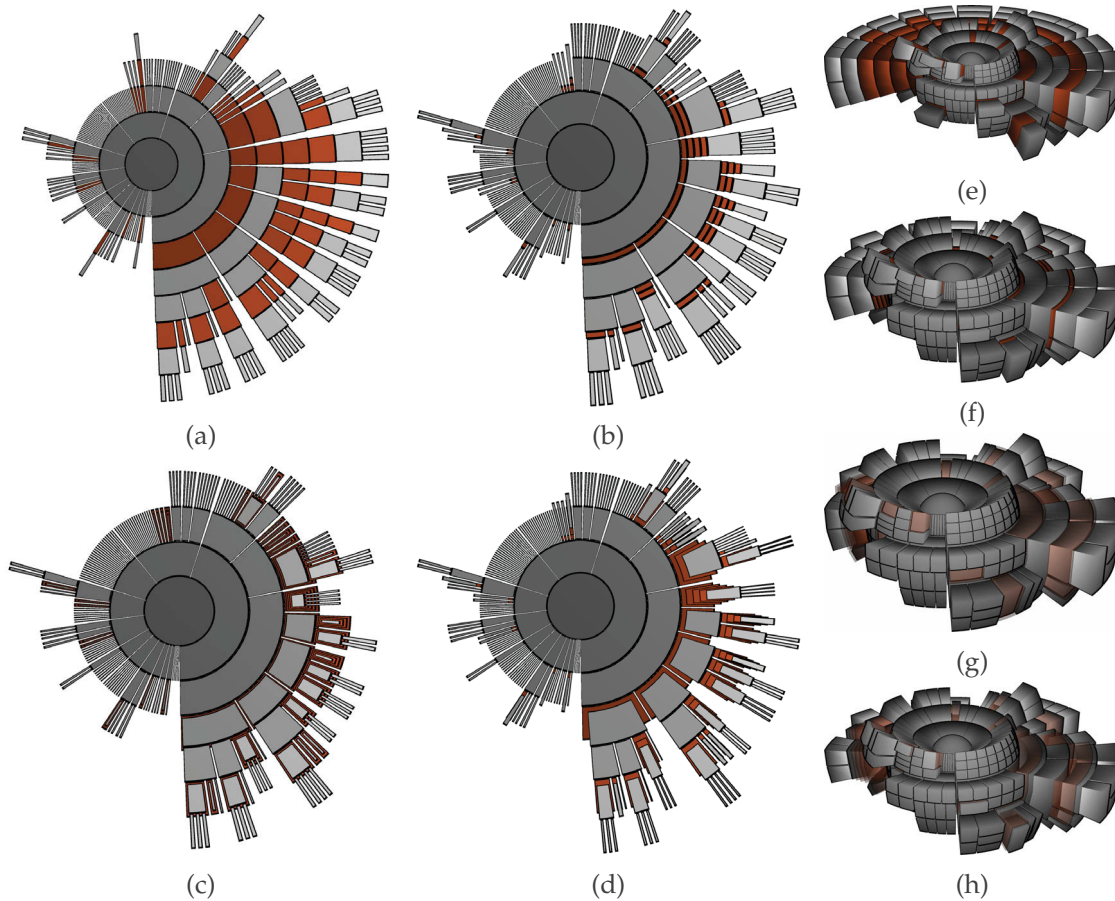


Figure 3.3.30: Different strategies to reduce the visual space assigned to deep narrow parts of the hierarchy. a) A normal 2D Sunburst using only adjacency and equal distances between nodes and their parents. b) A compressed 2D Sunburst using only adjacency but assigning different distances to narrow parts. c) A compressed 2D Sunburst using inclusion for narrow parts. d) A compressed 2D Sunburst using overlap for narrow parts. e) A normal 3D Sunburst (variant "Wheel") using only adjacency and equal distances. f) A compressed 3D Sunburst using adjacency and assigning different distances to narrow parts. g) A compressed 3D Sunburst using inclusion for narrow parts. h) A compressed 3D Sunburst using overlap for narrow parts. Nodes with only a single child (narrow parts) are highlighted in red.

use of transparency to make the overlapping nodes more visible. Yet, still the modified inner nodes appear to be much more in focus than their children. In this case, following the first strategy may be a more viable option.

While these strategies are discussed for compressing nodes with a single child the basic idea can be easily generalized for the first and third strategy (there are no intermediate steps for the second strategy). Therefore, the width of a node may be mapped inverse proportional to either its distance to its children (adjacency strategy) or to the overlap factor (overlap strategy). Furthermore, this approach is not limited to deaccentuate narrow parts of the hierarchy but may be used as a means for deaccentuation in general by calculating a degree-of-interest for each node as will be discussed in Section 4.3.

3.3.3.2 Attributes

The four design axes already provide some basic choices for the visualization of attributes, such as mapping them on the third degree (dimensionality), on color or shape (node representation), on the variable distance (edge representation) or on the size (layout). However, the user is often interested in more than one attribute. While it is possible to map multiple attributes on different visual variables their comparison remains difficult. Hence, a common way is the embedding of glyphs and charts allowing the simultaneous visualization of multiple attributes in a similar manner.

Comparing common charts such as product plots with implicit tree visualizations reveals that they share similar design decisions concerning the node representation and the layout. For example, Pie Charts and Pietrees rely on a subdivision of circles, whereas Mosaic Charts and Treemaps rely on a subdivision of rectangles. In this sense, the design space for product plots overlaps with the design space of implicit tree visualization. This overlap allows the recreation of different types of charts⁸ using the implicit tree visualization design space and vice versa as was shown for instance for Treemaps in [WH11]. Additional charts may only need some slight adjustment of existing design choices. For instance, Bar Charts are based on a subdivision of a rectangular area but not in a space filling way. Thus, it is only necessary to add an additional method to the layout axis to also capture Bar Charts.

On this basis, an embedding of charts into a tree visualization can be performed directly using the design space. Especially for implicit tree visualizations, such an embedding can be described as another inclusion step in which the space of a node is distributed across its attributes. Therefore, the cloning operator is used to create a copy of the node for each attribute which is then attached as a child to the original node. In this way, the newly created nodes can be laid out using the design space to create the final glyph. Here, the edge representation and the dimensionality restrict the nodes for which it is useful to embed charts. In case of a 2D adjacency based visualization, for all nodes charts may be embedded. Whereas in case of an inclusion based or 3-dimensional visualization the high occlusion of inner nodes advises to embed charts only for the leaves.

However, because of this similarity between tree visualizations and attribute charts an important aspect for the embedding is the maintenance of a clear visual separation to distinguish between structure and attributes in the combined visualization. In the fol-

⁸For an alphabetical listing of possible charts the interested reader is referred to [Har96].

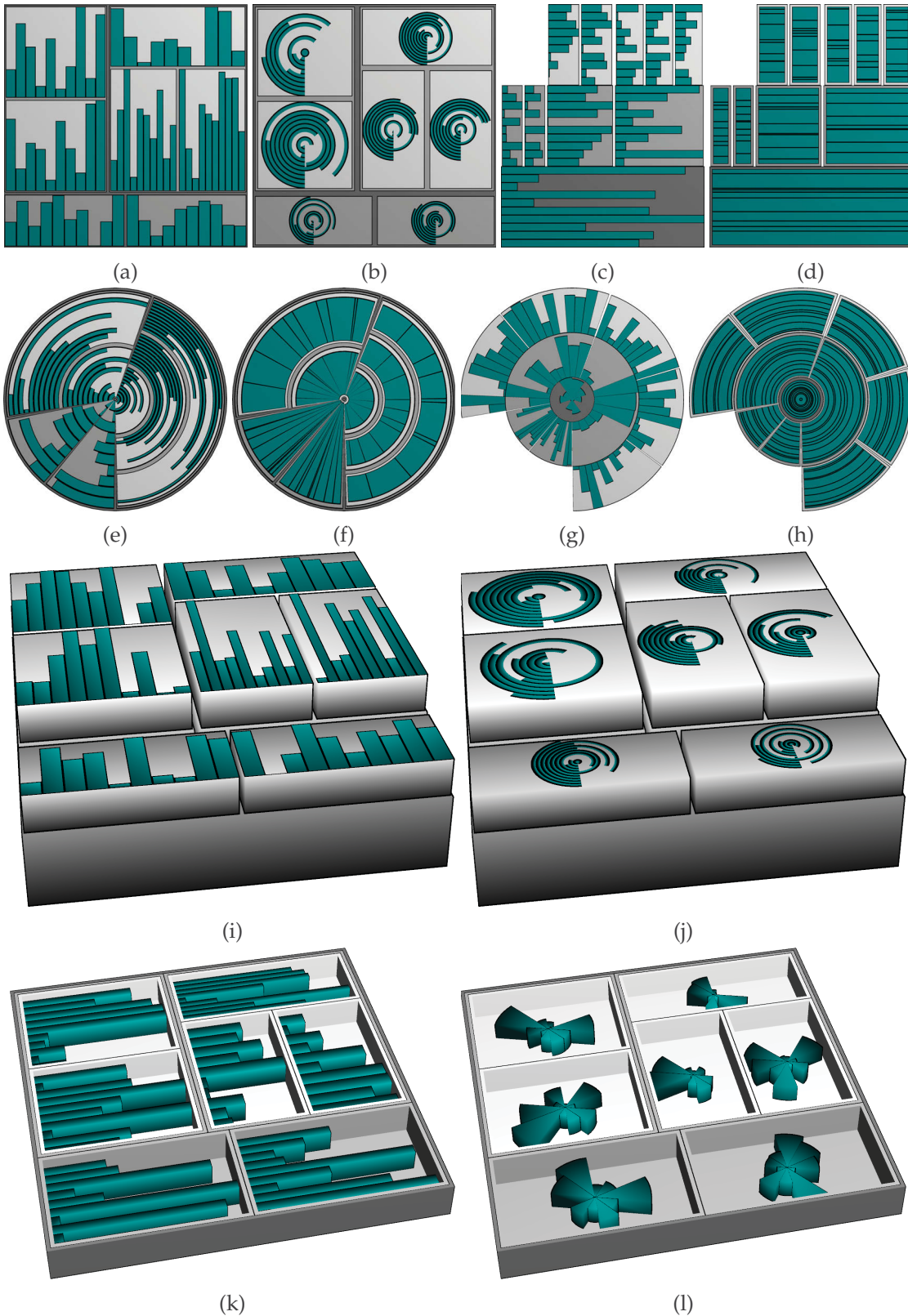


Figure 3.3.31: Embedding of charts to visualize multiple attributes. Treemaps with embedded (a) Column Charts and (b) Circular Bar Charts. Icicle Plots with embedded (c) Bar Charts and (d) horizontal Mosaic Charts. Pietrees with embedded (e) Bar Charts and (f) horizontal Mosaic Charts. Sunburst with embedded (g) Column Charts and (h) vertical Mosaic Charts. Steptree with embedded (i) Column Charts and (j) Circular Bar Charts. Boxtree with embedded (k) Bar Charts and (l) Sector Charts.

lowing, some examples for possible combinations along the 4 design axes are discussed and given in Figure 3.3.31.

Dimensionality: Represent attributes by using 2D or 3D. Figures 3.3.31i and 3.3.31j are based on a 3-dimensional visualization of the structure on which the attributes are mapped as a 2D "texture" on top of the leaves. Contrary, Figures 3.3.31k and 3.3.31l rely on a 2D visualization of the structure in which 3-dimensional extruded charts are embedded.

Node representation: Switch the complete primitive or adjust its surface properties. Figures 3.3.31b, 3.3.31j or 3.3.31l for instance switch from rectangular and cuboids to circles and cylinder primitives. Whereas in all figures the color is used to differentiate structure (gray colors) and attributes (cyan colors).

Edge representation: Use different geometrical relations for attributes. Figures 3.3.31c to 3.3.31g for example switch from adjacency for the structure to inclusion for the attributes.

Layout: Switch the layout or adjust its parameters. Figures 3.3.31a, 3.3.31c, 3.3.31e and 3.3.31g switch from a space filling layout to a "Bar Chart" layout leaving gaps. Whereas Figures 3.3.31d and 3.3.31h switch the orientation of the layout's subdivision.

While the embedding discussed so far shows multiple attributes for all nodes / leaves this may not be necessary in each case. Sometimes only attribute values of specific nodes or subtrees may be of interest. Hence, a further restriction of the embedding to only those nodes supports on the one hand the accentuation of these nodes. And on the other hand the remaining nodes may be represented in a more compact way providing additional display space for the nodes of interest. Such an idea is discussed amongst others in the Section 3.3.3.4.

3.3.3.3 Spatial Context

In existing techniques (see Section 2.2.3), the spatial reference is nearly exclusively handled by the layout assigning nodes positions close to their spatial location. In this way, the design space does not yet contain enough design choices to reflect the spatial reference in many other ways. An alternative is a 2 step approach: first designing an iconic representation of the hierarchy with the design space which is then embedded into a map display as it has been done for the point-based layout in the Figures 3.2.19 and 3.2.20.

3.3.3.4 Temporal Context

As discussed in Section 2.2.4.1 there are basically two different strategies for conveying the dynamics of the hierarchy with a structural focus:

- multiple drawings (animations and small multiples) to visualize the hierarchy for each time point, and
- supergraph based visualizations to provide a structural overview.

Despite animations, both visualization strategies tend to render nodes multiple times in a single visualization. Either the whole hierarchy as in the first case to convey struc-

tural changes. Or only the leaves as in the second case to communicate their associated attributes as time plots etc.

However, both strategies often render more objects than actually necessary. Small multiples always show the whole structure even if parts remain stable over all time points. Whereas supergraph based approaches distribute the space equally between all nodes even if subtrees do not show any interesting changes over time. Besides this information overload the visualizations of the first strategy are especially affected by the dynamics as they require comparable layouts (mental map) or smooth transitions often interfering with aesthetic layout criteria such as a good space utilization.

For both strategies the design space offers possible extensions to provide some first solution ideas for these problems. As the design space grants explicit access to the different design choices it opens up the black box of the visualization. In this way, the mapping may be freely adapted to either increase the overall space utilization or the comparability of the layout, or even the general space distribution.

Extensions for small multiples: When comparing layouts the most important aspect is to find the same node across different time points. In this sense, stabilizing a node's position over the course of time is the most common idea to support the mental map and to make layouts comparable. However, hierarchy layouts in general try to use the space as efficiently as possible leaving no gaps for removed nodes or nodes that will be added. This behavior is exemplified in Figure 3.3.32a showing a visualization similar to CodeFlows [CAT07]. The visualization uses layered Icicle Plots to communicate the structure whereas each layer encodes a different time point. Therefore, the whole hierarchy is cloned from the root (gray rectangle in the background) allowing the representation of multiple time points within a single visualization. For each time point the Icicle Plot is constructed by assigning missing nodes a visibility (node representation) and a weight (layout) of zero. In this way, the available space is only used for the existing nodes. While Icicle Plots maintain a linear order of nodes the space filling property of the layout complicates the comparison. Therefore, in CodeFlows ribbons were used to connect nodes in subsequent time points allowing their tracking.

Node representation: To lessen the problem of identifying nodes in each time point a first possibility is to assign each node a specific color according to their linear order using for instance a continuous color scale. This color coding reduces on the one hand the effort of finding a node as only subtrees with a similar color have to be investigated. And on the other hand, changes in the structure can be perceived as sudden breaks in the color.

Alternatively, the color can also be directly used to convey structural changes similar to the layering approach of the point-based layout shown in Figure 3.2.15 of Section 3.2.5. Therefore, nodes that have been added may be colored in red. Nodes that will be deleted in the next step may be colored in blue. Whereas, nodes that have moved in the hierarchy may be assigned a green color.

Layout: Furthermore, the assignment of layout weights can be adapted similar to the stabilization strategies of node movements described for general networks in Section 2.2.4.1. Using the same weight in all time points leads to a constant layout similar to a supergraph based layout and only the visibility of nodes will change from one time point to the other simplifying a comparison. Figure 3.3.32b shows the results us-

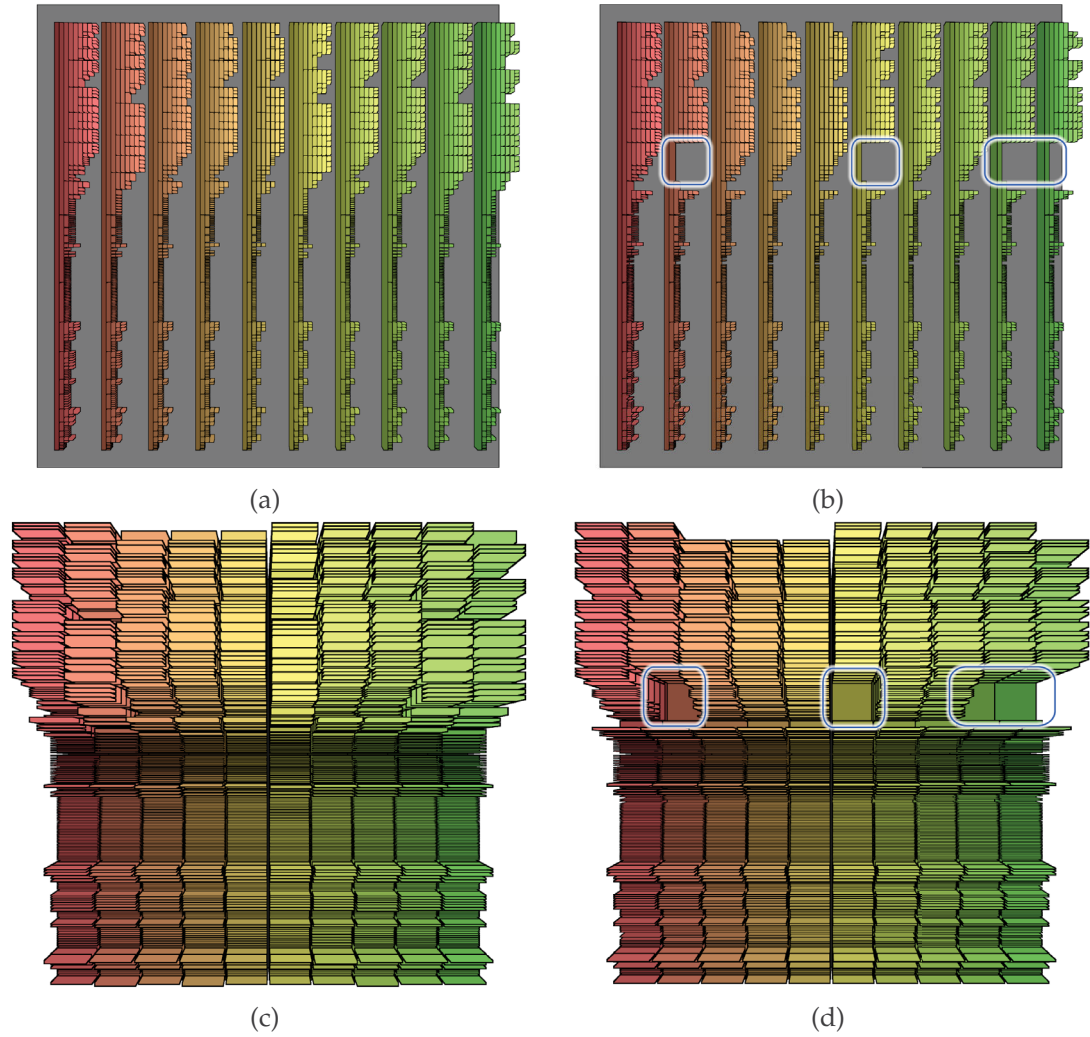


Figure 3.3.32: Layered Icicle Plots to convey changes in the structure using a space filling or a comparable layout. Each layer encodes the hierarchy for a different time point which is also mapped on the hue. Icicle Plots are drawn horizontally using a) a space filling layout and b) a comparable layout, and vertical using c) a space filling layout and d) a comparable layout. In the comparable layouts, gaps clearly point to missing nodes and subtrees. Some larger gaps are highlighted by a blue border.

ing such a constant layout. In this visualization changes are more clearly reflected by gaps in the visualization pointing to missing nodes or even whole subtrees. Both layouts may be combined into an intermediate visualizations by averaging the weights across time. Therefore, the mapping uses also information of prior and following time points to determine how much empty space should remain. In this way, gaps are not closed immediately but remain over some time points smoothing the general movement.

Dimension: This effect may be enhanced further by laying out the Icicle Plot not in 2D but using the third dimension as shown in Figures 3.3.32c and 3.3.32d. In this way, multiple occurrences of a node are not separated by other levels of the hierarchy anymore but are placed right next to each other. As the depth levels are now laid out along the z-axis instead of the x-axis the width of the visualization is not bound anymore by the maximum depth of the hierarchy but can be reduced to only some pixels. Additionally, by using the third dimension for the visualization of a single time point the x-axis is now only used to convey time. Hence, this adaptation allows a much tighter placement of time points compared to the original visualization increasing the number of time points to be visualized simultaneously.

Hybrid representation: Yet, still more nodes than necessary are visualized as not all nodes exhibit changes over time. For instance, the hierarchy in Figure 3.3.32 exhibits only a few larger changes but is mostly stable. Hence, it is useful to only show unstable parts separately for each time point while maintaining a single representation for stable nodes and subtrees. This idea basically describes a hybrid of a supergraph based approach for stable parts with small multiples for unstable parts. Therefore, first a script is applied to identify nodes and subtree featuring larger changes. This information is then used to clone only these nodes and embed them into a stable base visualization. The results of this hybrid visualization are depicted in comparison to the previous visualization in Figure 3.3.33 from different perspectives. In contrast to showing the whole hierarchy multiple times, this reduced hybrid view highlights the changes and thus simplifies their identification.

Because an Icicle Plot has been used as a base visualization for this example the overall layout of the visualization contains many long and thin cuboids. By switching the layout for the base visualization from a simple slice layout (1-dimensional subdivision) to a squarified layout (2-dimensional subdivision, see [BHvW00]) turns the Icicle Plot into a Steptree featuring cuboids of better aspect ratio. Yet, this switch also affects the embedded visualizations as shown in Figure 3.3.34. While the overall structure can now better be perceived the visualization now contains a local time axis for each embedded visualization. In this way, identifying the location of changes within the hierarchy becomes easier but determining simultaneous changes has become more difficult.

Optimizing supergraph based approaches: A general problem of embedding additional information into a base visualization is the space distribution. For instance in the previous visualization, the space is distributed equally between stable and unstable parts independently of the additional amount of information shown in the unstable parts. This also holds true when visualizing attribute changes for specific nodes such as the leaves.

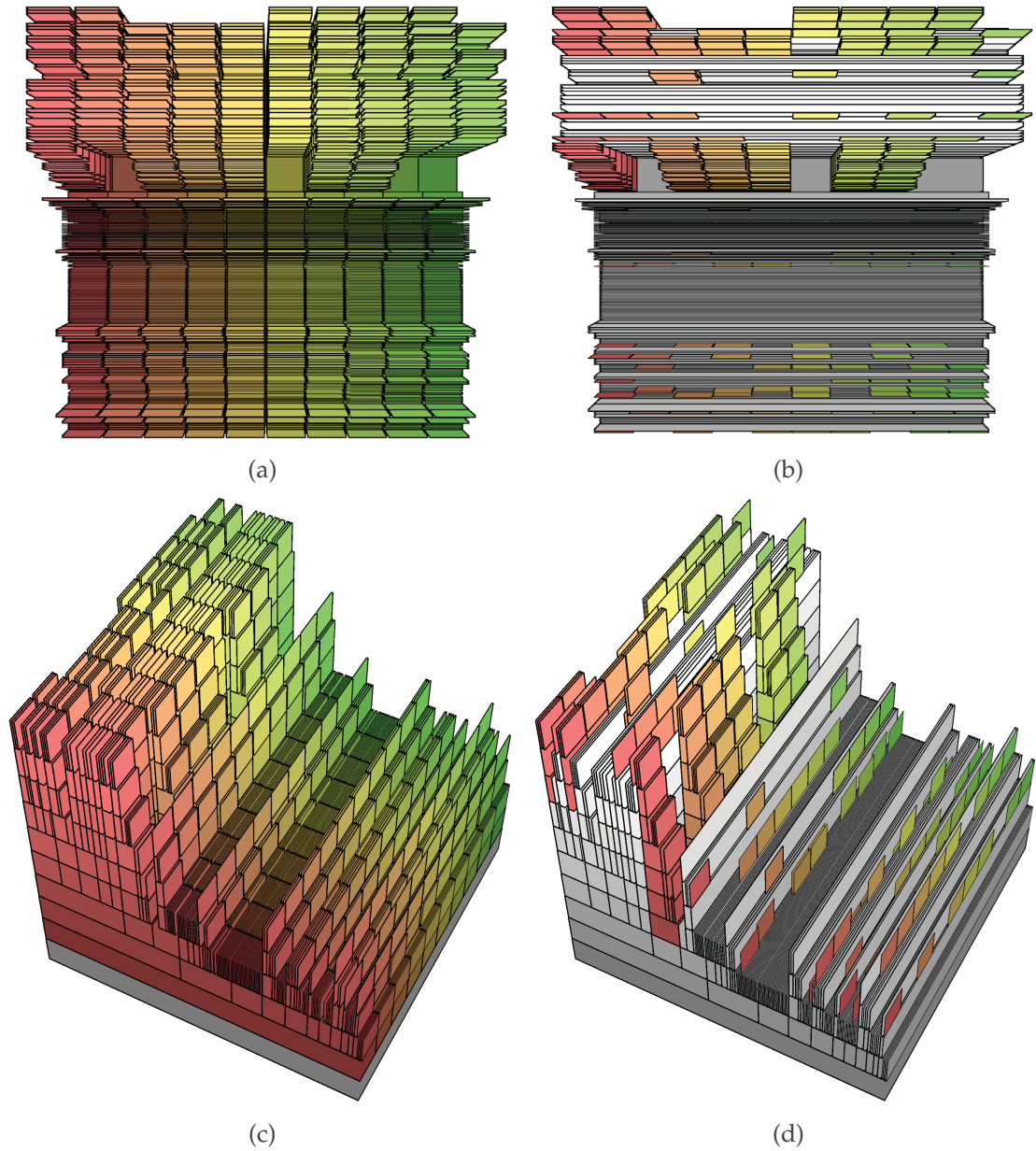


Figure 3.3.33: A filtered visualization based on layered Icicle Plots to convey changes in the structure. The original visualization as presented in Figure 3.3.32 is shown in a) from top and in c) from the side. A filtered version in which the hierarchy only splits up in multiple time points for subtrees featuring structural changes is shown in b) from top and in d) from the side.

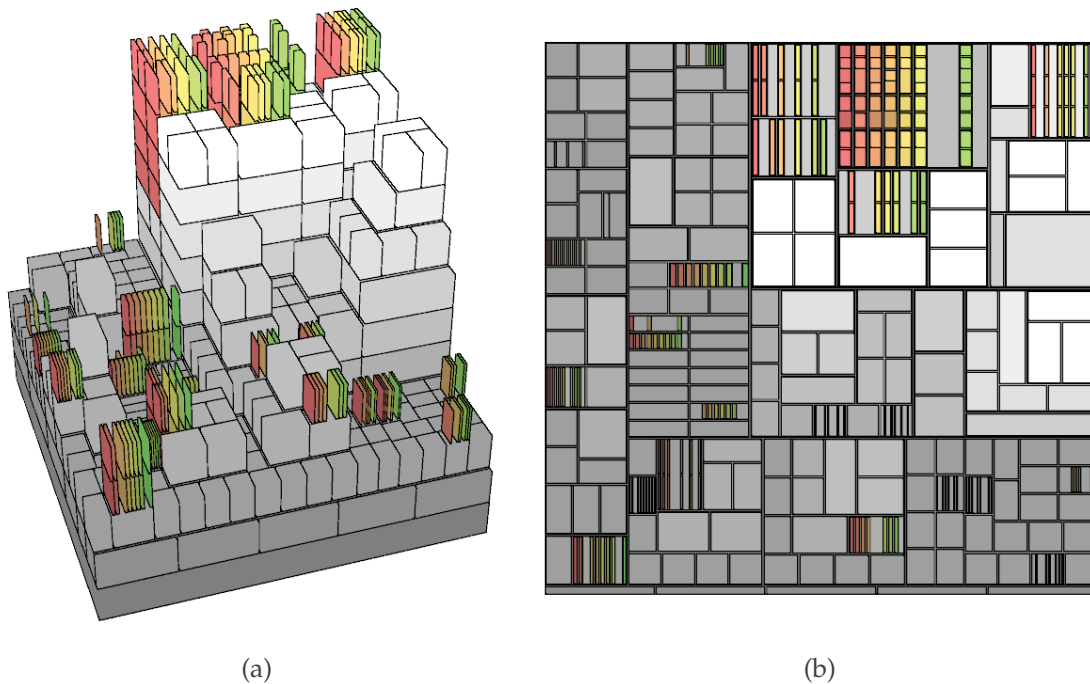


Figure 3.3.34: Unstable parts of the hierarchy are shown for multiple time points as layered Icicle plots embedded into a squarified Steptree. b) a side view and d) an orthogonal projection from top.

Not for all nodes the attributes may change significantly and thus may also be omitted or deaccentuated.

This problem is exemplified in Figure 3.3.35a in which the layout has tried to distribute the space equally between the leaves. Here, Sector Charts are used to visualize an associated time-varying attribute. Many charts show nearly full circles corresponding no changes at all. This is also reflected by the colors of the leaves communicating the variance of their attributes over time. In this way, more than half of the space is used to visualize redundant information.

Again, by adapting the layout weights the space distribution may be steered to highlight those leaves featuring a high variance. Therefore, first a script aggregates (e.g., summing up) the variance from the leaves to the root of the hierarchy. Then instead of using a subtree's size its overall variance (or any kind of importance measure) is used as a weight for distributing the space. The result in Figure 3.3.35d shows that the layout now favors the changing nodes. However, nodes with stable attributes are drawn very small if at all and thus very much hides contextual information. By using a linear combination of both attributes (subtree size and variance) the user is able to control the amplification of changing nodes. The Figures 3.3.35b and 3.3.35c reflect some steps in between still showing all stable nodes but only rendering them smaller.

The idea is similar to the Balloon focus [TS08] which was introduced to define and enlarge multiple foci within a Treemap. Yet, as the proposed adaptation of layout weights as well as the other discussed adaptations are specified directly in the mapping they can be freely adapted, exchanged and combined without the burden of a fixed function

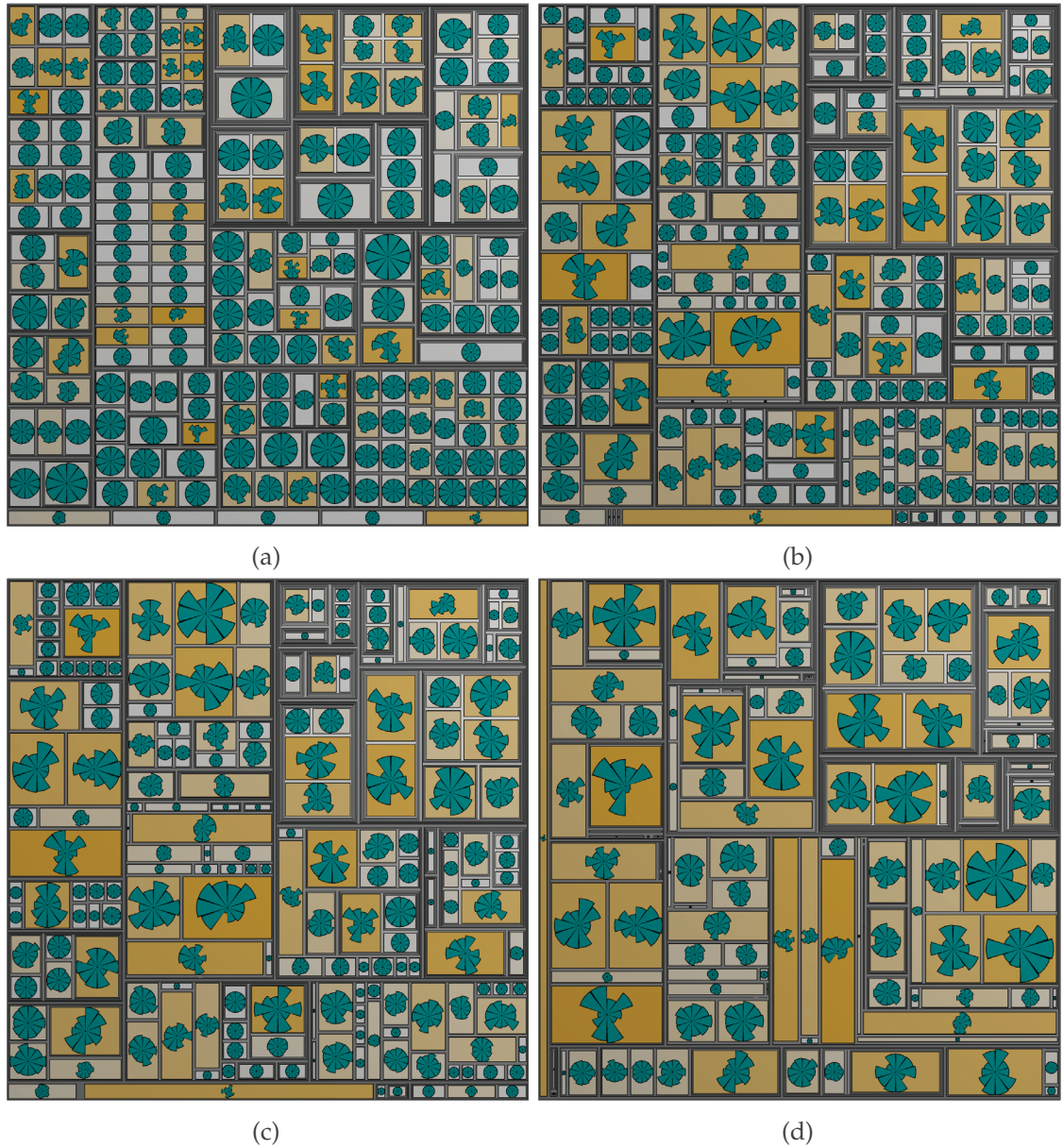


Figure 3.3.35: Embedding of charts to visualize time-varying attributes and adapting the space distribution according to the temporal variance. Sector Charts are used to communicate attribute changes. The variance of the attribute values over time are color-coded for each leaf with a bright color representing no changes and an orange color representing many or high changes. The figures show different space distribution strategies: a) distribution only according to the subtree size, b) distribution mainly according to the subtree size and partially by the variance within the subtree, c) distribution partially according to the subtree size and mainly by the variance within the subtree, and d) distribution only according to the variance.

pipeline that most existing visualizations rely on.

3.3.4 Conclusion

This section has introduced a design space for implicit tree visualizations based on 4 design axes: dimensionality, node representation, edge representation and layout. Plugging appropriate choices for each of these axes together allows rebuilding most of the existing techniques. The practicability of the design space has been shown by an implementation making the different design choices explicit and accessible. Additionally, a tuple based visualization design allows the realization of a visualization by defining a sequence of design decisions (tuples). Each design decision may depend on precalculated node attributes allowing not only global decisions but also the restriction of decisions to specific nodes. In this way, a user is not only able to tune an existing visualization to adapt it to the characteristics of his data set but also to mix different visualizations by combining their design decisions. This adaptability of the design space was then used to derive some new visualizations specifically tailored for the different aspects by identifying formerly unknown combinations of different design choices.

3.4 Summary

This chapter has introduced two novel visualization approaches for hierarchies. The first approach is based on a family of point-based layouts to provide a compact representation of large structures. In this context, requirements and limitations concerning the visualization of the different aspects (structure, attributes, spatial and temporal context) have been discussed. And the second approach is based on a design space for implicit visualization techniques to grant a more flexible way of handling all aspects of hierarchies. Both approaches show promising first results but need further evaluation. Especially the design space for which only some examples have been shown may yield many interesting and important new visualization ideas.

Yet, at the same time these approaches also show the limitations for a single visualization. The larger the graph becomes, the more difficult it is to address all aspects sufficiently. Thus, to be able to really cope with the different aspects it is essential to also have a rich set of reduction techniques. Hence, the next chapter explicitly targets the size of dynamic graphs.

4 Scalable Visualization of Dynamic Graphs

4.1 Introduction

As discussed in the previous sections, all visualizations are bound to a visual entity budget limiting the amount of information that can be represented simultaneously. To deal with larger graphs reduction strategies are applied to process these graphs in a way that enables their visualization. Generally, there are two reduction strategies that are often used in conjunction: *abstracting* the graph or selecting a *subset* of the graph.

As the overall size of the graphs depends on all aspects (structure, attribute, spatial and temporal context) reductions may target any aspect. Yet, there are some peculiarities considering the associated attributes and the spatial context of graphs. Generally, different attributes cannot be abstracted that easily without appropriate knowledge of the context resulting in selections often being the only viable option. Similar, for the spatial context often different levels of granularity are given that are based on the geography or a subdivision into administrative regions. In this way, a further abstraction is often not needed for the spatial aspect. Hence, the following discussions focuses primarily on more general strategies concerning the structural and temporal aspects of graphs.

Furthermore, static graphs have been studied extensively by the existing literature. Yet, their application to dynamic graphs is still an open research question. This chapter therefore introduces two new approaches for reducing the size of dynamic graphs:

- A new clustering for dynamic graphs abstracting either the structural or the temporal aspect while taking the respectively other aspects into account.
- A flexible and modular selection technique based on defining a Degree of Interest for each node, edge and time point of the dynamic graph.

4.2 Clustering based Abstraction of Structure and Time

Besides reducing the size of the graphs, abstraction strategies often try to group similar elements. In case of dynamic graphs, these elements may be nodes and edges or time points. Here, depending on the aspect (structure or time) to be abstracted different goals may be supported:

- For instance, by clustering on the one hand the nodes and edges of the dynamic graph according to similar trends of their attributes over time, temporal relationships between them may be identified. Based on these relationships stable, evolving or unreliable parts of this graph can be determined.

- On the other hand, by clustering time points according to a similar graph structure different temporal patterns may be determined allowing the differentiation of time intervals featuring smaller or larger changes or even the identification of recurring graph constellations.

Furthermore, such temporal relationships or patterns may be found on different levels of temporal and structural granularity. For example, in a communication network, patterns that distinguish weekday from weekend can be observed on a daily granularity, whereas the distinction between working and non-working hours relies on hourly patterns. The same goes for the structural granularity, which can be, for instance, per device, per household, or per city block to observe different connection patterns. Hence, different temporal and structural granularities have to be taken into account for the clustering of a dynamic graph.

Here, the clustering may be performed on its structural aspect grouping nodes and edges, its temporal aspect grouping time points, or on both aspects simultaneously grouping different sets of nodes and edges for different time points. As discussed in Section 2.3.1 the latter approach may provide the most versatile clusters but is both time consuming concerning its calculation time and difficult to steer to gain practical results. In this sense, it is most likely unsuited for a scalable visual analysis of dynamic graphs. The clustering of the structure is mostly performed individually for each time point making nearly no use of the temporal information. And a clustering of the temporal aspect has not yet been applied for the visual analysis of dynamic graphs.

Therefore, in this section two new clustering approaches for dynamic graphs are introduced that make use of the temporal information targeting either the structural or the temporal aspect. Both approaches follow two steps:

1. **Extraction** of the graph for a given structural and temporal granularity.
2. **Clustering** to abstract either the structural or temporal aspect of the graph.

In this way, these approaches provide a scalable visualization of large graphs by relying on the one hand on a given hierarchical organization of the graph allowing the user to select an appropriate granularity. And on the other hand, by introducing another, artificial level of abstraction by clustering elements of the graph not only to further reduce the size but also to show similarities between the elements such as nodes with similar temporal trends.

In the following, first the extraction for a selected granularity is described for a given hierarchical organization of structure and time. Then the structural and the temporal clustering approaches are introduced each in a separate section with its own visualization setup to accommodate the different aspects of the clustering results. Both clustering approaches and their interactive visualization setups are then showcased with a wireless mesh network with 297 nodes and 2'008 edges over 217'253 time points (minutes) spanning 5 months of data. All figures shown throughout this section are based on this data set.

4.2.1 Extraction of the Graph for a given Granularity

The identification of temporal patterns or relationships is sensitive to the chosen granularity in structure and time. For instance, if the time points of the dynamic graph are clustered at a higher temporal granularity (i.e., minutes or hours), the cluster results will capture recurring patterns in the graph's short term behavior. Whereas if the clustering is performed at a lower temporal granularity (i.e., months or years), the clustering will group similar long term trends of the graph's dynamics. Hence, the extraction of the graph on a structural and temporal granularity which is of particular interest for the user has to be performed prior to the clustering. This sequence of computation is also in line with existing approaches in clustering time series data, for example, for their calendar-based visualization [vWvdW99].

A dynamic graph is generally only recorded at a single level of granularity for structure and time, the finest level of granularity. Yet, for its observation space often different hierarchical organizations may be given that can be used for the visual analysis. For instance, nodes may be grouped according to their spatial reference regarding a geopolitical organization of space (districts, counties, states, countries) or time points may be organized in minutes, hours, days and months. Based on such a hierarchical organization, the dynamic graph is extracted for a specific structural and temporal granularity in two steps according to its two aspects (structure and time):

1. **Structural coarsening** of the graph on a given structural granularity.
2. **Temporal aggregation** of its dynamics on a temporal granularity.

This extraction process depends on two input parameters, the desired granularities in structure and time which can be modeled as equivalence relations \sim_s and \sim_t , respectively. Each of these equivalence relations divides the graph elements into equivalence classes that are considered the same on the selected granularity. For example, such a class may group nodes that are contained in the same clique or otherwise dense subgraph or belong to the same geopolitical district or county. Whereas for the temporal aspect time points may be classified as equal if they fall into the same interval (same hour or day). In this way, these equivalence relations and classes define the transformation of the graph from the finest, given level to the desired level of granularity.

As both steps, the coarsening and the aggregation, can be performed independently of each other, this raises the question whether to aggregate the time before coarsening the structure or vice versa. In case of performing the structural coarsening after the temporal aggregation, node movement between clusters from one time step to another will falsely induce a connection between these clusters if these time points get aggregated. This effect is illustrated in Figure 4.2.1 that compares the results of both orders of the two steps. Therefore, the structural coarsening is performed prior to the temporal aggregation in the following and described in more detail in this order.

Step 1 - Structural Coarsening: The structural coarsening is in itself a 2-step process that first coarsens the nodes and then coarsens the edges with respect to the coarsened nodes. It basically iterates over all time points $t_i \in T$ and coarsens each V_i as outlined in Algorithm 1.

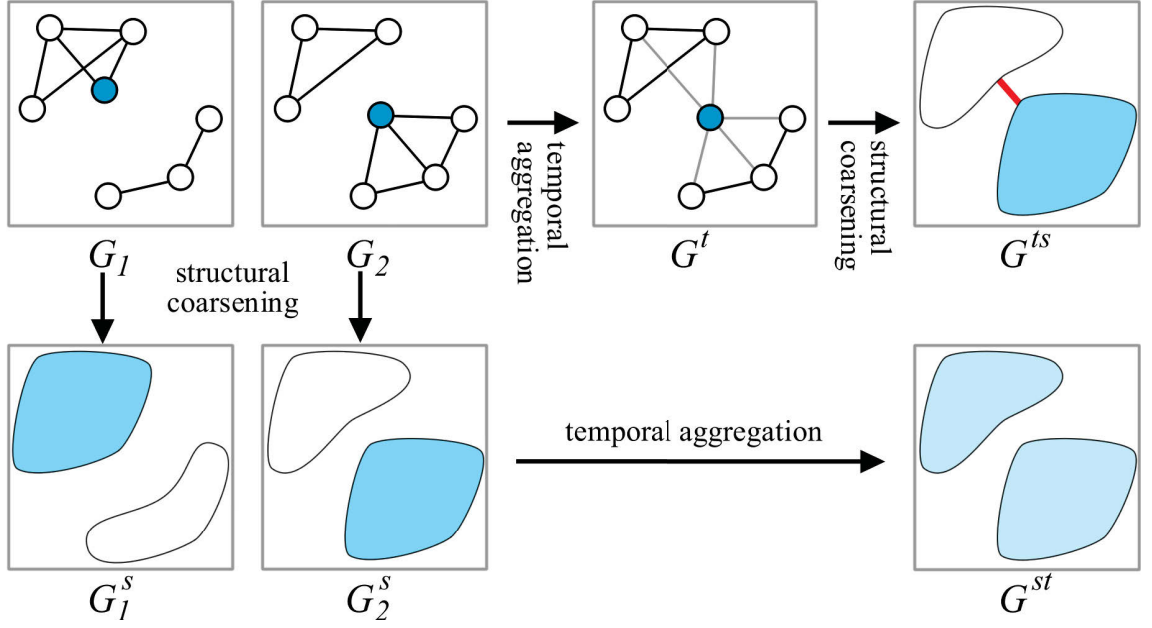


Figure 4.2.1: The effect of the two possible orders for structural coarsening and temporal aggregation exemplified. Both scenarios start with graphs G_1 and G_2 at two subsequent time points, which are equal except for the blue node that has moved. The first ordering performs the temporal aggregation into the supergraph G^t before the structural coarsening of this graph into G^{ts} . The second ordering is to first coarsen both graphs individually, yielding G_1^s and G_2^s , and then aggregate these into the supergraph G^{st} . As one can see, the first order of steps introduces a connection between the two clusters into the resulting graph for where the movement has occurred, whereas the second order of steps retains their disconnectedness throughout the aggregated time interval.

Algorithm 1 Structural Coarsening of the Node Set V_i w.r.t. \sim_s

<pre> 1: for $i : 0 \leq i \leq T$ do 2: $V_i^s \leftarrow \emptyset$ 3: $C \leftarrow \emptyset$ 4: for $v_i \in V_i$ do 5: if $v_i \notin C$ then 6: $C \leftarrow C \cup \{v_i\}$ 7: $v_i^s \leftarrow \{v_i\}$ 8: for $u_i \in V_i \setminus C$ do 9: if $v_i \sim_s u_i$ then 10: $C \leftarrow C \cup \{u_i\}$ 11: $v_i^s \leftarrow v_i^s \cup \{u_i\}$ 12: $V_i^s \leftarrow V_i^s \cup \{v_i^s\}$ </pre>	<div style="display: flex; justify-content: flex-end;"> <div style="margin-right: 10px;"> <div>\triangleright for all time points</div> <div>\triangleright set of meta nodes at structural granularity s</div> <div>\triangleright set of already processed nodes</div> <div>\triangleright for all nodes</div> <div>\triangleright if node was not processed yet</div> <div>\triangleright mark node v_i as processed</div> <div>\triangleright create a new meta node</div> <div>\triangleright for all remaining nodes</div> <div>\triangleright if nodes are equal w.r.t. \sim_s</div> <div>\triangleright mark node u_i as processed</div> <div>\triangleright add node to meta node</div> <div>\triangleright add meta node to V_i^s</div> </div> </div>
---	--

Note that for the sake of brevity, the looping through all node attributes in A_i and their aggregation into attributes A_i^s of the meta nodes have been omitted from this algorithm. Depending on the type of an attribute, its aggregation can take on many forms [BPC07]. Yet in most cases, meaningful results are obtained by simply computing the attribute values' average and providing their min/max values on demand.

Algorithm 2 Structural Coarsening of the Edge Set E_i w.r.t. V_i^s

```

1: for  $i : 0 \leq i \leq |T|$  do                                      $\triangleright$  for all time points
2:    $E_i^s \leftarrow \emptyset$                                         $\triangleright$  set of meta edges at structural granularity  $s$ 
3:   for  $v_i^s \in V_i^s$  do
4:     for  $u_i^s \in V_i^s \setminus \{v_i^s\}$  do                        $\triangleright$  for all meta node pairs  $(v_i^s, u_i^s)$ 
5:        $e_i^s \leftarrow \emptyset$                                     $\triangleright$  create a new meta edge
6:       for  $v_i \in v_i^s$  do
7:         for  $u_i \in u_i^s$  do                                      $\triangleright$  for all node pairs  $(v_i, u_i)$ 
8:           if  $(v_i, u_i) \in E_i$  then                              $\triangleright$  if they have an edge
9:              $e_i^s \leftarrow e_i^s \cup \{(v_i, u_i)\}$             $\triangleright$  add it to meta edge
10:      if  $e_i^s \neq \emptyset$  then                                    $\triangleright$  if edges were found
11:         $E_i^s \leftarrow E_i^s \cup \{e_i^s\}$                       $\triangleright$  add meta edge to  $E_i^s$ 

```

Once the node set V_i has been coarsened into V_i^s , its set of incident edges E_i can then be coarsened accordingly by carrying out Algorithm 2. This algorithm also omits the aggregation of edge weights W_i into weights of the resulting meta edges W_i^s , which can be averaged as well. For the aggregate value of edge weights between nodes that are subsumed by a meta node, an attribute is added to the meta node to keep this information for later analysis.

Overall, this step has a runtime complexity of $\mathcal{O}(|T| * |V|^2)$, which is due to its iteration over all time points and the looping through all node pairs in the worst case in which the coarsening leaves V unchanged. Note that this complexity captures the worst case for the principal algorithm presented above. It can either be outsourced into a preprocess or a more sophisticated coarsening technique can be used to achieve better runtime complexities especially for sparse graphs. For example, the Topological Fisheye [GKN05] has a complexity of $\mathcal{O}(|V| + |E|)$ with a precomputation running in $\mathcal{O}(|V| * \log |V|)$ for coarsening a single graph. Coarsening a dynamic graph will thus result in a complexity of $\mathcal{O}(|T| * (|V| * \log |V| + |E|))$. In both cases, the result is a structurally coarsened dynamic graph $DG^s = \{G_0^s \dots G_{|T|}^s\}$ with $G_i^s = (V_i^s, E_i^s, A_i^s, W_i^s, t_i)$.

Step 2 - Temporal Aggregation: The temporal aggregation is a rather straightforward process described in Algorithm 3 that accumulates graphs G_i and G_j if the given equivalence relation \sim_t holds true for the two time points t_i and t_j . This accumulation is performed by a supergraph (see Section 2.1) that is denoted in Line 10 by the symbol \oplus . According to [Arc09, BDKW07] where this operation is used to compute a difference or median graph, this operation runs in $\mathcal{O}(|V| + |E|)$. Thus the overall runtime complexity of this procedure is bounded by $\mathcal{O}(|T|^2 + |T| * (|V| + |E|))$, which is due to its looping through all pairs of time points and the calculation of a single supergraph in the worst case in which all time points are aggregated. The result of this step is a dynamic graph DG^t that is a sequence of supergraphs $G_i^t = (V_i^t, E_i^t, A_i^t, W_i^t, [t_{i_{min}} \dots t_{i_{max}}])$ defined over a time interval with $t_{i_{min}}, t_{i_{max}} \in T$.

Applying both steps, the structural coarsening and the temporal aggregation, for a given structural and temporal granularity results in the coarsened dynamic graph DG^{st} . As the structural coarsening and the temporal aggregation only need to be performed

Algorithm 3 Temporal Aggregation of the Dynamic Graph DG w.r.t. \sim_t

```

1:  $G^t \leftarrow \emptyset$                                  $\triangleright$  set of graphs  $G$  at temporal granularity  $t$ 
2:  $C \leftarrow \emptyset$                              $\triangleright$  set of already processed graphs
3: for  $G_i \in DG$  do                                $\triangleright$  for the graphs at all time points
4:   if  $G_i \notin C$  then                              $\triangleright$  if graph was not processed yet
5:      $C \leftarrow C \cup \{G_i\}$                       $\triangleright$  mark graph  $G_i$  as processed
6:      $G_i^t \leftarrow G_i$                             $\triangleright$  create a new supergraph
7:     for  $G_j \in DG \setminus C$  do                  $\triangleright$  for all remaining graphs
8:       if  $t_i \sim_t t_j$  then                      $\triangleright$  if time points are equal w.r.t.  $\sim_t$ 
9:          $C \leftarrow C \cup \{G_j\}$                   $\triangleright$  mark graph  $G_j$  as processed
10:         $G_i^t \leftarrow G_i^t \oplus G_j$               $\triangleright$  add  $G_j$  to supergraph
11:       $G^t \leftarrow G^t \cup \{G_i^t\}$               $\triangleright$  add supergraph to  $G^t$ 

```

once, the computation of the coarsened graph can be shifted to a preprocess. This coarsened graph is the input of the following two clustering approaches.

4.2.2 Clustering of the Structural Aspect

This section¹ introduces a novel structural clustering that clusters the supergraph of the dynamic graph according to time varying attributes associated with the nodes and edges. In this way, it makes full use of the temporal information contained in these time varying attributes. In the following, first the clustering is described and then the visualization of the resulting *clustered supergraph* that reflects the captured temporal relationships.

4.2.2.1 Clustering associated Temporal Attributes of the Supergraph

The goal of this approach is to support the identification and inspection of groups of nodes and edges that are similar according to their attribute values over time. For the automatic extraction of these substructures and their temporal relations, first the supergraph is calculated. On the temporal attributes associated with the nodes and edges of this supergraph a clustering for time series data which has been successfully used for instance for the analysis of spatio-temporal data [AAB⁺10, LWZZ09, ZJGK10] is then applied. The resulting clustering of the associated time series is finally used to group the corresponding nodes and edges of the supergraph. However, before these techniques can be applied, the following problems have to be solved on the supergraph of the dynamic graph:

1. The attributes for all nodes and edges have to be extracted to define the time series.
2. Their clustering has to be configured and carried out.
3. Based on the clustering, connected subgraphs have to be defined.

These three steps are described in the following.

¹Parts of this section have been published as “Supporting the Visual Analysis of Dynamic Networks by Clustering associated Temporal Attributes” 2013 in the IEEE Transactions on Visualization and Computer Graphics [HSCW13].

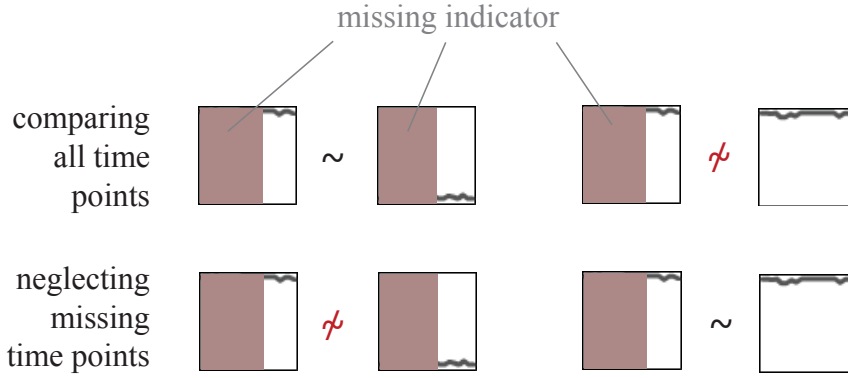


Figure 4.2.2: Handling of missing values during the similarity calculation: in the first row all time points are compared resulting in the first pair of time series being more similar. In the second row, time points with missing values are neglected resulting in the second pair of time series being more similar.

Step 1 - Extraction of Time Series: The time series to be clustered are associated as attributes with nodes or edges of a given dynamic graph. As not all attributes may be of interest, first those attributes have to be selected that should be used for the clustering. The supergraph contains all nodes and edges of the dynamic graph. Hence, for each of these nodes and edges a time series representing the values of each selected associated attribute can be extracted.

As nodes and edges may not exist at all time steps, missing values have to be marked by a particular value. A common way is the missing-indicator method introducing a placeholder value for each attribute that is not within the range of existing values. Thus, this placeholder is used to differentiate between missing and original values [DvdHSM06]. Yet, it has to be used with care for the clustering as it may lead to a biased analysis:

- For instance, two nodes that coexist for a short amount of time and are absent at the remaining time points will be grouped together because they show the same placeholder value most of the time as depicted in the first row of Figure 4.2.2. This grouping is rather independent of the real values they feature during the time they exist.
- An alternative to compensate for such a problem is to neglect time points at which at least one node does not exist. In this way, those nodes would be grouped with other nodes or edges sharing a similar course at only those time points they coexist as exemplified in the second row of Figure 4.2.2.

However, both approaches yield interesting information. The first allows for the grouping of coexisting subgraphs and thus the determination of parts of the graph that have been deleted or created at the same time. Whereas the second approach allows for the analysis of nodes and edges according to the behavior they show during the time they exist. In communication networks, this could be used to determine stable connections independently of their creation date. So this leads to another parameter that may be used to steer the analysis.

As for each node and edge each attribute value has to be gathered for each time point the runtime complexity of this step is bounded by $\mathcal{O}(|T| * (|V| + |E|))$. To filter out minor

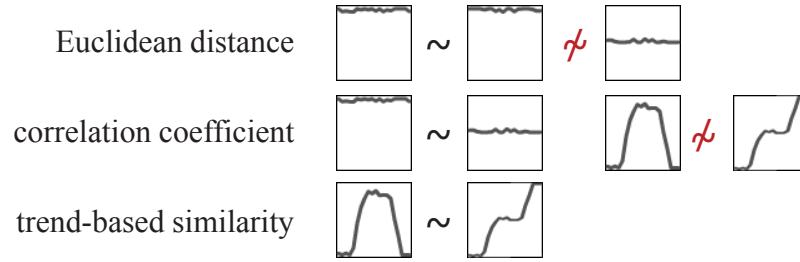


Figure 4.2.3: Examples of different similarity measures for the clustering: Euclidean distance based measures allow the grouping of nodes and edges with similar values over time. Correlation coefficient based measures allow their grouping according to their correlation. Trend-based similarity measures allow their grouping according to simultaneous changes.

fluctuations such as unwanted noise and to allow for an interactive analysis by decreasing the calculation time of the clustering, often a reduction of the time series is performed. A common way is the application of a simple low-pass filtering averaging the data values of multiple time steps and a subsampling resulting in a coarser and smoother temporal scale to be clustered as proposed in [vWvdW99]. This procedure is similar to selecting a coarser temporal granularity in the previous step but may be configured independently of the analyzed granularity. A more complex alternative is to extract perceptually important points to represent the main characteristics of the time series [ZJGK10]. Yet, its non-uniform distribution of time points complicates the similarity calculation during the clustering.

Step 2 - Clustering of the associated Time Series: Clustering time series requires two steps: determining a distance/similarity measure and selecting a method for clustering the time series based on this measure [WL05]. Measure and method can be chosen and configured independently of each other, enabling different ways to analyze the dynamic graph. Here, three different similarity measures are provided: the Euclidean distance, the correlation coefficient and a trend-based similarity measure [LWZZ09] as well as two clustering methods: a hierarchical and a k-means method.

The choice of the similarity measure strongly depends on the chosen attributes and on the analysis goals of the user as depicted in Figure 4.2.3. If he is interested in grouping the nodes and edges with regard to simultaneous changes over time, a trend-based measure would suffice. For example, this can be the case if he is looking for evolving topics in a co-authorship or citation network which is characterized by a simultaneous increase in the node degree or citation count. Yet, if he is interested in nodes and edges showing similar values over time, the Euclidean measure is better suited. This is generally a good first choice for attributes featuring only a single data range and that are thus comparable like the quality of connections between nodes in a communication network. To be able to compare different similarities these measures are normalized to the range [0,1] with 0 describing no similarity of the time series and 1 describing full similarity.

As a default clustering method, a hierarchical clustering method is used as it supports an interactive, more scalable exploration of the graph on multiple levels of detail. Here, a similarity threshold can be given to define an initial cut through the clustering hierarchy

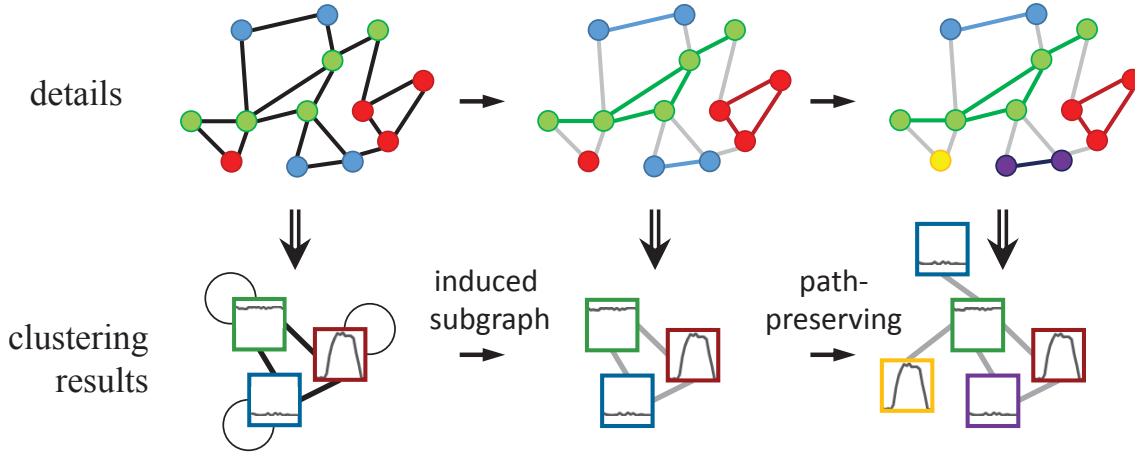


Figure 4.2.4: Defining connected subgraphs: The first row reflects the network structure, the second row the clustering results. The first column shows the clustering of the nodes according to their time series without considering the edges. The second column shows the results after extracting the induced subgraphs (edges are assigned the color of their cluster or gray). And the third column reflects the final path-preserving clustering which has split two clusters into its connected components (blue cluster into blue and purple clusters, red cluster into red and yellow clusters).

according to the intra-cluster similarity which captures the similarity of all time series in a cluster. In case of a k-means clustering, the number and centers of clusters have to be appropriately specified.

The runtime complexity of this process depends on the chosen similarity measure and clustering method. Here, the computation time for calculating the similarity between two time series is in most cases linear depending on the number of time points and thus has a complexity of $\mathcal{O}(|T|)$. Whereas the runtime complexity of the clustering method strongly varies between the different methods [XWI05]. For instance, k-means has a complexity of $\mathcal{O}(k * (|V| + |E|))$ and thus is nearly linear depending on the number of nodes and edges. Whereas in case of a hierarchical method, the runtime complexity is $\mathcal{O}((|V| + |E|)^2)$ because each pair of nodes and edges has to be compared. Hence, the overall complexity of this step is in the worst case using a hierarchical method $\mathcal{O}((|V| + |E|)^2 * |T|)$.

Step 3 - Defining connected Subgraphs: The result of the clustering is solely based on the time series extracted for the attributes. Thus, structural connections between nodes and edges are not considered. Furthermore, attributes may be either defined for nodes or edges. That means that each cluster represents a collection of possibly disconnected nodes or edges as shown in the first column of Figure 4.2.4. However, each cluster can be transformed into a subgraph as follows:

1. Extraction of the induced subgraph: In case of a cluster containing disconnected nodes often as a result of attributes defined only for nodes, all edges whose incident nodes are both contained in the cluster are added as demonstrated in the second col-

umn of Figure 4.2.4. For a cluster containing disconnected edges which may arise from attributes defined only for edges, the incident nodes are added.

2. Path-preserving clustering: The induced subgraph may still consist of multiple components that are similar according to their time series, but that are not structurally related. This can be exemplified by a simultaneous drop-out of nodes within a communication network. If these nodes are clustered together but are not connected, their drop-out may be a coincidence. However, if these nodes are connected, it is more likely that their simultaneous drop-out is caused by the same incident. Thus, to obtain a clustering in which each cluster represents a single connected subgraph (also called “path-preserving” clustering [Arc09]), all discovered clusters are split into connected components afterwards as done in the third column of Figure 4.2.4. If the analyst is interested in finding relations that are not covered in the structure but appear in the temporal clustering, the maintenance of the path-preserving property can also be skipped making this step optional depending on the analysis.

For extracting the induced subgraph the containing cluster needs to be identified only once for each node and edge of the graph. Whereas the clusters can be split into the connected components by using a breadth-first search which also checks each node and edge only once. Hence, both steps can be performed in $\mathcal{O}(|V| + |E|)$.

The runtime complexity of the entire clustering procedure is bound by $\mathcal{O}((|V| + |E|)^2 * |T|)$ resulting from the most complex step, the clustering of the associated time series. As the complexity of all these steps strongly depends on the chosen structural and temporal granularity, selecting a coarser scale leads to a significant acceleration as stated in [GW11, PIB⁺11]. Furthermore, parts of these computations can be shifted to a preprocess. For instance, the distances between each pair of time series can be precomputed and stored. By doing so, the complexity of the clustering of the supergraph can be reduced to the mere grouping of nodes and edges in an overall complexity of $\mathcal{O}((|V| + |E|)^2)$ during the analysis. In this way, these refinements of the process support the generation of multiple clustering results according to different granularities, similarity measures, clustering methods and parameters. In the following section, the resulting clustered supergraph is used for the visualization of the dynamic graph.

4.2.2.2 Visualization of the Clustered Supergraph

The goal of the visual design is to support the visual exploration of the dynamic graph concerning nodes with similar trends over time. Therefore, an overview and several detail views are provided. The supergraph provides the overview showing groups of nodes and edges being similar with regard to the temporal trends of their associated attributes. Clustering results and time series of interest are represented by detail views. In the following, the views are described in more detail.

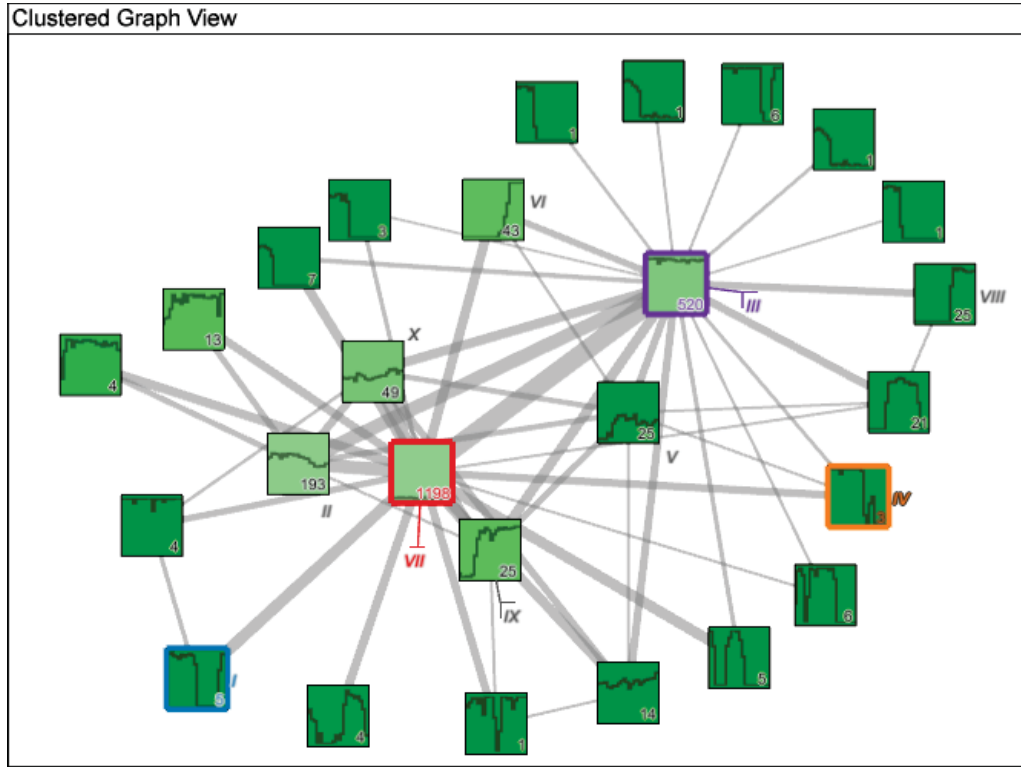


Figure 4.2.5: The clustered graph view provides an overview of the clustered super-graph. Nodes represent clusters abstracting subgraphs with a similar behavior over time and edges reflect connections between these clusters. Nodes are visualized as glyphs showing small time plots that represent the temporal trend of the associated cluster as well as its color-coded intra-cluster similarity.

Clustered Graph View

The *clustered graph view* aims at communicating the found temporal relationships of the clustering. It visualizes the clusters and their interrelations and allows the simultaneous visualization of clusters together with their major temporal trends. In this way, the user can differentiate between stable or unreliable parts of the graph.

Visual Design: The *clustered graph view* (see Figure 4.2.5) visualizes the clustered super-graph. The temporal trend of each cluster is represented by small glyphs as suggested by [SLN05, US09]. These glyphs show small time plots for a selected attribute. The similarity of the time series of each cluster (intra-cluster similarity) is mapped to the background color of its glyph. High similarity is represented by dark green and low similarity by white glyphs. The size of each cluster is reflected by a label in the lower right corner of its glyph. Furthermore, to reflect the structural aspect of the dynamic network, clusters containing nodes or edges that are connected within the network are connected by lines. Here, the width of these lines reflect the number of connections between the clusters.

To sum it up, the *clustered graph view* communicates the connectivity and intra-cluster similarity of clusters as well as the major temporal trend of each cluster.

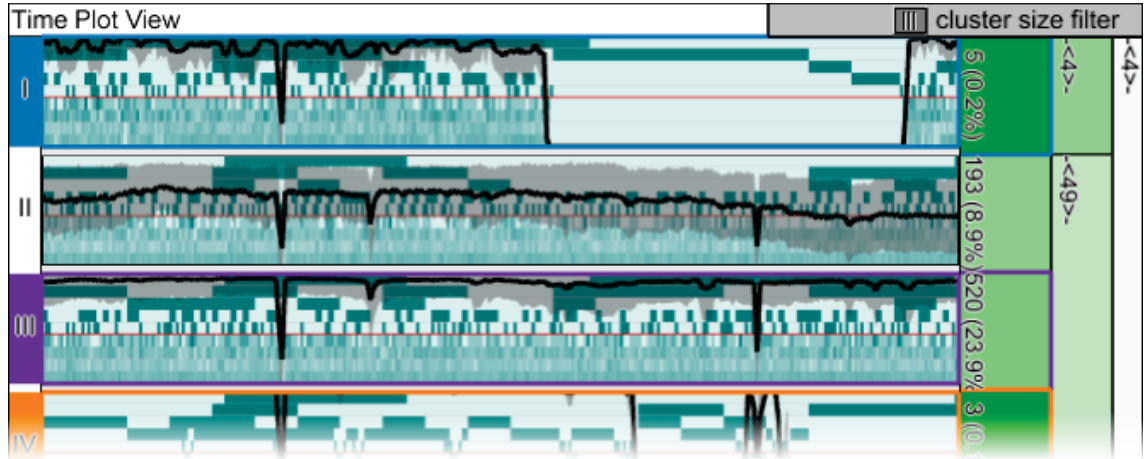


Figure 4.2.6: The time plot view provides a more detailed multi-scale time plot for each important meta node resulting from a clustering of the dynamic graph according to its structural (as in this case) or temporal aspect. The cluster hierarchy is shown on the right side of this view, also reflecting the intra-cluster similarity.

Interactions for Exploring the Structure: In case of a hierarchical clustering, a layer-based approach is utilized. The *clustered graph view* then represents a single layer/cut in the clustering hierarchy. A layer-based navigation allows the user to select a similarity threshold and thus the layer/cut to be visualized as well as an unbalanced drill-down by expanding and folding specific clusters on demand.

The user can also select multiple clusters for a closer inspection. These clusters are highlighted in the *clustered graph view* as well as in the following *time plot view* by a colored border. Therefore, a couple of distinguishable colors (e.g., qualitative color schemes in [HB03]) are provided to differentiate the selected clusters and allow their localization in both views. For each cluster, the contained substructure can also be drawn in the according color in the background of the *clustered graph view* as shown in Figure 4.2.18 allowing for a quick peek into the details of multiple clusters on demand.

Time Plot View

The aim of the *time plot view* is to provide a more detailed view of the clusters according to the temporal aspect at the chosen temporal granularity. Therefore, multiple time plots, one for each cluster, are shown simultaneously to allow their direct comparison to determine for instance simultaneous changes of different clusters. Furthermore, it provides visual hints pointing to temporal scales on higher granularity levels or time points that may provide additional information.

Visual Design: Each cluster visible in the *clustered graph view* is represented by a separate time plot in the *time plot view* (see Figure 4.2.6). The time plot shows either the average trend of the time series or the cluster representative. As the time series is often larger than the available amount of pixels, the data is typically shown on a coarser scale and thus details may be hidden. To support the user in finding those details even at finer

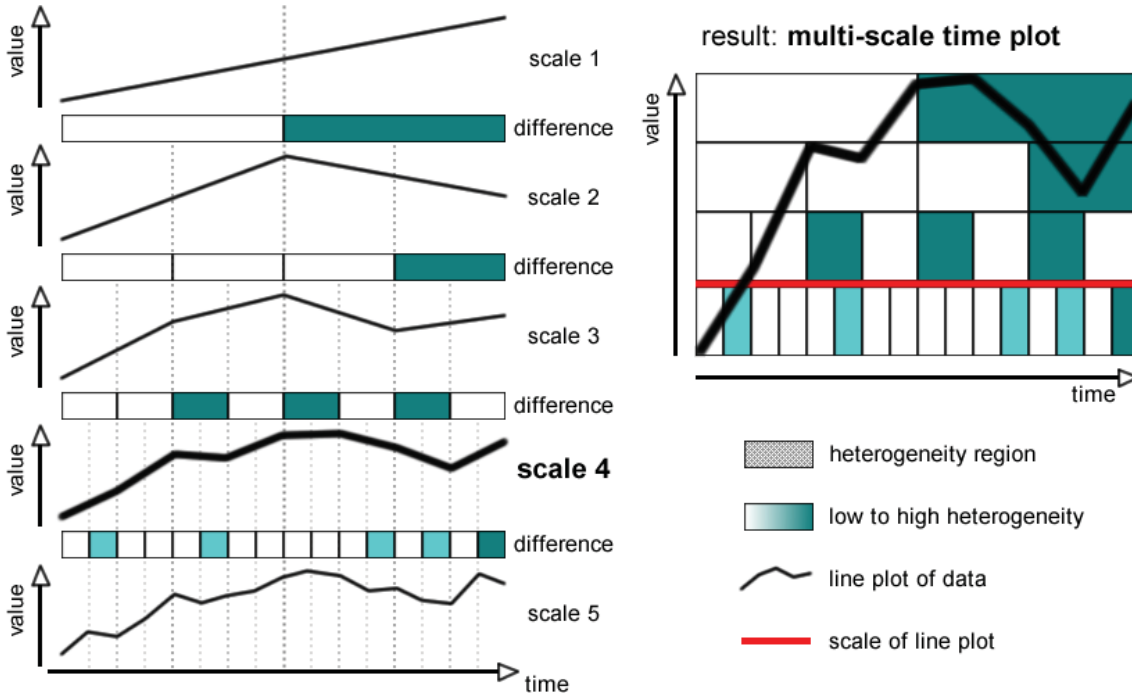


Figure 4.2.7: The construction of a multi-scale time plot according to [LSM⁺12]. Left side: Different scales of the time series. Between subsequent temporal scales, the heterogeneity is calculated and visualized as a color-coded band. Dark regions represent high heterogeneity and thus communicate high differences between the scales. Right side: These bands are stacked to form the background of the plot providing pointers to time intervals with more information. The scale above the red line is used to represent the time series.

scales, this view is enhanced by utilizing the idea of a multi-scale time plot as introduced in [LSM⁺12]:

First, for each time series a number of coarser scales are extracted for instance by iteratively applying the reduction described in the first step of Section 4.2.2.1. Between each pair of subsequent temporal scales, the heterogeneity is determined for example by calculating the differences in the slopes. This heterogeneity is then visualized as a color coded-band with dark regions representing high heterogeneity. This is shown on the left side of Figure 4.2.7. The background of the image is then constructed by stacking these bands with the coarsest scale on top. In this way, the background provides pointers to time intervals with more information than visible on coarser scales. For the visualization of the time series, a temporal scale with a resolution fitting the number of available pixels is chosen. This scale is communicated by a red line between the bands including this scale. The resulting multi-scale time plot is depicted on the right side of Figure 4.2.7.

Additionally, the standard deviation is visualized as a semi-transparent ribbon along the time series to communicate the similarity within the cluster over time. This allows the user to discern between clusters that are similar across the whole time series and clusters featuring a high dissimilarity only for a few time points.

With a hierarchical clustering, the time plots are accompanied by an *icicle plot* [KL83] showing the clustering hierarchy up to the selected layer. Each cluster is represented by

a colored rectangle representing its intra-cluster similarity. Here, the same color-coding is used as in the *clustered graph view* by mapping high similarity on a dark green color and low similarity on a white color. For those clusters currently visible in the *clustered graph view*, the corresponding rectangle also provides the number of nodes and edges it contains as an absolute number and as the percentage of the whole graph in brackets.

In this way, the *time plot view* provides insight into the temporal trends of each cluster as well as visual hints pointing to temporal scales on finer granularity levels or time points that may provide additional information. Moreover, together with the *icicle plot*, they show the similarity within each cluster.

Interactions for Exploring the Time Series: Besides zoom and pan for inspecting details in the time plots, filtering mechanisms are provided to cope with a high number of clusters shown in the *clustered graph view* allowing to visualize only the time series for those clusters that are most important to the user. For this purpose, he can either select a minimum number of nodes and edges a cluster has to contain or directly select the clusters to be analyzed in the *clustered graph view*. The *icicle plot* provides visual feedback on the filtering by showing the number of nodes and edges that were filtered in angle brackets.

Similarity View and Visual Comparison of Clustering Settings

The presented views communicate the information captured by the clustered supergraph. Yet, different clusterings that consider for instance different attributes or are based on different measures would typically offer further information. Hence, the user needs appropriate interactions to adapt the clustering to his analysis needs as well as visual cues for estimating the effects of different clustering settings.

Therefore, the user is allowed to define and configure two clustering settings at the same time. First, an active one used for calculating the clustered supergraph that is used for the visual exploration in the *clustered graph* and *time plot views*. And a second one used to provide visual cues of how the active clustering will be affected by changes to clustering parameters. At any time, the user can switch between the two settings allowing him to perceive the dynamic graph from different perspectives. For this purpose, a *similarity view* is provided.

Visual Design: The *similarity view* (see Figure 4.2.8) shows two clustering settings and provides controls for their adaptation. It visualizes a similarity matrix for each setting capturing the similarities between all extracted time series according to the selected similarity measure. To communicate possible clusters, the time series are ordered to form homogeneous blocks as described in [WTC08]. This relation is demonstrated in Figure 4.2.8 allowing the identification of two larger blocks as clusters. In this way, the similarity matrix supports:

- with a k-means clustering, the estimation of the number of clusters and the specification of cluster centers.
- with a hierarchical clustering, the extraction of an appropriate layer/cut.

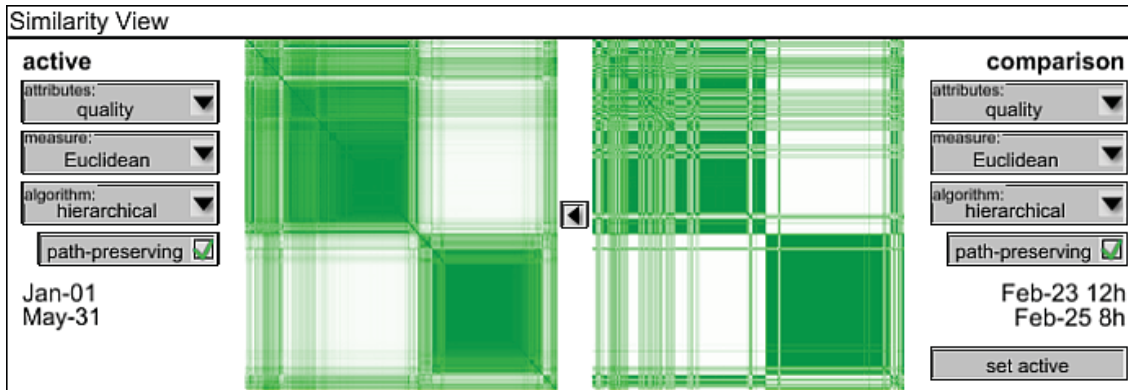


Figure 4.2.8: The similarity view shows the active clustering settings and the corresponding similarity matrix on the left side and the settings and matrix for a second clustering on the right side. High similarity of time series is mapped to dark green, low similarity to white. The second clustering was restricted to a smaller time interval (see Figure 4.2.17c for its selection). This resulted in a split of a large cluster visible from the breaks on the right side in the formerly homogeneous block.

The layout of the matrices is synchronized showing the same order of time series. In this way, similar matrices reflect similar clustering results while different results, e.g., a split of a cluster can be perceived as a break in one matrix compared to the otherwise homogeneous block of this cluster in the other matrix as exemplified in Figure 4.2.8.

Adaptation of the Views: While the *similarity view* allows for the comparison of two settings at the same time, the *clustered graph* and *time series views* only support the visual exploration of a single clustering result. Hence, two modes of visualizing both clustering settings are provided:

- Only one setting is visualized at any time. In this case, the *clustered graph* and *time plot view* are used as they have been described.
- One setting – the active clustering – is used as the basis of the visual exploration but additional visual cues according to the second clustering settings are embedded. For this purpose, the *clustered graph view* and the *time plot view* visualize the active clustering but are adapted as follows to also provide visual cues of the differences to the second clustering setting:

In the *clustered graph view*, the background of the glyphs used to represent the clusters is split into two parts (see Figure 4.2.9a). The lower left half is still showing the similarity within the cluster according to the similarity measure of the active clustering while the upper right half is showing its intra-cluster similarity according to the second clustering setting. The representation of the clusters in the *icicle plot* is handled accordingly. In this way, the similarity matrices as well as the glyphs communicate whether with a new clustering setting a current cluster would still form a cluster (for instance Figure 4.2.9b and 4.2.9c) or would break up (Figure 4.2.9d and 4.2.9e). Based on these visual cues, the user can estimate if a new clustering may yield additional information and thus if it is beneficial to calculate and explore a new clustering.



Figure 4.2.9: Adaptation of the glyphs for representing clusters to accommodate two clustering settings: (a) The background of the glyphs for representing clusters is split into two parts visualizing the intra-cluster similarity according to the selected similarity measure of both clustering settings. High similarity is mapped to dark green, low similarity to white. (b) and (c) exemplify a cluster that will remain after the switch, while (d) and (e) show a possible split of the cluster.

In summary, the *similarity view* provides an overview of two clustering settings and shows their differences. In this way, it allows an informed refinement of different clustering settings. Furthermore, the adapted views communicate changes in the similarity within the clusters. Altogether, all views – the *similarity view* as well as the *clustered graph view* and *time plot view* – provide valuable insight not only for the analysis of a single clustering but also for steering the refinement of the clustering settings.

Interactions for Refining the Clustering: For adjusting the settings, the following functionality is provided:

- choosing between different attributes, similarity measures and clustering methods as well as adapting specific parameters in the *similarity view* (see Figure 4.2.15 for adaptations of the number of clusters) allowing for the analysis of correlations between different attributes.
- selecting a subset of nodes and edges in the *clustered graph view* to be clustered according to different settings supporting the analysis of a specific cluster of interest.
- selecting a time interval in the *time plot view* (see Figure 4.2.17c) for which the time series are extracted to support the analysis of local trends and their comparison to global trends.

Adjustments of the settings of both clusterings are reflected in all views with different effects. Interactions with regard to the active clustering immediately lead to an update of the *clustered graph view* and the *time plot view*. However, adapting the second clustering only leads to an update of the background color of the glyphs in the *clustered graph view* and the *time plot view*. Therefore, the user can refine this clustering without losing his current view on the dynamic graph, compare the different settings and switch to this clustering if it shows significant changes.

Overall, the three views provide an overview as well as details of a supergraph clustering according to temporal trends of associated attributes. Furthermore, they reflect similarities and differences between two clustering settings. In this way, these views do

not only support the analysis of a single clustering result but also support an interactive and informed refinement of the clustering. This allows the analyst to determine global trends affecting the whole graph as well as to discover local trends only affecting specific substructures or time intervals and their comparison.

How this interplay of clustering, visualization and interactive refinement can support a visual analysis of a real world dynamic graph is discussed in Section 4.2.4 together with the following clustering of the temporal aspect.

4.2.3 Clustering of the Temporal Aspect

In this section, a first temporal clustering for dynamics graphs is introduced that groups time points which are similar according to the graph structure into states. By subsequently adding transitions between those discovered states this clustering finally results in a *state transition graph* capturing the temporal patterns.

4.2.3.1 State Transition Graph Generation

The goal of this approach is the identification of recurring temporal patterns in the graph's dynamics. Therefore, the dynamic graph is transformed into a state transition graph that captures recurring graph configurations in different states and the temporal sequence of states in transitions between them. A state transition graph promises to capture patterns in the graph's dynamics that would otherwise go unnoticed when viewed in a regular 1-dimensional fashion – i.e., as a linear sequence in a list of small multiples or in an animation. In this way, it is able to represent the graph's temporal progression in the complexity that it exhibits and not as a mere linear sequence. The procedure to extract such a graph is split into two steps:

1. Identification of the states by grouping similar graph instances into one state.
2. Adding transitions between two states if each subsumes at least one graph instance from subsequent time points.

In the following, both steps are discussed individually.

Step 1 - State Identification: Identifying states in the dynamics of a graph DG is essentially a clustering problem with states being clusters of time points rather than clusters of nodes. Similar to the clustering of the supergraph as discussed in Section 4.2.2.1 this demands for a distance/similarity measure and a clustering method from which multiple exists.

For this clustering, a distance function $DIST$ has to be defined that quantifies the difference between two graphs $G_i, G_j \in DG$. This difference can be broken down in the *structural differences* that aim to measure how much the actual graph structure (nodes, edges) differs, and in the *attribute differences* that aim to measure how much the attached attribute values (node attributes, edge weights) differ. As these differences can be calculated independently for nodes and edges, the distance function can be split up into four

individual calculations:

The **structural difference** between two **node sets** described by their symmetric difference, with the first term capturing all added nodes and the second term capturing all removed nodes from t_i to t_j :

$$\text{DELTA}_V(V_i, V_j) = |V_i \triangle V_j| = (|V_j| - |V_i \cap V_j|) + (|V_i| - |V_i \cap V_j|)$$

The **attribute difference** between two **node attribute sets**, with one DELTA_a for each $a \in A$:

$$\text{DELTA}_a(V_i, V_j) = \sum_{v \in V_i \cap V_j} |a_i(v) - a_j(v)|$$

The **structural difference** between two **edge sets** described by their symmetric difference, with the first term capturing all added edges and the second term capturing all removed edges from t_i to t_j :

$$\text{DELTA}_E(E_i, E_j) = |E_i \triangle E_j| = (|E_j| - |E_i \cap E_j|) + (|E_i| - |E_i \cap E_j|)$$

The **attribute difference** between two **edge weight sets**, with one DELTA_w for each $w \in W$:

$$\text{DELTA}_w(E_i, E_j) = \sum_{e \in E_i \cap E_j} |w_i(e) - w_j(e)|$$

These four components of the distance function DIST are each normalized to the interval $[0 \dots 1]$. In order to not obfuscate the overall description, the normalization of the function definitions above has been omitted, as it can easily be obtained by dividing all differences by the maximum value that occurs. They are combined into the overall distance function by computing a weighted sum using the weights α , β , γ , and δ , with all weights α through δ adding up to 1 to preserve the normalization:

$$\begin{aligned} \text{DIST}(G_i, G_j) = & \alpha * \text{DELTA}_V(V_i, V_j) + \sum_{a \in A} (\beta_a * \text{DELTA}_a(V_i, V_j)) + \\ & \gamma * \text{DELTA}_E(E_i, E_j) + \sum_{w \in W} (\delta_w * \text{DELTA}_w(E_i, E_j)) \end{aligned}$$

By adapting the weights of this distance function the clustering can be tuned to be more centered on structural or attribute related changes. Setting all structure related weights to 0 would allow the localization of events at which large changes in the associated attributes occur independently of any structural changes. For instance in a communication network where it is important to determine large drop-offs in the connection quality (attribute change) the addition of new subnetworks (structural change) may be of lesser concern. By using similar weights for all components, the influence of individual changes to the structure and attributes on the overall distance will be reduced and possibly drown out. That means, that smaller changes in the structure and the associated attributes may

only be perceived if they occur simultaneously, whereas larger changes may still be visible.

It has to be noted, that this distance function represents just a single possibility to calculate the difference between two graphs. Besides the weights, a different aggregation can be used to combine the individual components. For instance, a maximum combination of all differences may be able to maintain smaller changes concerning the different components in contrast to the weighted sum. Similar for each of the attribute difference, the individual differences of all nodes/edges may be aggregated differently, for instance using a squared sum emphasizing especially larger changes. In this sense, the distance function may be composed in a modular way similar to the DoI definition as will be introduced in Section 4.3.2 providing the analyst important parameters to steer the analysis.

The chosen distance function can then be used to parametrize a suitable clustering technique of which there are numerous in existence [BDKW07, XWI05]. In case of a hierarchical clustering method, graphs are clustered whose distance is below a given distance threshold d with:

- If this threshold is $d = 0$, it means that only identical graphs along the time line will be grouped into a state.
- If this threshold is $d = 1$, all graphs will be grouped into one single state – i.e., the *supergraph* over all time points.

In case of a k-means clustering method, the most similar graphs are grouped until the desired number of clusters is reached.

As the distance function compares the existence as well as the attribute values of each node and edge in both graphs, its computation is linear depending on the number of nodes and edges and thus has a runtime complexity of $\mathcal{O}(|V| + |E|)$ [BDKW07]. Depending on the complexity of the chosen clustering method [XWI05], the final runtime complexity of this step is bounded by $\mathcal{O}(|T|^2 * (|V| + |E|))$ as in the worst case, when using a hierarchical clustering, the distance has to be calculated between each pair of graphs in DG.

The result of the clustering are states $s = \{\dots G_i \dots\}$ that subsume a number of graphs G_i and is signified by a cluster representative, i.e. a representative graph or a mean graph (see Section 2.3.1.2). All of these states s define the node set called *STATES* of the state transition graph.

Step 2 - Adding Transitions: This step adds the transitions between the states. The procedure to do so is very similar to Algorithm 2, except that in this case transitions are added only if two states contain two graphs that stem from subsequent time points (cp. Line 7 of Algorithm 4).

The runtime complexity of this computation in its principal form as it is outlined above amounts to $\mathcal{O}(|T|^2)$ in the worst case of each individual time point being its own state. Yet, as a thoughtful implementation would have kept the information of which graph went into which state in Step 1, it could then instead simply traverse the time line, look-up this information, and assign transitions to the states that subsume subsequent graphs.

Algorithm 4 Adding Transitions to the States

```

1:  $TRANS \leftarrow \emptyset$  ▷ set of transitions
2: for  $s_1 \in STATES$  do
3:   for  $s_2 \in STATES \setminus \{s_1\}$  do ▷ for all pairs  $(s_1, s_2)$  of states
4:      $trans \leftarrow \emptyset$  ▷ create a new transition
5:     for  $G_i \in s_1$  do
6:       for  $G_j \in s_2$  do ▷ for all pairs  $(G_i, G_j)$  of graphs
7:         if  $i + 1 = j$  then ▷ if graphs are consecutive
8:            $trans \leftarrow trans \cup \{(G_i, G_j)\}$ 
▷ add them to the transition
9:       if  $trans \neq \emptyset$  then ▷ if subsequent graphs were found
10:         $TRANS \leftarrow TRANS \cup \{trans\}$  ▷ add transition

```

Such a speed-up would lower the runtime complexity to $\mathcal{O}(|T|)$.

The entire clustering procedure is depicted for a hierarchical clustering in Figure 4.2.10. Its runtime complexity is bound by $\mathcal{O}(|T|^2 * (|V| + |E|))$ resulting from the most complex step, the clustering of the time points. Similar to the structural clustering (see Section 4.2.2.1) this process can also be accelerated by choosing a coarser granularity as well as by precomputing parts of the computations such as the pairwise differences of all graph instances. In this way, the complexity of the temporal clustering can be reduced to the grouping of time points in an overall complexity of $\mathcal{O}(|T|^2)$. The result of these steps is a static state transition graph *STG* with the node set *STATES* and the directed edge set *TRANS*. Its visualization is discussed next.

4.2.3.2 Visualization of the State Transition Graph

The goal of the visual design is to support the visual exploration of the dynamic graph concerning recurring patterns in the graph's temporal evolution. Therefore, three interlinked views are provided targeting: the dynamics of the graph with respect to the chosen temporal granularity, the structure of the graph with respect to the chosen structural granularity, as well as the state transition graph identifying these temporal patterns. In the following, the views are described in more detail.

The Temporal Overview

The aim of the *temporal overview* is to give a static overview of the dynamics of the graph at the chosen temporal scale, yet before the more time-consuming clustering starts. It thus serves as a first, simple visual feedback, whether the graph exhibits any dynamics of interest at the chosen temporal granularity. If this view already shows unexpected behavior, such as sudden drops in the graph complexity or in the overall connection quality, the chosen temporal scale may be a good candidate for further exploration to find out about the observed behavior.

Visual Design: The *temporal overview* (see Figure 4.2.11) shows the entire graph condensed into a single numerical value for all time points at temporal granularity t in

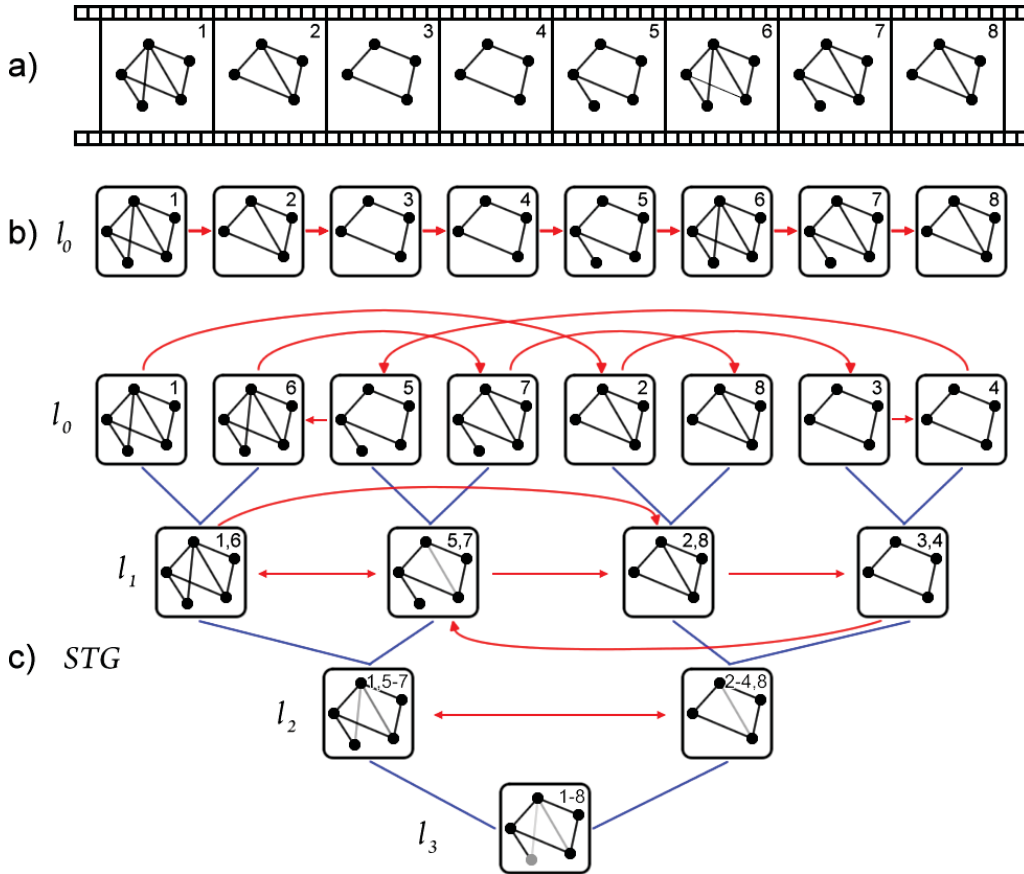


Figure 4.2.10: Overview of the state transition graph generation using a hierarchical clustering to identify the states. a) shows the dynamic graph as a sequence of snapshots. b) shows the lowest level of the hierarchical clustering in original order with red links describing the transitions. c) reflects the cluster hierarchy showing the stepwise reduction of states and aggregation of transitions. The final state transition graph is then based on a horizontal cut through this hierarchy according to a given similarity threshold.

a time-value-plot. The numerical value at each time point can be chosen by the user and can either subsume structural properties or attributes/weights. An example for the former are structural measures that capture different aspects of the graph topology [BB05, PD08] and which can be used to give a concise overview of a graph's topological evolution [JSS⁺13]. Examples for the latter are averages of the given node attributes or edge weights, computed over the entire graph. Note that these attribute values are taken from the meta nodes and meta edges of the structurally coarsened graph DG^s .

Interactions for Exploring the Temporal Aspect: The *temporal overview* is linked to the other views to facilitate the temporal exploration of the given graph along the linear time line. Selecting a time point by simply clicking in the view will not only highlight the graph configuration at this time point in the following *structural overview*. But also the state in the state transition graph into which this particular time point was clustered is highlighted. Furthermore, the clustering can be restricted to a smaller time interval by selecting it in this view.

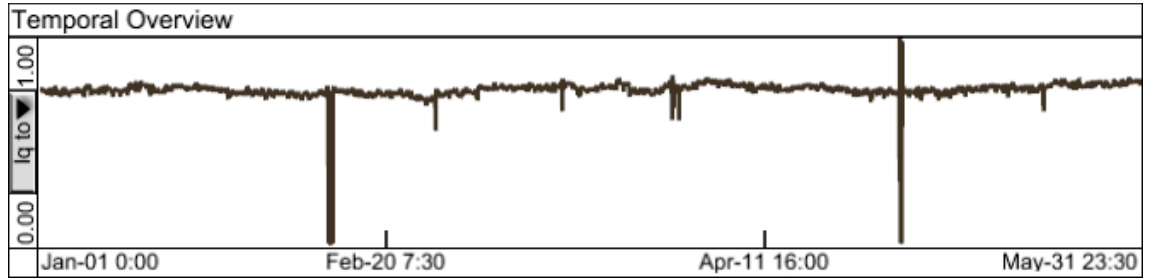


Figure 4.2.11: The temporal overview of the dynamic graph displays averaged attribute values over the entire graph for each time point of the chosen temporal scale as a time-value-plot. Each numerical value may subsume structural properties or average attributes/weights. In this case, an averaged edge weight is plotted at a half-hourly temporal scale.

The Structural Overview

The *structural overview* seeks to provide the same information about the graph structure, as the *temporal overview* provides for the dynamics: It gives the user a static overview of the dynamic graph at the chosen structural granularity. This is done before the possibly time-consuming clustering and provides visual feedback to the user, whether the chosen structural granularity shows any interesting topological patterns. Examples for such patterns are outliers, such as singular stable nodes in a fluctuating neighborhood of instability, or particular graph topologies, such as stars or cycles that are worthwhile to explore in depth.

Visual Design: The *structural overview* (see Figure 4.2.12) shows the entire evolution of the graph condensed into one static supergraph at structural granularity s in a node-link diagram. In case a time point or state has been selected, only those nodes and edges of the super graph are visualized that exist at the selected time point or state. Whereas the remaining nodes and edges are smoothly blended out allowing the comparison of different time points and states. In this way, this view builds upon the most common approaches for dynamic graphs: super graph and animation-based visualizations.

As it was the case for the *temporal overview*, either structural properties or numerical node attributes and edge weights can be depicted, in this case by color-coding them in the representation. An example of the former is the number of time points a node/edge is present out of the number of all time points at temporal granularity t . In this way, nodes/edges in the graph, which appear only shortly, will be shown in a less prominent color than those that are more stable and exist in the graph over many time points. Whereas examples for the latter are again averaged attribute values and edge weights, but in contrast to the temporal overview these are averaged over all time points of the coarsened graph DG^t and not over the entire graph structure.

Interactions for Exploring the Structure: The view is linked to the other views, so that selecting a node or edge in the *structural overview* highlights all time points in the *temporal overview* at which the particular node/edge is present. Furthermore, all states are marked

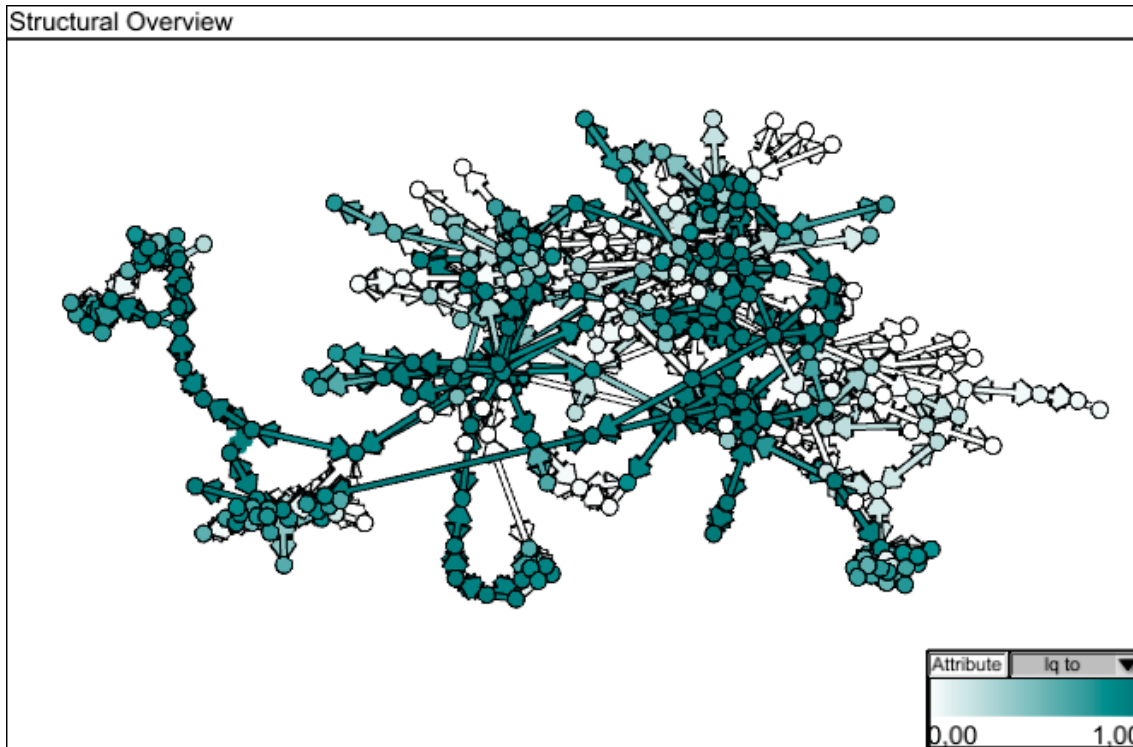


Figure 4.2.12: The structural overview of the dynamic graph visualizes its supergraph over all time points at the chosen structural granularity by a node-link representation. An averaged attribute of interest is mapped to the color of the nodes and edges.

in the following *state transition graph view* in which this node/edge exists. Similar to the *temporal overview*, the clustering may be refined to a subgraph by selecting it in this view.

The State Transition Graph View

The *state transition graph view* aims to provide a branching view of temporal patterns in the graph's dynamics by showing the dependencies between the states. Unlike the *temporal overview* with its linear display of the dynamics, it permits the user to identify cyclic graph patterns, dead-end states that lead nowhere but back to the state a particular graph configuration developed from, or star-shaped behavior with a common “normal” state in the center and a number of rare exceptional states gathered around.

Visual Design: This view (see Figure 4.2.13) shows the resulting state transition graph *STG* in a directed node-link-diagram, as it is the standard form of representation for such graphs [BBG⁺09, PIB⁺11]. To maintain the relative temporal order of states, a modified force-directed layout is used that anchors the states on the left or right side of the drawing area depending on the average time of their occurrences by adding additional attraction forces similar to the magnets introduced in [SF12]. That means, states that occur primarily in the beginning will be situated on the left side whereas states that occur more often in the end are situated on the right side. In this way, the temporal sequence can roughly be perceived and the shown sequences of states can be analyzed in their temporal context. To communicate the frequency of each state, the number of time points it subsumes,

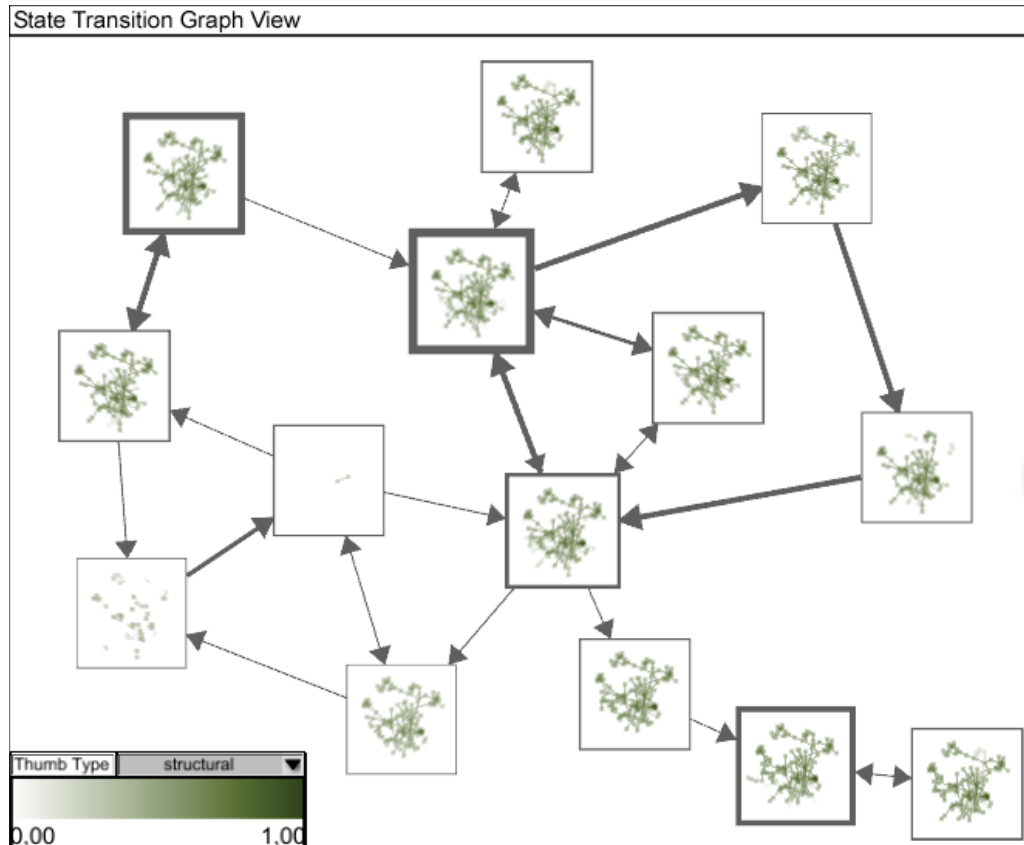


Figure 4.2.13: The state transition graph view shows the extracted temporal patterns of the graph's dynamics with all the cyclicity and complexity that it exhibits on the chosen level of structural and temporal granularity. Each state subsumes similar graph configurations with respect to the chosen threshold and shows a glyph of their representative graph. The border thickness of each state reflects the number of subsumed time points and the width of each transition reflects how often it occurs over time.

is mapped on its border thickness. Transitions between states are handled accordingly by mapping their frequency on the line width.

On this basis, a temporal pattern is defined as a connected subgraph of the state transition graph. Depending on the structure of these subgraphs different temporal patterns can be captured:

- **Recurring temporal patterns** consist of at least one state that occurs multiple times in the graph's dynamics.
- **Sequential temporal patterns** are paths in the state transition graph describing possible dependencies between states.
- **Cyclic temporal patterns** are cycles in the state transition graph describing recurring sequences of states.
- **Branching temporal patterns** consist of states having more than one incoming or outgoing transition describing alternatives or synchronization events.

Based on these patterns one may derive predictions. For instance, if a state is always following another state (a simple sequential pattern), the occurrence of the latter may serve as an indicator for the occurrence of the prior.

To support the interpretation of these temporal patterns, the states are represented by different types of glyphs communicating either a state's structural or temporal aspect.

- The **structural glyphs** show the representative graph of each state. Their layout follows the layout of the supergraph shown in the structural overview, so that a node will appear at the very same position in all representative graphs it is part of and in the *structural overview*. This eases the comparison of the different states among each other and with the supergraph shown in the *structural overview*. The layout is subsampled onto the smaller drawing area of the glyph. This is done by counting the number of nodes falling into each sample and color-coding the sample accordingly, so that higher node counts are mapped to a more intensive color.
- Whereas, the **temporal glyphs** provides a linear view of the distribution of time points in a small 1-dimensional heat map. Therefore, the temporal scale is binned into equal sized intervals and the number of time points falling into each interval is mapped onto the color.

Depending on which aspect is of interest to a user, he can switch the display between the two, as well as choose from other attributes to color-code onto them.

Interactions for Exploring the Temporal Patterns: The *state transition graph view* serves not only to reflect the oftentimes intricate behavior of dynamic graphs, but also as an interaction canvas to steer the analysis and bring up details in the other views. For example, it can be used to select states or transitions of interest and see them highlighted in the *temporal* and *structural overviews*. For a state, it marks the time points subsumed by the state in the *temporal overview* and highlights its representative graph in the *structural overview*. For a transition, it marks the time points between which the transition occurs in the *temporal overview* and highlights the changes in the graph topology it entails in the *structural overview*. This is done by mapping both states incident to the selected transition on different colors to allow for their comparison in the context of the *temporal* and *structural overview*.

4.2.4 Use Case

The use case² concerns a data set of link qualities in a wireless mesh network. In this network, nodes describe devices such as WLAN routers establishing wireless connections that are encoded in the edges. The data was collected in the *OpenNet*³, a wireless community network located in Rostock, North Germany. The network covers parts of the city and some villages in the surrounding region.

²Those parts of this use case section that are based on the clustering of the structural aspect have been published as "Supporting the Visual Analysis of Dynamic Networks by Clustering associated Temporal Attributes" 2013 in the IEEE Transactions on Visualization and Computer Graphics and resulted from a joint work with Till Wollenberg who was in charge of the application of this approach for the use case.

³More information in German language available at <http://www.opennet-initiative.de/>

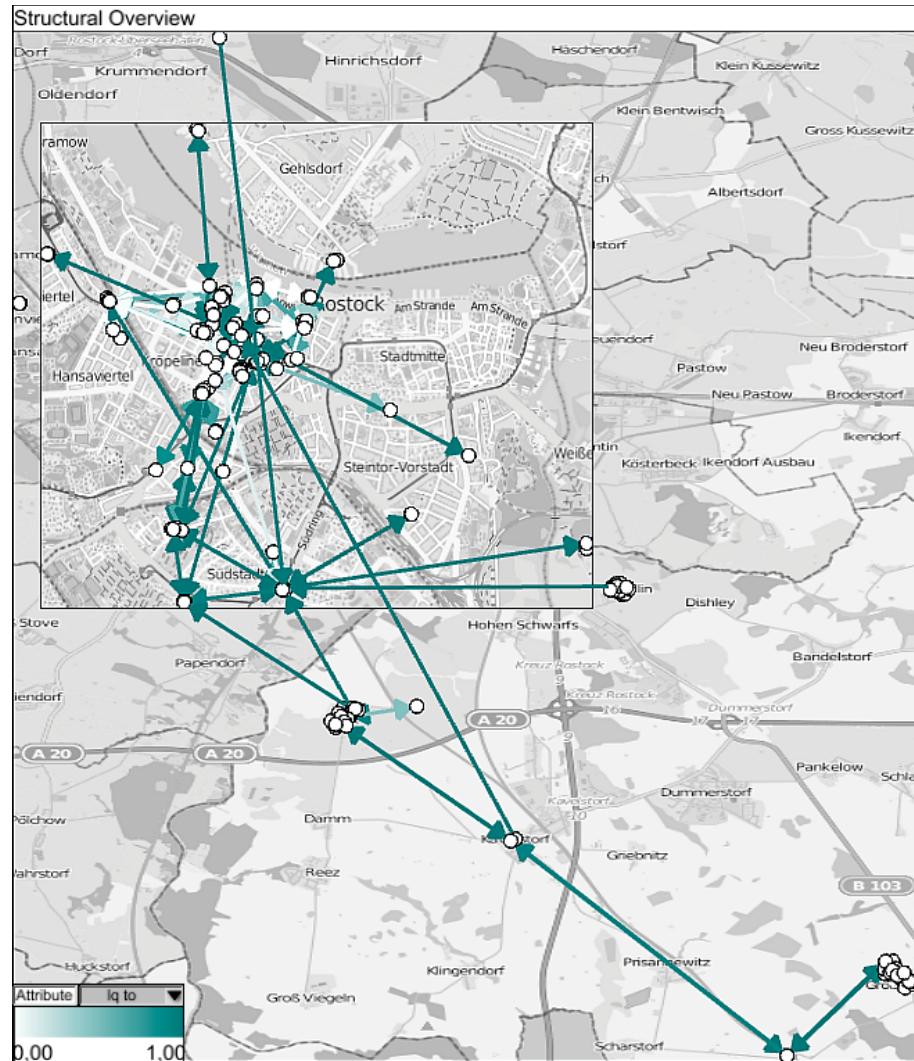


Figure 4.2.14: Structural overview of the OpenNet network. As almost all nodes have a fixed geographic position they are laid out on top of a map (© OpenStreetMap contributors). The link quality of their connections (edges) is mapped on the color. Dark represents high quality and light low quality. The region around the city center of Rostock is enlarged providing a more detailed view while maintaining the overall structure following the idea that will be presented in Section 5.2. Densely populated clusters are situated in the city center and three villages south of Rostock while other regions are sparsely populated.

4.2.4.1 Data Description

To give an impression of the network structure, Figure 4.2.14 shows the OpenNet nodes and links overlaid on a map in the *structural overview*. Almost all nodes are fixed in place and for the majority of nodes, the geographic position is available as a latitude and longitude tuple. It can be seen that the network has some densely populated clusters in the city center and in three villages south of Rostock while it is sparsely populated in other regions. It can also be seen that the clusters are connected with one or more long-distance links.

The network is based on IEEE 802.11 Wireless LAN technology and uses an extended

version of the Optimized Link State Routing protocol (OLSR, [CJA⁺03]). The original protocol classifies links as either working or non-working. This does not work well for wireless networks where intermediate packet loss rates are perfectly normal. The extended version OLSR-LQ [Lop13] accommodates this by associating a quality value with each link. A link's quality value LQ describes the probability of a successful packet transmission on this link. A perfect link, i.e., a link showing no packet loss at all, has an LQ of 1.0. Whereas a poor link, i.e., a link with a high packet loss, has an LQ near 0.0.

OLSR works proactively and belongs to the link-state class of routing protocols. As a result, each node receives information about all existing links and their qualities in the entire network. From the received information, each node builds-up a weighted graph representing the network with the weights reflecting the link qualities. This graph is then used to compute optimal paths to each node in the network.

For collecting the data set, the current list of known links was obtained from a particular node within the OpenNet every minute over a five-month period. The data collection took place from January 1st 2011 until May 31st 2011. During this time, the network graph contained on average 172 nodes and 762 edges. The supergraph of all 217'253 recorded snapshots contains 297 nodes and 2'008 edges. For 1'342 time points, no data is available due to outages of the recording node. The data set is described in more detail in [Wol12] and is publicly available⁴.

In this data set, existing links (edges) are described explicitly along with their respective link quality. If a link was not working at a particular point in time, it is missing in the respective snapshot. However, if this link was working at some time during the data collection period, it is included in the supergraph. In this case, it is convenient to have an explicit statement about the quality of any particular link at any particular point in time. Therefore, if a link from the supergraph is missing in a particular snapshot, the respective link is added and a zero quality is associated with it, which is semantically equivalent to a non-existing link.

In general, each link has a basic quality magnitude mainly determined by factors such as distance and antenna type. Beyond that, the link quality varies over time. These fluctuations may be caused by interference originating from other radio transmitters, strong load on the respective link or weather effects (e.g., increased attenuation due to moisture or antenna misalignment due to wind). When a particular link within a path deteriorates, the OLSR software normally selects an alternative path automatically. However, if no alternate links of good quality are available, a sudden strong deterioration of a single link may limit the overall connectivity of entire network segments.

For the following analysis, the dynamic network is abstracted on 4 structural levels the router level, the district level, the county level, and the state level using the geographical positions and on 11 temporal levels that are commonly used in this domain ranging from minutes to months allowing the identification of temporal patterns with different period lengths. While this approach represents a rather discrete sampling of scales, it serves as a first overview for identifying specific scales showing interesting patterns. On demand, further scales can be defined by the user to accommodate for patterns not fitting into the given scales.

⁴A subsampled and anonymized version of the data set can be found under: http://opsci.informatik.uni-rostock.de/index.php/Network_quality_measurements

4.2.4.2 Analysis Goals

The general goal of the OpenNet members is to maintain a network quality satisfying all users. Therefore, they need to identify low quality links that may cause low network performance as well as to react quickly to sudden drops in the link quality. Considering the number of links and time steps of the recorded data set, this may become a challenging task. In general, the OpenNet members are interested in analyzing the data on multiple levels:

- They want to monitor the network status on a *global level* and identify general problems early.
- They also want to localize the source of sporadic problems and determine which parts of the network were affected by them. In this case, their focus is on a *local level*.
- After detecting such recurring problems, they want to identify possible indicators to *forecast* them. These indicators may then help to propose recommendations to adapt the network such as rerouting the communication around such affected areas, as proposed in [WM09].

All goals are addressed separately in the following three sections.

4.2.4.3 Analysis on a Global Level

In order to solve the first problem, the OpenNet members want to identify which links persistently show a high quality, which links are fluctuating, and which links show a constantly low quality. Commonly, they would have to analyze each time point separately to filter out those links that frequently have a low quality or links that exhibit sporadic shifts in quality.

In order to reduce the burden of analyzing all time steps separately, a common approach would be to look at the supergraph only as shown for instance in the *structural overview* of Figure 4.2.14. Therefore, the time series data from each node and edge would need to be mapped to a few attributes that can be visualized in parallel. For example, the mean and the standard deviation of a link's quality could be mapped to color and style attributes of a line. In practice, this approach is problematic because time series mostly cannot be described adequately by a small number of attributes. In Figure 4.2.16b, an example for this problem is shown. Here, clusters *I* and *IX* show an almost identical mean and a similar standard deviation while it can be easily seen that their behavior over time is quite different.

In contrast, by using a structural clustering, the resulting clustered supergraph serves as a compact overview allowing the integration of time plots which show for all cluster the temporal trends of their attributes over time in more detail. In this way, with the new approach, this goal can be reached much easier and without this ambiguity.

As they are interested in links showing generally a similar trend over time, a structural clustering at the finest structural scale and a coarser temporal scale such as days is a good first choice for a first parametrization of the clustering. Furthermore, an attribute and a similarity measure have to be chosen to extract the nodes and edges they are looking

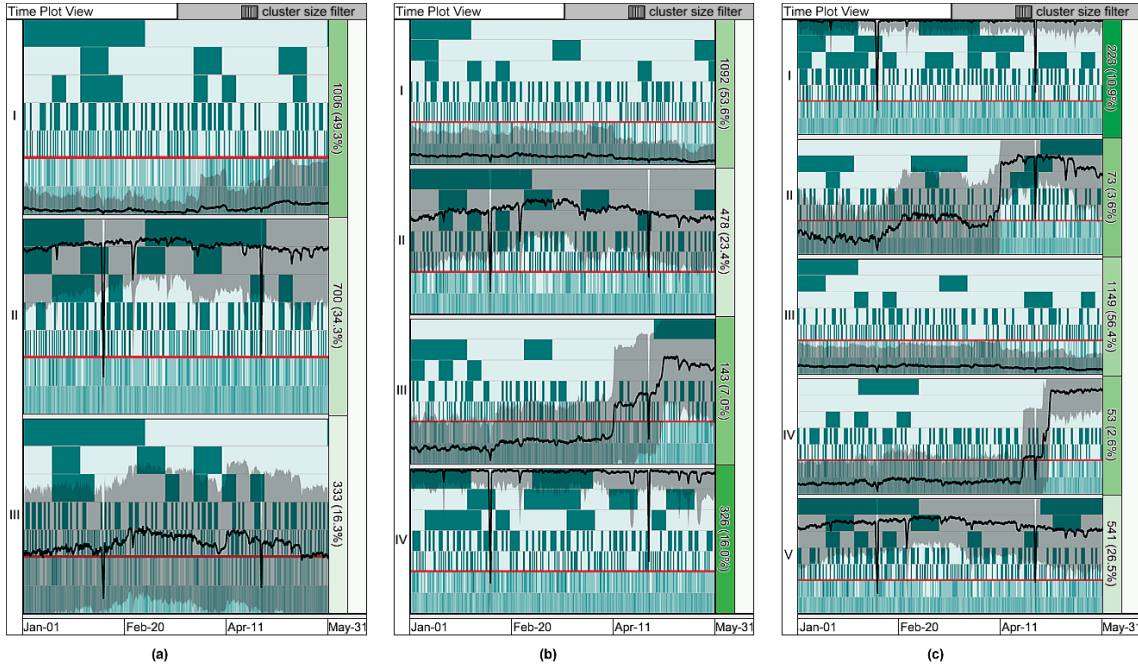


Figure 4.2.15: The temporal trends for three different structural clustering settings based on k-means. (a) $k = 3$: The averaged link qualities support the assumption of three distinct quality levels with cluster *I* containing poor links (low values), cluster *II* good links (high values), and cluster *III* average links (intermediate values). Clusters *II* and *III* exhibit a low intra-cluster similarity which implies that they contain divergent time series, which is confirmed by switching to a larger k . (b) $k = 4$: The intra-cluster similarity of all clusters increases and a new trend arises. Cluster *III* seems to contain nodes and edges that belong to a part of the network that was connected to the core network in April 2011 for the first time. (c) $k = 5$: A further increase of k improves the overall intra-cluster similarity. Yet, still some clusters with a high intra-cluster dissimilarity remain. The previous mentioned cluster *III* splits into the new clusters *II* and *IV*.

for, i.e., the nodes and edges that exhibit a similar behavior. As they are interested in grouping them according to their connection quality over time, the Euclidean distance measure calculated for this attribute is a good choice.

The network monitoring and graphic tools currently used within the OpenNet distinguish three levels of reliability of the links (good, average, poor) based on their link quality. Therefore, a k-means clustering method is used to start the analysis with an initial $k = 3$. This choice is also supported by looking at the *similarity view* depicted in Figure 4.2.8. There, one can see two large homogeneous blocks representing two distinct clusters and a few smaller ones.

The time plots of the three resulting clusters are shown in Figure 4.2.15a. The averaged link qualities support the initial assumption of three distinct quality levels. Cluster *I* exhibits low LQ values corresponding to poor links, cluster *II* shows high LQ values and thus captures good links, and cluster *III* contains average links. However, clusters *II* and *III* exhibit a low intra-cluster similarity as can be seen from the rectangles on the right side showing light colors. Their intra-cluster dissimilarity implies that they contain multiple,

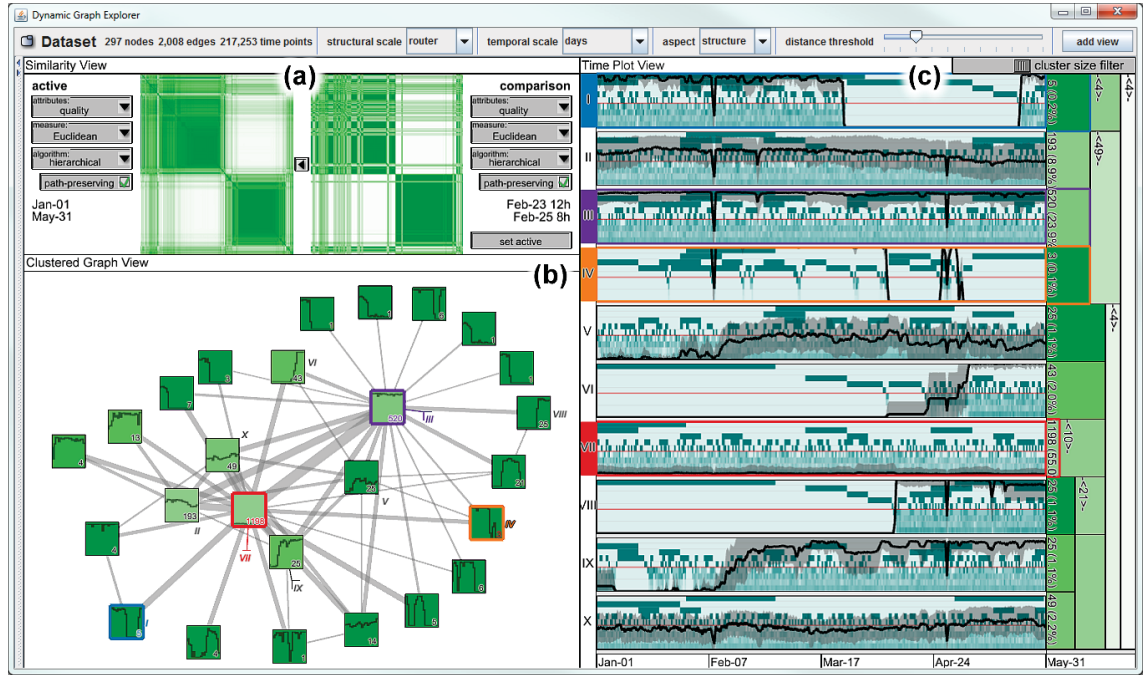


Figure 4.2.16: A view setup for the analysis of a hierarchical structural clustering according to a similar link quality. It consists of a similarity view (a), a clustered graph view (b), a time plot view (c) and menu bar to configure the clustering (selection of structural and temporal granularity etc.). Clusters (I, III, IV and VII) selected for a closer inspection are highlighted in different colors in the meta graph and time plot view.

divergent time series.

This implication is confirmed by choosing different, larger k as depicted in Figures 4.2.15b and 4.2.15c for $k = 4$ and $k = 5$. With increasing number of clusters, the intra-cluster similarity of all clusters rises. While the “good” and “poor” clusters remain, the divergent time series are captured in the emerging clusters.

The clusters III in Figure 4.2.15b as well as II and IV in Figure 4.2.15c seem to contain nodes and edges that belong to a part of the network that was connected to the core network in April 2011 for the first time. This can be seen from the fact that the elements of both clusters show a constantly low (close to zero) LQ value until April while their average LQ value is high and shows only few fluctuations afterwards.

Even with increasing k , some clusters remain heterogeneous. So, by switching to the hierarchical clustering method, instead of specifying the number of clusters to be extracted, a desired level of intra-cluster similarity can be defined directly. The result of the hierarchical clustering is shown in Figure 4.2.16. Generally, two groups of clusters can be differentiated:

- The first group shows a low variance in their LQ values over time. Yet, each cluster has a distinct quality level. This group contains the clusters II, III, VII and X. Here, cluster III captures the steadily good links, clusters II and X the average links and cluster VII the constantly poor links.
- The second group contains clusters showing a higher variance over time, i.e., they

exhibit different quality levels at different time intervals. This group consists of the clusters *I*, *IV*, *V*, *VI* and *IX*. For example, clusters *VI* and *IX* capture the part of the network that was connected to the core network in April 2011. As the links were not present until April 2011 they show a constant quality of 0.0. Yet, after they were established, they exhibit generally high LQ values and thus fall into the good quality level.

Both groups describe interesting features to be analyzed further. However, the second group represents only a minority of nodes and edges (as indicated by the labels in their glyphs in the *clustered graph view* and in the *icicle plot*) and is therefore of lesser interest for the OpenNet members. In contrast, the majority of nodes and edges are located in the first group. When looking at the clusters *II* and *III*, one can see that one third of all nodes and edges exhibit a constantly good or high quality while the cluster *VII* shows that nearly 50% of all nodes and edges in the network have a consistently low quality.

The large amount of low quality edges does not imply a low network quality in general because a weak link only has an effect on the network quality if there are no better links available to reach a certain destination. Weak links have a significant impact if they are the only connections between substructures in the network. The impact is particularly strong if the connected substructures have a similar high quality otherwise. In order to distinguish weak links with significant impact on the network from links which are weak but redundant, the path-preserving clustering is very useful. During the path-preserving clustering, this constellation results in a split of the cluster containing these substructures into multiple components.

The four selected clusters in Figure 4.2.16 illustrate the different cases. In the *clustered graph view*, the cluster *I* (colored blue in lower left corner) is connected to another cluster with high quality. Yet, both are connected to two other clusters that have a constantly low quality. Further investigation shows that these two clusters contain three nodes which are located in the west of the city center at the border of the OpenNet network. With this information at hand, OpenNet members could try to improve the links that connect these nodes with the remaining network. The figure also shows an orange colored cluster (*IV*, lower right corner) that is connected to the low-quality cluster *VII* (red). However, this cluster is likewise connected to a cluster of nodes and edges that show a constantly high quality (*III*, purple color). Thus, the periods of low quality in this orange colored cluster cannot be explained solely by the quality of the intermediate clusters but require further investigation.

4.2.4.4 Analysis of Local Problems

Identifying such sporadic changes in the quality of the clusters such as temporary outages is important to address the second analysis problem. A traditional approach to identify such local outages would be to look for significant drops in the quality in the individual time series of every node and edge. For assessing the scope of an outage, the user would then need to compare the different time series to find nodes and links with similar behavior. Considering the large number of links within the OpenNet this would be a time-consuming task. Alternatively, animations or small multiples could be used to identify sudden drops in the link quality. However, when analyzing large time spans this

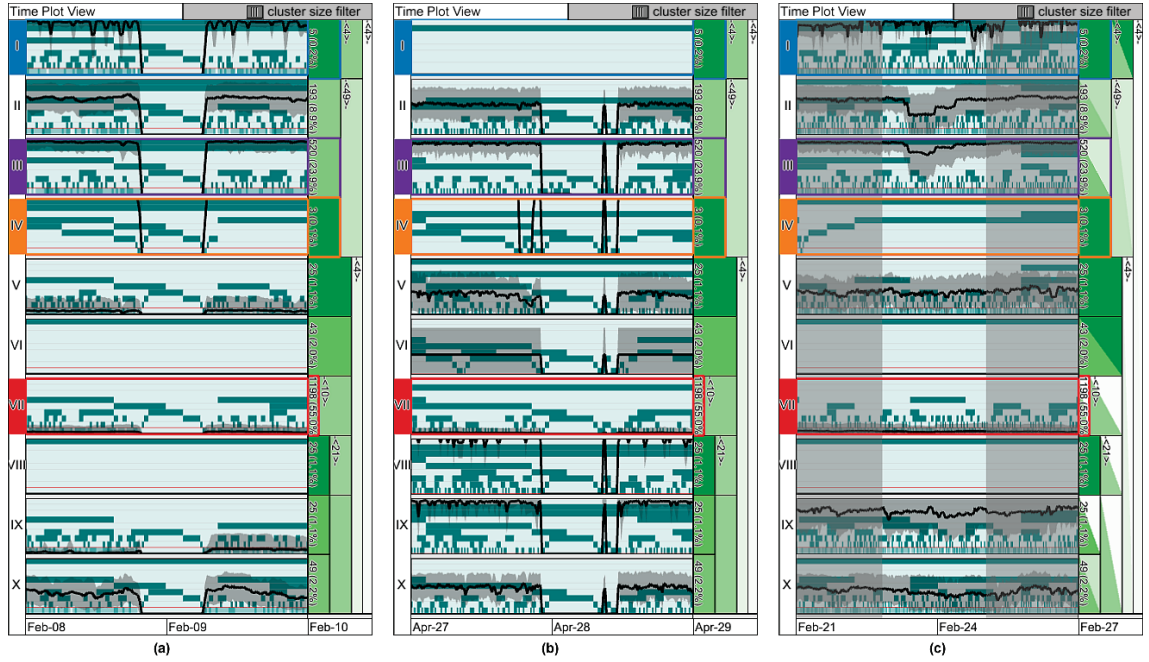


Figure 4.2.17: A zoomed-in view of three time intervals showing drop-offs in the quality of clusters with a generally good quality otherwise. (a) and (b) show a global drop-off of all clusters simultaneously. In both cases the recording node was disconnected and thus these intervals contain no data at all. On the other hand, (c) shows a larger drop-off only in the clusters *II* and *III*. Its cause has to be a local event. A time interval around this event is selected for closer inspection (time points outside of this interval are faded-out in gray). The icicle plot on the right reveals that the intra-cluster similarity of both clusters in the selected time interval is lower (light color of the upper right triangle) than for the complete interval (darker green color of the lower left triangle). If a new clustering is calculated on the basis of this time interval the degradation of their intra-cluster similarity will lead to a split of these clusters into multiple components.

would become time-consuming as well. Assuming that each time step is only shown for a second, it would take two and a half days to animate through all 217'253 time steps.

With a structural clustering, identifying sporadic, strong changes of link qualities and estimating their scope becomes much easier. Based on the global overview described in the previous section, local events in the different clusters concerning important links or nodes can be identified. Links and nodes are important if their outage would lead to a separation of parts of the network from the remaining core network. Due to the structure of the OpenNet, such effects usually affect local subgraphs. In this context, the second group of clusters seems interesting because of the high variance in their LQ values over time. However, as this group contains only a small number of nodes and edges, their influence on the overall functionality of the network is limited. Hence, this group is of lesser importance for the OpenNet members. They are more concerned about the clusters *II* and *III* of the first group showing a steadily average to good LQ over time. In this way, these two clusters describe the core network and a drop in their quality can have a more severe effect on the network.

By looking at the clustering as shown in Figure 4.2.16c, one can see that despite their



Figure 4.2.18: The supergraph is clustered according to the time interval as selected in Figure 4.2.17c. The time points outside of this interval are faded out in gray. The clusters are laid out on top of a map (© OpenStreetMap contributors). Cluster VII which shows a generally good link quality is highlighted in purple and the clusters (III, IV, V and VI) showing the drop-off are highlighted in red to orange. The network view reveals that these four clusters completely contain three connected villages dropping out at the same time. This was caused by a power outage in that region.

generally good quality both clusters show drops in the quality that are interesting for a further inspection. In the *time plot view*, a zoom-in is performed on the three largest peaks visible in clusters II and III (see Figure 4.2.16). The result of the zoom operation is shown in Figure 4.2.17. The peaks were caused by two different incidents. The two largest peaks are the result of a disconnection of the data recording node and thus only point to missing data. This is visible in Figure 4.2.17a and 4.2.17b as this event has a global effect on the time series of all clusters. Yet, there is a peak visible for only two clusters in February that is not caused by the data gap and thus shows a more local incident.

Next in order to narrow down the cause of the peak, a new structural clustering of the network is calculated by selecting the time interval around the peak (Figure 4.2.17c) and switching to a finer temporal granularity, in this case: hours. The result of this refinement is shown in Figure 4.2.18. The *similarity view* and the *time plot view* show that the peak was caused by a subset of nodes and edges that completely lost their connection while other parts of the OpenNet remained functional. Clusters III, IV, V and VI (colored in orange to red) consist entirely of disconnected network elements. The *clustered graph view* (Figure 4.2.18) reveals that these four clusters contain nodes and edges located in three villages south of Rostock. Further investigation reveals that all three villages were disconnected from the core network on February 9th due to a power outage affecting some nodes that connected the villages with the city network (cluster VII highlighted in purple).

4.2.4.5 Forecasting Recurring Problems

This simultaneous drop out of three villages describes a major incident affecting larger parts of the network. In this sense, it is an important goal to determine if such a problem has occurred multiple times and if it is possible to forecast such an event. As the disconnection of the villages has shown, severe problems having a high influence on the network may already be found on a coarser structural scale such as the district level. In this way, smaller fluctuations, for instance the disconnection of singular nodes, can be filtered out.

Recurring problems are generally difficult to identify with common techniques as most of them focus primarily on the structure and present the graph's dynamics in a linear way as an animation or as a series of small multiples. Using animations for instance, the analyst would have to go back and forth through time to first compare different graph instances and finally to identify recurring graph states. Identifying cycles or branches in the dynamics becomes even more time-consuming as it involves the search for sequences of different states. Yet by using a temporal clustering, the analyst is being relieved of this tedious work as the derived state transition graph already contains these states and directly reflects cycles and branches in its structure. Instead of searching the temporal patterns, the analyst is now able to concentrate on the actual analysis of these patterns, for instance to predict rare events always following a specific state.

For determining the different states the quality of a connection is as important as its (non-)existence. Therefore, equal weights are assigned to the structural and attribute differences concerning the connection quality for the distance function (see Section 4.2.3.1). To steer the compromise between a few but more general states and many detailed states the distance threshold for the clustering has to be selected. In this case, a threshold of 0.2 proved to be sufficient as it filtered out smaller fluctuations in the connection quality but remained detailed enough to capture disconnections of districts.

Based on the selected parameters, the state transition graph is extracted for a temporal scale of half an hour, which is quite detailed but also filters out smaller fluctuations. The result of this extraction is depicted in Figure 4.2.19. It shows three frequent states (indicated by the thicker borders and marked with s_1 – s_3) following a nearly linear sequence and a few rare states contained in some cycles. Analyzing the structural overview according to the glyphs of the three frequent states reveals that the network is almost completely functional most of the time. Besides some missing edges, all districts and villages are connected and thus a communication may be established between all of them. While these states represent the preferred status of a communication network, they are of lesser interest for the goal of finding outages. Yet, on a finer structural scale they could still exhibit connection problems within the districts. Switching to the finest structural scale – the router level – as shown in Figure 4.2.20 reveals that there are indeed some larger changes compared to the previous view. These changes eventually resulted in a split of the state s_2 . But altogether, the frequent states remained unchanged capturing only different development stages of the network. In this way, the grouping of similar time points into much fewer states supports the analyst in ruling out time points of lesser concern and allows him to concentrate on a much smaller number of time points.

From the remaining rare states, two states (s_6 and s_8) show larger abnormalities. Dur-

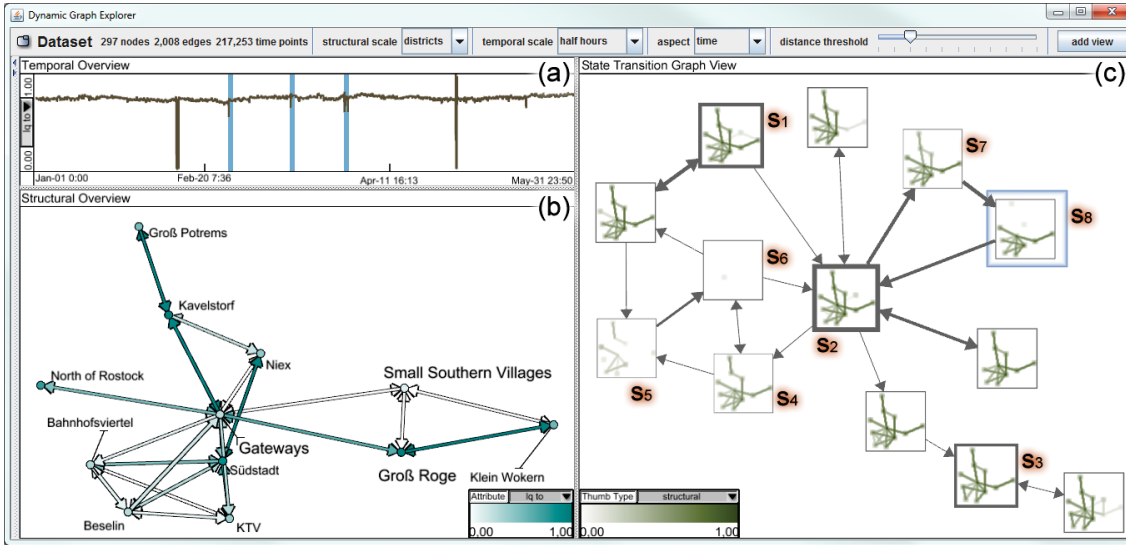


Figure 4.2.19: A view setup for the analysis of a temporal clustering for exploring branching patterns of the dynamic network. It consists of a temporal overview (a), a structural overview (b), a state transition graph view (c) and menu bar to configure the clustering (selection of structural and temporal granularity etc.). A state capturing the disconnection of three villages and its subsumed time points have been highlighted in blue color in the temporal overview and the state transition graph view.

ing the time points of state s_6 the whole network seems to be offline as its glyph shows only an empty rectangle in both Figures 4.2.19 and 4.2.20. These time points are also very prominent in the temporal overview showing a drop-off of the quality to zero. State s_8 depicts a more confined, local event visible in the upper part of its glyph. This state actually captures the disconnection of the three villages and after it has been selected the *temporal overview* reveals that it has occurred three times during the recorded time interval (time intervals highlighted in blue color). As the city center contains considerably more nodes and edges than these villages, this event is nearly drowned out in the glyph at the router level (Figure 4.2.20). Here, working at multiple granularities proves useful as their disconnection is perceived better on the coarser district level (Figure 4.2.19). With common techniques such as animations, the analyst may also be able to identify these events. But he would have to jump back and forth through time to analyze the network at different time points to first determine problems and then he would have to compare them manually to finally find similar recurring patterns. Here, the state transition graph relieves the analyst of this tedious back and forth by already providing a higher level summary of these patterns. This is especially important for the next step.

After recurring problems have been identified, such as the two aforementioned disconnections, the OpenNet members are interested in indicators pointing to a potential decline in the network quality for specific areas. Based on such indicators, the routing algorithm could automatically reroute the communication around those areas just before the connection quality significantly deteriorates. With common methods, the analyst would have to manually check all time points prior to the found problems and compare them for similarities. Yet, these temporal relations are directly captured in the structure

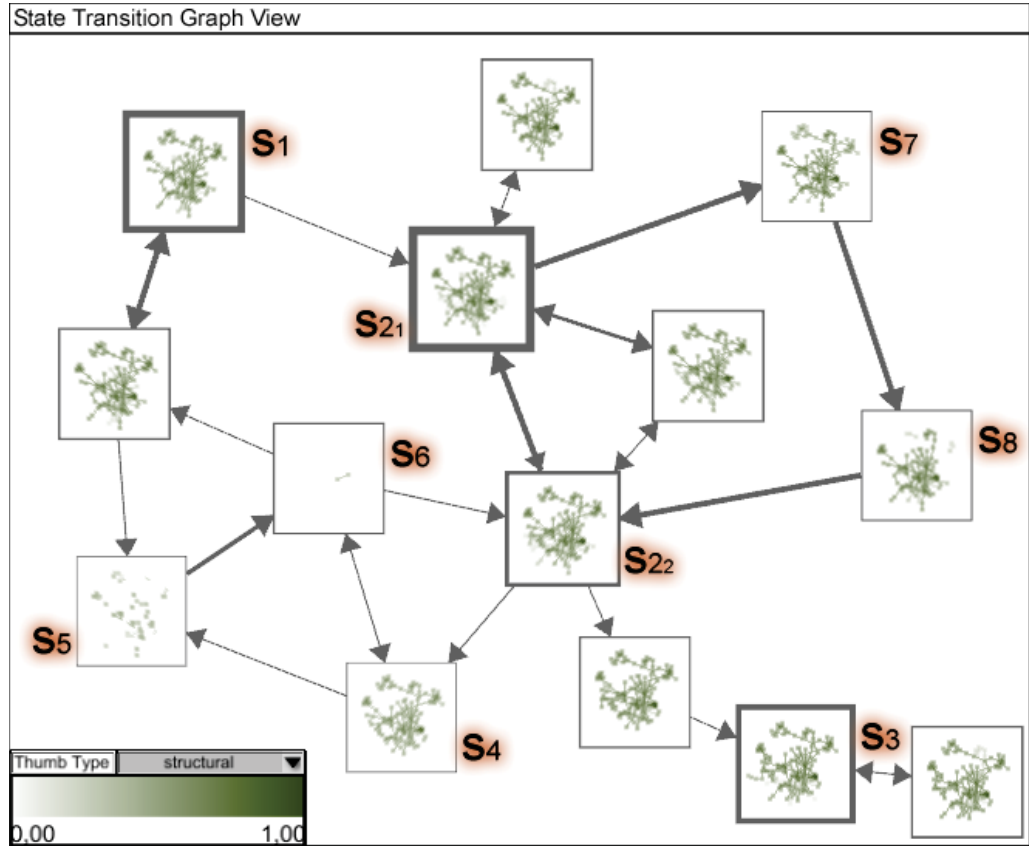


Figure 4.2.20: The state transition graph generated for the finest structural scale showing the individual routers at an half hourly temporal scale. In comparison to Figure 4.2.19, the structural glyphs at this finer level show a more detailed but also denser and cluttered topology revealing additional differences between the states leading to a split of the former state s_2 in two states s_{21} and s_{22} .

of the state transition graph. As shown in Figures 4.2.19 and 4.2.20, both problematic states are found in cycles in which they are preceded by singular states. In case of state s_6 , which occurred during the time intervals of both s_1 and s_2 , it is preceded by the states s_4 and s_5 . However, the state s_4 occurs before and after the problematic state, as it is reflected by the bidirectional transition between s_4 and s_6 and also leads to another state. In this way, s_4 is no definite indicator, whereas the state s_5 leads exclusively to the problematic state. Yet, s_5 already shows large parts of the network being unavailable and may thus already be too late. Contrary, state s_8 is always appearing after state s_7 , which only shows a slight drop-off in the connection quality. Thus, the occurrence of a network configuration similar to those subsumed by s_7 may pose as a first indicator to predict outages such as s_8 .

4.2.5 Conclusion

In this section, a novel structural and temporal clustering for large dynamic graphs has been presented that either groups nodes and edges by the temporal trend of their associated attributes or time points by their graph structure. An appropriate visual design based on the clustered supergraph or on the generated state transition graph provides a

scalable overview of the dynamic graph and its inherent branching behavior over time as well as detailed information on the groups of nodes, edges or time points and their trends or patterns. The integration of visual cues communicating the effects according to different clustering settings does not only allow for showing the clustered dynamic graph, but also for estimating the information yield when adapting clustering settings. Coupled with specific interactions, these visualizations allow for an scalable exploration as well as an interactive and informed refinement of the clustering. Altogether, this interplay of computation, visualization and interaction supports the user in discovering and analyzing global and local trends affecting different parts of the graph as well as to forecast recurring problems. A first use case study for a wireless communication network with data captured over a long period of time has shown its scalability as well as its usefulness and advantages over existing techniques.

Overall, the proposed reduction approaches provide the necessary means to freely explore large dynamic graphs in a scalable way. Yet, they pose no replacement of common techniques but are useful additions that work in tune with them. Actually, the visualization setups of both approaches build upon common techniques such as small multiples, time plots or super graphs to depict the temporal and structural aspect as first overviews but also for more detailed views on the clusters, states and transitions. As these approaches effectively reduce the number of nodes and edges as well as time steps to be explored they actually support the existing techniques to be again a feasible option for the analysis of large dynamic graphs. For instance, as a super graph aggregates all graph states into a single static image it is often not an appropriate technique for identifying particular recurring temporal patterns. Yet, the fragmentation of the time axis, which is imposed by the state identification step of the temporal clustering, ensures that the super graph is only calculated over similar graph configuration when analyzing specific states. During their analysis, animation techniques are furthermore utilized for a smooth transition yet not between a large number of individual time points but between the identified states.

Still, for larger data sets where most of the details have been abstracted away, one might wish for more guidance towards specific patterns of interest. Without such guidance, it can become quite tedious to explore the graph's dynamics in their entirety. A possible basis for such a guidance is introduced in the following section allowing the definition of interesting nodes, edges and time points according to their properties.

4.3 A Modular Degree-of-Interest based Selection for Dynamic Graphs

While abstraction approaches as described in the previous section grant an overview of the general structure as well as of the general dynamics of large dynamic graphs, accessing specific details in a large dynamic graph requires appropriate selection methods. On the one hand, direct selections allow to focus on elements (subgraphs or time intervals) that are currently visible in an abstract overview. But at the same time, they require a possible time-expensive calculation of that overview and only allow the selection of those elements that are actually visible and thus accessible.

To counter these problems on the other hand, often more indirect selection methods are used that are based on the characteristics of nodes, edges, and time points. A prominent example for such an indirect selection method is the utilization of DoI functions to capture the user interest in the nodes, edges and time points. With such a DoI function interesting elements are then extracted and visualized without the need of an overview first [vHP09]. Yet, overviews may also be enhanced to guide to interesting elements [GST13]. In this way, DoI functions enable a scalable visualization of large graphs by specifying which parts of the graph to be shown.

While DoI functions have been studied extensively for graphs, they are mostly given as final, fixed monolithic solutions tailored to a specific application domain with specific tasks and specific types of graphs as a brief survey of existing DoI-based graph visualization approaches in Section 4.3.1 shows. Yet, to be able to specify the interestingness regarding the complete diversity of these graphs – their structure, their associated attributes as well as their spatial and temporal context – which may include to perform tasks not anticipated by the fixed functions a more flexible DoI definition is necessary.

Therefore, this section⁵ introduces a novel modular DoI definition that exceeds the existing DoI-based approaches in two aspects:

- it can be utilized to hierarchically compose and recompose DoI functions on the fly, and
- it explicitly incorporates means to capture the dynamics of a graph.

Based on this modular DoI definition, a semi-automatic approach consisting of a user-driven definition and parametrization and an automatic evaluation of the DoI function supports the selection of specific elements even for larger graphs. Therefore, a visual analysis setup that shows the dynamic graph according to the DoI values and that permits the interactive adjustment and steering of the DoI computation in all its aspects is presented. This effectively breaks open the “black box” of DoI computation, as it not only enables a user to see whether a part of the graph exhibits any dynamic feature of interest, but also to explore which feature(s) concretely lead to the display of a part of the graph.

The scalability and utility of this DoI-based analytical process are then illustrated by taking a closer look at the top authors of the largest connected component of the DBLP

⁵Parts of this section have been published in “A Modular Degree-of-Interest Specification for the Visual Analysis of Large Dynamic Networks” 2014 in the IEEE Transactions on Visualization and Computer Graphics [AHSS14].

data set [Ley09] with its 914'492 nodes and 3'802'317 edges over 22 time points (years). It serves to give an impression of how different DoI definitions are built, adjusted, and used to gain insight in the dynamics of this large graph – something that would be hard to do with any other approach existing to date.

4.3.1 Existing DoI Specifications for Graphs

One of the first DoI functions was specified by Furnas in 1986 for static hierarchies [Fur86]. It already considered two aspects of the data: the user interest in each node (inverse to the distance to a currently selected focus node) and an attribute also referred to as an a priori interest inherent in each node (inverse to its distance to the hierarchy's root). These two values are summed and then used to generate a contiguous subtree of high interest by pruning subtrees of lower interest. In this way, the limited screen real estate is used to convey only the subtree of highest interest. Furnas' overall idea is still the same until today and was merely extended in subsequent research to cope with

- other a priori interest values (attributes) than the distance to the root,
- networks instead of trees,
- multiple user-selected focus nodes instead of a single one,
- and dynamic instead of static graphs.

Other a priori interest values than the rather specific distance to the root are regularly defined to capture interest according to certain application domains or associated attributes. Examples are the definition of a priori interest for nodes representing concepts in ontologies [HZ07] and importance values for nodes corresponding to given search terms [vHP09]. It is very common that a single DoI function combines multiple such a priori interest values, whose influence on the overall DoI value can be adjusted through adaptable weights. As a result, a user can no longer distinguish which of the a priori values triggered a node to receive a high combined DoI value. To communicate the cause for a node's high DoI value, most approaches simply use different colors [CSP⁺06, CLWM11]. Furthermore, it has been observed that the combined DoI values may not necessarily be "well distributed", with lower values at the leaf nodes and higher values closer to the root node, as it would be needed for pruning [vHP09]. To solve this problem, a priori interest values are often aggregated bottom-up, so that they are guaranteed to be monotonously increasing towards the root.

For networks, this requirement of strictly increasing interest values cannot be guaranteed as there is no singular (root) node towards which to aggregate them. To nevertheless counter the problem of heterogeneously distributed interest values, DoI-based approaches use, for example, a diffusion of DoI values across the network [vHP09] to even them out. Script-based approaches can be used to tailor such a diffusion process, for example, to only diffuse the values to every other node, as it makes sense for bipartite networks [SJUS08]. The same problem of a missing singular reference node also affects the representation of networks that are to be reduced according to a DoI function: For the

pruning of subtrees, it was always apparent where details were removed (at the bottom). Yet, this is no longer the case for networks in which nodes may get removed or collapsed into meta nodes in various places. In the latter case of a partially collapsed network, glyphs are often used to highlight meta nodes among the uncontracted nodes and thus to inform a user where information is hidden from his view [HC04, RPD09]. In the former case of nodes having been removed or filtered, no meta nodes exist in which to embed this information. In order to nevertheless indicate where subgraphs were removed, graph cues and signposts can be added to point towards invisible parts of the network and thus to provide a handle for navigating the abstracted context [MSDK12, PvH12, vHP09].

For multiple focus nodes, the challenge lies first and foremost in the computation of the distance between each node and the selected foci in order to determine each node's DoI value. While this is straightforward for a single focus, there is no standard way to compute a single distance value to multiple points in a graph – regardless of whether the geodesic distance is used or any other distance metric, such as a geometric distance [HZ07] or a similarity metric [CLWM11]. In either case, the most common approaches are to use the distance to the closest focal node [HC04] or to use the average distance to all focal nodes [MSDK12], while some also provide other aggregations such as the sum or the maximum of the distances [Noi95]. To reduce the number of distance computations, the number of focal nodes is often bound by a maximum quantity, which is realized by decreasing the influence of all existing foci whenever a new focus is selected [ds09, HC04, HZ07]. For their depiction, graphs with multiple focus nodes can no longer simply be laid out by using the geodesic distance to the focus, as it is done, for example, in [RZT⁺09]. Instead two other approaches prevail: those that aim to assign them prominent positions through an adapted force-based layout [Osa01] and those that simply use node size, color or label size to highlight them [CN02, ds09, HZ07] in a regular layout.

For dynamic graphs, DoI values are computed for each time point individually. Here, the same problem can occur as when generalizing to networks: the DoI values of a node can vary a lot from one time point to the next. In case of an animated view over time, this can lead to pop-up artifacts in the visualization when nodes are of high interest at one time point and of low interest at the next time point. The first DoI-based approach for dynamic trees did not directly counter this problem, but instead put all focus nodes into a set of high interest nodes that remained the same for all time points [CSP⁺06]. In this way, pop-up artifacts will still occur, but not for focus nodes as these will always be of high interest and thus visible. In [RPD09], a different approach was presented for dynamic hierarchical compound graphs. It uses a temporal relaxation to distribute DoI values from and to previous and future time steps to limit such visual effects. After treating sudden changes with either method, both approaches rely on animations to transition smoothly from one time point to the next. To navigate through time, [CSP⁺06] utilizes the DoI values as part of a complex time slider that denotes time points at which specific events have occurred for a set of focus nodes.

In summary, it can be observed that the many existing DoI-based visualization approaches are all provided as final, fixed monolithic solutions that are tailored to a specific application domain with specific tasks and specific types of graphs. Possibilities to adjust them rarely go beyond the adaptation of a few weights for their individual terms. While this simplification prevents the user from being exposed to the full complexity of a comprehensively defined DoI function, it also limits his ability to express his own interest for a particular task or a particular input graph that was not anticipated by the visualization designers.

4.3.2 A Modular DoI Definition for Dynamic Graphs

The aim of this section is to introduce a DoI definition that is able to cope with the changing demands of interactive visual analysis of dynamic graphs from various domains. This is achieved by bringing two novel ideas to the field of DoI functions for graphs. The first is a modular way of assembling DoI functions from predefined functional components. In this way, a DoI function can be flexibly adapted to new application domains or newly found graph characteristics by simply exchanging or reconfiguring its individual components. These components can be grouped into four categories, which are named depending on the practical role they play in the overall DoI definition:

- *DoI generators* provide the base DoI values that specify the user interest in individual characteristics of graph elements. The corresponding function blocks are called **specification components**.
- *Unary DoI operators* transform a single given DoI value – e.g., to amplify or reduce it. The corresponding function blocks are thus called **transformation components**.
- *N-ary DoI operators (1 graph element, n DoI values)* combine two or more DoI values defined for a single graph element. The corresponding function blocks are therefore called **combination components**.
- *N-ary DoI operators (n graph elements, 1 DoI value)* propagate a DoI value across neighboring graph elements. The corresponding function blocks are therefore called **propagation components**.

The second novel idea is the incorporation of the temporal aspect in the DoI definition. This is achieved by providing specific components that allow the user on the one hand to define something as interesting, because of its temporal dynamics, and on the other hand to define something as interesting, because it was/will be interesting in past/future time steps. The former is embodied in particular specification components, the latter is made available through a temporal propagation component – both are novel in the domain of DoI functions for graphs.

These four types of components are described in the following sections, before they are finally brought together in an example of how to express Furnas' original DoI function. Here, in accordance with established DoI-based graph visualization approaches [Noi95], a DoI value is defined as a real number in the interval $[0 \dots 1]$, with 0 expressing *no interest* and 1 expressing the *highest degree of interest*.

Table 4.3.1: Categorization of basic sources for specifying interest.

Time Points Domain	current (one time point)	immediate changes (two consecutive time points)	higher order changes (three or more consecutive time points)
structure (node / edge)	presence / absence	addition / removal	age, frequency,...
values (attribute / weight)	identity, normalization,...	change, rate of change,...	moving average, trend,...

4.3.2.1 Specification Components

Overall, the specification $spec(x_i)$ captures the user interest in a graph element $x_i \in V_i \cup E_i$ at time point $t_i \in T$ based on some property of that element, for example, an attribute $attr(x_i)$. It consists of two functions: a computation function $comp : V_i \cup E_i \mapsto \mathbb{R}$ that calculates a single numerical value for a graph element and the interest function $inter : \mathbb{R} \mapsto [0 \dots 1]$ that maps this value to the interval $[0 \dots 1]$ in accordance to the user interest. Hence, any specification component can generally be expressed in the form $spec(x_i) = inter(comp(x_i))$. This is noteworthy, as most existing DoI approaches use computed node/edge properties directly in their DoI function, without explicitly defining the user interest over each property's value range. These two notions are decoupled to help the user distinguish between *what* influences his interest in a graph element (computation function) from *how* it influences it (interest function).

The Computation Function $comp$ evaluates structural properties and attribute values for a given graph element $x_i \in V_i \cup E_i$ and computes a single numerical value to express them. As it is evaluated before the interest function is applied, it allows for capturing graph dynamics that span multiple time points, deriving a singular value from them, before defining one's interest on top of such derived values. Depending on the number of time points that $comp$ takes into account, three cases are differentiated and shown in Table 4.3.1. They are discussed in the following:

Current / One time point: This first case is usually used for static graphs, for which there are no time points to consider. Structurally, this means to specify a DoI in terms of the presence or absence of graph elements (usually edges). In terms of a graph element's numerical attributes, the computation function is usually an identity function or performs at most a normalization of these values. The latter is frequently used in order to be able to apply the same DoI function across different time points with potentially very different absolute node/edge counts, as well as across differently sized input graphs. Overall, this case can be expressed for a given attribute $attr$ as $comp(x_i) : attr(x_i) \mapsto \mathbb{R}$.

Immediate changes / Two consecutive time points: This case contains all computation functions, which are defined on two subsequent time points. Structurally, this can be either an addition of a graph element from the last time point t_{i-1} to the current time point t_i , or a removal that will occur at the next time point t_{i+1} . Changes of a numerical attribute are determined in a straightforward manner by, for example, computing the

difference between the two values or the rate of change between them. Again, depending on whether to compute the change in hindsight or foresight, this can be expressed for a given attribute $attr$ as

$$\begin{aligned} comp(x_i) : attr(x_{i-1}) \times attr(x_i) &\mapsto \mathbb{R} \quad (\text{hindsight}) \\ comp(x_i) : attr(x_i) \times attr(x_{i+1}) &\mapsto \mathbb{R} \quad (\text{foresight}) \end{aligned}$$

Higher order changes / Three or more consecutive time points: This case subsumes all computation functions, which compute a development of the graph over more than two consecutive time points. With respect to the graph structure, this can be, for example, the age of a graph element, i.e., for how long it is already present, or any other measure over multiple time points, such as the frequency with which an element appears [AERD10]. Whereas the development of attribute values and weights of a graph can be assessed in the manner known from time series data, e.g., by computing moving averages over multiple time points or trends. For a given attribute $attr$, as well as a number p of past time points and a number f of future time points to include, this case can be expressed in general as

$$comp(x_i) : attr(x_{i-p}) \times \dots \times attr(x_i) \times \dots \times attr(x_{i+f}) \mapsto \mathbb{R}.$$

The Interest Function $inter$ defined on top of the computation $comp$ realizes the mapping from real values to DoI values. It can be envisioned as “carving” the interesting parts from the value range of a computed numerical property. This includes, that it also maps any undefined attributes to 0 to express no interest in them. While the interest function can take on any required shape or form, there are a number of commonly used ones that are briefly listed in the following, where x stands for any previously computed numerical value $comp(x_i)$.

Gaussian function: $inter(x) = e^{-(x-\alpha)^2/\beta}$

This function is used to pinpoint a certain value of interest and produce a smooth decline of interest with increasing distance to this value. The value of interest can be set through the parameter α , the gradient of its decline is governed by the parameter β with $\beta > 0$.

2-Sided exponential function: $inter(x) = \beta^{|x-\alpha|}$

This function is also used to focus on a specific value, but with a steep decline towards smaller values. The value itself can be set via the parameter α , while the rate of the decline is governed by the parameter β with $0 < \beta < 1$.

Sigmoid function: $inter(x) = 1 / (1 + e^{-\beta*(x-\alpha)})$

This function is used to express interest in only high values with a smooth decline at a certain threshold. The threshold can be set by the parameter α , whereas the rate of the decline is governed by the parameter β with $\beta > 0$.

Piecewise constant function: This function is used to define intervals of different interest levels. There is no fixed set of parameters for its specification, as the number of intervals may vary.

Utilizing a Combination of Computation and Interest Functions to Capture Interactive Selections So far, only instances, in which properties of the graph data and its dynamics are captured, have been discussed. Yet, the two-part DoI specification is much more

versatile than that and it is even able to express such aspects as the interactive selection of graph elements by encoding the selection logic in the computation function and an interest in more recently selected elements in the interest function.

For a simple selection, it needs merely a running global counter *count* that is increased by 1 each time a selection is made and an attribute *clicked_at*(x_i) to store the current *count* when element x_i gets selected. The computation function can then be used, for example, to limit the selected elements to only those of the n last selection operations

$$last_n(x_i) = \begin{cases} clicked_at(x_i) > count - n & : \quad clicked_at(x_i) \\ else & : \quad 0 \end{cases}$$

The interest function defined on top of that computation can be, for example, a decay function similar to [dS09]:

$$decay_d(last_n(x_i)) = \begin{cases} last_n(x_i) > 0 & : \quad d^{count - last_n(x_i)} \\ else & : \quad 0 \end{cases}$$

Depending on the decay factor d , the resulting DoI values can range from all 1's for $d = 1$ (multiple foci if $n > 1$) to only the last selected element having a DoI value of 1 for $d = 0$ (single focus). A decay factor between 0 and 1 can be used to balance these two approaches by fading-out older selections and thus ensuring that only a limited number of elements is selected at all times. In the remainder of this paper, the term $select_d(x_i)$ is used as a placeholder for any such DoI specification $decay_d(last_n(x_i))$ that captures an interactive selection with decay factor d over the last n graph elements that were clicked.

Every specification component $spec(x_i)$ is in itself already a very simple DoI function $doi(x_i)$. The remaining components merely take DoI functions – such simple ones as $spec(x_i)$, but also more complex ones as they result from applying the following components – as an input to yield more powerful DoI functions.

4.3.2.2 Transformation Components

Functional components falling in this category are mainly used to modify a previously specified DoI function by emphasizing certain parts, cutting off others, or simply inverting it. Thus, a transformation component can be realized through any function, which fulfills $trans : [0 \dots 1] \mapsto [0 \dots 1]$. The two transformations that are mainly used are an **inversion** and a **scaling** of a DoI.

Inversion function: $inv(doi(x_i)) = 1 - doi(x_i)$

Inverting a DoI function is suitable, if the opposite of the user interest can be specified more easily than the actual interest. For example, when the user is not interested in all values above a certain threshold, as expressed with the sigmoid transfer function, but instead in all values below that threshold, then the inversion of the sigmoid can be used to express this.

Scaling function: The scaling of DoI values is suitable, for example, to lessen their influence or to ensure a guaranteed minimal influence, even if the values are low. The two forms that are often used, the multiplication and the exponentiation, can be expressed in one function $scale_{const,exp}(doi(x_i)) = const * doi(x_i)^{exp}$ with $const \in [0 \dots 1]$ and $exp \in \mathbb{R}$.

4.3.2.3 Combination Components

Since a DoI definition is usually based on multiple attributes, properties, dynamics, etc., combination components can be used to form more complex patterns from multiple DoI definitions capturing different features to accommodate the diversity of the graph. Such patterns can even have different temporal dependencies, i.e., a pattern defining a linear function can be combined from a value at some time point (first column of Table 4.3.1) and a slope (second column of Table 4.3.1). Combinations thus realize a mapping $comb : [0 \dots 1]^n \mapsto [0 \dots 1]$. Numerous useful combination functions are known [BPC07]. However, the most common ways to incorporate multiple DoI terms in one function are either **Min/Max combinations** (often used in DoI functions for multivariate data, such as the FDL [DGH03]) or **weighted sums** (often used in DoI functions for network data).

Min/Max combinations: The minimum combination is used to express that all input functions must return a high DoI value, in order for the combined function to also return a high DoI value. In this regard, it can be viewed as the fuzzy-logical AND operator on the input functions. The maximum combination works the other way around by returning a high DoI value if at least one of the input functions has a high value. It is thus comparable to the fuzzy-logical OR operator.

Weighted sum: This type of combination aims to balance the influence of the n individual DoI terms by multiplying them with weights $w_1, \dots, w_n \in \mathbb{R}$ before simply adding them up. To ensure that the resulting function value will still be within the range of $[0 \dots 1]$, the weighted sum is divided by the sum of all weights. Depending on how the weights are chosen, weighted sums can be used to express a variety of different operations, such as averages ($w_k = 1/n$) or the maximum likelihood estimator ($w_k = 1/\sigma_k^2$), both with $k \in [1 \dots n]$.

4.3.2.4 Propagation Components

The DoI functions defined so far are always bound to one specific graph element x_i at one specific time point t_i . The region around them – structure-wise, as well as time-wise – is not yet considered. To realize the goal of not only pinpointing elements of interest, but also showing them in their immediate context, propagation functions are used to disseminate high DoI values to surrounding graph elements and time points. Consequently, the propagation of DoI values across the graph topology and along the temporal axis are discussed individually.

A Structural DoI Propagation distributes DoI values across multiple graph elements for each time point individually. It is basically determined by four functions:

- An **input DoI function** $doi(x_i)$ whose values are to be gathered for a graph element x_i .
- A **distance function** $dist_{attr}(x_i, y_i) : (V_i \cup E_i)^2 \mapsto \mathbb{R}$ that takes two graph elements from the same time point t_i as an input and measures w.r.t. the edge attribute $attr$ how far they are separated from each other as a real value.

- An **edge attribute** $attr(x_i)$ to be used for computing the distance function, with only positive edge attributes being allowed: $attr(x_i) : E_i \mapsto \mathbb{R}^+$. If the used edge attributes represent capacities, where large values actually stand for a tighter connection than small values do, they have to be transformed into costs. This can simply be done by computing $attr' = 1/(attr + 1)$.
- A **drop-off function** $drop(doi(x_i), dist_{attr}(x_i, y_i)) : [0 \dots 1] \times \mathbb{R} \mapsto [0 \dots 1]$ that is monotonic decreasing, so that at larger distances the DoI value recedes.

These functions are then combined with a maximum to preserve the highest DoI value, so that no element of high interest will be drowned out by a neighborhood of lesser interest:

$$prop_s(doi(y_i), x_i) = \underset{y_i \in V_i \cup E_i}{MAX} drop(doi(y_i), dist_{attr}(x_i, y_i))$$

For its computation, different distance functions are possible. Since no differentiation between nodes and edges is done and the DoI values defined for nodes have to be diffused also to their incident edges and vice versa, a generalized distance function is necessary. For this, the distance between two nodes d_{vv} is defined as the usual geodesic distance $gd(x_i, y_i)$ between them. The distance between an edge and a node d_{ve}/d_{ev} is computed as the minimum distance between the two incident nodes to that edge and the node plus half of the edge attribute, which mimics a node lying halfway on the edge. And the distance between two edges d_{ee} is reduced to the former case. All in all, the distance function is defined as:

$$dist_{attr}(x_i, y_i) = \begin{cases} x_i \in V_i \wedge y_i \in V_i & : d_{vv}(x_i, y_i) \\ x_i \in V_i \wedge y_i = (v_1, v_2) \in E_i & : d_{ve}(x_i, y_i) \\ x_i = (v_1, v_2) \in E_i \wedge y_i \in V_i & : d_{ev}(x_i, y_i) \\ x_i = (v_1, v_2) \in E_i \wedge y_i \in E_i & : d_{ee}(x_i, y_i) \end{cases}$$

$$d_{vv}(x_i, y_i) = gd(x_i, y_i)$$

$$d_{ve}(x_i, y_i) = MIN(d_{vv}(x_i, v_1), d_{vv}(x_i, v_2)) + attr(y_i)/2$$

$$d_{ev}(x_i, y_i) = MIN(d_{vv}(v_1, y_i), d_{vv}(v_2, y_i)) + attr(x_i)/2$$

$$d_{ee}(x_i, y_i) = MIN(d_{ve}(v_1, y_i), d_{ve}(v_2, y_i)) + attr(x_i)/2$$

A Temporal DoI Propagation distributes DoI values across multiple time points for each graph element individually. It consists of an input DoI function $doi(x_i)$, but x_i 's DoI values are in this case to be gathered from all time points t_j with $1 \leq j \leq |T|$ and scaled w.r.t. t_i using the given drop-off function:

$$prop_t(doi(x_j), x_i) = \underset{j=1}{\overset{|T|}{MAX}} drop(doi(x_j), t_j - t_i)$$

The distance between two time points is simply their difference. If it is negative, the time point under consideration lies before t_i – otherwise after it. This way, the drop-off function can be defined differently for negative values than for positive values, allowing the user to adjust the spreading of DoI values independently. Thus a foreshadowing of the occurrence of an interesting element can be handled in another way than its influence in retrospect. As a result, the propagation across time prevents sudden jumps in the

DoI values and smoothes any process that relies on these values, i.e., prevents pop-up artifacts in animations. But, as the example in the following section shows, propagations can also be used for other purposes.

4.3.2.5 Example: Realizing Furnas' DoI Function

Since this approach combines many features of existing DoI functions, it is not surprising that most of them can be expressed by a combination of the functional components described above. To illustrate the overall idea of how this is done, this section shows how to formulate Furnas' original DoI function using this novel approach.

Furnas' DoI function has two parts: the a priori importance $API(x) = -dist(x, root)$ and the distance $D(x, y) = -dist(x, y)$ of a node x to a focus node y , which are summed to yield:

$$DOI(x|y) = -dist(x, root) - dist(x, y)$$

The value range of this function spans $[-3 * max_height \dots 0]$ where max_height denotes the maximum height of the tree in question. Consequently, a DoI value of $-3 * max_height$ represents the lowest interest and is produced when x and y are both leaves at depth max_height and $root$ is their least common ancestor. On the other end of the spectrum, a DoI value of 0 stands for the highest interest, which results for $x = y = root$. This kind of an open ended DoI value range that depends on the size of the input graph and that can be observed for many existing DoI functions is an aspect that cannot be modeled with the fixed $[0 \dots 1]$ interval of DoI values.

While the pros and cons of open or closed value ranges are debatable, a fixed closed interval is an absolute necessity in this case in order to make the different functional components compatible with each other and thus to permit for their flexible combination. Yet, even though the very same DoI values of Furnas' DoI function cannot be produced, its functionality can nevertheless be captured in the range of the $[0 \dots 1]$ interval. Note that the index i has been dropped from the following DoI definition, as Furnas' DoI function does not deal with dynamics in the tree.

$$doi(x) = 1/3 * inter(height(x)) + 2/3 * prop_s(select_0(y), x)$$

$$inter(height(x)) = 1 - height(x) / max_height$$

$$dist_{attr}(x, y) = gd(x, y) \text{ with } attr(x) = 1$$

$$drop(select_0(y), dist_1(x, y)) = 1 - dist_1(x, y) / (2 * max_height)$$

The overall DoI function is a weighted sum of two individual DoI functions: one defined over the height of a node x in the tree and another one defined over the selection status of a node y . The weights make the contribution of each function to the overall DoI value explicit and thus also adjustable if necessary. They can be observed in Furnas' DoI function, for example, in the case of the minimum DoI value of $-3 * max_height$ to which the a priori interest contributes $-max_height$ (the maximum distance to the root) and the distance to the focus node contributes $-2 * max_height$ (the maximum distance between two nodes).

The first function is a straightforward encoding of Furnas’ a priori interest, with the computation function being $height(x) = dist(x, root)$ and the given interest function $inter(height(x))$ mapping it inverse linearly on the interval $[0 \dots 1]$. The latter ensures that the root at $height = 0$ receives the maximum DoI value of 1 and the deepest leaves receive the minimum DoI value of 0.

The second function $select_0(y)$ captures the selection status of a node y according to its specification in Section 4.3.2.1. The selection status of each node (either 0 or 1) is then propagated to all other nodes using the common geodesic distance $gd(x, y)$ with all edge attributes set to a constant 1 and a linear drop-off function that assigns 1 if the current node itself is selected and 0 if the current node is at the largest distance to the selected node. This procedure is necessary, as the interactively selected focus node is not known “a priori”. Thus, it is not possible to precompute all distances to it and store it as a node attribute, as it was for the height. Instead, the structural propagation is used to automatically readjust the DoI values every time the selection status changes.

As a result, the overall DoI function behaves similar to Furnas’ DoI function and yields the maximum DoI value of 1 for the root node, if it is also the focus node, and 0 for a leaf at the deepest level and furthest away from the focus node. Yet, in contrast to Furnas’ DoI function, its behavior can very easily be changed, now that all its components are exposed. For example, to turn the DoI function into one that handles multiple foci in the way that DOITrees [HC04] do, one can simply change $select_0(y)$ to $select_1(y)$. This is possible, because the closest focus node that has been least affected by the drop-off function and thus has the highest DoI value will be the natural result of the maximum combination that realizes the structural propagation.

As this example shows, it is not only possible to build one’s own DoI function with the proposed components, but also to reproduce existing DoI functions with them. This provides endless combination and adjustment opportunities, which on the one hand are absolutely necessary to express the user interest as precisely as possible to form a meaningful functional base for the subsequent visualization. On the other hand, all these different options also create additional complexity that the user must master in order to make fullest use of the novel approach. The following section discusses these points, illustrating a realization of a visual analysis setup built around the modular DoI definition concept.

4.3.3 An Interactive Visual Analysis Setup based on the DoI Values

With the modular DoI definition, the previous section detailed a new approach to define how much interest each graph element receives at each point of time. Once the DoI function is computed, the resulting DoI values provide an extremely valuable annotation to the data that can be used in many more aspects than just for reducing the network. Focusing on the DoI, common architectural details and design issues as they were already discussed in previous sections and are found in most existing graph visualization systems, such as TugGraph [AMA09] or CGV [TAS09] are only described briefly. Instead, this section illustrates specifically, how the visual analysis setup makes extensive use of the DoI values to provide DoI-based representations and interaction mechanisms. For

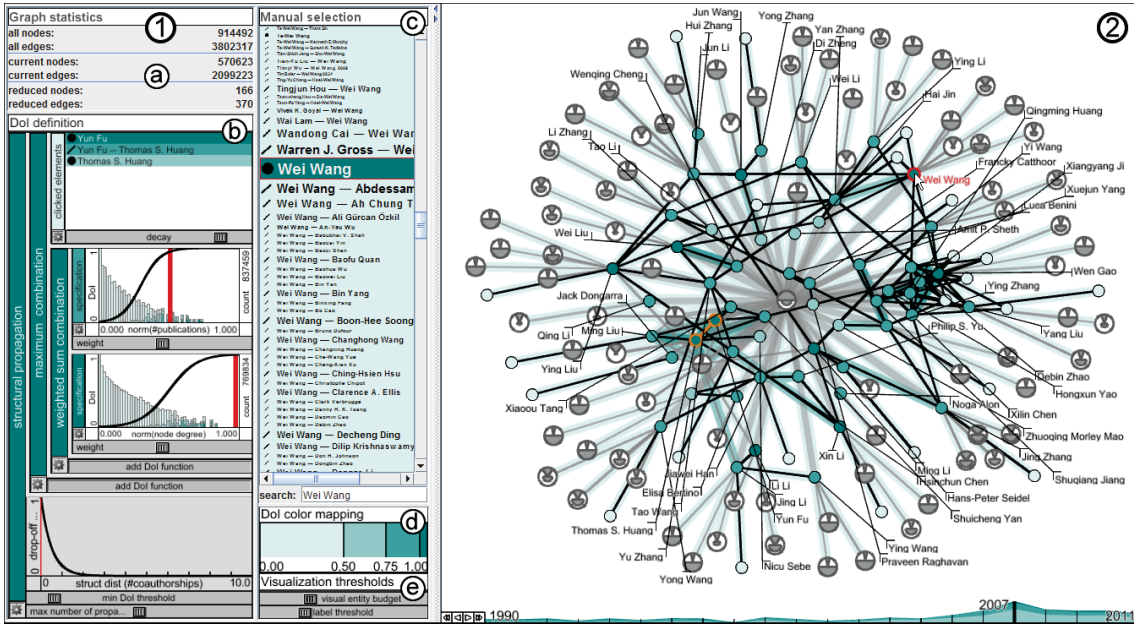


Figure 4.3.21: The overall visual setup with its two main views: (1) the DoI view and (2) the graph view. The graph view shows a snapshot of the DBLP data set for the year 2007, which has been reduced according to the defined DoI function. The DoI view is partitioned in five panels (a)-(e). Panel (a) gives some numerical details on the reduction. Panel (b) shows the DoI function $doi(x_i) = prop_s(\max(\{select_{0.85}(y_i), doi_{topAuthors}(y_i)\}, x_i))$. The term $doi_{topAuthors}(x_i) = sum(\{inter_1(norm(publications(x_i))), inter_2(norm(node_degree(x_i)))\})$ selects the most active authors, while the *select*-term governs how graph elements picked by the user in panel (c) are treated. Here, they are added as focus nodes (*MAX* combination) and shown in orange in the graph view. The colors of all other elements are given by the adjustable color scale in panel (d) and the amount of reduction can be fine-tuned in panel (e). The labels of the DoI components in panel (b) are currently colored to reflect their individual DoI values for the hovered node “Wei Wang”, which is in highlighted in red.

this, it relies on two distinct views:

- The *DoI view* serves the purpose to interactively define and refine a DoI function that captures the user interest by exposing its functional components as reorderable, hierarchically stacked visual components.
- The *graph view* shows the result of applying the defined DoI function by agglomerating graph elements of lesser interest on a global overview level, while highlighting graph elements of high interest on a local individual node level.

The overall setup consisting of these views is exemplified in Figure 4.3.21. The next two sections will discuss these two views, followed by a description of how the views are interlinked to support the user in defining and adjusting the DoI function to express his interest in detail.

4.3.3.1 The DoI View

The purpose of the *DoI view* is to interactively define *what will be shown* of the graph by giving access to the DoI function and a few other parameters. This view goes well beyond the usual simple adjustment of some weights by exposing the entire DoI function to the user and allowing for its complete reconfiguration. The view consists of five panels that each has an important role in the course of the DoI definition:

The **graph statistics** (Figure 4.3.21, panel (a)) provide a brief numerical overview on the current reduction of the graph by showing how many graph elements exist (overall, in the current time step, and in its reduced form). These numbers give the user a concrete idea of how big the graph is and how much of it is actually hidden from the *graph view* – in other time points, as well as inside the meta nodes, into which elements of lesser interest are contracted.

The **DoI definition** (Figure 4.3.21, panel (b)) permits the user to adjust the DoI function that forms the basis of the reduction into meta nodes. The view is an interactive, hierarchically structured graphical representation that directly encodes the different DoI components of the currently used DoI function. Each functional component defined in Section 4.3.2 corresponds to a matching visual component that can be plugged into the *DoI view* and that provides the necessary handles to adjust a component's parameters. Some of their realizations are shown in Figure 4.3.22. Depending on the complexity of the DoI function, the embedded component views may require a large part of the available screen real estate. To counter this drawback, the user can switch to a reduced view that only shows the left-hand labels of the component views in an Icicle-plot-like fashion and hides the visualizations and parameters on the right-hand side. While this reduces the width of the *DoI view*, the height can be reduced by folding known branches of the hierarchically composed view to a mere label as well. This feature is exemplified in one of the later screenshots in Section 4.3.4, where known DoI terms from previous visual analysis steps are folded and only the ones that have been newly added in the current step are shown in full detail.

The **manual selection** (Figure 4.3.21, panel (c)) gives access to individual graph elements for adding or removing them manually as high-interest nodes through the *select_d*-term in the DoI function. This access is quite important, as it permits the user to not only express his interest in a particular behavior through the DoI function for *identification tasks* (indirect look-up), but also in certain graph elements for *localization tasks* (direct look-up). For this, a compact list-based view shows the search result for a search term typed at the bottom. The entries in the list are highlighted in the color that corresponds to the respective DoI value computed for them. In case it cannot be determined from the label itself, a small icon in front of each entry denotes whether an element is an edge or a node. Since there can be a large number of matching results, a fisheye distortion is used that couples the applied magnification with an entry's DoI value, so that elements with higher interest are shown in a larger font. To support the user in selecting the smaller entries of lesser interest, a table-lens enlarges the rows under the mouse cursor for an

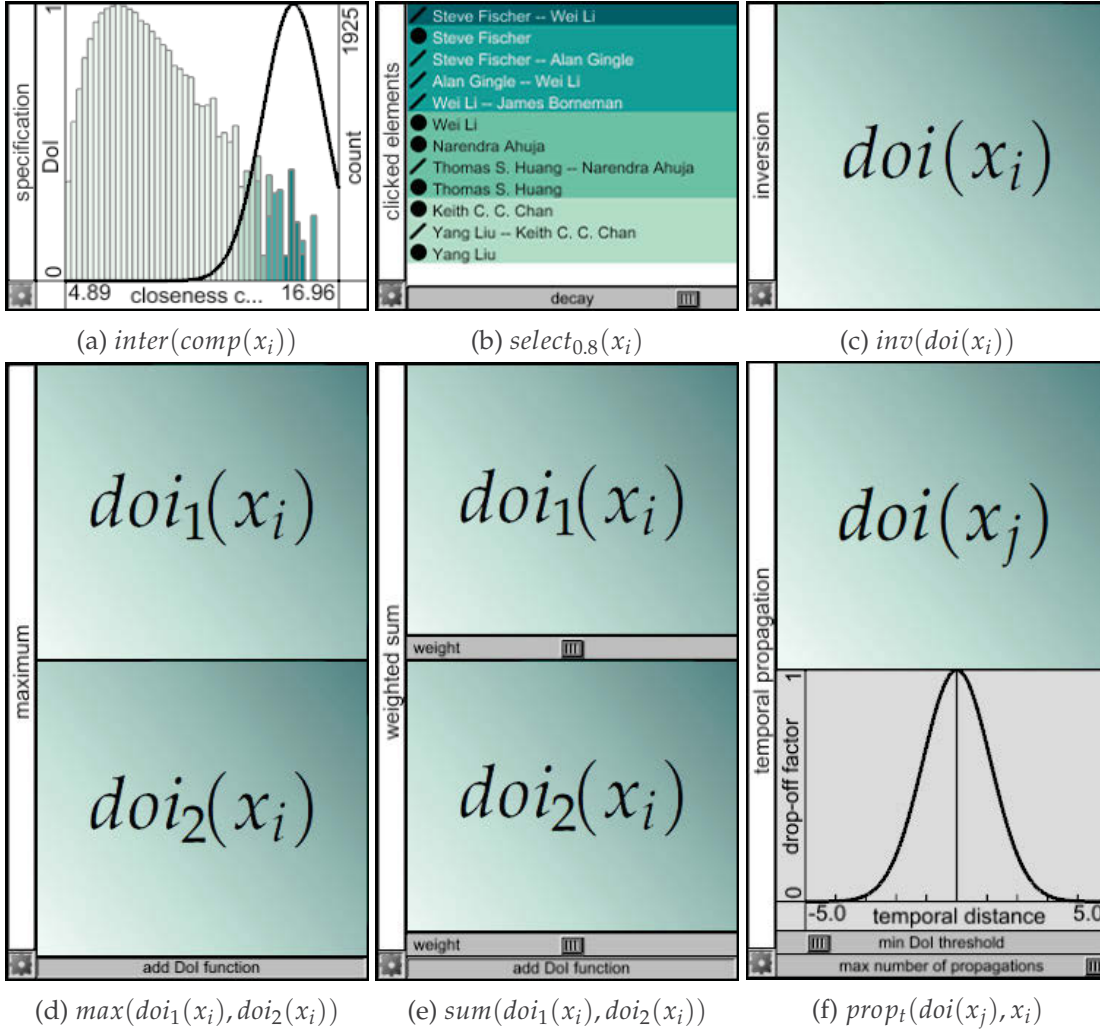


Figure 4.3.22: Different DoI components and their visual counterparts. Specification components assigning high DoI values to graph elements (a) according to their properties (in this case, high closeness centrality) or (b) if they were clicked. A transformation component (c) inverting the given DoI function $doi(x_i)$. Combination components returning (d) the maximum DoI value or (e) the weighted sum of the two given DoI functions $doi_1(x_i)$ and $doi_2(x_i)$. And a propagation component (b) distributing a given DoI function $doi(x_j)$ across multiple time points.

easier selection. The effect of the selection depends on how the $select_d$ -term is used in the DoI function: a simple $MAX(doi, select_d)$ -combination will result in an addition of the selected element, whereas a $MIN(doi, INV(select_d))$ -combination will remove selected elements.

The **DoI color mapping** (Figure 4.3.21, panel (d)) allows the user to change how the DoI values are mapped onto color. In order to help distinguish elements of different interest more clearly, the full interval of DoI values $[0 \dots 1]$ is partitioned into multiple ranges. This way, one can set a clear threshold between two ranges of different interest and they will get assigned two clearly distinguishable colors, according to the range of DoI values in which they lie – something that would be hard to do with a continuous color scale. The number of the ranges and the interval of DoI values they span can be adjusted by the user directly within the interactive color scale. The color mapping set by this means is used throughout the *DoI view* and the *graph view*.

The **visualization thresholds** (Figure 4.3.21, panel (e)) are controlled via sliders that effectively permit to tune the size of the shown graph in the *graph view*. As the graph is too large to be shown in its entirety, it has to be reduced according to the computed DoI values. Therefore, the edge(s) whose averaged DoI values of its two incident nodes and itself is minimal across the entire graph are contracted. Another option would be to make the probability of an edge to be contracted inversely proportional to the summed DoI values and then choose random edges for contraction w.r.t. this probability to yield a sampling effect. A contracted edge and its two nodes are subsumed by a meta node that receives the maximum DoI value of all contained elements. The contraction is performed until the number of remaining nodes and edges falls below the visual entity budget that the user defines via the first of the sliders. Such a visual entity budget is usually a better way to limit the size of the graph than a fixed cutoff with respect to the DoI value, as it maximizes what is shown no matter how skewed the distribution of DoI values is. The second slider governs the number of labels in the *graph view*.

4.3.3.2 The Graph View

While the previous section detailed how the *DoI view* helps to determine *what will be shown* (i.e., the reduced graph), this section discusses *how it is shown* in the *graph view*. Generally, it represents the reduced structure by a node-link representation and uses animation to convey changes in the layout caused by switching the time points or adapting the DoI function, see Section 2.2.4.1 for more details. In the following, especially those aspects are discussed that tie in closely with the DoI values.

The **positioning** is done for each time point individually, yet in a fashion that ensures some stability across time points, so that browsing along the time axis gives a coherent impression of the overall behavior. For this, a simple base layout is combined with an additional stabilization mechanism. For the base layout, the Fruchterman-Reingold force-directed layout [FR91] is employed to incrementally generate a layout for each time point by taking the resulting layout of the previous time point as a starting point for the

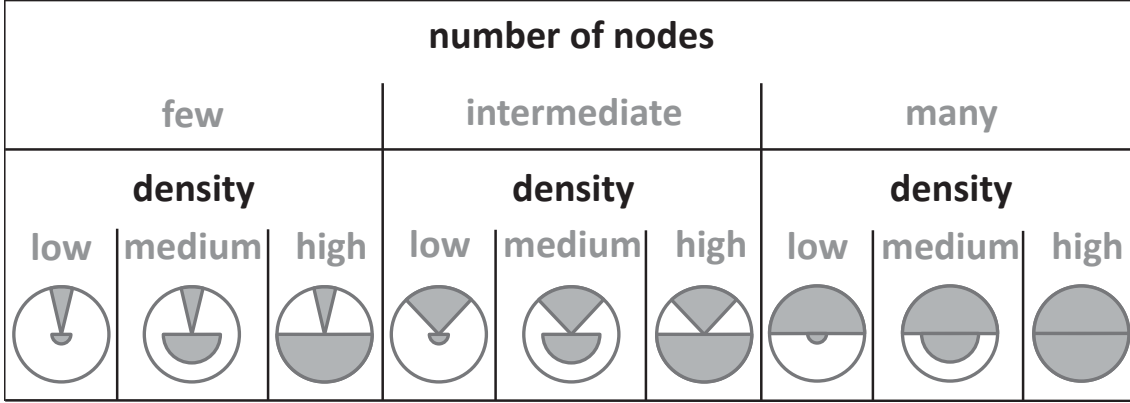


Figure 4.3.23: Different appearances of the glyph encoding contracted subgraphs. The size of the subgraph is mapped to the angle of the circle section at the top. The density (number of edges out of all possible edges) is mapped to the radius of the circle section at the bottom. Thus, the glyph at the very left encodes a small set of nodes with minimal connectivity, whereas the full circle at the right represents a large clique.

next time point's layout. As a stabilization mechanism, it uses pinning weights, as it has been suggested in [FT08]. By choosing the pinning weight of a node to be proportional to its DoI value, a node's position becomes automatically more rigid, the higher its DoI value is. This way, nodes of higher interest can be followed more easily, while nodes of lesser interest are still allowed to adapt more freely, as they do not need to be tracked individually. In order to enhance the trackability of high-interest elements already during the force-directed layout, spring constants are assigned to the edges that are proportional to the largest DoI value of its incident nodes. This mechanism is similar to the heat model used in [Osa01]. Optimally, it creates a fisheye-like effect around nodes with high DoI values as it elongates its incident edges. But in cases of densely connected subsets of detail nodes, this effect is neutralized since all edges receive similarly high spring constants.

The **shapes** of the nodes are probably the most prominent feature to tell apart detail nodes and meta nodes. While detail nodes are shown in the form of small circles, meta nodes are given a more distinct look that makes it immediately clear that they are representing multiple contracted nodes instead of a single detail node. Therefore, meta nodes are represented by glyphs. These glyphs give some basic node/edge statistics on the respective collapsed subgraph they represent. It consists of a circle section on top of a semicircle and can have different appearances depending on the size and density of the subgraph, as schematically illustrated in Figure 4.3.23. The circle section at the top encodes information about the number of subsumed nodes, which is mapped logarithmically on its angle $\alpha = 180^\circ * \frac{\log(\#nodes)}{\log(\max\#nodes)}$. Whereas, information about the subsumed edges is mapped in the form of the subgraph's density (the ratio of existing edges vs. all possible edges) logarithmically on the radius of the lower semicircle $r = r_{full} * \frac{\log(\#edges)}{\log(\#nodes * (\#nodes - 1) / 2)}$.

The **color** follows the color mapping that is set in the *DoI view*. Detail nodes are color-coded directly according to their DoI value, while the color-coding of the edges is shown

via halos [LS08b] around the edges to not interfere with the topological information they convey. To better distinguish detail nodes from meta nodes that form the context of the observed details, meta nodes and edges that lead up to them are colored in gray.

The **labeling** prioritizes nodes with higher DoI values for labeling overall and direct labeling specifically, so that high interest nodes are always labeled with best possible attribution of the label to them. This labeling is achieved by utilizing the particle-based approach described in [LSC08]. This approach starts with the detail node with the highest DoI value and stopping when the labeling threshold set in the *DoI view* is reached. This has the effect that nodes with higher DoI values also have a higher chance of getting a direct label, since not many other labels have been placed yet. Whereas nodes with lower DoI values are then more likely to receive eccentric labels as all the close spots are potentially already taken by previously placed labels. This also helps to set high-interest nodes apart from nodes with lower DoI values, for which labels are only available on demand.

The **interaction** facilities provided by the *graph view* are mainly the standard interaction techniques, such as zooming, panning, and relocating nodes. While these are not influenced by the DoI, they nevertheless affect the DoI-based view. For example, a zoom-in will lead to fewer visual objects being displayed in the zoomed-in part of the view. This leaves in turn more space for additional labels, which are automatically added to previously unlabeled nodes in this case. Other, more exotic couplings of standard interaction with the DoI values are easily imaginable. For example, when relocating a node, one could set the mouse movement speed inversely proportional to the node's DoI value, similar to the idea of pseudohaptics [LBE04]. This way, nodes with higher DoI values would be harder to relocate not only for the layout algorithm due to the pinning weight, but also by the user who would thus get an additional feedback about the node's DoI value.

The **dynamics** of the graph can be accessed via a DoI-based time slider in the form of a stacked graph [BW08] at the bottom of the *graph view*. It supports all the usual interactions, such as stepping through the time points one by one or playing an animation over all of them in both directions. Its overall height reflects the number of all elements having a non-zero DoI value at a given time point. It is subdivided into horizontal bands of the different color intervals chosen in the color scale in the *DoI view*. These bands are stacked from the lower DoI values to the highest ones, so that a user can judge the interestingness of a time point from the size of its topmost layers encoding the number of elements with high DoI values. To achieve that even the few elements of the highest DoI interval are visible, element counts are not simply added up in the stacked graph, but their DoI values instead. This way, elements with a DoI value of zero are not taken into account and elements with higher DoI values get more emphasis. The encoding as a stacked graph informs a user about time points at which larger numbers of interesting graph elements occur. With this information, the user can pinpoint a time point of high interest and directly jump to it with a single mouse click.

4.3.3.3 Supporting DoI Definition and Adjustment with Presets and Linking

The *DoI view* and the *graph view* provide the interactive and graphical tools a user needs to pursue a DoI-based visual analysis of a dynamic graph. As powerful as these tools are, their flexibility introduces additional complexity that a user has to master before the tools become useful. To aid in this process, two additional mechanisms are offered to support the user:

- for the initial definition of a DoI function, **pre-configured DoI presets** to choose from can be defined and visual cues for their parametrization are provided, and
- for the interactive adjustment of a DoI function, **the views are linked** to allow for a peek into the DoI computation for “debugging” and fine-tuning.

The **pre-configured DoI presets** and visual cues provide help for defining meaningful DoI functions and thus a sensible distinction between detailed focus nodes and contracted context nodes in the *graph view*. Presets are preassembled and preconfigured DoI definitions that can be selected as starting points and building blocks for one’s own definition. As different domains and different types of input graphs require different presets, it is not possible to give a comprehensive list of such function blocks. For example, network intrusion detection requires different components than citation network analysis and trees require different functional DoI terms than a bipartite graph or a general network, as in each case the distinction between focus and context is made differently. Hence, DoI presets have to be tailored to the application domain and the graph type of the input data to be semantically meaningful to a user from a particular area. For example, during the application of this approach for the use case from Section 4.3.4, ready-made presets that capture the common interests in bibliometric research and citation analysis were defined and stored for later reuse.

Once selected, the presets can then be adapted to the specific characteristics of the graph to be analyzed by using the visual cues. These are given in the form of a histogram of the distribution of attribute values in the background of the DoI specification components. This way, a user can make an informed decision when adjusting the interest function directly on top of the values of the computation function, as they are shown in the histogram. Furthermore, the histogram provides feedback of how many nodes in each histogram bar are already assigned high DoI values through the currently specified DoI function. This is encoded in the histogram by means of stacked bar charts, which use the same global color ranges as they are defined in the color mapping.

The **linking between the views** ensures that the user can investigate in detail which functional component caused particular DoI values in the *graph view*. By hovering over an element in the *graph view*, it gets marked red and the *DoI view* changes the background color of the labels of all DoI components to reflect its DoI value. One can then track the high DoI values from the “root” of the DoI composition hierarchy down to the individual feature(s) that contributed the most to the resulting DoI value. In Figure 4.3.21, it apparently stems from the weighted sum combination, which is easily identified by its darker color in the functional DoI hierarchy, whereas for example the component of the clicked elements is colored in a lighter shade. Furthermore, the red marker at distance 0



Figure 4.3.24: A view of the stacked graph along the timeline for the DoI function $doi_{filter}(x_i) = \min(\{doi_{topAuthors}(x_i), inter_3(clustering_coefficient(x_i))\})$, which captures top authors with low clustering coefficients. It can be seen that this does not yet pinpoint the trend of an increasing number of namesakes starting in 2006/2007.

at the bottom of the structural propagation component indicates that the high DoI value originates indeed from the highlighted element itself and was not propagated from another element. This marker denotes the spatial or temporal distance at which the element lies that propagated its DoI value to the hovered element. Histograms feature a similar marker to show where the investigated element lies in the value spectrum of a particular attribute. How this interlinking is used in a real world example is shown in the following section.

4.3.4 Use Case

Citation networks and co-authorship networks are an important subject of study as their analysis can reveal interesting features about a scientific community and expose emerging research topics [New04]. Especially when analyzed over time, they allow for observing the evolution of a scientific community. One source for such a co-authorship network particularly for the computer science research community is the DBLP database [Ley09] that contains around 2 million publications as of today. Because of the size of this network, most analysis attempts are based merely on statistical evaluations of these networks or consider only publications from a specific community or specific conferences [BD10, CGS⁺11, EL05, HH06, NSP03]. Such statistical evaluations represent the network in an abstracted, numerical way that yields at most lists of important authors or highly influential publications without their neighborhood or any other structural information. In the following, first experiences depicting the application of the novel DoI-based visualization approach for the DBLP co-authorship network (snapshot from June 2012) are discussed.

4.3.4.1 Data Description

In co-authorship networks, the nodes correspond to authors and an edge exists between two nodes iff the authors, whom these nodes represent, have published a paper together. Such graphs are extracted from the DBLP database for each year from 1990 to 2011. These graphs are cumulative, which means that each graph contains all authors and co-authorships that have occurred up to that year. This makes sense, as two persons having coauthored a publication once will stay coauthors forever. Additionally for the graph at each time point, the nodes are assigned the number of papers published by its corresponding author in that year and the edges are associated with the number of papers the two incident authors have published together in that year. Based on these attributes, it is possible to extract the network of a particular year by taking only nodes and edges into account, which have an attribute value > 0 in that year. As proposed in [NSP03], the following discussion focuses solely on the largest connected component of the network,

as this component alone makes up 94.2% of the overall data set. This results in a dynamic network with 22 time points (years) containing 914'492 nodes and 3'802'317 edges.

4.3.4.2 Analysis Goals and First Steps

In the analysis of co-authorship networks, the main interest lies often on top authors. According to [EL05, NSP03] and [WF94, ch.5], these authors can be identified by a number of different characteristics, such as a high number of published papers and a high number of coauthors, which corresponds to a high node degree. These two aspects are easily combined into a DoI function by means of a weighted sum combination that takes both of them into account. Manually selected focus nodes are then added through another term, such as the one given in Section 4.3.2.1. As a last step, a structural DoI propagation is applied to ensure that the immediate neighborhoods of top authors and selected nodes also receive high DoI values and will thus be shown in their context. For the year 2007, the result is depicted in Figure 4.3.21.

This takes co-authorship network analysis as far as it would be possible with a fixed monolithic DoI function that has been developed to address this scenario. Maybe some other characteristics of top authors would also be included and one could change their weight and thus their influence on the combined DoI function. The result will always be the same: this works fine up to the year 2005, but starting around 2006/2007 one can observe that the visualization is taken over by a large number of Asian names. The difficulty with these Asian names is that many of them get listed as top authors with a large number of publications and coauthors, because they actually represent more than one person. For example, the author(s) *Wei Wang*⁶ (highlighted in Figure 4.3.21) actually subsumes over 40 different individual persons. The cause of this known problem is the use of the author's name as the key for identifying a person in the DBLP database, which also leads to the problem that authors appear multiple times due to different spellings [Ley02]. A tool with a monolithic DoI function cannot adapt to such specifics of individual data sets and will thus show incorrect results.

With the new approach, however, a visual analysis of the top authors does not have to end here, because the DoI function can be altered to filter out the namesakes and yield a correct view of the top authors. This is detailed in the following section.

4.3.4.3 Further Refinement through DoI Modification

For this analysis a two-step approach is used, which aims first at finding a DoI function that pinpoints the namesakes and then subtracts them from the initial DoI function in order to get a view that is unperturbed by this phenomenon.

To select these namesakes, it is necessary to find characteristics allowing to discern them from regular authors. A first such characteristic is based on the observation that authors who have published with the same person are also likely to have published together [New04]. As a result, this person with whom they have published will have a higher clustering coefficient, meaning his immediate neighborhood is well connected. Contrary to this, coauthors of namesakes are not as well connected, because the multiple

⁶<http://dblp.uni-trier.de/pers/hd/w/Wang:Wei.html>

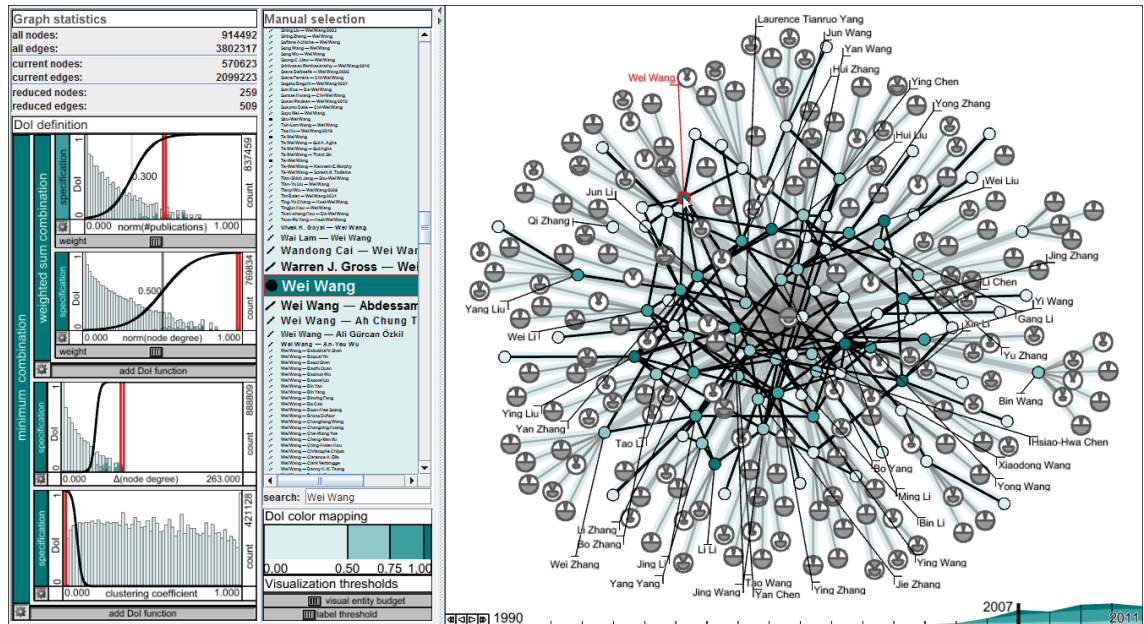


Figure 4.3.25: A full view of the DBLP data set for 2007, capturing all top authors with low clustering coefficients and a high increase of node degree over time, which is expressed through the DoI function $doi_{filter}(x_i) = \min(\{doi_{topAuthors}(x_i), inter_3(clustering_coefficient(x_i)), inter_4(change(node_degree(x_i)))\})$. The stacked graph along the timeline shows that this refined DoI function better fits the observed starting point and the selected author(s) *Wei Wang* have correctly been identified as multiple persons by the function.

“overloaded” persons belong to different scientific communities, and so do their coauthors.

The problem with using this characteristic alone can be observed already in the stacked graph along the timeline, which is shown in Figure 4.3.24: while the trend of a growing number of namesakes was identified to begin in 2006/2007, the clustering coefficient captures a number of authors throughout the entire time interval. It turns out that this characteristic is somewhat too generous in declaring people as namesakes, as there is another category of people, who also fit this characteristic, but are not namesakes: advisors who have published with generations of graduate students with only few joint publications between these students due to temporal separation and thus also resulting in a low clustering coefficient. Advisors can be told apart from namesakes by their slow increase of coauthors, which is likely to correspond to a few new grad students each year. Whereas namesakes exhibit an almost unreal surge of new coauthors each year that is due to the combined research collaborations of multiple persons subsumed under a single name. So, both aspects – the low clustering coefficient and the high increase in node degree (2nd column of Table 4.3.1) – are combined together with the original characteristics of top authors – a high publication count and a high node degree. The result of this DoI combination is shown in Figure 4.3.25, again with *Wei Wang* highlighted. From the stacked graph along the time axis, one can clearly see how this phenomenon starts in 2006/2007, as it had been observed before. The resulting DoI function pinpoints the

4.3 A Modular Degree-of-Interest based Selection for Dynamic Graphs

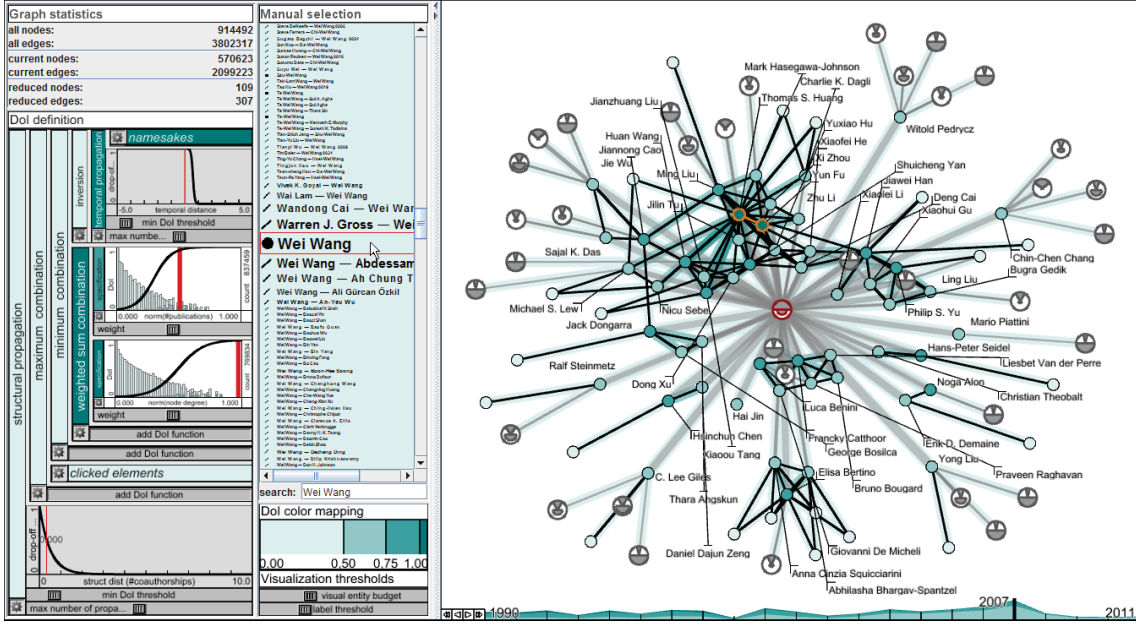


Figure 4.3.26: The same view as in Figure 4.3.21, but this time with the found namesakes from Figure 4.3.25 having been subtracted from it. This has been achieved by combining both DoI functions into $doi(x_i) = prop_s(\max(\{select_{0.85}(y_i), \min(\{inv(prop_t(doi_{filter}(y_j)), y_i), doi_{topAuthors}(y_i)\})\}, x_i)$.

namesakes well enough to use it in combination with the initial DoI function to yield a tidied-up view of the top authors.

To filter out these namesakes, they are subtracted from the top authors. This subtraction can be performed through a minimum combination of the top author selection from Figure 4.3.21 and the inversion of the namesake selection from Figure 4.3.25. However, the selection of namesakes is based on a temporal feature (the change of node degree) that only leads to a high DoI value for a node in a specific year if the steep increase actually occurred in that year. As a result, authors are classified as namesakes in one year, while showing an unsuspicious behavior in their node degree in other years. Hence, a name-sake is defined as such starting from the year it was identified as one. Therefore, before the actual subtraction, a temporal propagation is utilized that distributes high DoI values corresponding to a found namesake to all future years. The result of the subtraction is shown in Figure 4.3.26. As a result of the number of namesakes being reduced, there is now room in the visual entity budget to show actual top authors who have previously been hidden. To demonstrate the subtraction, *Wei Wang* has again been highlighted to show that while this node fulfills all aspects of a top author (the weighted sum component in the *DoI view* is shown in dark cyan), it also conforms to the characteristics of a namesake (folded function block “namesakes” is colored even darker).

Establishing and verifying such a tailored DoI function would have been very cumbersome without this new DoI approach that permitted its interactive derivation with immediate visual feedback. The use case also illustrated the concerted use of all three of the newly introduced concepts: the modular DoI specification to iteratively refine the DoI function, the specification component capturing the dynamics of graph elements, and the

propagation of DoI values over time to spread a once identified namesake to all future time points. Now that such a DoI function has been interactive identified, it would be possible to use it in a pre-computation or data cleansing step to identify namesakes in future co-authorship input data that matches authors by their names.

4.3.5 Conclusion

In this section, a flexible modular DoI specification for dynamic graphs has been introduced, which goes beyond the established fixed monolithic DoI-based approaches. While DoI-based techniques are usually used for generating focus+context visualizations, the flexibility and scalability of this approach is used to employ it for supporting the visual analysis of large dynamic graphs. This is made possible by providing a user with the capability to express dynamic features of interest and to assemble these features into more complex patterns. Nodes and edges that fit a pattern (e.g., a certain specified dynamic) are then put into focus of an otherwise abstracted context representation of the remainder of the graph. As such, this novel approach indeed permits to analyze details such as local changes, while at the same time maintaining an abstracted overview of the context. Therefore, a simple but fast contraction approach is used that iteratively groups uninteresting graph elements. In this way, it scales well to larger graphs and hence can cope with the large size of the discussed use case while still maintaining structural connections between interesting graph elements that are important for the mental map.

4.4 Summary

This chapter has introduced two different strategies to deal with the scalability of graph visualizations. The first strategy uses abstraction mechanisms to either cluster nodes and edges according to similar trends of their associated attributes or time points regarding similar network configurations. In this way, the first strategy provides an abstract overview of the dynamic graph. Whereas the second strategy uses DoI functions to select specific graph elements of interest according to structural, temporal and attribute related properties and shows them in detail while abstracting the context. While the second strategy may not need an abstract overview as provided by the first strategy, their combination as well as their extensions are still valuable options to further investigate.

Both strategies made extensive use of multiple linked views, for instance different overviews for the structural and temporal aspect to select an appropriate granularity for performing the abstraction, detail views to show temporal trends of clusters, or even the different visual components for the DoI specification to accommodate all aspects of the graph. While such a variety is of importance for pursuing a thorough visual analysis, it is also a burden for the user to choose from. Hence, the user has to be supported by allowing him on the one hand to freely switch the visualization at any time during the analysis and to provide him on the other hand a more informative overview of his graph data to steer and coordinate the different views. First solutions for these problems are discussed in the next chapter.

5 Flexible Visualization of Graphs in Space and Time

5.1 Introduction

In the previous section, a variety of techniques has been introduced for the visualization of graphs regarding their different aspects – structure, attributes, spatial and temporal context (diversity) – and their size (scalability). Yet, as a visual analysis session is a sequence of different tasks, a user usually cannot decide for a single visualization once and for all, but has to switch between different visualizations to be able to always use the one that is best suited at a given stage of analysis. The same holds true if the user selects different subsets of the data for further analysis, as these may have different characteristics demanding for a dedicated visualization.

In this sense, a flexible switch as well as a tight linking between different techniques focusing on different aspects and showing different subsets are needed. But most techniques focus on individual visualizations, and very few concern strategies to interactively shift between them and combine them.

Hence, in the following sections two different approaches are introduced to provide the user the flexibility needed for pursuing a full-fledged analysis of his graph:

- that starts with a base visualization concerning only a subset of aspects such as structure or time and then provides a local switch to embed the respective others in a smooth and seamless manner, or
- that utilizes a single view showing all aspects simultaneously as an overview for synchronizing multiple coordinated views each showing details for the different aspects.

5.2 An In Situ Visualization for Switching and Combining Dynamic Graph Visualizations

Typically, multiple coordinated views are used to communicate the different aspects of the data, but a seamless switch from one visualization to the next is rarely supported. Therefore, in this section¹ a novel embedding approach – termed *in situ visualization* – is introduced that addresses the following requirements:

¹Parts of this section have been published in “*In Situ exploration of large dynamic networks*” 2011 in the IEEE Transactions on Visualization and Computer Graphics [HSS11].

- A local embedding of visualizations allowing a local adaption for any subset of the graph data.
- Preservation of the context or mental map during an embedding within a graph visualization.
- Possible recursive application as it may be needed for repeated refinements of the visualization.

This approach draws upon well-established concepts, such as focus+context [Fur86], semantic lenses [BSP⁺93], and portals [OW00], and thus does not confront the visualization users with an entirely new paradigm. It enables the user to shift the analysis focus for individual regions of a base visualization to a different one by switching between visual representations while maintaining the context and thus supporting the user's mental map. This is an important aspect especially for structured data such as networks, as the continuity of the spatialization used by the underlying base visualization should be preserved [SZG⁺96] – be it the structural continuity of the network or the temporal continuity of a time axis. This novel approach achieves this by linking embedded in situ visualizations and contextual base visualization.

Therefore, first a classification of the existing techniques is proposed in Section 5.2.1 to support the selection of an appropriate base visualization. Then, the mechanism for the embedding of various visualizations for dynamic graphs is described in Section 5.2.2. Although this approach is not restricted to specific aspects of a graph it is demonstrated for its structural and temporal aspect.

The usefulness of the in situ visualization concept for dynamic graphs is illustrated by two exemplary use cases in Section 5.2.3. Here, the approach is applied to dynamic graphs from two different application fields: model versioning and mesh networks. The expert feedback given in both cases confirms that the adaptability of the in situ visualization makes it a good fit for these highly interactive analysis scenarios, which require to tailor the view to the given data in a number of subsequent exploration steps.


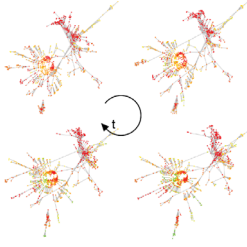
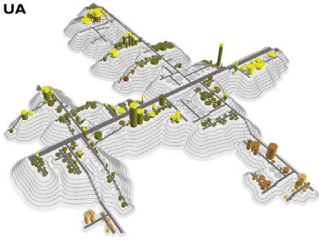
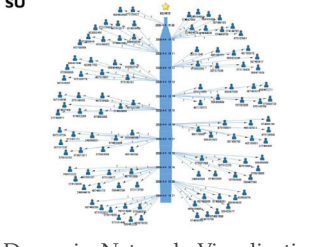
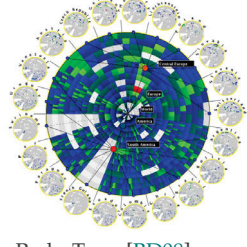
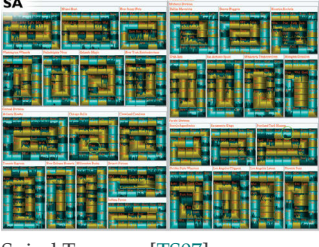
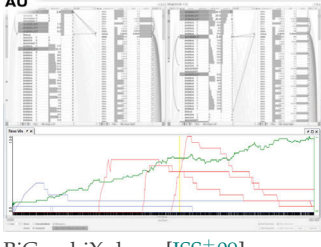
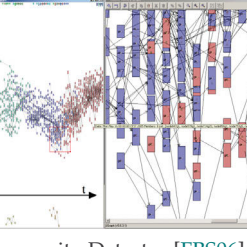
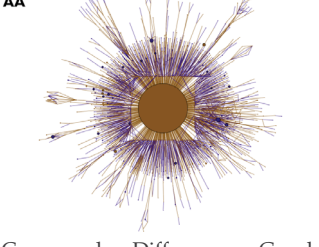
5.2.1 Classification

Current visualization techniques try to find a compromise for what to show at what level of detail between the huge graph structure on the one hand and the large number of time points on the other hand. As discussed in Section 2.3, common ways to cope with the size of these graphs are using selections and abstractions. A selection maintains individual details often at the cost of losing the overview of the whole, whereas an abstraction preserves the overall characteristics at the expense of the details. In terms of a compromise between structure and time this means the more detail of one of these aspects is shown the fewer details can be shown for the other aspect. If a visualization focuses rather on the temporal aspect, more visual entities are allocated for time and less for the structure, and vice versa. In order to achieve this, selection and abstraction can be applied independently for both aspects of the data.

As this is such a fundamental property of visualizations for large dynamic graphs and an important decision to make when deciding for one of them, it can be used as a natural classification system for these techniques, as shown in Table 5.2.1. This classification

5.2 An In Situ Visualization for Switching and Combining Dynamic Graph Visualizations

Table 5.2.1: A classification of visualization approaches for large dynamic graphs.

		time		
		unreduced	reduced	
			selection	abstraction
structure	unreduced	UU  only suitable for large displays [EGK ⁺ 03]	US  Online Dynamic Graph Drawing [FT08]	UA  Software Cities [SL10]
	reduced	SU  Dynamic Network Visualization in 1.5D [SWW11]	SS  TimeRadarTrees [BD08]	SA  Spiral Treemap [TS07]
	abstraction	AU  BiGraphiXplorer [JSS ⁺ 09]	AS  Community Detector [FBS06]	AA  Coarsened Difference Graph [Arc09]

provides a compact overview of the available techniques (compare their listing in Section 2.2.4), as most visualizations for large dynamic graphs fall into one of the table cells. The shown technique in each cell serves as a representative example of a cell's specific combination of reducing structure and time. The individual rows and columns of the table are shortly described in the following.

Unreduced Structure/Unreduced Time: The first row and the first column keep the structure and the time unreduced respectively, so that these visualization techniques contain a lot of details. Especially the combination of both leads to huge layouts for large dynamic graphs. They are thus only suitable for very large displays, such as display walls [EGK⁺03] or poster printouts [HKHB07], which have the necessary space and resolution to show them. Otherwise, if only one aspect is left unreduced, visualizations can be made to fit a regular display by using a drastic reduction of the respective other aspect – e.g., by selecting merely a single time point for which to show the full graph or a single node for which to show its complete development over time.

Selected Structure: Selecting only a small part of the graph structure leaves more space for representing the dynamics of this smaller part. Depending on how small a selection is

made, its development can either be shown over the entire time line or only for a reduced set of time steps. The selection can range from just a single node as in [SWW11] which is thus able to display many time steps, to large selections which also require a reduction of the time aspect, for example by abstracting it [GW06]. Another often encountered variant is the selection of only a certain kind of nodes, i.e., only leaf nodes of a tree for which to show the temporal development [TS07].

Abstracted Structure: Visualization techniques that abstract the graph structure of the graph and use these abstractions for visualization, can likewise abstract away either more or less structural details, thus giving more or less space to the temporal aspect. An approach for a rather heavy abstraction of the graph structure is to capture it with different graph metrics [BB05, PD08], each expressing some structural property of the graph and plotting them over the course of the entire time line, e.g., in graph complexity plots [JSS⁺09]. Techniques that abstract away less of the structure are for example graph clustering techniques [BC01]. They show only clusters of nodes and their interrelations – either by a layering [FBS06] showing a few time steps at once, by animating its evolution [FT04] showing only a single time step at a time, or by small glyphs showing the general trend of each cluster (see Section 4.2.2). Clustering and metrics can also be combined by clustering the graph at first and then computing cluster metrics such as the average number of nodes per cluster, which are then shown in time-value plots [FBS06].

Selected Time: When cutting down on the number of time points, many techniques select only a certain time interval or even just a single time point for which to show the graph. Visualizations of single time points usually make use of animation, which is interactively steered by a slider on the time axis, with which a user can pick a time point of interest [FE01]. Multiple time points from a time intervals can be represented by multiple small drawings [SLN05] or layers in which changes between subsequent time steps are highlighted [Bra01b]. A wide range of different layering approaches exist for 2D [TA08], 3D [BC03], and even concentric arrangements [BD08]. Independent of the representation, all techniques try to maintain the user’s mental map of the graph structure as good as possible for which different concepts exist [BIM12, DGK01, DG02, FT08, PHG07].

Abstracted Time: When time gets abstracted, it can be done via temporal clustering to group time steps that are similar with respect to some kind of measure (see Section 4.2.3), or simply by generating a single large supergraph over all time steps. The latter reduces the temporal aspect to an attribute value for each node and edge which can be incorporated into the visualization – e.g., the time point a node was created or last modified. One way to incorporate it is by using the third dimension, e.g., by mapping a node’s creation time onto its height [SL10].

Combination: As it was already mentioned, reduction of both aspects can be combined to meet the constraints of the visual entity budget. Some techniques actually interweave reduction of both aspects so that one depends on the other – e.g., by first constructing the supergraph over all time steps (abstraction of time) and then clustering connected subgraphs which are simultaneously present (abstraction of structure) as done in [Arc09]. Also, combinations of selection and abstraction of the same aspect can be observed in some cases. For instance, the Gephi graph visualization platform [BHJ09] allows the user to first select a time interval (selection of time points) for which then a supergraph

is computed (abstraction of time). Another example are focus+context techniques for graphs which combine the selection of a focus region with the abstraction of the context region [LA93].

The list of given examples is far from exhaustive, but even this selection of visualization techniques gives already a good impression of their diversity. Furthermore, the classification itself can be used for a preliminary decision which visualization techniques to use for a given task. As an abstraction maintains the overview of the data at the expense of the detail, techniques utilizing it are mostly suitable for overview tasks. Whereas a selection preserves details while trading in the overview, which makes techniques using a selection approach more suitable for detail-on-demand tasks. So, if for example an overview of the structural aspect is needed while details of the temporal development shall be explored, a technique from the category (AS) should be considered. The visualization strategy presented in the following section draws upon this classification and utilizes it as a practical concept to integrate different techniques.

5.2.2 In Situ - A new Approach for Large Dynamic Graphs

The classification given in Section 2.2 concerns individual visualization techniques that all have their individual strengths at showing the graph structure, the time aspect or a certain compromise between both. In the light of visual analysis for which not a single best-suited visualization exists, it is only natural to attempt to patch different visualizations together that correspond to the local characteristics of the data. This has already been done for both of the two aspects time and structure independently. As discussed in Section 2.2.1.4 for the structural aspect, several visualizations are combining different representational paradigms such as explicit node-link representations, implicit space-filling techniques, and matrix views to better display subgraphs of certain topologies [HFM07, RMF09]. Similar ideas were followed in the visualization of temporal data, where for instance high detailed line plots and bar charts are often used in an enlarged region of interest and less detailed color coding for the context [BG03, BSM04, MMKN08]. All these approaches share the same idea, as they adapt the visualization locally, *in situ*, bringing to bear the structural or temporal aspect of the data. The in situ visualization concept presented here is a generalization of these approaches which is able to locally combine visualizations in the structural and in the temporal domain likewise.

The *Visual Analytics Mantra* proposes a well-established guideline for the iterative visual analysis of large data sets: “Analyse First, Show the Important, Zoom, Filter and Analyse Further, Details on Demand” [KMSZ06]. While the mantra describes this iterative analysis as a series of computations (“Analyse First”), visualizations (“Show the Important”), and interactions (“Zoom and Filter”), it does not relieve the user of deciding which concrete technique to use at each step. Nevertheless, this choice of different techniques is not to be understood as a burden, but instead as a powerful opportunity for the user to focus the analysis on specific aspects of the data. Especially when facing an unknown data set for which it is not known in beforehand which aspect may be important and should thus be focused in the visualization, being able to switch and locally adapt different techniques is extremely helpful for a swift analysis. In this case, an in situ visualization is a powerful tool, as it allows the user to locally reconfigure or switch the representation.

In the following, the in situ visualization is presented as a means to pursue the Visual Analytics Mantra by a stepwise local refinement of an initial suitable overview or base visualization. At each refinement step, it allows the user to **select** different subsets of data, then **choose** an appropriate and desired visual representation for each selection, and finally to **embed** these representations right in the place where the selection was performed. The selection as well as the visual representation can be altered at any time allowing the analysis of different subsets by different representations.

5.2.2.1 Selection of Data Subsets

In the in situ visualization, the selected area within a base visualization serves at the same time as the selection mechanism and as the drawing area in which to embed the visualization later on. Therefore, a detached selection mechanism using sliders or input fields for threshold values would not work in our case and thus an immediate selection mechanism that works within the base visualization should be used. This is usually a rectangular or a freeform selection tool that can be applied right on top of the visualization for both aspects – structure and time. Hence, targets of selections can be either nodes, edges, and subgraphs for the structural aspect, or time points and intervals for the temporal aspect. Since the different visualization techniques all show the data from different perspectives, they allow and forbid different types of selection. For example, aggregated representations do not allow the selection of individual nodes or time steps, or even both as it is the case for class (AA) in Table 5.2.1. Yet, they allow the user to select parts of the graph structure and the temporal domain depending on their characteristics such as a peak in the number of nodes or edges for which the corresponding time point(s) shall be selected.

More complex selection criteria which are hard to perform interactively via mouse clicks can furthermore be (pre)computed in the “Analyse (first)” step, so that elaborate statistical measures are readily available as node/edge attributes for interactive selection. This rather abstract view on selections for the in situ visualization refinement allows the perception of the visual analysis process as described by the mantra as a sequence of such selections: starting from an overview, subsets of interest are subsequently chosen until a desired piece of information is found. This provides a general interface between the different selection steps allowing the user to recursively redefine a previous selection and all other selections that have been defined in its context will adapt accordingly. Another interesting observation is that the nestedness of in situ visualizations provides a good sense of their provenance, as the different steps of their creation are clearly visible.

However, in some cases such an immediate selection may lead to ambiguities: in a Treemap for instance which already shows the child-parent-relation through nesting the nodes it is not clear if to extract all nodes within the selected area or only the leaves as only these are actually visible and thus explicitly selected. To resolve this problem, approaches for refining the selection are necessary.

In Situ Approach: First, this problem can be solved by directly using the in situ visualization. Therefore, this decision is delayed by extracting all items at first, visualizing them, and allow the user to refine the selection within this visualization. For instance, a

sunburst mapping the depth of the nodes directly to the layout enables the user to easily select distinct levels of that tree for further, recursive embeddings. Thus this strategy in itself already brings with it the means to hierarchically refine the selection without the need for additional mechanisms. However, this approach increases the nesting depth of embedded visualizations resulting in a faster decrease of the available screen space for these visualizations.

Filtering Approach: Another solution is to interactively define selection mechanisms that select from all items contained in the selected region only those of interest (compare the DoI selection approach introduced in Section 4.3). This definition can be facilitated by detached mechanisms such as sliders or input fields allowing the user to specify more concretely what to include in the selection and what to discard. Another way to specify more concretely what to select is to temporarily embed an in situ visualization which is specifically chosen to be able to refine the selection, before releasing it again and continuing the visual analysis with the selection made. This procedure saves screen space but it leads to in situ visualizations which do not necessarily show all underlying data items anymore, as they have been refined. As this violates the original mechanism of the in situ visualization, it has to be signalized to the user that a filtered and not the complete subset is shown in the embedded visualization. For instance, a small icon on top of the embedded visualization is included which also serves as a button for temporary showing the filter definition allowing the user to redefine the selection.

Which solution to use is left to the choice of the user. One challenge though, is to select suitable and compatible visualization techniques to be displayed inside the selected area in order to investigate its characteristics and to select further. This issue is discussed next.

5.2.2.2 Choice of Visualization Techniques

As much power as the free choice of visualization techniques provides to the visual analyst, choosing a suitable one from the many techniques available remains challenging. This is where the classification introduced in Section 5.2.1 comes in as a helpful tool, as it gives a high-level overview of all possible visualization techniques for dynamic graphs. Hence, a thumbnail version of Table 5.2.1 is provided as a first selection menu as shown in Figure 5.2.1a for choosing the class of interest, depending on whether time or structure is in the analysis focus. Then, in a second step, only techniques of the chosen class of visualizations are presented to choose from. Yet, this free choice approach requires the user to know when a visualization of which class could and should be used.

Hence, an alternative approach is necessary that provides a more guided access to the visualization techniques. In contrast to the first, it must not only suit the data to be shown and the selection to be carried out next, but also the stage the analysis is currently in. For example, an in situ visualization of an overview-on-demand inside of a detail view, as it is realized in [JH07], would be somewhat counterintuitive in the context of the Visual Analytics Mantra, which rather proceeds from overview to detail. Thus, it makes sense to use the mantra to cut down on the number of applicable visualization techniques. For example, for a selection of only a dozen nodes, no aggregation and no further selection

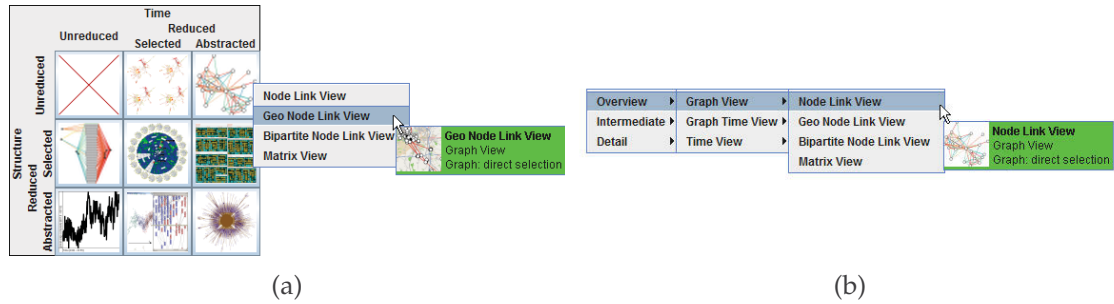


Figure 5.2.1: Different menus for the selection of a visualization to be embedded: (a) matrix menu based on the classification introduced in Section 5.2.1. (b) hierarchical menu based on the current exploration phase and aspect.

would be necessary, as the analysis has apparently already reached a fairly detailed level on the structural aspect, allowing for techniques from classes (UA) or (US) – depending on how the temporal aspect shall be treated. Overall, three phases can be differentiated and each phase corresponds to a specific set of suitable visualizations.

Analyse First, Show the Important: In this first phase, there is too much data, which would not fit in the visual entity budget without being abstracted by mapping multiple data items to a single visual entity. This is usually done by running analysis methods to determine sensible abstractions, such as temporal or structural clustering. Hence, appropriate classes of visualizations for this phase are (AU), (UA), or (AA) which reflect the abstraction in their visual encoding. As techniques of these classes provide a first overview of the data, they are termed **overview techniques**.

Zoom, Filter and Analyse Further: After a first abstracted visualization of the data is given, subsets of interest can be selected for a more thorough examination. However, a single selection may not be enough to reduce the data to fit in the visual entity budget, so that additional abstractions may have to be calculated for this subset before visualizing it. Classes that can be used for this phase are (AS) and (SA) from the Table 5.2.1. As these visualizations are still abstract but can display more details because of the selection, they are called **intermediate techniques**.

Details on Demand: Through multiple iterations of the previous phase a sufficiently small subset can be selected so that very detailed visualizations can be used. This allows the application of the most detailed visualization techniques contained in the classes (SU) and (US), depending on which aspect has been drilled-down to the detail level. If both aspects have been drilled down, techniques from the class (SS) can be chosen. Accordingly, they are named **detail techniques**.

Based on this separation of techniques into different phases a second selection menu is proposed that first groups the visualizations according to the exploration phase into overview, intermediate and detail techniques. The visualizations within each group are then furthermore categorized according to their class – whether time or structure is in

the analysis focus. In a last step, only techniques of the chosen class of visualizations are presented to select from. The resulting selection menu is depicted in Figure 5.2.1b.

On the basis of these two selection menus, the user can decide between free and guided access to the visualization techniques. Here, both schemes allow a very space-efficient display of the visualization techniques that can possibly be applied after a selection was made. If a technique is selected that turns out not to be suitable due to extensive visual clutter, it is always possible to switch to a different technique or to enlarge the selected region and with it the embedded visualization by using any of the methods described above. After a visualization is chosen, it has to be embedded in the area of the selection, which is detailed next.

5.2.2.3 Embedding of the Visualization

Placing a visualization inside another visualization is not uncommon, even though a local adaption of a view is often not perceived as an embedding while it could very well be used in such a way. Common approaches are:

Multiple Coordinated Views: e.g., by placing different visualizations inside a matrix visualization [MDH⁺03]

Focus+Context: e.g., by using semantic lenses [BSP⁺93] or portals [OW00]

Overview+Detail: e.g., via transient overlays, as described in [JH07]

One of the most important requirements for an embedding within a graph visualization is to preserve the context [SZG⁺96]. This cannot be guaranteed by multiple coordinated views, which generally arrange visualizations side by side, or by overview+detail approaches which just superimpose one visualization with another, leaving only the focus+context techniques. A second requirement would be the ability of the approach to cope with the recursive nature of its application, as it is needed for repeated refinements via selection. While lenses can be stacked on top of one another, this is far from trivial, as conflicts between the different lenses may arise [TFS08a]. Whereas for portals, multiple levels of nesting have been shown to work without such problems [WOA⁺01]. Hence, the in situ visualization is based on a portal-based embedding with some extensions that go beyond the original portal concept, such as the ability to connect nodes inside and outside of the portal to preserve the overall graph structure.

Portal-Based Approach: As described in [OW00], portals are defined as two-dimensional regions of a view showing another view. In this case, the extent of the selection made in a view corresponds directly to the region of the portal. That means that the data shown inside the portal is exactly the data positioned in the region of the base view now being occupied by the portal. Hence the position, size, and shape of the portal define the data inside it. This allows the user to change the data shown inside a portal simply by dragging, resizing, or reshaping the portal to include or exclude certain regions of the base visualization. Additionally, it is also possible to open up multiple portals by making multiple selections allowing the user to analyze different regions simultaneously using different visualization techniques, as it can be seen in Figure 5.2.2. Making subsequent and

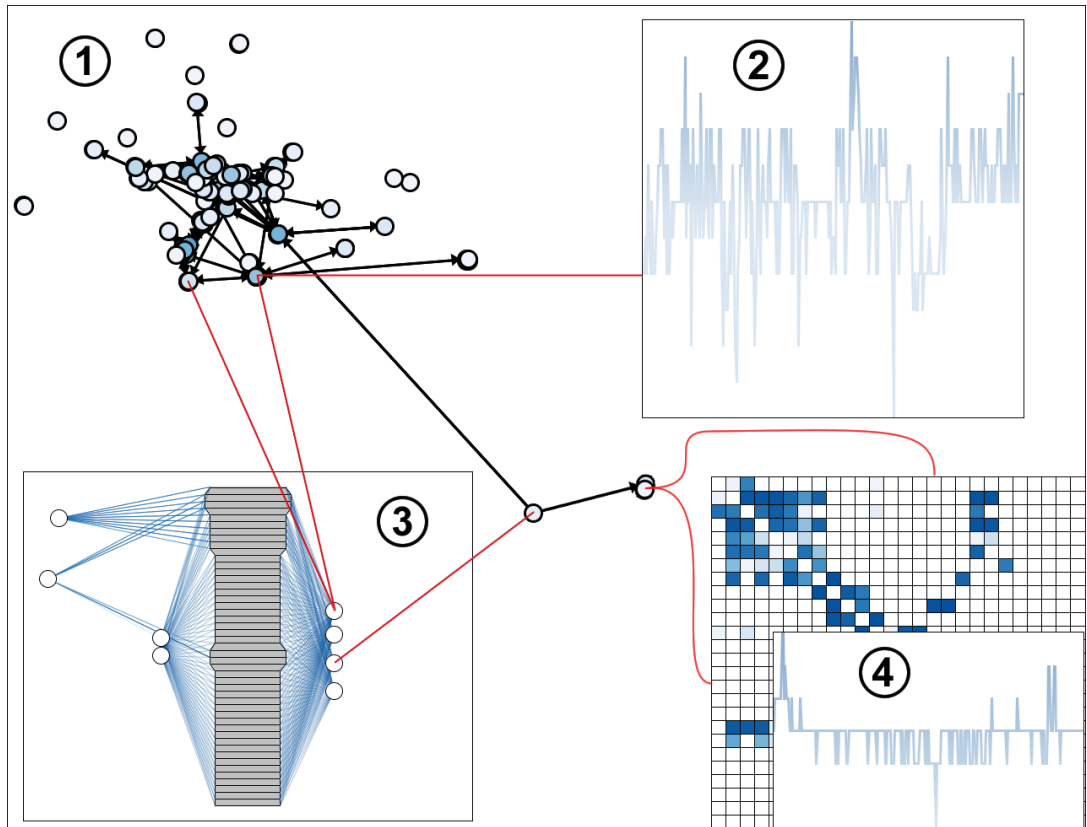


Figure 5.2.2: Example showing the in situ visualization. 1: base visualization showing a node-link layout of the supergraph and multiple embedded visualizations. 2: in situ visualization showing a complexity plot for the underlying subgraph. 3: in situ visualization showing a 1.5D visualization of the underlying subgraph, connecting links are overlaid in red by the base visualization. 4: recursive use of in situ visualization to show a complexity plot for a subgraph selected in a matrix view.

multiple selections, and choosing different views for them provides exactly the needed functionality to carry out the iterative analysis process of going from an overview to the detail on demand, while at the same time maintaining the context of each analysis step.

Extending Portals to In Situ Visualizations: Portals have the limitation that they are defined to be completely independent from their base visualization. This may suffice for unrelated data items within a data set, but it does not go far enough for graph visualization, where complex relationships must be maintained even across multiple levels of nesting. Therefore, the strict compartmentalization of the portals has been relaxed in order to allow for a mutual adaptation. These relaxations are **portal awareness** (the base visualization knowing about the portal and its contents), **context awareness** (the portal knowing about the base visualization and its contents), and **overlays** (the base visualization being allowed to draw on top of the portal).

Portal Awareness: As portals may interfere with the existing edge routing in the base view, it is of importance for the base view to have information about the portal firstly as a whole to be able to reroute edges around the portal, if they do not lead to nodes within

it. Secondly, the base view should also be aware about the layout of the data inside of the portal, as it is certainly different from the original positioning in the base view and thus requires an adaptation of the connections leading to them. Otherwise, after opening up a portal, it would no longer be possible to determine to which nodes inside of a portal the edges lead. This modification is illustrated by red links in Figure 5.2.2 connecting the nodes of the base visualization (a node-link representation) with the relayouted nodes in the 1.5D visualization in Selection 3 and the correct row/column of the matrix visualization in Selection 4.

Context Awareness: Similarly, it makes sense for the portal to have knowledge about the surrounding base visualization. This allows the visualization within the portal for instance to take the positions of connected “outside nodes” into account when computing a layout. This makes sense when, for example, nodes that are connected to the base view are automatically laid out at the border of the portal, facing the right direction, so that connecting it to the base view will not require to route edges all the way around the portal or across it. The 1.5D visualization of Selection 3 of Figure 5.2.2 for instance placed all nodes with connections to the base visualization on the right side to minimize their edge lengths. Another example is to align the orientation of a time plot with the orientation of the time axis in the base view, making it easier to follow the time axis.

Overlay: To achieve the tight interlinking and connection between parent and portal, it is finally necessary, to extend the portal concept in another aspect: originally, portals manage their drawing area themselves with no interference from the parent. Yet, to provide continuous links between both, the parent is allowed to overlay the portal with additional graphics, such as edges. This is exemplified by the red links visible in Figure 5.2.2 connecting nodes from the base node-link representation and from the 1.5D visualization in Selection 3.

To apply this generic approach in practice, a number of functional issues have to be solved such as the handling of arbitrary selections and the different possibilities to deal with higher space demands for an embedded visualization than is actually available.

5.2.2.4 Practical Considerations for the In Situ Visualization

The greatest concern for a visualization of large data sets is the available screen space. Thus, the in situ visualization tries to use the space as efficiently as possible by not arranging visualizations of subsets side-by-side in multiple linked views, but by embedding them so that each data item is shown only once on the screen and does not increase the overall space demand. Therefore, the in situ visualization utilizes the selected area within a base visualization as the drawing area in which to embed the visualization. Yet, while data selections can be of arbitrary shape, visualizations tend to be optimized for a rectangular drawing area. Besides the shape, there are also a number of constraints complicating this embedding, including aspect ratio, size and connectivity. In case of multiple selections having been defined in the base view where each shall be analyzed for itself these aspects can be accompanied by overlapping selection areas. This raises the chal-

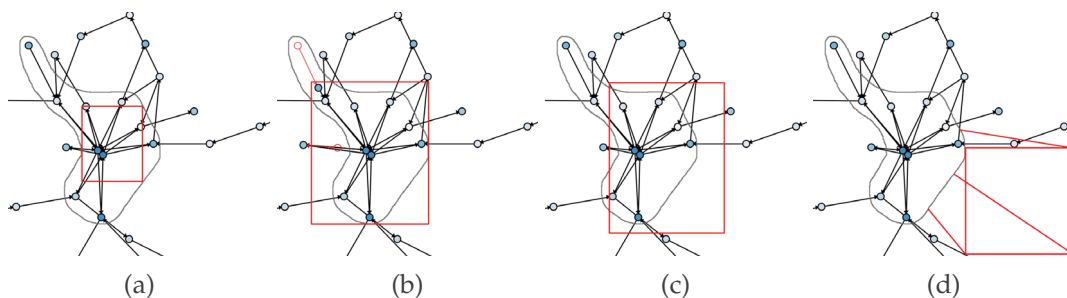


Figure 5.2.3: Different adjustment strategies for embedding rectangular visualizations in arbitrary shaped selections of a node-link visualization: (a) inscribing the largest rectangle of a desired aspect ratio, (b) adjusting the layout by relocating all unselected nodes to positions outside of a circumscribed rectangle, (c) adjusting the selection to encompass also any empty space that can be gathered around it, and (d) eccentric placement in a more suitable spot with additional connections to the original selection to maintain the relation.

length of finding suitable drawing areas for an embedding of visualizations despite the possible differences between selection area and visualization. For the sake of brevity, the following discussion assumes non-overlapping, contiguous selections of arbitrary shape in a node-link visualization and rectangular visualizations to be embedded. Solutions for cases in which these assumptions do not hold, will be sketched at the end of this section.

The general approach to solve this problem is an adjustment of the selected region. The smaller this adjustment is, the closer the embedding will be to the actual selection made. For example, inscribing the largest rectangle into the selection as a drawing area for an embedding allows a full realization of the *in situ* property. Yet, the more constraints have to be taken into account, the less likely it becomes that a suitable drawing area can be found *in situ*. The three adjustment strategies described in the following are discussed along this line: from **inscription**, and **adaptations** of layout or selection, to **eccentric positioning** as a last resort – and thus ranging from *in situ* to *ex situ*.

Inscribing in the Selection: The most obvious strategy is certainly to inscribe the largest axis-parallel rectangle into the selection, as it is shown in Figure 5.2.3a. For its computation, algorithmic geometry provides algorithms, such as the $O(n \log^2 n)$ divide&conquer algorithm from [DMR97]. These algorithms are fast enough to quickly determine whether a large enough inscribed rectangle exists or not. Usually, even simpler heuristics can be used, as it is not of outmost importance to find the largest inscribed rectangle, but a rectangle that is large enough. In case no such rectangle can be determined, the adaptation has to move on to one of the next strategies.

Adaptation of the Layout and/or the Selection: This strategy tries to establish a rectangular drawing area which is larger than the inscribed rectangle. The first way of achieving this is by defining such a larger rectangular region on top of the selection and then modifying the layout of the base visualization by “pushing” all unselected nodes out of the boundaries of the rectangle and by “pulling” all selected nodes inside. This is illustrated in Figure 5.2.3b and realized as a force-directed approach that places an invisible

node at the center of the rectangle, which repels all unselected nodes inside of the rectangle and attracts all selected nodes on the outside. The limits of this approach are obvious, as it can only be applied if the node positions are not used to encode any attribute, as for instance the *Semantic Substrates* [SA06] do. Additionally, to maintain the mental map, the transformation of the layout should be animated to help a user identify and match nodes before and after the layout adaptation.

The second way to achieve a larger rectangular drawing area is to start with the inscribed rectangle and extend it outwards in all four directions until it hits an unselected node. This way the layout remains preserved, but at least none of the empty space around a selection is wasted. It is exemplified in Figure 5.2.3c.

Both ways achieve a larger rectangular area than by simply inscribing a rectangle. They do this at the cost of slightly violating the in situ principle, as they use more drawing space than the selection actually grants them. Yet, they are still overlapping the original selection to a large extent, thus making the immediate connection between both nevertheless obvious.

Eccentric Positioning: This strategy positions the drawing area for the in situ visualization somewhere in the base visualization but in contrast to the previous strategies not on top of the selection, as shown in Figure 5.2.3d. Therefore, the position of the largest empty rectangle is computed – e.g., with a fast matrix search as proposed in [AS87]. Then the visualization can be embedded there and linked via connector lines to the original selection. This approach clearly violates the in situ principle, yet it may be necessary to resolve an otherwise fruitless attempt to find enough screen space for an embedding. Additionally, it allows the comparison of the same subset with different visualization techniques as well as with the base visualization.

Even though these adjustment strategies have been illustrated for the example of node-link representations, they can be either applied directly or with slight modifications to other types of visualizations as well. Yet, as their realization depends on the concrete visualization technique, not every technique allows for all three strategies to be used. If inscribing a rectangle or adapting the layout/selection are not possible, at least an eccentric positioning can be used. This is obviously the case for visualizations in which data items overlap, e.g., Beamtrees [vHvW02], and where thus no designated space can be reserved for embedding a visualization. The only exception to the applicability of eccentric positioning are space-filling representations which do not leave any free space outside of the selection to use for an embedding.

Dealing with the possibility of overlapping or discontinuous selections is relatively straightforward. Overlapping selections which shall be analyzed for themselves are subtracted from each other so that only the regions which one selection inhabits individually are used for embedding. Discontinuous selections can be handled in a number of ways: through a layout adaptation with a rectangular area in which all items from the selections are pulled together, via an eccentrically positioned rectangle which is connected to all of the selections, or by dealing with each selected region by itself and connecting the found rectangles with lines indicating their relation. Besides these two issues, it can also occur that a non-rectangular visualization shall be embedded, e.g., a radial one such as

the TimeRadarTrees. In this case, other shapes can be used as well – e.g., inscribed circles or polygons.

As a result of the recursive embedding of visualizations and the limitations of the above described approaches these embedded visualizations can be rather small. To make enough room for an embedding in these cases, additional techniques are needed for increasing their sizes. Each embedded visualization can of course be treated as a small window by itself, which permits to zoom and pan independently of the base visualization, allowing the user to analyze the enclosed parts of the data and embedded visualizations in more detail. The drawback of this independence is the inability to maintain the context, as connections to nodes within the embedded visualization can no longer be drawn if these nodes are invisible when only a zoomed-in part of the data is shown. Hence other methods are needed, that allow the user to locally increase the drawing space of an embedded visualization within the base visualization while simultaneously preserving the context.

One example of such a method is the fisheye distortion as it is introduced in [Fur86]. As the in situ visualization allows the combination of very different visualization techniques there are different demands to a fisheye transformation. For instance, when applying fisheyes to node-link layouts the topology of the graph has to be preserved, calling for a *Topological Fisheye View* [GKN05] in case of a node-link representation or for the *Balloon Focus* technique [TS08] in case of a Treemap. Additionally, the recursive nature of the in situ visualization has to be taken into account as well. The *Variable-Zoom* [SZG⁺96] or the *Continuous Zoom* [DBHH94] serve this purpose, as they allow the combination of different fisheyes along a given hierarchy. They only differ in how the magnification factors are calculated, which the variable-zoom does independently for each level and the continuous zoom accumulates from the lowest level. This yields a more stable interaction with the variable-zoom, as changes only occur locally in a single embedded view and have no influence on the parent view – yet to enlarge a deeply nested view, each level has to be enlarged separately. In contrast, the continuous zoom just needs to enlarge the view of interest however deeply nested it may be, because the bottom-up accumulation of magnification factors resizes all parent views as well. As this significantly reduces the effort of enlarging embedded views, the latter approach is applied along the in situ visualization.

After a technique is chosen and the data is visualized in situ, the connections between base visualization and embedded visualization have to be maintained. In case of a graph visualization, this means to maintain connections between nodes, and in case of a time visualization this means to maintain the orientation and order of the time axis from the parent view. Especially connecting edges from a node-link base visualization with an embedded visualization of a different form raises some questions which need special approaches to deal with them – none of which is complicated, yet each of them has to be specifically considered in one way or another. An example for an embedded graph view would be the connection with a matrix representation, as a matrix offers not just one possible location to connect with a node, but actually four of them: at the beginning and the end of a node's row and column – basically on all four sides of the visualization.

5.2 An In Situ Visualization for Switching and Combining Dynamic Graph Visualizations

One solution to this issue is to connect to the two closest borders of the matrix and thus only to either end of the corresponding row and column. When embedding a time visualization into a node-link representation the edges leading to the time visualization can be split and connected with each time point generating a 1.5D visualization [SWW11]. Yet, a high number of connections can result in extensive clutter on top of the embedded portal preventing the analysis of the selected subset. Therefore, additional techniques such as edge bundling, transparency and highlighting of a subset of the connections are necessary to cope with possible visual clutter. However, maintaining connections should not be underestimated, as it is an effort well worthwhile: connections are important for path-based analysis tasks and can be used for edge-based navigation [TAS09].

As an example of how an actual implementation of the in situ visualization can look like and what it is capable of, the next section will step through two use cases using in situ visualizations.

5.2.3 Use Cases and User Feedback

In this section the flexibility of the in situ visualization is demonstrated by utilizing it for the analysis of two dynamic graphs from different application areas and by reflecting feedback from experts of these areas. The first graph stems from the area of model versioning, the second from the field of mesh networks. Based on these data sets, a small user study was conducted to get a first impression of the usefulness of the in situ visualization.

5.2.3.1 Model Versioning

In biology, graphs are used to model different kinds of biological systems. An important case of biological modeling is to establish models of biochemical reaction networks. Biochemical reaction networks are often described as bipartite graphs, where the nodes correspond to chemical compounds and reactions and the edges link the compounds to their respective reactions in which they take part. Due to new insights from experiments in the laboratories, these models are subject to continuous development and adjustment. Thus, it is important for the scientific community to track these model changes to support their refinement and the construction of new models. State-of-the-Art model repositories such as the *BioModels Database* [LDR⁺10] are thus equipped with versioning systems which keep track of the structural changes of the models, very much like an SVN system does for program code. As these changing model structures are nothing else than time-varying networks, the in situ visualization was coupled with the *BiVeS Framework* (see <http://bives.sf.net>) which was specifically designed to provide specialized version control for biological models.

In the following, one exemplary data set that was captured using the BiVeS framework is used. This reaction network has 162 nodes and 236 edges captured over 227 time points in 15 branches. Additionally, with each node two attributes are associated. The first attribute is describing a parameter which is used for initializing simulations of this model. The second attribute, shared also by the edges, specifies if a node or edge is present in a version. While this dynamic graph may not be complex in its size, the branching temporal aspect neglects the utilization of many very common visualization methods. For

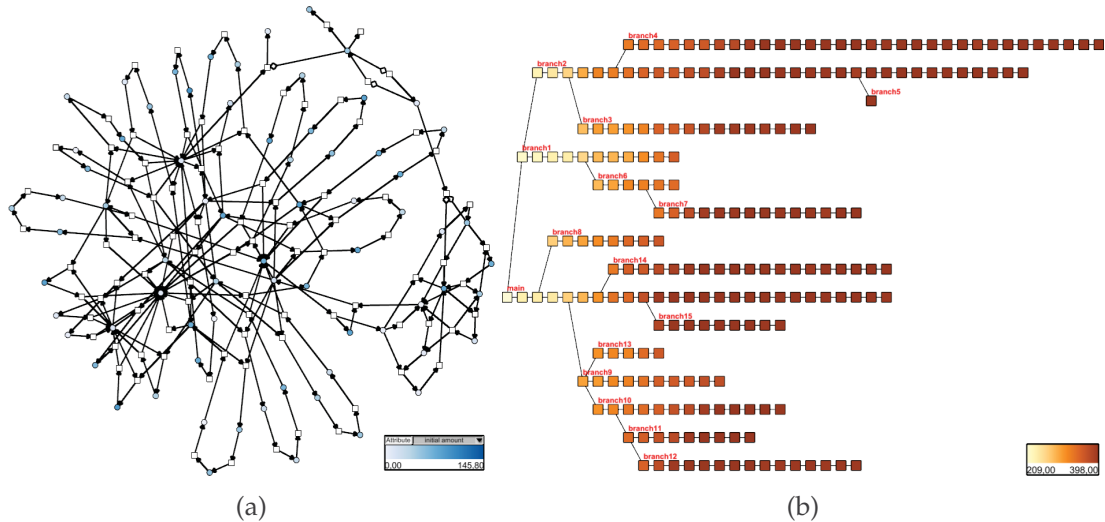


Figure 5.2.4: Possible base visualizations of the versioning history of a biological model. (a) shows the supergraph over all revisions as an overview of the structural aspect. An initial simulation parameter for this model is mapped on the color of the nodes. (b) shows the branching temporal aspect as an overview, whereby color corresponds to the structural complexity of each model revision.

instance, animations and complexity plots usually used for conveying the dynamics both rely on a linear order of time points and thus cannot compensate the branching time. Similar, a supergraph based overview as shown in Figure 5.2.4a aggregating the model over all revisions of all branches may not be sufficient for a user interested only in specific branches.

Here is where the flexibility of the in situ visualization comes into play. At the beginning, an appropriate base visualization has to be chosen as an overview of the data set. Two possible candidates are shown in the Figures 5.2.4a and 5.2.4b showing either the structural or the temporal aspect. Already at this first step, the user decides whether the temporal aspect or the structural aspect of the graph should be in the focus.

Focus on the temporal aspect: In this case Figure 5.2.4b represents a good starting point as its similar to the state transition graph based visualization as discussed in Section 4.2.3 permits it to reflect the branching temporal aspect. With such a view, it is now possible to get a first overview of the complex temporal development of the model. It can be seen, that multiple versions of this model have been branched out, generating their own time line with revisions to only that particular branch. The structural aspect of the model at each revision is reduced to a numerical graph complexity value that is color-coded at the individual time steps similar to a complexity plot. It is clearly visible that the structural complexity of the model is increasing over time on all branches, before it then remains nearly constant at the end.

In this way, this view allows the selection of interesting time points (versions) or branches for a detailed look, for instance to discover what exactly has changed in between the different versions at the end. By selecting specific branches common visualizations can then be used and embedded such as animating a graph layout across the different revi-

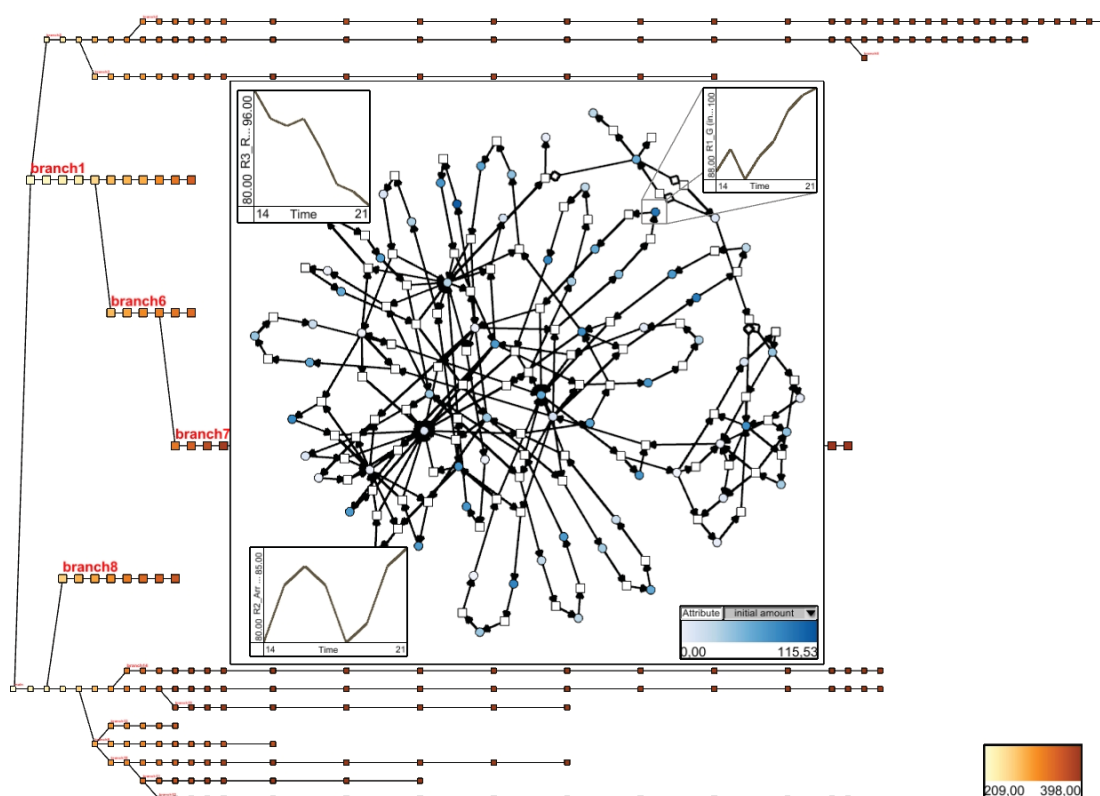


Figure 5.2.5: Visualization of the revision graph of a biological model. The base visualization is showing the branching temporal aspect as an overview. A selection is made on a number of consecutive time steps of `branch7` showing a supergraph visualization of these model revisions. A fisheye is utilized to enlarge this selection. Nodes of interest are selected and thus substituted with a time-value-plot – in one case using eccentric positioning.

sions of a single branch or showing its supergraph. Such a detailed view is opened up as an in situ visualization in Figure 5.2.5 by making a rectangular selection on `branch7` and switching to a supergraph view inside that selection. To take a further look at additional details of the individual nodes, they can also be selected to embed detail views – in this case time-value-plots showing the initial simulation parameter. It is clearly visible from these plots that in different revisions of that model, different parameter settings have been tried in order to adapt the behavior of the model to fit the natural behavior of the biological system it encodes.

Focus on the structural aspect: When starting on the other hand with an overview of the structural aspect as shown in Figure 5.2.4a, it is possible to support the search for a specific revision. If for instance all versions containing a specific reaction or compound have to be found, the user can select this specific reaction in the supergraph of the base visualization and embed a view showing the branching time of the versioned model as shown in the top right of Figure 5.2.6. Here, the same visualization as shown in Figure 5.2.4b is embedded which was formerly used as an overview of the temporal aspect. Yet, in this combination it shows in which versions the selected node exists (dark color) and where it is missing (light color). This procedure can also be applied to subgraphs as

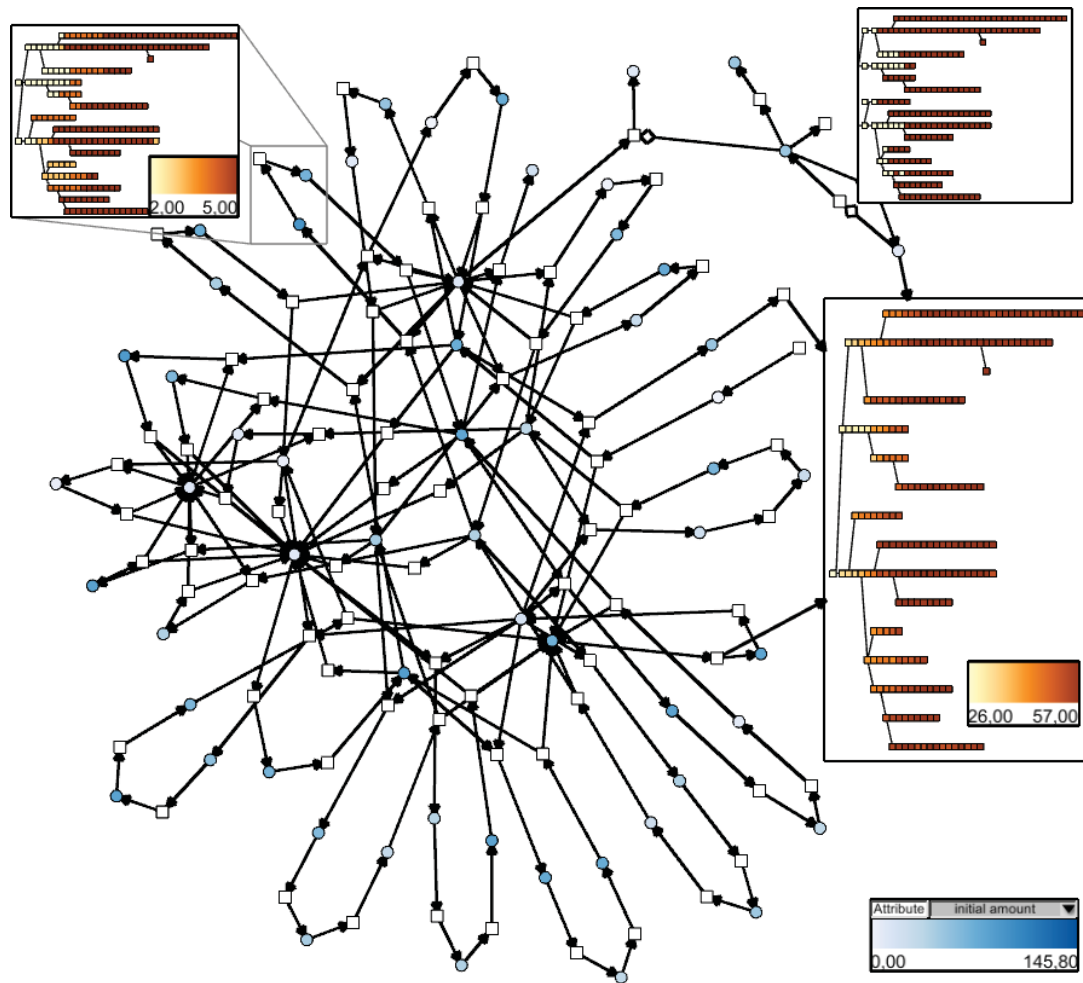


Figure 5.2.6: Visualization of the general structure of a biological model. The base visualization is showing the supergraph over all model revisions. Subgraphs of interest are selected and substituted with a branching time view – in one case using eccentric positioning – showing their existence in the different versions.

shown in the top left for 3 nodes using an eccentric positioning and for a well-connected subgraph on the right.

The next example shows that the in situ visualization scales up to larger data sets as well before reflecting some expert feedback.

5.2.3.2 Mesh Networks

As described in Section 4.2.4 for the OpenNet network, mesh networks consists of network devices such as routers (nodes) and wireless or cable-based connections between them (edges). The following discussion is also based on this data set containing 297 nodes and 2'008 edges along 217'253 recorded snapshots. In the same manner as for the previous use case, at the first step of the analysis, the user can choose freely between very different perspectives by focusing on the structural aspect or on the temporal aspect for the overview in the base visualization as depicted in Figure 5.2.7.

5.2 An In Situ Visualization for Switching and Combining Dynamic Graph Visualizations

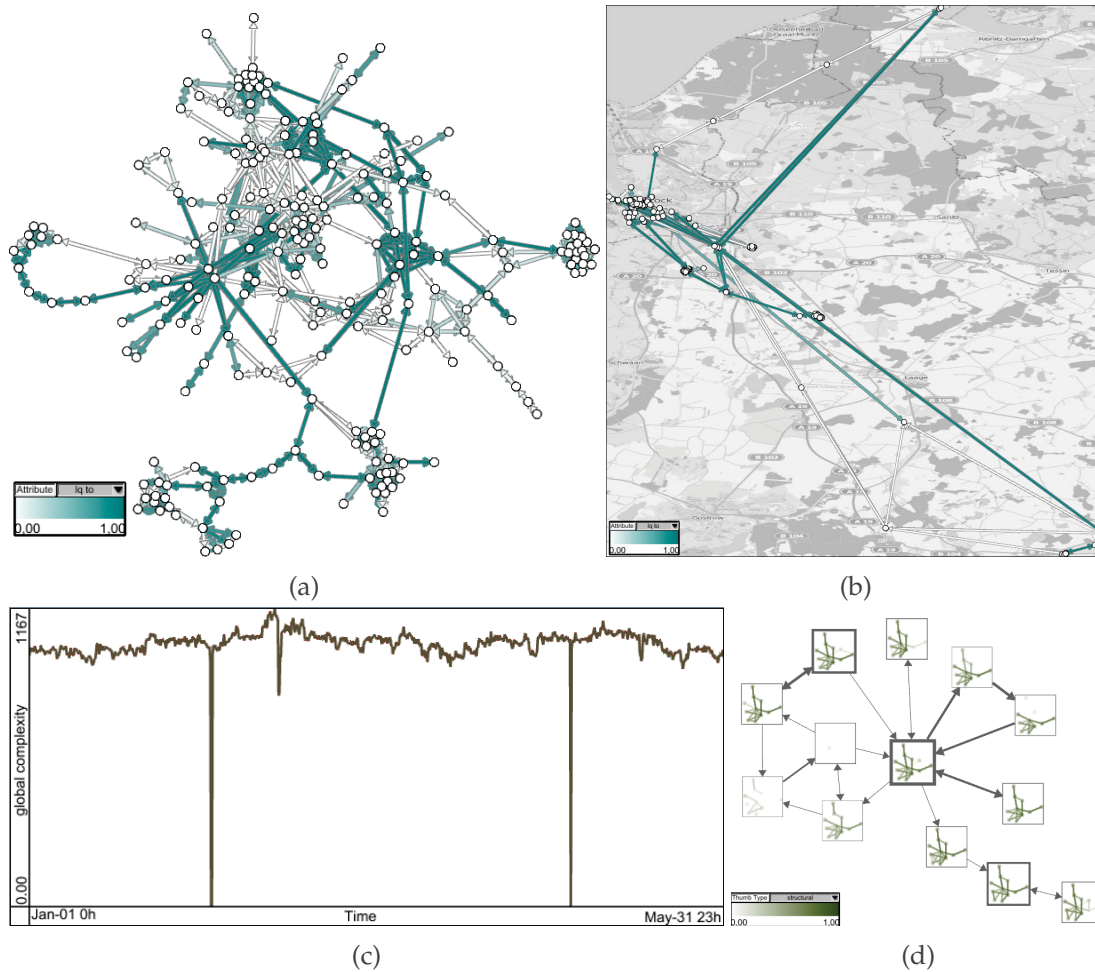


Figure 5.2.7: Possible base visualizations of the OpenNet mesh network. A structural overview based on the supergraph (a) using a force-directed layout and (b) showing its spatial context on top of a map (© OpenStreetMap contributors). A temporal overview based on (c) a linear complexity plot and (d) a non-linear state transition graph.

Focus on the structural aspect: As the OpenNet members are mainly interested in connecting all routers to provide a good network quality satisfying all users, a supergraph can be used as a good initial base visualization. In this way, all connections available during the gathered period can be shown for a more thorough analysis with their connection quality color-coded onto the edges. Here, a force-directed layout as shown in Figure 5.2.7a may support the identification of structural weaknesses such as only few connections between well connected subgraphs. However, it is difficult to orientate in such a layout as with each router often a specific position is associated in and around Rostock.

As the network nodes have geographical positions, these can also be used to layout the graph as it is done in Figure 5.2.7b providing a more intuitive view on the network. Yet, due to the geography of the city, the nodes of this network are very irregularly distributed: there are dense clusters of nodes in the inner city and in the villages around Rostock, whereas in between there are no nodes and thus only sparse connections. While

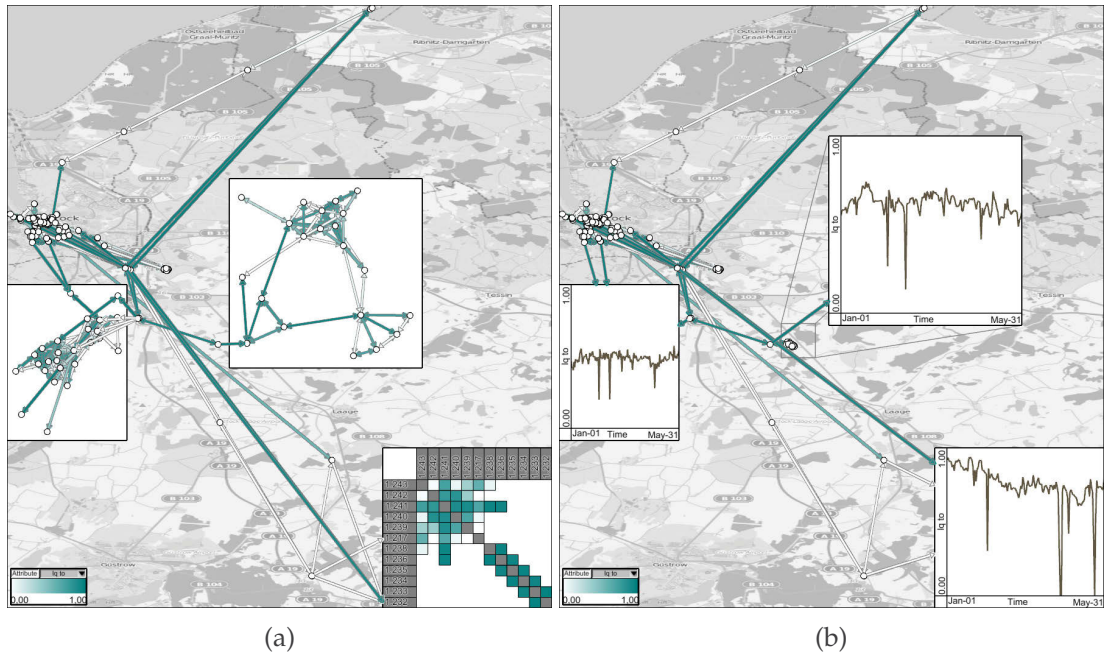


Figure 5.2.8: Visualization of the general structure of the Opennet in its spatial context (map © OpenStreetMap contributors). (a) Node-link and matrix visualizations are embedded to better reflect the structure of spatially clustered subgraphs in the villages around Rostock. The overlay capability of the in situ visualization is used to maintain the connections across different representations. (b) Complexity plots are embedded to reflect the temporal developments of the same subgraphs.

this allows for a very natural orientation in the network, the dense clusters make a detailed visual analysis difficult.

Here, the flexibility of the in situ visualization to locally switch the visualization helps to combine the advantages of both layouts. It facilitates to change the representation of the clusters to another one which better reflects the structure, as it is done in Figure 5.2.8a for the three villages around Rostock using different in situ visualizations and the available empty space around. However, for a detailed view of the city center, where no additional space is available, different adaptations have to be used such as fisheye transformations or eccentric positioning. In this way, it is possible to focus locally on the structure while maintaining the spatial context of the network on a global level.

Alternatively, the visualization in the embedded views may also be switched to represent the temporal aspect as exemplified in Figure 5.2.8b. Such a switch allows not only to break down the overall dynamics of the network to the different villages but also their comparison. In this sense, it is visible that the selected villages on the left and on the top share a similar course over time and thus are structurally more closely related as they exhibit similar peaks/valleys over time. Whereas the third village in the lower right shares only one of the valleys at the beginning of the recording period but shows more drop-offs towards the end of the recording. By selecting time intervals around such valleys the cause of these drop-offs can be localized as it is done in the following for the complete network.

Focus on the temporal aspect: Therefore, a temporal overview such as the complexity plot shown in Figure 5.2.7c can be used instead of a supergraph visualization. By selecting time intervals of low structural complexity, which are especially of interest, those time points can be identified where not enough redundant connections are present and bottlenecks emerge. A local switch to a view focusing on the structural aspect then helps to localize where the problems have occurred as performed in Figure 5.2.9 for the three largest valleys.

These embedded in situ visualizations clearly show that the first and third valley represent drop-offs concerning the complete graph as all connections show very low attribute values. As discussed in Section 4.2.4 these drop-offs were caused by a disconnection of the data recording node. Whereas, the second in situ visualization captures the more local event concerning the drop-out of the smaller villages in the south east of Rostock.

Alternatively to the linear representation of time as shown by the complexity plot, the non-linear state transition graph which was introduced for dynamic graphs in Section 4.2.3 can also be used as a base visualization. As this visualization already groups states with a similar network configuration (see Figure 5.2.7d) it simplifies the selection of interesting states for a closer look. Here, states were extracted based on the similarity of the network on the district level which is also shown in the glyphs. By selecting specific states such as the aforementioned disconnection of the data recording node (left) or the drop-out of the villages (right) it is not only possible to see the corresponding network configuration in its spatial context. But the embedded in situ visualizations can also show the network on a finer structural granularity showing directly which routers are affected as exemplified in Figure 5.2.10. To support the localization of missing routers, the most common state of the OpenNet (center) has also been selected and switched to a map display.

Altogether, these two use cases have demonstrated the versatility of the in situ visualization to cope with very different scenarios. They have also shown the flexibility gained by applying and embedding different visualization on the one hand. But on the other hand, changing the order of these visualization also increases the flexibility and allows the support of different goals.

5.2.3.3 User Feedback

The previous sections described two use cases utilizing the in situ visualization. In this section, the results of a first user study are presented. Therefore, a qualitative user study was performed with 13 participants including nine experts from both domains and four visualization experts. The user study started with a brief introduction and training phase with the in situ visualization as well as a multiple coordinated views system, both using the same techniques. The multiple coordinated views system was simulated by opening a new window for each selection made. After the training phase, the participants were asked to complete a couple of tasks, e.g., identifying unreliable access points in the OpenNet network or comparing the model structure of two different versions of a biological model. During the user study, they were encouraged to think loud and comment what they like and dislike for both systems. The following lists noteworthy observations from

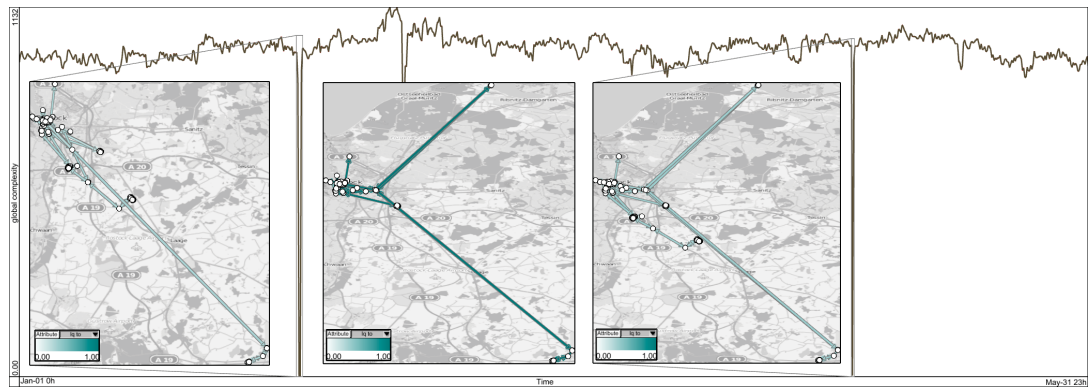


Figure 5.2.9: Linear representation of the temporal course of the OpenNet using a complexity plot. In situ visualizations showing the network in its spatial context (map © OpenStreetMap contributors) are embedded to investigate larger peaks in the complexity plot, in two cases using eccentric positioning.

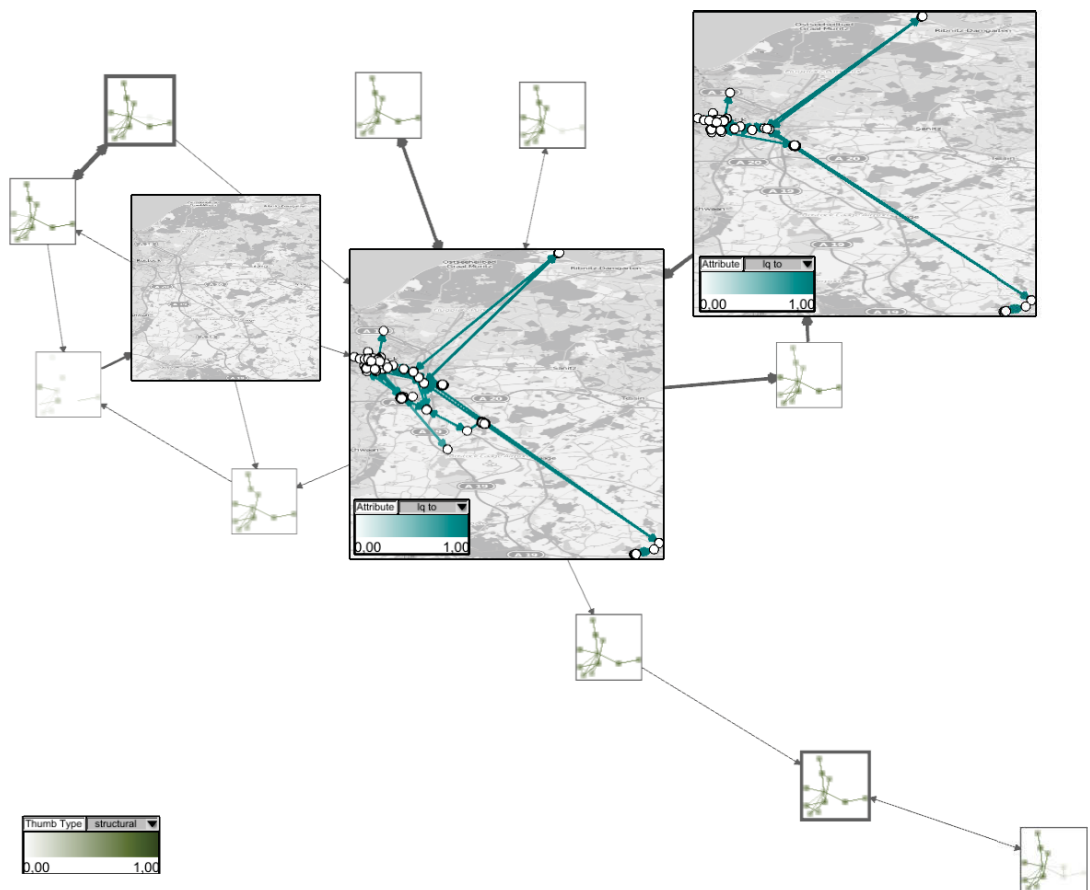


Figure 5.2.10: Non-linear representation of the temporal course of the OpenNet based on a state transition graph. In situ visualizations showing the network in its spatial context (map © OpenStreetMap contributors) are embedded to investigate three different states.

this user study:

Most candidates preferred to start with the in situ visualization as they felt it was more intuitive and flexible. *“The data I want to see is where it is”* was stated by multiple candidates. Especially for the OpenNet data set, they appreciated the direct embedding as it maintained the spatial relations between the nodes and thus supported their orientation in the data. Generally, the overlay of connections between base and embedded visualization was highlighted by many participants as this also supported them in maintaining their overview. They also favored the lens like behavior where no switching from one view to another is necessary allowing them a fast exploration of the data. One participant compared multiple views and in situ to the interaction with mouse and touch pad: *“it just feels more immediate”*. Yet, some of the candidates felt a bit overwhelmed by the freedom the in situ visualization provided them and indicated that they need more training to use its full potential.

Overall, all candidates liked to see the detailed view of the selected data next to the selection. However, some complained that they want to see both the embedded view as well as the base visualization as they were unsure if both views really show the same data and therefore preferred the eccentric positioning of the detail views. Also, when comparing different parts of the data, many participants used the eccentric positioning to move detail views of these parts closer together creating their own multiple views arrangement.

In general, the in situ visualization and multiple coordinated views performed equally well. Identification tasks were solved slightly faster with the in situ visualization whereas multiple views performed better when comparing different subsets of the data. Yet, almost all participants said that they want to have the choice at any moment to switch from multiple views to in situ and vice versa.

While these results are far from being statistically conclusive, they already indicate that the in situ visualization represents a useful complement for common approaches such as multiple coordinated views. It also highlights the importance of eccentric positioning which describes a possibility to combine the intuitive access to the data given by in situ visualizations with the comparability of multiple coordinated views further increasing the flexibility of accessing the data.

5.2.4 Conclusion

In this section, a flexible strategy for the visual exploration of large dynamic graphs was introduced. As discussed in the previous chapters, a large variety of rather different visualization techniques is necessary to cope with the diversity and the size of these graphs. And there is no unified one-view-fits-all visualization that serves analysis needs in all aspects equally well.

To resolve this problem the novel in situ visualization was introduced that allows a combination of these techniques. In contrast to existing approaches such as multiple coordinated views that put different views side-by-side, the in situ visualization is based on a direct embedding that is also able to support the mental map during the visual analysis proceeding from an overview all the way to the details. In addition, it allows the user to shift seamlessly between different analysis foci and visual representations

at any time. Limitations regarding the available screen space can often be resolved by employing standard algorithms and heuristics from algorithmic geometry or by using distortion techniques. The resulting scalability of this visualization was shown in two different use cases. They also highlighted the flexibility gained by the in situ visualization as with even a small set of different visualizations a very diverse set of analysis goals can be pursued just by changing the order of their embedding.

The in situ visualization is accompanied by a novel classification of dynamic graph visualizations that groups visualization techniques based on how they approach the graph structure and the temporal context. In this way, this classification helps to reduce the number of visualization to select from at any stage during the analysis. Yet, the final choice of an appropriate visualization is still left to the user. To further support him in this choice the following section introduces a new abstract overview of the data set and all its aspects.

5.3 Steering the Analysis of Multiple Aspects with a Synchronized Approach

When dealing with an unknown data set, the decision for a specific visualization can become difficult as on the one hand it is not known in before which aspects actually exist and to which extent they are to be found in the data set. For instance, is there a single associated attribute or a large multitude? And on the other hand, it is not obvious which of these aspects hold any interesting insights. Hence, the user often has to use trial-and-error to select an appropriate visualization as well as its parametrization, for example the level of detail for each aspect or which attribute to encode. Especially, when using multiple different visualizations, it can become difficult to keep track of relations between the shown subsets of the data.

In this section², a new visualization – the synchronization view – is therefore introduced. Its contribution to the analysis process is two-fold:

- It provides a compact overview of the existing aspects and their extent.
- As well as analytical means to identify interesting aspects to be analyzed in detail.

It is based on transforming a given data set into a multipartite graph in which each partition corresponds to a specific aspect and connections between partitions capture existing combinations of these aspects in the given data set. In this way, the aspects can be shown and explored side-by-side facilitating their combined and synchronized analysis. Furthermore, it enables traditional and novel forms of navigating the graph data to interactively seek out and investigate behavior of interest across different aspect combinations. On this basis, the user can perform a more informed choice to select an appropriate visualization.

This approach is exemplified with case studies from the political domain, featuring various election data and voter sentiment data from polls for three administrative levels of Germany on three temporal levels. Apart from serving as a running example throughout this section, some of the findings made in the data are reproduced using this visual analysis approach at the end of this section.

5.3.1 Transformation into a Multipartite Graph

To provide a simultaneous and synchronized visualization of all aspects of the graph, every aspect is first divided into one or more partitions also referred to as levels in the following.

- For the structure, nodes and edges are handled as separate levels to maintain the characteristics of multipartite graphs having no connections between elements of the same partition. Furthermore, it allows a simple extension to hyperedges connecting more than two nodes for instance in case of a reaction model.

²Parts of this section have been published in “A Visualization Approach for Cross-level Exploration of Spatiotemporal Data” 2013 at the International Conference on Knowledge Management and Knowledge Technologies [SHS13].

- Both the spatial and temporal context are split into multiple levels according to the levels of granularity they exhibit. For example, the spatial context may be separated into countries, states and counties, whereas the temporal context may be split into the levels years, months, and days.
- At last, for each associated attribute an additional level is introduced.

For each level, all existing instances are gathered. It can thus hold numerical data, categorical data, or nominal data. If a level contains too many instances, as it would be the case for a continuous numerical variable, its instances can be binned on demand into adequately, yet not necessarily uniform sized intervals as they suffice for a concrete analysis. In this way, showing the different levels and their instances already provides a first overview of the existing aspects and their extents.

Next, existing combinations of instances across the levels in the given data are added as edges to the multipartite graph which are referred to as tuples. Such a tuple describe for instance where a node is located and which attribute values it has at a specific moment in time. It is noteworthy, that it is not necessary that a tuple features an instance for each level and thus connects all levels of the multipartite graph. For example, an edge does not have any values for a node attribute and vice versa. Furthermore, nodes may be assigned to locations at different granularities as it is the case for the election use case. Here, nodes are hierarchically organized and each depth level of the hierarchy correspond either to countries, states or counties. With this information it is now possible to not only give an overview of the individual levels but also of their combinations allowing the identification of interesting combinations to look into.

The following synchronization view and the visual analysis mechanisms built on top of it are grounded in this interpretation of the data.

5.3.2 Visual design

The above way of perceiving the aspects of the graph is reflected in a novel view, called *synchronization view*. It realizes the same decoupling of the levels to be able to represent and navigate them simultaneously. While it provides a way to access the graph and get an overview of its aspects, adjoined *data views* are needed to more naturally represent the structure of the graph, as well as its spatial and temporal context. Hence, besides common graph visualizations such as node-link views a map view and a calendar view are provided alongside the synchronization view because of their importance for the election use case. Each of these components of the visual design – the synchronization view and the data views – are introduced and discussed in the following.

5.3.2.1 Synchronization view

Typically, one does not know where in structure and space, when in time, and which attribute exhibits behavior of interest. Searching through all possible level combinations in structure, space, time, and attributes is tedious and time consuming. Therefore, the synchronization view shows all these levels simultaneously. To this end, it reflects the multipartite model by aligning all instances of each level in a horizontal *band*. To distinguish between different instances in a band, appropriate labeling and color-coding is

5.3 Steering the Analysis of Multiple Aspects with a Synchronized Approach

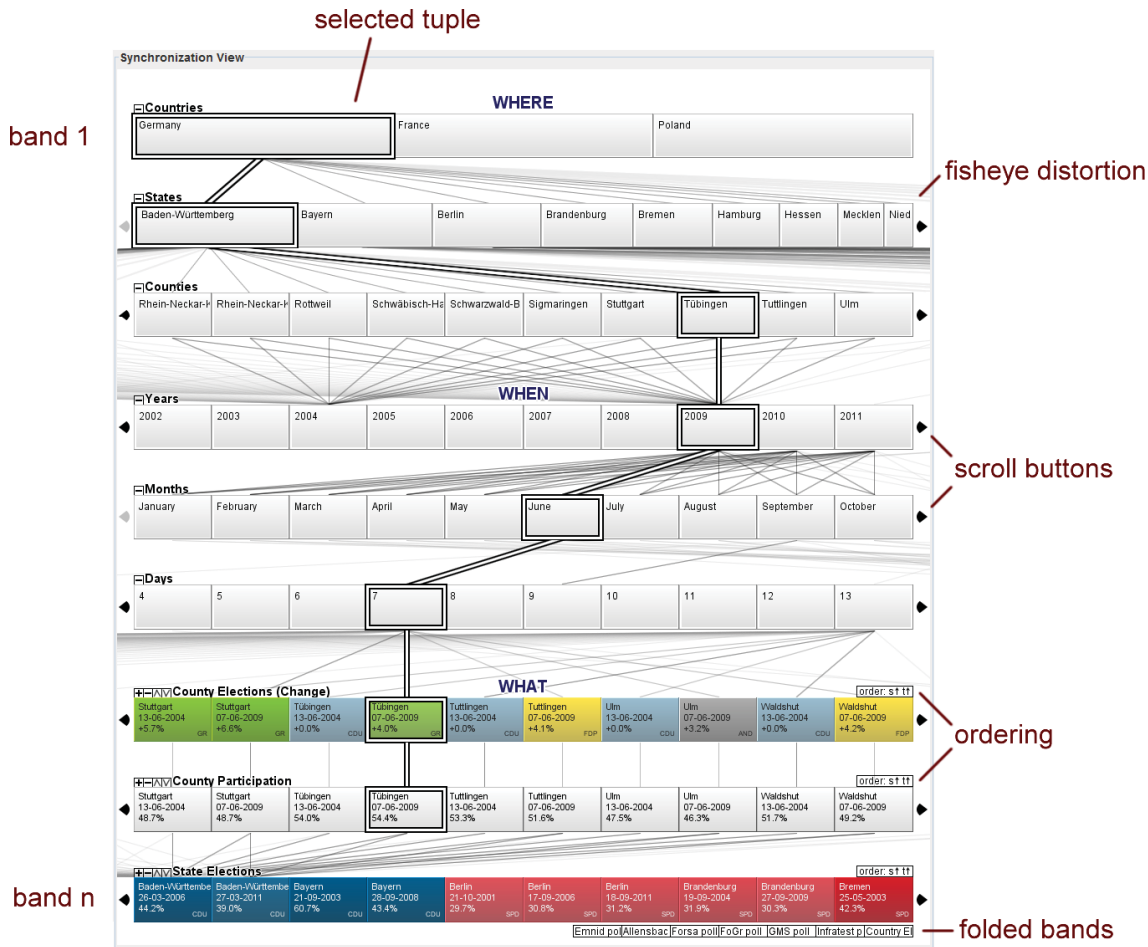


Figure 5.3.11: The synchronization view showing multiple levels of space (where), time (when), and different attributes (what). Each level is represented by a horizontal band of all instances of that level. The hue of the colors is used to encode categorical values, such as parties, and quantitative values, such as percentages, is reflected through their saturation. Data tuples are shown as edges connecting instances on all levels a tuple contains. Scroll buttons and fisheye-like distortion allow the user to navigate and explore each level.

applied. The bands are laid out in parallel and the tuples are shown as edges connecting the instances from all bands a tuple contains. This basic setup³ is shown in Figure 5.3.11.

It is this connection via tuples that realizes the actual synchronization of the aspects, as levels are shown simultaneously, even though they may never occur together in a tuple. This is made explicit through the linking: iff a link is connecting two instances, they co-occur. This way, attributes gathered on a yearly basis are only linked to the level of years, while monthly data is connected to both, months and years, as a month must be part of some year as well.

Constructing a view from the proposed separation of all aspects in multiple levels poses a scalability challenge, if the number of levels or the number of instances per level grow large. The former problem is solved by allowing the user to interactively adapt

³As each node in the depicted use case is related to exactly one location and vice versa, the structural aspect has been neglected from the visualization to simplify the illustration.

the number of shown bands by folding those of lesser interest and by duplicating others of higher interest. For example, in an investigation on a monthly granularity, bands of finer temporal scales (weeks, days, etc.) and structural, spatial and attribute-related bands that are associated only with those finer temporal scales can be folded. Yet, a band that relates to many other bands can be duplicated to see its direct relation to more than two neighboring bands similar to parallel coordinates. In case of a large number of instances, the user can scroll horizontally through the bands and apply a fisheye distortion, to maximize only instances that are currently explored and minimize all others. This is illustrated in Figure 5.3.11 for the spatial level of states.

With this view, a first approach is introduced for an overview of all aspects on all levels of granularity. It relieves the user from having to choose an appropriate visualization to investigate the data on, before actually seeing the possibly unknown data. And even if the user already knew the data and which aspects to look at, this approach avoids the need to switch back and forth between them in case of a comparative investigation involving data on two or more levels.

However, the separation of levels has dissolved the common spatial and temporal contexts in which a node would usually be embedded. This could be a node-link layout for the structural relations, a map or a globe for the spatial context, as well as a calendar or timetable for the temporal context. While the bands contain all the data, they do not show it in their “natural” frame of reference. Therefore, linked data views are added to complement the synchronization view.

5.3.2.2 Data views

The data views aim at presenting the data in the more conventional ways the user is accustomed to. They address the structure as well as the spatial and the temporal context independently by supplying different graph views (see Section 2.2), a map view [MK97] and a calendar view [vWvdW99] to capture them. As graph visualizations have already been extensively discussed, the following description focus mainly on the map and the calendar view showing the data on a per region or per time point basis rather than for each node and edge individually. These views provide the contexts in which the node and edge attributes are represented – usually by color-coding them for the regions of the map or the time slots in the calendar.

The implications of providing these views are apparent: they cannot show all the data, as the synchronization view can. Hence, the user has to choose a spatial granularity to show in the map view, a temporal granularity to show in the calendar view, and an attribute to encode in both views. Since these choices can be made from within the synchronization view upon inspecting all levels, they are now informed and no longer based on trial-and-error.

Both data views observe hierarchical dependencies between the levels. For example, spatial levels often exhibit a hierarchical dependency expressing the inclusion of administrative regions – i.e., countries, states, counties. Whereas levels that organize space in grid squares would be independent of these administrative levels, but form a hierarchical dependency among themselves. In case of such a hierarchical order among spatial and/or temporal levels, it depends very much on the semantics of the data if and how

they can be projected onto other levels. This applies to all three of such possible projections:

- **Aggregation** of low level data to higher levels can be done iff an appropriate aggregation function is given. This is important as aggregating values is highly domain-dependent with many different methods in existence [BPC07].
- **Duplication** of high level data at lower levels is possible iff the data given for the whole also applies to its parts. For example, this is the case for countrywide election results that hold true for all parts of the country and thus also for all states and counties.
- **Registration** of data between independent levels is only possible iff adequate mapping functions are given. For example, to map zip code level data to the county level and vice versa, one must know how much each zip code area contributes to each county, because at least in Germany, they do not strictly nest.

These projections have also to consider cases in which multiple nodes are assigned to the same region making it necessary to first aggregate them to gain information for each region. To prevent misinterpretation, the map or calendar in the view is by default colored gray to mark an invalid choice of levels if the attribute is not given for their particular combination. If one wants to project data from one level onto another, the data views are conceptually able to do this by color-coding the projected data instead of graying out the level. In this case, one has to make the user aware of the projection, so that the projected data is not mistaken as actually collected data. This can happen easily, for example, when looking at higher level data (e.g., country level election results) on lower levels (e.g., county level) one may find that all counties appear to have voted the same. Yet, in fact merely the overall voting result of the whole country has been duplicated and color-coded in each individual county.

The synchronization view and the data views are mutually linked, so that interactive selections and adaptations in one view are reflected in the others as well. This enables their concerted use for an exploratory analysis as it is further facilitated by the interaction concept presented in the following section.

5.3.3 Interactive exploration

The newly introduced synchronization view is built so that it effectively reflects the separation of all aspects into a multitude of interlinked levels. It provides per-level visualization by aligning the bands and per-tuple visualization by connecting them accordingly. As such, it extends well beyond the data views that can only provide traditional per-level representations. Having the versatility of the multipartite graph embedded in the visualization technique, this section discusses the different modes of exploratory analysis that can be pursued by using this view. As a result of this discussion, a tailored exploration mechanism is provided by utilizing a combination of two orthogonal modes of interaction: the novel *tuple-based exploration* together with the common *level-based exploration*.

5.3.3.1 Modes of exploratory analysis

In the spirit of Bertin's Levels of Information [Ber81], it is common especially in the context of spatio-temporal data to define the analysis interest with respect to its extent:

- **point-based extent**, e.g., for a particular node or point in time and space the associated value instance is sought;
- **local extent**, e.g., for contiguous subset of nodes or points in time and space the development of value instances (temporal decline, spatial spread, etc.) is of interest;
- **global extent**, e.g., for all nodes or points in time and in space the overall behavior (distribution, global extremes, etc.) of value instances is to be determined.

In case of multiple granularities, a “point in time and space” specifies value instances on a variety of levels, as a date specifies a year, a month, and a day and an address specifies a country, a state, a county, a city, a zip code, etc.

Standard methods, such as node-link representations, maps or bar charts, can cope with this multitude of data only partially: either by reducing the number of instances (usually by selecting a few instances of interest, such as a specific point in time) or by reducing the number of data items to show per level (usually by summarizing the value instances of entire levels with a few statistical measures) as discussed in Section 2.3. In either case the standard methods omit parts of the data (levels, value instances) to be able to show the rest. An access to all individual value instances on all levels (global extent without abstraction) is not possible from within these visualizations.

In contrast, the synchronization view can be used for such a global access without abstracting individual values. As such, it enables a user to follow two modes of exploration that cannot be pursued with standard techniques alone:

1. It provides a global view of all levels. Based on this full view, it permits an **exploration of levels** by making an informed selection of levels of interest to be shown in the adjoined traditional views (graph, map and calendar).
2. It presents a global view of all tuples. By this, it is possible to **explore the tuples** across all levels directly in the synchronization view.

The following two sections describe these two modes of exploration and the particular interaction needed for them.

5.3.3.2 Level-based exploration

The level-based exploration steers the selection of the individually shown levels in adjoined data views directly from within the synchronization view. In terms of interaction, it comprises of the conventional browsing of multilevel data level by level [EF10]. This can be thought of as cutting *horizontally* along specified levels through the set of tuples. In contrast to existing solutions, this is not a mere slider or mouse-wheel interaction performed on top of the data views, but a navigation of levels in the synchronization view that automatically updates the adjoined views through linking mechanisms. The difference is the informed choice of a level, as one does not have to go through all possible

level combinations one by one to see whether they exhibit behavior of interest. Instead, the synchronization view shows all levels simultaneously and the user can pick level combinations of interest directly based on what he discovers in this overview. This is not only a convenient addition to the data views, but mutually benefitting to both data and synchronization views. The reason is that the power of the synchronization view of showing all levels comes for the price of a very compact representation that aligns all levels as 1-dimensional horizontal bands. Yet, these bands may not fully capture more complex behavior, such as the spatial or temporal spread of a certain pattern, which may be identified in the bands, but can more easily be verified in the data views.

5.3.3.3 Tuple-based exploration

The tuple-based exploration presents a new way of exploring multilevel data: tuple by tuple. It can be thought of as the orthogonal counterpart to the level-based exploration, as it cuts *vertically* along a specified tuple through the set of levels. This exploration mode is motivated by the fact that multilevel patterns are not to be found on individual levels. Or as it has been observed and formulated in the mid-nineties by Ahl and Allen [AA96, p.76]: “What makes levels interesting is the relationship between them.” This relationship is encoded in the tuples that connect the levels in the multipartite data model. To find out all there is to know about this relationship, one must explore all tuples. Unfortunately, often the sheer number of tuples makes this a challenging undertaking, as there are too many to display them all at once without creating clutter and also to browse through all of them one by one. This is the downside of providing a detailed global view and two mechanisms are provided to solve this issue: *sorting* and *pinning*.

The basic assumption that underlies both is that the user is not just looking at the tuples in general, but has instead a partial idea of what interests him. This can be either a *relative interest* (e.g., all tuples exhibiting a high voter turnout in recent years) or an *absolute interest* (e.g., all tuples relating to a specific party in the years 2004 and 2005).

The **relative interest** can be translated into a partial **sorting** order for tuples. If the user looks for multiple criteria, these have to be prioritized to express in which the user is foremost interested – e.g., the user wants to sort first for high voter turnouts and only if values coincide, the more recent shall precede earlier tuples. Once sorted according to a user’s criteria, the browsing would start with tuples of highest interest to the user – in the example, this would be the tuple with the highest voter turnout, from the most recent year if there were multiple tuples with the same turnout. In this way, it is unlikely that the user will have to browse all tuples to explore the relations of the levels with respect to his particular interest. It is obvious that this method stands and falls with the availability of suitable sorting methods.

While sorting is trivial for numerical and ordinal levels, in particular the sorting of structural, spatial and nominal attribute levels is challenging. Usually, different application domains employ different orderings that are targeted towards their specific exploration tasks. This makes it hard to provide a general ordering strategy. So, to acknowledge that other application scenarios may require other ordering strategies, in the following a simple north-east to south-west ordering for the geospatial domain and an alphabetical ordering for nominal data values is assumed in the context of this concep-

tual discussion. In the synchronization view, any sorting order (e.g., temporal first, then by value) and its direction (ascending, descending) is shown in a small box at the top-right side of each band. Once, such an intra-level ordering has been determined, it is conceptually also possible to automatically determine an inter-level order that helps to show patterns among tuples more clearly [MTJ12, PWR04].

The **absolute interest** can be translated into fixing, in the following referred to as **pinning**, certain instances on some levels. In the above example, the specified party on the level “party” and the years 2004 and 2005 on the temporal level “year” get pinned. This results in a filtering of tuples to only those that contain these instances and thus cutting down on the large number of tuples in a data set. The browsing of tuples would then only encompass those that run through these particular instances. Furthermore, pinning works also on the attribute levels, so that attribute instances of interest can be pinned to narrow down the number of tuples to those of all nodes or edges at those times and places that instance occurred. If the number of tuples is still too large, pinning can of course be coupled with sorting to impose an additional order on the tuples.

As the views are linked, the instances of the currently explored tuples are highlighted in the all data views. On the other hand, a simple click on a node in a graph view, on a region in the map view or on a day, month, or year in the calendar view will trigger a pinning of the clicked instance on the appropriate structural, spatial or temporal level in the synchronization view.

The interplay of the views and their use together with the described interaction concept are brought to life in the following section, which presents a first realization of them.

5.3.4 Implementation & case studies

The realization of such a novel visualization concept, which deviates in many aspects from established representations, cannot be done without user feedback along the way. Hence, users were included throughout the process of developing this approach, as this is an established way in visualization design to prevent from building ad-hoc solutions [TM04, LD11]. When involving users and gathering their feedback, it is important to do so in the context of data they can relate to. Since the opinions of a diverse group of people was to be included and to prevent tailoring the prototype to a specific application domain, a number of different, yet related data sets were chosen that most people have some basic knowledge about – various election results and voter polls from Germany. In the following, the design of the implementation is described, a description of the used data is given, and some cross-level findings that were made in the data while test-driving the visualization are reproduced.

5.3.4.1 Implementation based on user feedback

The implementation relies on Java 6 with the standard Java2D functionality for the synchronization view and the calendar, as well as on the ApacheTM Batik SVG Toolkit for the rendering of the map view. It implements the multipartite data model and transforms input data to this model by partitioning it into levels and instances. The exploration strategies make heavy use of this data model for their fast interactive realization. For ex-

ample, it would be very cumbersome to implement a pinning on the original data items (e.g., on full dates or addresses).

From the early stages on, eight users from different departments including a domain expert from the political sciences gave iterative feedback on the realization. Four out of these eight users had some prior exposure to information visualization. In short sessions, they used the visualization in various configurations and setups. Afterwards, structured interviews were performed to evaluate their experiences with the prototype. To rule out learning effects, the real data was slightly distorted by a random process, so that the data was still reasonably realistic, yet no participant could thus claim prior knowledge of it. Besides the free exploration of the data, two recurring tasks had to be solved by each participant with each presented setup: a generic overview task and a specific localization task. The two main aspects that were investigated with their help were the view setup (i.e., whether additional data views are needed and whether they should be simultaneously visible or be shown one by one) and the tuple-based exploration (i.e., how the ordering affects the utility of the tuple-based exploration and which orderings to provide).

With respect to the first design question, one user summarized the feelings of most participants as he does not *“want to miss any of these views as they all have their benefits.”* This underlines that besides the synchronization view, the data views are necessary as a simultaneous display of the missing spatial and temporal contexts. Having then been given a simultaneous display of both types of views, many participants highlighted the importance of the synchronization view as a central element of the view setup, despite having to master the learning curve attached to such a novel view. One participant stated that *“the synchronization view provides a good way to coordinate the analysis.”* In this way, the back and forth between the views seems to be a powerful feature, as it allows for switching seamlessly between a more general overview and more detailed views of the data.

The benefit of the ordering of bands in the synchronization view was also highlighted, even if it only vaguely reflected a user’s partial interest. The reason is that because an ordering forms blocks of similar tuples, they can thus easily be skipped in bulk when browsing them. So even if the tuples, which are the most interesting to a user, are not sorted to the very top, the user can fast-forward through the ones of lesser interest and quickly reach them. The possibility to freely switch the ordering at any time was received very well by the participants: *“Being able to change the order in which the data is represented makes it very easy to find extraordinary features in the data.”* Our domain expert valued the idea of scrolling through the tuples in a particular order and thus seeing the relations between the different aspects of the data within a single view: *“Normally, we can only see a couple of data aspects, but never all of the data we have gathered. Their interplay is then often only captured within statistical evaluations of the data. Yet, having the possibility to really see this interplay together with the data within a single visualization is a very promising feature.”*

In addition to the user feedback, design insights from projects having similar aims were also collected, such as the SOLAP interface by Beaulieu and Bédard [BB08] or the early use of Polaris for exploring hierarchical data [STH02]. Altogether, this has helped to put the conceptual design on a broader basis.

5.3.4.2 Data description

The used data for the user-driven design, for all examples and screenshots in this section, as well as for the following case studies is based on various data sources. It contains German election data compiled from different online sources, such as the statistical offices of the states, as well as poll data from different institutes taken from the website <http://www.wahlrecht.de/umfragen> and reaching back as far as 1998. Overall, this data consists of 16 levels⁴ structured in three hierarchical space levels (countries, states, counties), three hierarchical time levels (years, months, days), and ten attribute levels covering different results and aspects of elections. This data is comprised of the percentage achieved by each party in elections and polls, and its delta of percentage points as compared to the last election/poll. Additionally, the voter participation numbers for elections where they were available have been included. Election results are available on country, state, and county level, while polls are only available on country level and for a few selected states at selected time points – e.g., a week before a state election.

Elections and polls are conducted on a specific date, which is encoded as a temporal reference in their tuples. Yet, their results remain in effect until the next election or poll yields a new outcome – i.e., a country is governed by the leading party/coalition until a new election is held, and journalists refer to the most recent poll until a new poll is published. Hence, the data from the actual day they were gathered was extended to the entire time interval until the next election/poll comes into effect. When a tuple is selected in the synchronization view, all other data instances are connected to it, if its date lies within the interval in which they are valid. For data that is truly given for specific dates, their extension into intervals and their connection would not be performed and the tuples are simply shown as they are.

5.3.4.3 Case studies

Through the exploration of this election data, two interesting patterns of voter behavior were revealed, which would be cumbersome to find without this approach. These patterns relate to two events in recent years: the Fukushima Daiichi nuclear disaster (temporal effects) and the Stuttgart21 railway project controversy (temporal and spatial effects).

Fukushima Daiichi nuclear disaster: The first example concerns the change of voter sentiment and is depicted in Figure 5.3.12. The shown synchronization view contains monthly or weekly poll data collected by different survey institutes. All poll related levels relate only to the country level, which is why all other spatial levels have been folded away. The data levels encode the party with the highest increase in voter popularity since the last poll. They are sorted in descending order of percentage points of each increase. If instances coincide, these are sorted in descending order of the time points they are associated with. From this sorting, a clear pattern emerges on the left side of the

⁴The structural aspect has been neglected on the one hand because of the redundancy resulting from a 1 to 1 relation between nodes and locations and on the other hand to not confuse the user with two different structures – the graph structure and the multipartite structure of the synchronization view.

5.3 Steering the Analysis of Multiple Aspects with a Synchronized Approach

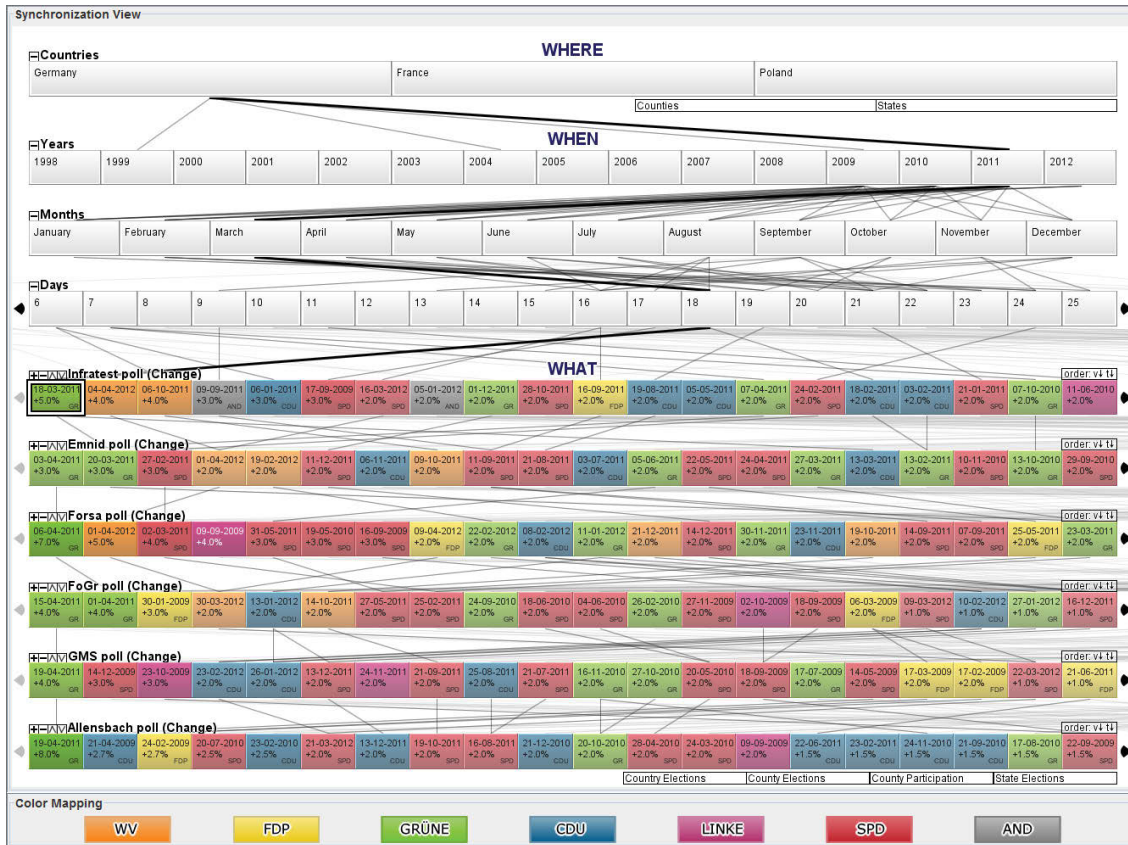


Figure 5.3.12: Investigating the largest increase in voter popularity for each poll in the synchronization view. All attribute levels (all polls) are sorted to show the instances of largest increase gained by a party on the left side. A clear pattern emerges there: across all polls, the Green party (green) has the largest increases among all instances. When looking at the dates of these polls, one can notice that they are from shortly after the Fukushima disaster.

synchronization view. It shows the biggest increase for the Green party throughout all the different polls, with a huge jump of up to 8 percentage points in the Allensbach poll (bottom band). Pinning individual value instances reveals that these polls were all taken shortly after the Fukushima disaster on March 11, 2011.

To confirm this finding, the data bands are reordered with respect to time only and the year, month, and day to the date of the Fukushima disaster are pinned. According to the described extension to intervals, the pinned date connects to all prior polls, as March 11, 2011 falls into the respective intervals for which these poll results are considered valid. The result of this adaptation can be seen in Figure 5.3.13. The calendar view shows the increase in voter popularity from the weekly poll results of the first band, the Infratest poll. One can observe in the synchronization view that the disaster took a few days to arrive in the political debate, as the Emnid poll (second poll band from the top) still shows the largest increase for the governing party (blue) for March 13, 2011, two days after the disaster. But as the temporal sorting of the data levels already hints at and as browsing through the tuples with the tuple-based exploration confirms: the popularity of the Green party increases steadily over the next weeks. This was most certainly due to an increased support for their stance against nuclear power. After four weeks, the impact

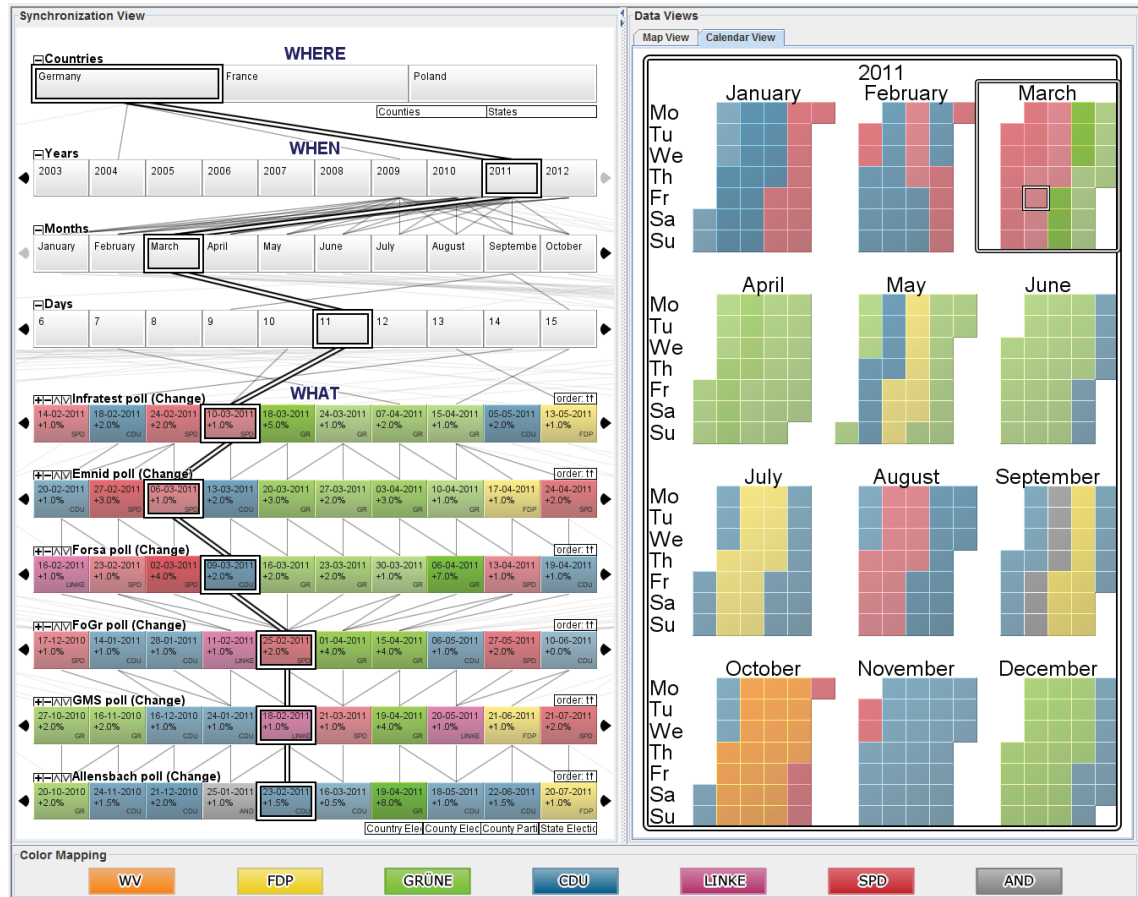


Figure 5.3.13: Effect of the March 11, 2011 Fukushima Daiichi nuclear disaster on German voter behavior. The synchronization view shows a strong increase in popularity for the oppositional Green party (green) in polls following the disaster. This trend was only broken after the ruling parties themselves announced to end nuclear energy production in Germany. The calendar view shows this period of time (4 weeks) for the weekly Infratest poll from the first band in the temporal context of 2011 at large.

of the event became weaker, which can be explained with the fact that the ruling parties in Germany followed suit and decided to abandon nuclear power, so that the Green party lost its edge over them.

Stuttgart21 railway project controversy: The second example is shown in Figure 5.3.14 and presents a finding regarding the controversial railway and urban development program Stuttgart21. Even though it concerned mainly the German state Baden Württemberg, it sparked debate and grassroots protests all across Germany. The county election in Stuttgart and the state election for all of Baden Württemberg held in 2009 and 2011, respectively, reflect the strong discontent of the local population with this project. This can be seen in the synchronization view, where the state and county are pinned, as well as the date of the 2009 county election in Stuttgart. The synchronization view shows a similar pattern to the first example: the winner of the county election held on that date was the Green party that opposed the project (first attribute band). This was not directly

5.3 Steering the Analysis of Multiple Aspects with a Synchronized Approach

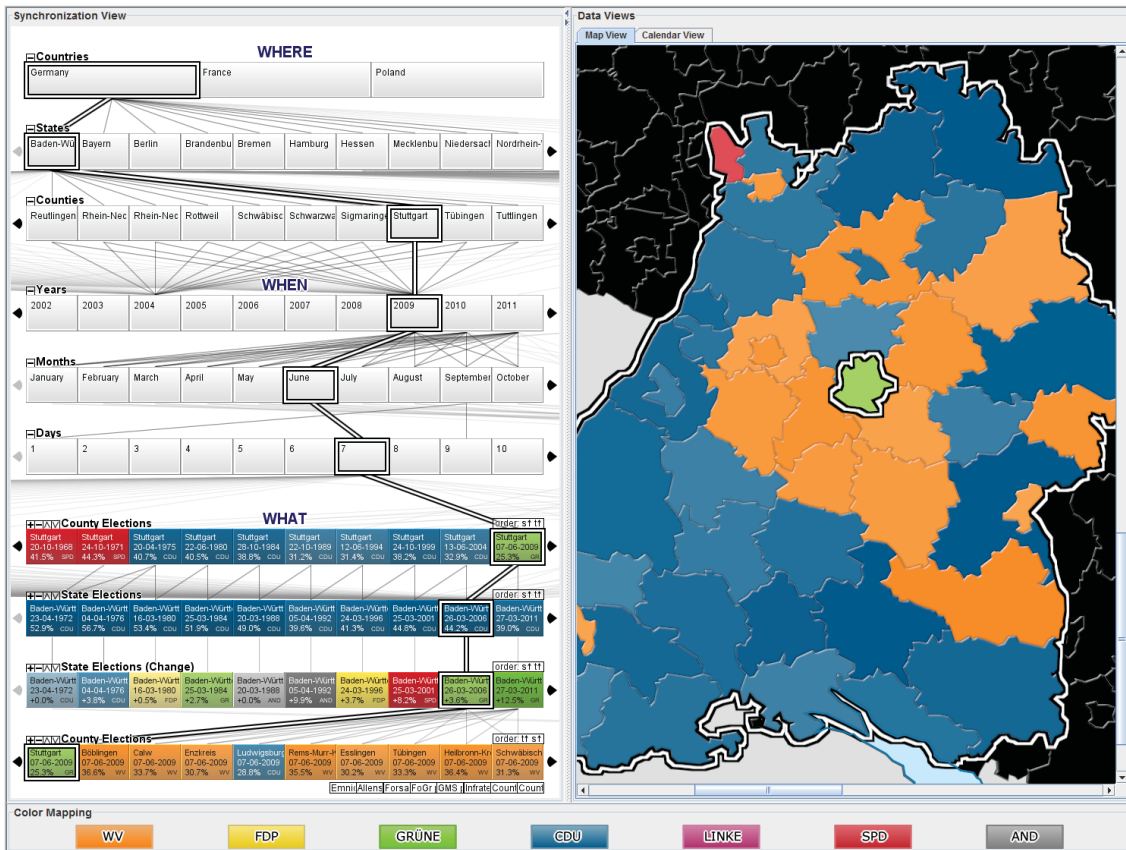


Figure 5.3.14: Effect of the controversial governmental development project Stuttgart21 on local voter behavior in the German state Baden Württemberg. The synchronization view shows the gains of the Green party (green) in response to the project's start of construction. The map view indicates a similar trend in Stuttgart's neighboring counties where oppositional citizens' action committees (orange) won the elections in the same period of time.

reflected in the state election two years later (second attribute band, the instance on the right side of the selected one) where the Christian Democratic Union (CDU, shown in blue) that traditionally governs the state received the most votes. Yet, its impact can be seen in the change of voter popularity (third attribute band, the instance on the right side of the selected one) where the Green party had the largest increase of 12.5 percentage points. Together with a loss on the side of the CDU, this actually led to the situation that the CDU could not form a majority coalition. Instead, the Green party formed a coalition and is now heading the state.

Another interesting observation can be made in the remaining band (fourth attribute band), which is a duplicate of the first attribute band, but differently sorted. The first band was sorted first by space (alphabetical by county names) to bring together all of Stuttgart's elections and second by time to have these elections line-up in ascending order from earliest (left) to latest (right). Yet, the fourth band is sorted first by time to bring together all county elections of the same date and second by space (ascending by each county's distance to Stuttgart) to have these elections line-up in ascending order from those closest to Stuttgart (left) to those farthest from Stuttgart (right). It can be seen from the ordering of the fourth band that on the day of county elections in 2009, numerous

counties close to Stuttgart have voted oppositional citizens' action committees (shown in orange) into office. While it can only be hinted at by the linear arrangement of the band, this observation can be verified in the map view. This effect is hardly coincidental, as it is very strong around Stuttgart and decreases with distance, so that it is very likely also a consequence of the Stuttgart21 project.

5.3.5 Conclusion

Discoveries and findings, which are based on a multitude of aspects, are very difficult to make using traditional approaches alone. Therefore, in this section a synchronization across structure, attributes and the spatial and temporal context was introduced which describes a first step towards a solution of this problem. This synchronization is based on a versatile multipartite graph-based data model for capturing and aligning information on multiple levels stemming from the different aspects. The visualization built on top of this data model passes the model's versatility on to the user to use it for interactively exploring data on multiple levels simultaneously with tailored exploration approaches. In this way, it provides the user an overview of the data and its aspects as well as a first way to identify interesting level combinations. On this basis, the user can now perform a more informed decision of which visualization to use and how to parametrize it. Consequently, the variety of existing visualization is not such a large burden anymore.

5.4 Summary

This chapter has introduced two novel visualization paradigms for utilizing the existing visualization techniques to their full potential and thus grant the user the flexibility to cope with the diversity and the size of dynamic graphs during their analysis. Therefore, first a new way of combining different visualization techniques was introduced that not only allows the simultaneous use of multiple visualizations showing more details or providing different foci. But also enables a local, in situ switch between representations at any time during the analysis while maintaining useful information such as connections from nodes in the base visualization to nodes in the embedded in situ visualizations. In this way, it maintains on the one hand the context of the analysis process and supports on the other hand the mental map of the user.

To help the user handle the overwhelming number of existing visualizations and their possible parametrization a novel visualization was proposed. This synchronization view provides an abstract overview of the data granting a summary of the aspects and their extents as well as to identify interesting combinations to look into by means of sorting and pinning. In this way, it not only allows an initial selection of appropriate data views but also their synchronization along the analysis process.

Altogether, both approaches describe first steps towards the flexibility a user needs to perform a full-fledged analysis. Yet, there are still some interesting problems to be addressed in future work as summarized in the next chapter that may further help the user.

6 Conclusion

6.1 Summary

This thesis addressed the visual analysis of graphs which is an important topic in many fields. Their analysis often entails a diversity of different aspects such as their structure and multiple associated attributes in their spatial and temporal context, and has to cope with thousands to millions of nodes, edges and time points. As a brief overview of the related work in Chapter 2 has shown, a wide variety of different visualization techniques has been proposed to deal with the different aspects and the size of these graphs. Yet, still some combinations of aspects – especially the embedding of the graph structure within its spatial context is an open research topic – as well as suitable approaches for handling the size of these graphs have not been sufficiently addressed yet. Furthermore, because the graph size neglects to show every information in a single image, each of these visualizations has to make a compromise on which level of detail each aspect is shown. In case of dynamic graphs, these compromises range from supergraph based visualizations that abstract the temporal aspect to complexity plots that abstract the structural aspect only showing the respectively other in full detail. In this way, each visualization provides a specific and fixed view on the graph data supporting a different analysis goal of the user. Hence, during a full-fledged analysis the utilization of multiple visualization techniques is necessary to adequately support a changing user focus.

Based on these observations, three major challenges for the visual analysis of graphs were identified and addressed in this thesis:

- The diversity concerning the different aspects of the graph,
- the scalability concerning the size of the graph,
- and the flexibility concerning the changing focus of the user.

6.1.1 Diversity

To deal with these challenges it is necessary to have suitable visualization techniques for each aspect as well as for different combinations of them that also scale well with regards to a large graph structure or temporal axis. At the same time, these techniques should not be too specific in their utilization as each further technique increases the costs for their implementation and necessitates more complex means for a flexible switch between them. To close some of these gaps specifically concerning the different aspects, two novel and more general approaches have been introduced for trees in Chapter 3 and discussed along the different aspects.

The first approach is based on a family of layouts all following a similar layout scheme. These layouts scale well for hundreds of thousands of nodes as they represent nodes by the smallest possible primitives – points – in combination with a fixed space-filling placement. The layouts of this family basically differ in their suitability for wider or deeper trees. And especially, the fixed placement of these layouts allows for an embedding of hierarchies into its spatial context. It also facilitates a comparison of different hierarchies and time points as the layout is very stable by design. Yet, the dense representation of the structure at same time limits the number of attributes and time points to be shown simultaneously.

For the second approach common design principles of implicit tree visualization were identified as independent design axes to derive a design space. This design space not only captures most of the existing implicit visualization but allows the creation of new ones by former unknown combinations of design decisions. That the design space can actually be employed for a rapid prototyping of new visualization techniques has been proven by an exemplary implementation. The underlying tuple-based visualization design in concert with a scripting capability and multiple data operators provides a very flexible prototyping tool. This flexibility does not only allow the user to define global design decision influencing the whole hierarchy but also to restrict decisions to specific and possibly more important nodes and thus create hybrid visualizations. For instance, existing techniques were adapted to better reflect multiple attributes or the temporal course of specific nodes while maintaining an overview of the remaining structure. Furthermore, this functionality along the different design axes provides multiple ways to fine tune a given visualization to specifically address and communicate an aspect of interest.

Because of their generality, it is possible with both approaches to adapt to different analysis goals concerning the aspects of a given hierarchy. Yet, it became very obvious that even with these more general approaches the size of the graph still affects the possibility to communicate every information at the same time.

6.1.2 Scalability

This problem affects especially dynamic graphs as despite their importance for the visual analysis there do not exist enough adequate approaches for their reduction. Hence, in Chapter 4 two different strategies – abstractions and selections – were investigated to devise scalable approaches for dynamic graphs.

To reduce the overall size of a dynamic graph new abstraction techniques were introduced that cluster either the structural or the temporal aspect of the graph with regard to the respectively other. For clustering the structural aspect, first the supergraph is calculated and for each node and edge a time series for a specific attribute is extracted. Those subgraphs of the supergraph that feature a similar time series are grouped together. To maintain the structural information the so found clusters are then split into its connected components. In this way, the clustering not only reduces the size of the graph but also helps identifying temporal relationships between subgraphs.

For clustering the temporal aspect, the time points are grouped instead into states that show a similar graph configuration. To depict the temporal course of the time points, states containing subsequent time points were connected by transitions resulting in an

abstract state transition graph. This state transition graph provides a compact overview of the dynamics of a graph and allows the identification of recurring and cycling patterns. Based on such patterns it is then for instance possible to determine indicators to forecast problems in the future.

To select only specific parts of a graph, a general and modular Degree-of-Interest function definition was introduced. In contrast to existing DoI-based approaches for graphs, this technique is not bound to a specific task as the underlying DoI function can be altered on the fly and takes all – the structure, attributes as well as the temporal aspect – into account. Therefore, common components for the definition of the DoI were extracted from the literature allowing the specification of basic interest functions and their subsequent combination into more complex functions. On this basis, existing DoI functions can be rebuild but also new functions can be assembled and extended to cope with changing analysis goals.

An application of these techniques to real world use cases has shown that they can support the user in the visual analysis of larger graphs. Yet, especially for abstraction techniques, the choice of the aspect to be reduced strongly influences the interrelations to be found in the resulting visualization.

6.1.3 Flexibility

To actually support the user in this choice between the different aspects, in Chapter 5 two quiet different approaches were introduced. The first approach – the In Situ visualization – actually aims at providing this choice not only globally by allowing the user to switch between different visualizations as done by most multiple coordinated views setups but also locally by allowing him to embed visualizations directly in each other where necessary. Therefore, its embedding is based upon common techniques such as portals and enhanced to maintain the structural and temporal continuity that are important for the user’s mental map. These enhancements contain the portal awareness, context awareness, and overlays that allow the base visualization and the embedded visualizations to adapt themselves to the respectively other as well as to maintain important connections which would be otherwise hidden under the embedded visualizations.

To allow the user a more informed choice of an appropriate visualization to start with or to embed and to support him orientating himself in his data, the goal of the second approach is to provide an overview of the graph with all its aspects. It therefore utilizes a multipartite graph model in which each partition contains a specific aspect and the edges capture the relations between the values of multiple partitions. Here, for each aspect multiple partitions may be used to distinguish for instance nodes and edges or different attributes. Using this multipartite graph model, an overview was introduced visualizing each partition as a scrollable band and thus is able to communicate the graph data in its entirety, yet in a very abstract way. It gives the user a first impression of possible interesting aspects and serves at the same time as a synchronization mechanism for additional visualizations providing a more natural and detailed view on the data.

6.2 Discussion and Future Work

This thesis has identified three major challenges for a thorough visual analysis of graphs and introduced several concepts for solving them. While these concepts cover first steps towards a complete solution there are still open research questions to address in the future. These questions basically concern current technical limitations of the proposed concepts, their application to other use cases as well as their generalization or specialization for instance to other graph classes.

6.2.1 Technical Limitations

In the following the limitations are discussed according to the three challenges – diversity, scalability and flexibility:

Diversity related limitations

Many of the techniques to be found in the literature as well as in this thesis are utilizing specific characteristics of the data aspects. For instance, nearly all tree layouts – including the point-based layout and the implicit tree visualization design space discussed in Chapter 3 – exploit the hierarchical organization of the graph structure to calculate the final layout. This also holds true for some more general graph layouts that first calculate a spanning tree to devise a graph layout. Yet at the same time, this utilization poses specific requirements to the input graph to be able to visualize it using these techniques.

This problem also affects the other aspects and especially the spatial and temporal context. In this thesis the assignment between nodes and their spatial references is considered rather fixed and does not change much over time. While such a fixed assignment is no necessity for the embedding of the point-based layout into the spatial context (see Section 3.2.4) and node movements are also taken into account during the structural coarsening (Section 4.2.1) of dynamic graphs, a full-fledged analysis of such spatio-temporal phenomena often requires further visual cues such as a direct representation of the movement trajectories. An example for a visualization of trajectories was introduced in [TSA12] which stacks them in 3D on top of each other. Their visualization is an important topic not only for graphs but for spatio-temporal data in general.

Similar, most visualizations for dynamic graphs are restricted to a linear time axis and do not cope with more complex time axes such as a branching axis which is for instance necessary in the versioning of bio-chemical reaction models. Here, the state transition graph based visualization introduced in Section 4.2.3 represents an exception. While this visualization approach is based on first transforming a linear time axis into a state transition graph using clustering techniques, this processing step may be omitted in case a more complex time axis is already given. Furthermore, considering stochastic simulations of these models, another interesting kind of time axes is based on multiple runs having not a single linear time axis but multiple axes at the same time. Here, the state transition graph may be used again for summarizing the time points by applying the clustering to all time axes at the same time. How such an application may look like is exemplified in Figure 6.2.1 showing the state transition graph generated for two runs.

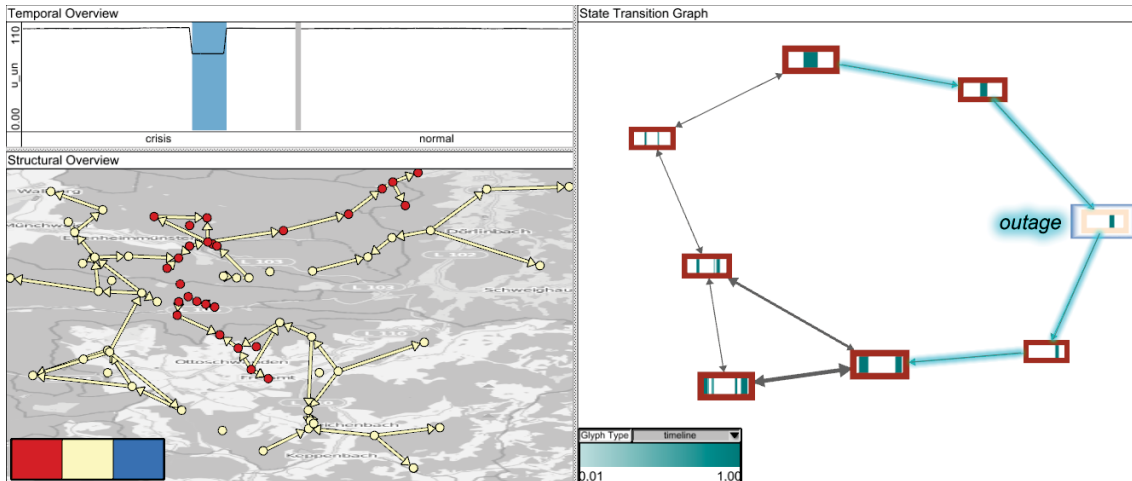


Figure 6.2.1: State transition graph based visualization of multiple simulation runs of a power grid. The state transition graph has been calculated for two different simulation runs (see the temporal overview showing a crisis and a normal run side by side). Except for an outage (covered in the highlighted state) in the crisis run both runs share a similar temporal course reflected by the same states. The structural overview reflects which nodes were causing the crisis in their spatial context (map © OpenStreetMap contributors).

Hence, following such ideas focusing more on the temporal aspect is also an important endeavor to follow in the future.

Scalability related limitations

To deal with a limited screen real estate, for most of the presented visualizations appropriate reduction techniques exist or have been introduced such as the clustering or DoI-based filtering. Yet, because of the novelty of the synchronization view as it was introduced in Section 5.3 the available screen space is still a limiting factor for its usability. Here, the interaction cost for investigating the data in its entirety is increasing with the number of partitions (aspects and especially attributes) and the contained data instances. Hence, future enhancements may aim at compacting the synchronization view to gain a truly powerful first overview of the data. This can principally be done in two ways: by reducing the number of partitions or by reducing the number of instances/tuples. Yet at the same time, all of the information necessary for the current exploration should be preserved. For instance, levels can be automatically folded away if they do not co-occur in any tuple within the currently selected levels. Likewise, the number of shown tuples can be reduced by bundling all tuples into ribbons that have no value instance in common with the currently selected tuples. The result would look similar to a Sankey Diagram [RHF05] or a Parallel Sets visualization [KBH06], but with the tuples currently under scrutiny and all related tuples being shown as individual lines.

Besides the visual scalability, the complexity of the reduction techniques is also of importance for the interactive analysis. Especially, the calculation of the similarity matrix necessary for the hierarchical clustering that is used at the moment can become very time consuming. While most of the calculations may be done in a preprocess, changing the clustering settings often requires a new calculation. Hence, the integration of more robust

and sophisticated clustering methods may be beneficial for the analysis. In this regard, the adaptation of caching and querying strategies as introduced in [AW12] for an interactive clustering of time series according to different time intervals may also be beneficial for an interactive analysis of larger dynamic graphs.

In particular for larger data with many different levels of granularity in structure and time, one might wish for more guidance towards granularity levels or patterns of interest. Without such guidance, it can become quite tedious to explore graphs in their entirety. While for numerical time series data, first guidance approaches towards granularities of interest have already been proposed [LSM⁺12], similar methods are so far unexplored for dynamic graph data but may be expressed using the modular DoI function definition. Their combination with graph clustering approaches seems to be a promising goal.

Flexibility related limitations

It has to be noted in this context that such a combination stands and falls with the ability to specify meaningful DoI functions that exactly reflect a given user interest. Hence, more flexible ways are needed for the specification of features that are intricately placed in a high dimensional, multi-faceted attribute space and thus more adequately addressing the diversity of the graph. While it may be possible to capture them with a combination of multiple DoI components as provided at the moment, this may require a large number of base DoI functions and can be very tedious to put together. Hence, it would be desirable to provide components in which such complex patterns can be specified directly, for example, by selecting a 2D region in a scatterplot of two different attributes ($inter(comp_1(\dots), comp_2(\dots))$) or of one attribute vs. the time axis ($inter(comp(\dots), T)$). Another way to ease the specification of complex patterns in a high-D feature space would be to also allow for defining DoI values based on principal components, as proposed in [SKK06]. In this way, complex patterns are predefined by expert users relieving the normal user of specifying them.

Considering the flexible switch between visualizations, another useful extension would be a method for predicting if a prospective visualization will be visually cluttered and/or suits a given visualization task. At the moment, the selection of a suitable visualization relies on the knowledge of the user and much trial and error. With such a method given, the user can be provided with additional information and thus be supported in his flexible choice of a suitable visualization technique. For the point-based layout a first numeric evaluation based on the number of filled pixels was already discussed in Section 3.2.2.1 for the selection of an appropriate preset. Yet, a generalization for other techniques is still an open research question. Just recently a design space for visualization tasks was introduced in [SNHS13] that allows a numeric evaluation of a visualization's suitability for a given task. A combination of such a general design space with the identified graph tasks (see [APS14, KKC14b, LPP⁺06]) also represents a useful direction for future work.

6.2.2 Application

While all the presented approaches have been demonstrated by the visual analysis of a specific use case, their applicability has still to be inspected for a more diverse set of use cases featuring other characteristics concerning their structure, associated attributes as

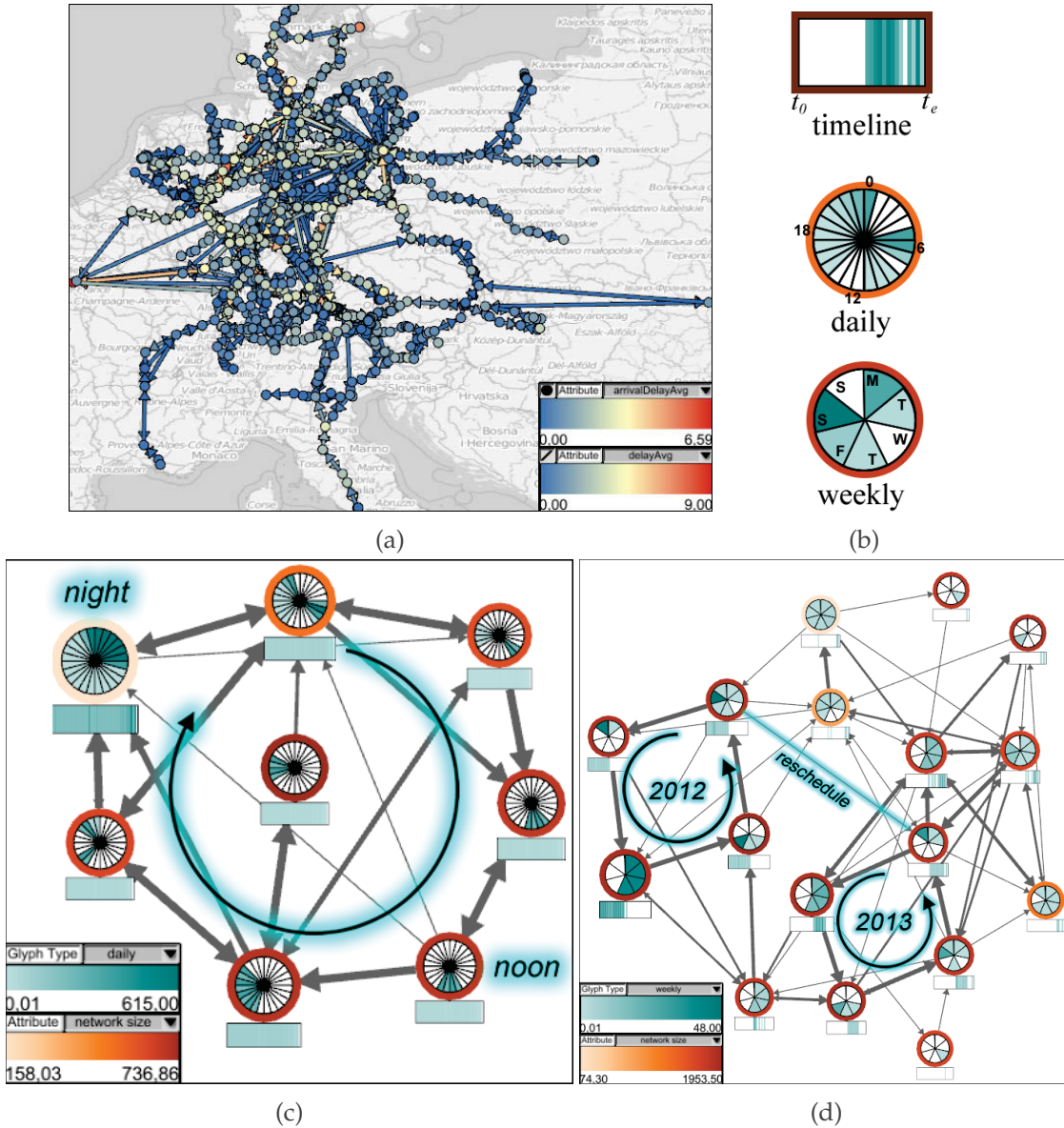


Figure 6.2.2: State transition graph visualization of temporal patterns for a high-speed train connection network in Germany. (a) the structural overview shows the train connection network with nodes representing the train stations and edges the train connections in their spatial context (map © OpenStreetMap contributors). The average delay is color-coded on both nodes (in this case the arrival delay) and edges. (b) different temporal glyphs used in state transition graph view. Timeline glyph provides a linear view of the distribution of time points. Radial glyphs provide an overview of recurring temporal patterns relative to a given temporal cycle, such as daily (24h hour periods) or weekly (7 day periods). (c) the state transition graph generated at an hourly temporal scale shows a cyclic temporal pattern capturing the daily cycle with states representing specific intervals of the day reflected by daily glyphs. (d) the state transition graph generated at a daily temporal scale captures totally different temporal patterns. Here, the radial glyphs visualize specific days of the week. Two large cycles are visible each capturing the main weekly pattern of the train services for 2012 and 2013. The reschedule event between these cycles is also visible as a single transitions.

well as their spatial and temporal context. Their application to other use cases can reveal other interesting temporal patterns as depicted for example in Figure 6.2.2. Here, two state transition graphs were generated for a high-speed train connection network at two different temporal levels of granularity. While a switch between different levels has also been performed in Section 4.2.4.5, the differences between the resulting state transition graphs were only slightly. Here, each state transition graph captures totally different cyclic recurring temporal patterns.

Furthermore, in a co-authorship or citation network for instance, evolving topics can for instance be discovered by a structural clustering of the nodes and edges with regard to their coexistence over time. This is feasible by extracting an appropriate time series for the existence of each node and edge. In this way, it does not only allow the clustering according to attributes over time but to their coexistence. Such a clustering would then separate groups of nodes and edges that were already connected over a longer period of time and groups that share connections only very recently. The latter case corresponds to newer or evolving topics.

Yet, even if a use case meets all the requirements of the technique to be used, it does not necessarily mean that the technique will grant useful insights into the data. Social networks often consist of different communities each having its own history and development. Here, it is unlikely that the whole dynamic network exhibits any recurring states. Hence, generating a state transition graph considering the whole structure will most likely result in a linear ordering of states limiting the utility of this approach. But when applied to individual subgraphs or communities the state transition graph may reveal interesting patterns in the history of these communities. For example, the visualization community tends to have stronger connections concerning the communication between researchers during special events such as the VIS conference that occurs every year. In this case, it might be more useful to extract state transition graphs only for specific communities that can be identified using structural clustering methods as a first step. In this way, it may be interesting to investigate the application of multiple reduction steps concerning different settings as well as different aspects.

Another example in which multiple reduction steps may be of importance concerns the analysis of biological reaction networks. Here, the amount of reactants is changed by their participation in different reactions. Even if two reactants are involved in the same reaction their amount may change rather differently, in one reaction their amount may rise simultaneously whereas in another reaction one is rising and the other one is dropping. In this way, it is difficult to formulate a similarity measure that may be applied for a structural clustering of these reactants limiting its use. However, by first generating a state transition graph for each reactant, two reactants can then be considered similar if they exhibit the same states and transitions which is the case for reactants that are involved and thus changed in the same reactions.

6.2.3 Generalization and Specialization

Besides the evaluation of the proposed visual analysis approaches with different use cases, further future research questions may concern their generalization to other problems. For instance, both the point-based layout as well as the implicit tree visualization

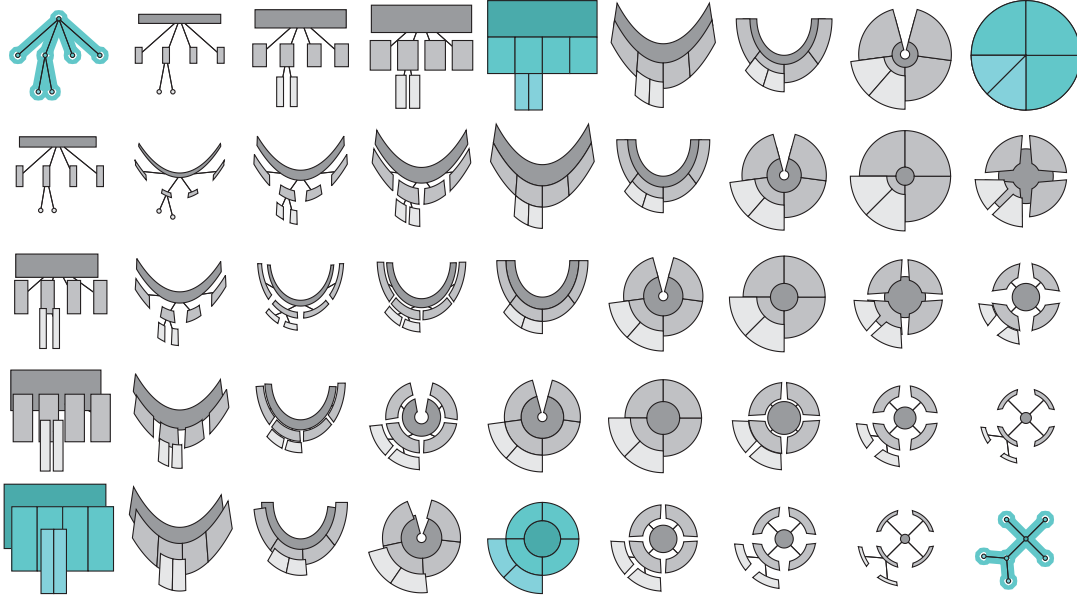


Figure 6.2.3: Smooth blending of 6 tree visualizations. The cyan visualizations represent the following presets from the upper-left to the lower-right: “X-Mas tree” layout, Icicle Plot, Pietree, cascaded Treemap, Sunburst, radial layout. Based on these presets blended visualizations (gray) are calculated by interpolating between different visualization parameters such as node size, displacement or curvature.

design space are both defined for trees. Moreover, the proposed design space covers only a subspace of all existing tree visualizations based on discrete choices of the identified design dimensions. Here, a first generalization may target to also include explicit tree visualizations into the design space before trying to transfer it to general graphs. As a matter of fact, some first steps have already been undertaken in this direction. Just recently, a novel approach [SAM13] was introduced that is able to capture both explicit and implicit tree visualizations, yet in a descriptive way by subdividing the tree layout pipeline. Further ongoing work (also of this thesis’ author) investigates a blending of tree visualizations as depicted in Figure 6.2.3 that is based on a smooth interpolation along multiple visualization parameters including a smooth switch from implicit to explicit visualizations.

Even if a visualization is already designed for general graphs and thus should work with any input data set, their utilization for other more specialized graph classes still holds interesting challenges for future extensions. Each specific class has its own semantics holding additional information not necessarily accounted for when dealing with general graphs such as different node sets in multipartite graphs or different kinds of relations between nodes in clustered graphs. Hence, when transferring a general approach to a specific graph class this can involve further adaptation steps.

For example, the modular DoI definition was introduced for the selection in rather general dynamic graphs only concerning nodes, edges and time points. Yet, when analyzing larger graphs an often observed preprocessing step concerns the clustering of these graphs. Hence, most often instead of a general graph a clustered graph is given holding much more information that can be used for specifying a user’s interest. In this sense, the user may also want to specify his interest on clusters or time intervals rather

than singular nodes, edges or time points. Thus, for navigating within such a clustered graph, the DoI may be aggregated to clusters as well as distributed to the contained graph elements [GST13]. Because of its modularity, the proposed DoI definition can be easily adapted to clustered graphs by implementing specific components dealing with clusters. For instance, interesting clusters may be specified according to the number of graph elements they contain using a specifically designed computation function. The aggregation of their DoI values reflects a special kind of structural propagation only regarding the parent-child relation of the clustering hierarchy instead of incident edges and thus can be captured by an appropriate distance function.

All in all, both the generalization and the specialization represent interesting research directions for future work holding much potential to further support a thorough visual analysis of graphs.

Bibliography

- [AA96] Valerie Ahl and Timothy F.H. Allen. *Hierarchy Theory: A Vision, Vocabulary, and Epistemology*. Columbia University Press, 1996. Page 76. url: <http://cup.columbia.edu/book/978-0-231-08480-2/>.
- [AAB⁺10] Gennady Andrienko, Natalia Andrienko, Sebastian Bremm, Tatiana Schreck, Tobias and Von Landesberger, Peter Bak, and Daniel Keim. Space-in-time and time-in-space self-organizing maps for exploring spatiotemporal patterns. *Computer Graphics Forum*, 29(3):913–922, 2010. doi:10.1111/j.1467-8659.2009.01664.x.
- [ADD⁺04] David Auber, Maylis Delest, Jean-Philippe Domenger, Philippe Duchon, and Jean-Marc Fédou. New Strahler numbers for rooted plane trees. In *Proceedings of the Colloquium on Mathematics and Computer Science Algorithms*, pages 203–215, 2004. doi:10.1007/978-3-0348-7915-6_21.
- [AERD10] James Abello, Tina Eliassi-Rad, and Nishchal Devanur. Detecting novel discrepancies in communication networks. In *Proceedings of the IEEE International Conference on Data Mining*, pages 8–17, 2010. doi:10.1109/ICDM.2010.145.
- [AMA07] Daniel Archambault, Tamara Munzner, and David Auber. TopoLayout: Multi-level graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):305–317, 2007. doi:10.1109/TVCG.2007.46.
- [AMA09] Daniel Archambault, Tamara Munzner, and David Auber. TugGraph: Path-preserving hierarchies for browsing proximity and paths in graphs. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 113–120, 2009. doi:10.1109/PACIFICVIS.2009.4906845.
- [AMM⁺07] Wolfgang Aigner, Silvia Miksch, Wolfgang Müller, Heidrun Schumann, and Christian Tominski. Visualizing time-oriented data - A systematic view. *Computers & Graphics*, 31(3):401–409, 2007. doi:10.1016/j.cag.2007.01.030.
- [APP11] Daniel Archambault, Helen Purchase, and Bruno Pinaud. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):539–552, 2011. doi:10.1109/TVCG.2010.78.
- [APS14] Jae-wook Ahn, Catherine Plaisant, and Ben Shneiderman. A task taxonomy for network evolution analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):365–376, 2014. doi:10.1109/TVCG.2013.238.
- [Arc09] Daniel Archambault. Structural differences between two graphs through hierarchies. In *Proceedings of Graphics Interface*, pages 87–94, 2009. url: <http://dl.acm.org/citation.cfm?id=1555880.1555905>.
- [AS87] Alok Aggarwal and Subhash Suri. Fast algorithms for computing the largest empty rectangle. In *Proceedings of the Symposium on Computational Geometry*, pages 278–290, 1987. doi:10.1145/41958.41988.

- [ASST07] James Abello, Hans-Jörg Schulz, Heidrun Schumann, and Christian Tominski. Interactive Poster: CGV – Coordinated Graph Visualization. In *Poster Compendium of the IEEE Conference on Information Visualization*, 2007. url: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.496&rep=rep1&type=pdf>.
- [ATMS⁺11] Jae-wook Ahn, Meirav Taieb-Maimon, Awalin Sopan, Catherine Plaisant, and Ben Shneiderman. Temporal visualization of social network dynamics: Prototypes for nation of neighbors. In *Proceedings of the International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 309–316, 2011. doi:10.1007/978-3-642-19656-0_43.
- [AvH04] James Abello and Frank van Ham. Matrix zoom: A visual interface to semi-external graphs. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 183–190, 2004. doi:10.1109/INFVIS.2004.46.
- [AW12] Zafar Ahmed and Chris Weaver. An adaptive parameter space-filling algorithm for highly interactive cluster exploration. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 13–22, 2012. doi:10.1109/VAST.2012.6400493.
- [AWP97] Keith Andrews, Josef Wolte, and Michael Pichler. Information PyramidsTM: A new approach to visualising large hierarchies. In *Proceedings of the IEEE Conference on Visualization*, pages 49–52, 1997. url: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.3929&rep=rep1&type=pdf>.
- [BB05] Danail Bonchev and Gregory A. Buck. Quantitative measures of network complexity. In Danail Bonchev and Dennis H. Rouvray, editors, *Complexity in Chemistry, Biology, and Ecology*, chapter 5, pages 191–235. Springer, 2005. doi:10.1007/0-387-25871-X_5.
- [BB08] Veronique Beaulieu and Yvan Bedard. Interactive exploration of multi-granularity spatial and temporal datacubes: Providing computer-assisted geovisualization support. In *Proceedings of the GIScience Workshop on GeoSpatial Visual Analytics*, 2008. url: <http://www.geoanalytics.net/GeoVisualAnalytics08/a17.pdf>.
- [BBD08] Michael Burch, Fabian Beck, and Stephan Diehl. Timeline Trees: Visualizing sequences of transactions in information hierarchies. In *Proceedings of Advanced Visual Interfaces*, pages 75–82, 2008. doi:10.1145/1385569.1385584.
- [BBDW14] Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. The state of the art in visualizing dynamic graphs. In *Proceedings of the Eurographics Conference on Visualization—State-of-the-Art Reports*, 2014. doi:10.2312/eurovisstar.20141174.
- [BBG⁺09] Jorik Blaas, Charl Botha, Edward Grundy, Mark Jones, Robert Laramée, and Frits Post. Smooth graphs for visual exploration of higher-order state transitions. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):969–976, 2009. doi:10.1109/TVCG.2009.181.
- [BC01] Ralf Brockenauer and Sabine Cornelson. Drawing clusters and hierarchies. In Michael Kaufmann and Dorothea Wagner, editors, *Drawing Graphs: Methods and Models*, chapter 8, pages 193–227. Springer, 2001. doi:10.1007/3-540-44969-8_8.
- [BC03] Ulrik Brandes and Steven R. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. *Information Visualization*, 2(1):40–50, 2003. doi:10.1057/palgrave.ivs.9500037.

- [BCD⁺10] Anastasia Bezerianos, Fanny Chevalier, Pierre Dragicevic, Niklas Elmqvist, and Jean-Daniel Fekete. GraphDice: A system for exploring multivariate social networks. In *Proceedings of the Eurographics / IEEE - VGTC Conference on Visualization*, pages 863–872, 2010. doi:10.1111/j.1467-8659.2009.01687.x.
- [BCS04] Thomas Bladh, David A. Carr, and Jeremiah Scholl. Extending tree-maps to three dimensions: A comparative study. In *Proceedings of the Asia-Pacific Conference on Computer-Human Interaction*, pages 50–59, 2004. doi:10.1007/978-3-540-27795-8_6.
- [BD08] Michael Burch and Stephan Diehl. TimeRadarTrees: Visualizing dynamic compound digraphs. *Computer Graphics Forum*, 27(3):823–830, 2008. doi:10.1111/j.1467-8659.2008.01213.x.
- [BD10] Maria Biryukov and Cailing Dong. Analysis of computer science communities based on DBLP. In *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries*, pages 228–235, 2010. doi:10.1007/978-3-642-15464-5_24.
- [BDKW07] Horst Bunke, Peter J. Dickinson, Miro Kraetzl, and Walter D. Wallis. *A Graph-Theoretic Approach to Enterprise Network Dynamics*, volume 24 of *Progress in Computer Science and Applied Logic*. Birkhäuser, 2007. url: <http://www.worldcat.org/oclc/187161827>.
- [BDL⁺10] Vladimir Batagelj, Walter Didimo, Giuseppe Liotta, Pietro Palladino, and Maurizio Patrignani. Visual analysis of large graphs using (X,Y)-clustering and hybrid visualizations. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 209–216, 2010. doi:10.1109/PACIFICVIS.2010.5429591.
- [Bed01] Benjamin B. Bederson. PhotoMesa: A zoomable image browser using quantum treemaps and bubblemaps. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 71–80, 2001. doi:10.1145/502348.502359.
- [Ber81] Jacques Bertin. *Graphics and Graphic Information Processing*. Walter de Gruyter, 1981. Page 12. doi:10.1515/9783110854688.
- [Ber83] Jacques Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1983. url: <http://www.worldcat.org/oclc/9684543>.
- [BETT99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999. url: <http://www.worldcat.org/oclc/39181634>.
- [BG03] Dominique Brodbeck and Luc Girardin. Trend analysis in large timeseries of high-throughput screening data using a distortion-oriented lens with semantic zooming. In *Poster Compendium of the IEEE Symposium on Information Visualization*, pages 74–75, 2003. url: <http://www.macrofocus.com/publications/infovis2003-poster.pdf>.
- [BHJ09] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, 2009. url: <http://www.aaai.org/ocs/index.php/ICWSM/09/paper/viewFile/154Forum/1009>.
- [BHvW00] Mark Bruls, Kees Huizing, and Jarke van Wijk. Squarified treemaps. In *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, pages 33–42, 2000. doi:10.1007/978-3-7091-6783-0_4.

- [BIM12] Ulrik Brandes, Natalie Indlekofer, and Martin Mader. Visualization methods for longitudinal social networks and stochastic actor-oriented modeling. *Social Networks*, 34(3):291–308, 2012. doi:10.1016/j.socnet.2011.06.002.
- [BLS99] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes: A survey*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 1999. url: <http://www.worldcat.org/oclc/40654789>.
- [BPC07] Gleb Beliakov, Ana Pradera, and Tomasa Calvo. *Aggregation Functions: A Guide for Practitioners*. Number 221 in Studies in Fuzziness and Soft Computing. Springer, 2007. doi:10.1007/978-3-540-73721-6.
- [BPF14a] Benjamin Bach, Emmanuel Pietriga, and Jean-Daniel Fekete. GraphDiaries: Animated transitions and temporal navigation for dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):740–754, 2014. doi:10.1109/TVCG.2013.254.
- [BPF14b] Benjamin Bach, Emmanuel Pietriga, and Jean-Daniel Fekete. Visualizing dynamic networks with matrix cubes. In *Proceedings of the Annual Conference on Human Factors in Computing Systems*, pages 877–886, 2014. doi:10.1145/2556288.2557010.
- [Bra94] Andreas Brandstädt. *Graphen und Algorithmen*. Leitfäden und Monographien der Informatik. Teubner, 1994. url: <http://www.worldcat.org/oclc/48576503>.
- [Bra01a] Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001. doi:10.1080/0022250X.2001.9990249.
- [Bra01b] Jürgen Branke. Dynamic graph drawing. In Michael Kaufmann and Dorothea Wagner, editors, *Drawing Graphs: Methods and Models*, chapter 9, pages 228–246. Springer, 2001. doi:10.1007/3-540-44969-8_9.
- [BSM04] Ragnar Bade, Stefan Schlechtweg, and Silvia Miksch. Connecting time-oriented data and information to a coherent interactive visualization. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 105–112, 2004. doi:10.1145/985692.985706.
- [BSP⁺93] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: The see-through interface. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, pages 73–80, 1993. doi:10.1145/166117.166126.
- [BSW02] Benjamin B. Bederson, Ben Shneiderman, and Martin Wattenberg. Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. *ACM Transactions on Graphics*, 21(4):833–854, 2002. doi:10.1145/571647.571649.
- [BVB⁺11] Michael Burch, Corinna Vehlow, Fabian Beck, Stephan Diehl, and Daniel Weiskopf. Parallel edge splatting for scalable dynamic graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2344–2353, 2011. doi:10.1109/TVCG.2011.226.
- [BW08] Lee Byron and Martin Wattenberg. Stacked graphs – geometry & aesthetics. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1245–1252, 2008. doi:10.1109/TVCG.2008.166.
- [BY08] C.C. Bilgin and B. Yener. Dynamic network evolution: Models, clustering, anomaly detection. Technical report, Rensselaer Polytechnic Institute, NY, 2008. url: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.161.6375&rep=rep1&type=pdf>.

- [CAT07] Fanny Chevalier, David Auber, and Alexandru Telea. Structural analysis and visualization of C++ code evolution using syntax trees. In *Proceedings of the International Workshop on Principles of Software Evolution*, pages 90–97, 2007. doi:[10.1145/1294948.1294971](https://doi.org/10.1145/1294948.1294971).
- [CGS⁺11] Nan Cao, D. Gotz, Jimeng Sun, Yu-Ru Lin, and Huamin Qu. SolarMap: Multifaceted visual analytics for topic exploration. In *Proceedings of the IEEE International Conference on Data Mining*, pages 101–110, 2011. doi:[10.1109/ICDM.2011.135](https://doi.org/10.1109/ICDM.2011.135).
- [CJA⁺03] Thomas Clausen, Philippe Jacquet, Cédric Adjih, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot. Optimized link state routing protocol. RFC 3626, Internet Engineering Task Force, October 2003. url: <http://hal.inria.fr/inria-00471712>.
- [CLWM11] Tarik Crnovrsanin, Isaac Liao, Yingcai Wu, and Kwan-Liu Ma. Visual recommendations for network navigation. *Computer Graphics Forum*, 30(3):1081–1090, 2011. doi:[10.1111/j.1467-8659.2011.01957.x](https://doi.org/10.1111/j.1467-8659.2011.01957.x).
- [CN02] Stuart K. Card and David Nation. Degree-of-interest trees: A component of an attention-reactive user interface. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 231–245, 2002. doi:[10.1145/1556262.1556300](https://doi.org/10.1145/1556262.1556300).
- [CSP⁺06] Stuart K. Card, Bongwon Sun, Bryan A. Pendleton, Jeffrey Heer, and John W. Bodnar. TimeTree: Exploring time changing hierarchies. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 3–10, 2006. doi:[10.1109/VAST.2006.261450](https://doi.org/10.1109/VAST.2006.261450).
- [CXP⁺12] Ross Curtis, Jing Xiang, Ankur Parikh, Peter Kinnaird, and Eric Xing. Enabling dynamic network analysis through visualization in TVNViewer. *BMC Bioinformatics*, 13(1):204–216, 2012. doi:[10.1186/1471-2105-13-204](https://doi.org/10.1186/1471-2105-13-204).
- [DBHH94] John Dill, Lyn Bartram, Albert Ho, and Frank Henigman. A continuously variable zoom for navigating large hierarchical networks. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, pages 386–390, 1994. doi:[10.1109/ICSMC.1994.399869](https://doi.org/10.1109/ICSMC.1994.399869).
- [DG02] Stephan Diehl and Carsten Görg. Graphs, they are changing. Dynamic graph drawing for a sequence of graphs. In *Proceedings of the International Symposium on Graph Drawing*, pages 23–31, 2002. doi:[10.1007/3-540-36151-0_3](https://doi.org/10.1007/3-540-36151-0_3).
- [DGH03] Helmut Doleisch, Martin Gasser, and Helwig Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of the Symposium on Data Visualisation*, pages 239–248, 2003. url: <http://diglib.eg.org/EG/DL/WS/VisSym/VisSym03/239-248.pdf>.
- [DGK00] Stephan Diehl, Carsten Görg, and Andreas Kerren. Foresighted graphlayout. Technical Report A/02/2000, Universität des Saarlandes, 2000. url: <http://scidok.sulb.uni-saarland.de/volltexte/2005/347>.
- [DGK01] Stephan Diehl, Carsten Görg, and Andreas Kerren. Preserving the mental map using foresighted layout. In *Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization*, pages 175–184, 2001. doi:[10.1007/978-3-7091-6215-6_19](https://doi.org/10.1007/978-3-7091-6215-6_19).
- [DMR97] Karen L. Daniels, Victor Milenkovic, and Dan Roth. Finding the largest area axis-parallel rectangle in a polygon. *Computational Geometry*, 7(1-2):125–148, 1997. doi:[10.1016/0925-7721\(95\)00041-0](https://doi.org/10.1016/0925-7721(95)00041-0).
- [dR13] Ernst de Ridder. Information system on graph classes and their inclusions. <http://www.graphclasses.org/>. Accessed: 2013-05-07.

- [dS09] Tricia d’Entremont and Margaret-Anne Storey. Using a degree of interest model to facilitate ontology navigation. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 127–131, 2009. doi:[10.1109/VLHCC.2009.5295284](https://doi.org/10.1109/VLHCC.2009.5295284).
- [DS13] Cody Dunne and Ben Shneiderman. Motif simplification: Improving network visualization readability with fan, connector, and clique glyphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3247–3256, 2013. doi:[10.1145/2470654.2466444](https://doi.org/10.1145/2470654.2466444).
- [DvdHSM06] A. Rogier T. Donders, Geert J.M.G. van der Heijden, Theo Stijnen, and Karel G.M. Moons. Review: A gentle introduction to imputation of missing values. *Journal of Clinical Epidemiology*, 59(10):1087–1091, 2006. doi:[10.1016/j.jclinepi.2006.01.014](https://doi.org/10.1016/j.jclinepi.2006.01.014).
- [EDG⁺08] Niklas Elmqvist, Thanh-Nghi Do, Howard Goodell, Nathalie Henry, and Jean-Daniel Fekete. Zame: Interactive large-scale graph visualization. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 215–222, 2008. doi:[10.1109/PACIFICVIS.2008.4475479](https://doi.org/10.1109/PACIFICVIS.2008.4475479).
- [EF97] Peter Eades and Qing-Wen Feng. Multilevel visualization of clustered graphs. In *Proceedings of the Symposium on Graph Drawing*, pages 101–112, 1997. doi:[10.1007/3-540-62495-3_41](https://doi.org/10.1007/3-540-62495-3_41).
- [EF10] Niklas Elmqvist and Jean-Daniel Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2010. doi:[10.1109/TVCG.2009.84](https://doi.org/10.1109/TVCG.2009.84).
- [EGK⁺03] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz and Dynagraph – Static and dynamic graph drawing tools. In *Graph Drawing Software*, pages 127–148. 2003. doi:[10.1007/978-3-642-18638-7_6](https://doi.org/10.1007/978-3-642-18638-7_6).
- [EHK⁺04] Cesim Erten, Philip J. Harding, Stephen G. Kobourov, Kevin Wampler, and Gary Yee. Exploring the computing literature using temporal graph visualization. In *Proceedings of Society of Photo-Optical Instrumentation Engineers Conference Series*, pages 45–56, 2004. doi:[10.1117/12.539245](https://doi.org/10.1117/12.539245).
- [EL05] Ergin Elmacioglu and Dongwon Lee. On six degrees of separation in DBLP-DB and more. *SIGMOD Record*, 34(2):33–40, 2005. doi:[10.1145/1083784.1083791](https://doi.org/10.1145/1083784.1083791).
- [FBS06] Tanja Falkowski, Jorg Bartelheimer, and Myra Spiliopoulou. Mining and visualizing the evolution of subgroups in social networks. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 52–58, 2006. doi:[10.1109/WI.2006.118](https://doi.org/10.1109/WI.2006.118).
- [FE01] Carsten Friedrich and Peter Eades. The Marey graph animation tool demo. In *Proceedings of Graph Drawing*, pages 396–406, 2001. doi:[10.1007/3-540-44541-2_37](https://doi.org/10.1007/3-540-44541-2_37).
- [FH04] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. doi:[10.1023/B:VISI.0000022288.19776.77](https://doi.org/10.1023/B:VISI.0000022288.19776.77).
- [FP03] Jean-Daniel Fekete and Catherine Plaisant. TreeML DTD describing a tree structure for visualization. In *InfoVis Contest*, 2003. url: <http://www.cs.umd.edu/hcil/iv03contest/datasets.html>.

- [FR91] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software-Practice & Experience*, 21(11):1129–1164, 1991. doi:[10.1002/spe.4380211102](https://doi.org/10.1002/spe.4380211102).
- [FT04] Yaniv Frishman and Ayellet Tal. Dynamic drawing of clustered graphs. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 191–198, 2004. doi:[10.1109/INFVIS.2004.18](https://doi.org/10.1109/INFVIS.2004.18).
- [FT08] Yaniv Frishman and Ayellet Tal. Online dynamic graph drawing. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):727–740, 2008. doi:[10.1109/TVCG.2008.11](https://doi.org/10.1109/TVCG.2008.11).
- [Fur86] George W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 16–23, 1986. doi:[10.1145/22627.22342](https://doi.org/10.1145/22627.22342).
- [GDC10] Derek Greene, Dónal Doyle, and Pádraig Cunningham. Tracking the evolution of communities in dynamic social networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, pages 176–183, 2010. doi:[10.1109/ASONAM.2010.17](https://doi.org/10.1109/ASONAM.2010.17).
- [GDLP09] Emilio Giacomo, Walter Didimo, Giuseppe Liotta, and Pietro Palladino. Visual analysis of one-to-many matched graphs. In *Proceedings of Graph Drawing*, pages 133–144, 2009. doi:[10.1007/978-3-642-00219-9_14](https://doi.org/10.1007/978-3-642-00219-9_14).
- [GFV12] Helen Gibson, Joe Faith, and Paul Vickers. A survey of two-dimensional graph layout techniques for information visualisation. *Information Visualization*, 12(3-4):1–34, 2012. doi:[10.1177/1473871612455749](https://doi.org/10.1177/1473871612455749).
- [GGWW06] Marco Gaertler, Robert Görke, Dorothea Wagner, and Silke Wagner. How to cluster evolving graphs. In *Proceedings of the European Conference on Complex Systems*, volume 6, 2006. url: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.124.500>.
- [GJ79] Michael Garey and David Johnson. *Computers and Intractability – A Guide to the Theory of NP-completeness*. W.H.Freeman, 1979. url: <http://www.worldcat.org/oclc/4195125>.
- [GKN05] Emden R. Gansner, Yehuda Koren, and Stephen C. North. Topological fisheye views for visualizing large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):457–468, 2005. doi:[10.1109/TVCG.2005.66](https://doi.org/10.1109/TVCG.2005.66).
- [Gla08] Dieter Glaser. Zeitexpandiertes Graphenclustern -Modellierung und Experiment. Master’s thesis, Universität Karlsruhe, 2008. url: http://illwww.iti.uni-karlsruhe.de/_media/teaching/theses/files/da-glaser-08.pdf.
- [GMSW10] Robert Görke, Pascal Maillard, Christian Staudt, and Dorothea Wagner. Modularity-driven clustering of dynamic graphs. In *Proceedings of the International Conference on Experimental Algorithms*, pages 436–448, 2010. doi:[10.1007/978-3-642-13193-6_37](https://doi.org/10.1007/978-3-642-13193-6_37).
- [GN02] Michelle Girvan and Mark E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002. doi:[10.1073/pnas.122653799](https://doi.org/10.1073/pnas.122653799).
- [GOV⁺11] Enrico Gutzzeit, Stephan Ohl, Jörg Voskamp, Arjan Kuijper, and Bodo Urban. Automatic wood log segmentation using graph cuts. In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics*, pages 96–109, 2011. doi:[10.1007/978-3-642-25382-9_7](https://doi.org/10.1007/978-3-642-25382-9_7).

- [GP07] Markus Gross and Hanspeter Pfister, editors. *Point-Based Graphics*. Morgan Kaufmann Publishers, 2007. url: <http://www.worldcat.org/oclc/162131393>.
- [GST13] Stefan Gladisch, Heidrun Schumann, and Christian Tominski. Navigation recommendations for exploring hierarchical graphs. In *Proceedings of the International Symposium on Advances in Visual Computing*, pages 36–47, 2013. doi:10.1007/978-3-642-41939-3_4.
- [GW06] Marco Gaertler and Dorothea Wagner. A hybrid model for drawing dynamic and evolving graphs. In *Proceedings of Graph Drawing*, pages 189–200, 2006. doi:10.1007/11618058_18.
- [GW11] Yi Gu and Chaoli Wang. TransGraph: Hierarchical exploration of transition relationships in time-varying volumetric data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2015–2024, 2011. doi:10.1109/TVCG.2011.246.
- [GXTL10] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis & Applications*, 13(1):113–129, 2010. doi:10.1007/s10044-008-0141-y.
- [Har96] Robert L. Harris. *Information Graphics: A Comprehensive Illustrated Reference : Visual Tools for Analyzing, Managing, and Communicating*. Management Graphics, 1996. url: <http://www.worldcat.org/oclc/34779053>.
- [HB03] Mark Harrower and Cynthia A. Brewer. Colorbrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003. doi:10.1179/000870403235002042.
- [HC04] Jeffrey Heer and Stuart K. Card. DOITrees revisited: Scalable, space-constrained visualization of hierarchical data. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 421–424, 2004. doi:10.1145/989863.989941.
- [HDM98] Ivan Herman, M. Delest, and Guy Melançon. Tree visualisation and navigation clues for information visualisation. *Computer Graphics Forum*, 17(2):153–165, 1998. doi:10.1111/1467-8659.00235.
- [HE98] Mao Lin Huang and Peter Eades. A fully animated interactive system for clustering and navigating huge graphs. In *Proceedings of the International Symposium on Graph Drawing*, pages 374–383, 1998. doi:10.1007/3-540-37623-2_29.
- [HEF⁺13] Christophe Hurter, Ozan Ersoy, Sara I. Fabrikant, Tijmen R. Klein, and Alexandru C. Telea. Bundled visualization of dynamic graph and trail data. *IEEE Transactions on Visualization and Computer Graphics*, 2013. PrePrints. doi:10.1109/TVCG.2013.246.
- [HFM07] Nathalie Henry, Jean-Daniel Fekete, and Michael J. McGuffin. NodeTrix: A hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1302–1309, 2007. doi:10.1109/TVCG.2007.70582.
- [HH06] Tze-Haw Huang and Mao Lin Huang. Analysis and visualization of co-authorship networks for understanding academic collaboration and knowledge domain of individual researchers. In *Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation*, pages 18–23, 2006. doi:10.1109/CGIV.2006.20.
- [HivWF11] Danny Holten, Petra Isenberg, Jarke J. van Wijk, and Jean-Daniel Fekete. An extended evaluation of the readability of tapered, animated, and textured directed-edge representations in node-link graphs. In *Proceedings of the Pacific Visualization Symposium*, pages 195–202, 2011. doi:10.1109/PACIFICVIS.2011.5742390.

- [HKHB07] Bruce W. Herr, Weimao Ke, Elisha Hardy, and Katy Börner. Movies and actors: Mapping the internet movie database. In *Proceedings of Information Visualisation*, pages 465–469, 2007. doi:10.1109/IV.2007.78.
- [HKPS04] Roland Heilmann, Daniel A. Keim, Christian Panse, and Mike Sips. RecMap: Rectangular map approximations. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 33–40, 2004. doi:10.1109/INFVIS.2004.57.
- [HP14] Jeffrey Heer and Adam Perer. Orion: A system for modeling, transformation and visualization of multidimensional heterogeneous networks. *Information Visualization*, 2014. doi:10.1177/1473871612462152.
- [HSS10] Clemens Holzhüter, Hans-Jörg Schulz, and Heidrun Schumann. Enriched heatmaps for visualizing uncertainty in microarray data. In *Poster Compendium of the Eurographics Workshop on Visual Computing for Biomedicine*, 2010. url: <http://www.informatik.uni-rostock.de/~hs162/pdf/vcbm10abs.pdf>.
- [HTC10] Christophe Hurter, Benjamin Tissoires, and Stéphane Conversy. Accumulation as a tool for efficient visualization of geographical and temporal data. In *Proceedings of AGILE Workshop Geospatial Visual Analytics: Focus on Time*, 2010. url: <http://www.lii-enac.fr/~conversy/research/papers/geovat2010-accumulation.pdf>.
- [HVvW05] Danny Holten, Roel Vliegen, and Jarke J. van Wijk. Visual realism for the visualization of software metrics. In *Proceedings of the IEEE International Workshop on Visualizing Software for Understanding and Analysis*, pages 27–32, 2005. doi:10.1109/VISSOF.2005.1684299.
- [HvW09] Danny Holten and Jarke J. van Wijk. Force-directed edge bundling for graph visualization. In *Proceedings of the Eurographics / IEEE - VGTC Conference on Visualization*, pages 983–998, 2009. doi:10.1111/j.1467-8659.2009.01450.x.
- [HZ07] Peter Hüsken and Jürgen Ziegler. Degree-of-interest visualization for ontology exploration. In *Proceedings of the International Conference on Human-Computer Interaction*, pages 116–119, 2007. doi:10.1007/978-3-540-74796-3_12.
- [IKIY02] Takayuki Itoh, Yasumasa Kajinaga, Yuko Ikehata, and Yumi Yamaguchi. Data Jewelry Box: A graphics showcase for large-scale hierarchical data visualization. Technical Report RT0427, IBM Research, 2002. url: <http://domino.research.ibm.com/library/cyberdig.nsf/0/65ee18867a4cf82385256b870028fa47>.
- [IYIK04] Takayuki Itoh, Yumi Yamaguchi, Yuko Ikehata, and Yasumasa Kajinaga. Hierarchical data visualization using a fast rectangle-packing algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):302–313, 2004. doi:10.1109/TVCG.2004.1272729.
- [IYT+13] Masahiko Itoh, Daisaku Yokoyama, Yoshimitsu Tomita, Satoshi Kawamura, Masashi Toyoda, and Masaru Kitsuregawa. Visualization of passenger flows on metro. In *Poster Compendium of the IEEE Conference on Visual Analytics Science and Technology*, 2013. url: http://www.tkl.iis.u-tokyo.ac.jp/top/modules/newdb/extract/1303/data/VAST2013_1.pdf.
- [JDK10] Ilir Jusufi, Yang Dingjie, and Andreas Kerren. The network lens: Interactive exploration of multivariate networks using visual filtering. In *Proceedings of the International Conference on Information Visualisation*, pages 35–42, 2010. doi:10.1109/IV.2010.15.

- [JH07] Mikkel R. Jakobsen and Kasper Hornbæk. Transient visualizations. In *Proceedings of Australasian Conference on Computer-Human Interaction*, pages 69–76, 2007. doi:10.1145/1324892.1324905.
- [JKKS12] Ilir Jusufi, Christian Klukas, Andreas Kerren, and Falk Schreiber. Guiding the interactive exploration of metabolic pathway interconnections. *Information Visualization*, 11(2):136–150, 2012. doi:10.1177/1473871611405677.
- [Joh93] Brian Scott Johnson. *Treemaps: Visualizing hierarchical and categorical data*. PhD thesis, University of Maryland, 1993. url: <http://search.proquest.com/docview/304079297/abstract>.
- [JS91] Brian Johnson and Ben Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the IEEE Conference on Visualization*, pages 284–291, 1991. doi:10.1109/VISUAL.1991.175815.
- [JSS⁺09] Mathias John, Hans-Jörg Schulz, Heidrun Schumann, Adelinde M. Uhrmacher, and Andrea Unger. Exploring time-varying hypergraphs. In *Poster Compendium of the IEEE Conference on Information Visualization*, 2009. url: http://www.informatik.uni-rostock.de/~schumann/papers/2008+/abstract_Poster_InfoVis09.pdf.
- [JSS⁺13] Mathias John, Hans-Jörg Schulz, Heidrun Schumann, Adelinde M. Uhrmacher, and Andrea Unger. Constructing and visualizing chemical reaction networks from pi-calculus models. *Formal Aspects of Computing*, 25:723–742, 2013. doi:10.1007/s00165-011-0209-0.
- [KBH06] Robert Kosara, Fabian Bendix, and Helwig Hauser. Parallel sets: Interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):558–568, 2006. doi:10.1109/TVCG.2006.76.
- [KG06] Gautam Kumar and Michael Garland. Visual exploration of complex time-varying graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):805–812, 2006. doi:10.1109/TVCG.2006.193.
- [KK89] Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989. doi:10.1016/0020-0190(89)90102-6.
- [KKC14a] Natalie Kerracher, Jessie Kennedy, and Kevin Chalmers. The design space of temporal graph visualisation. In *Proceedings of the Eurographics Conference on Visualization—Short Papers*, 2014. doi:10.2312/eurovisshort.20141149.
- [KKC14b] Natalie Kerracher, Jessie Kennedy, and Kevin Chalmers. Tasks for temporal graph visualisation. *arXiv preprint*, 2014. Accessed: 2014-04-25. url: <http://arxiv.org/abs/1402.2867>.
- [KKZ12] Andreas Kerren, Harald Köstinger, and Björn Zimmer. ViNCent : Visualization of network centralities. In *Proceedings of the International Conference on Information Visualization Theory and Applications*, pages 703–712. SciTePress, 2012. url: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.220.2484>.
- [KL83] Joseph B. Kruskal and James M. Landwehr. Icicle Plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983. doi:10.1080/00031305.1983.10482733.

- [KMSZ06] Daniel A. Keim, Florian Mansmann, Jorn Schneidewind, and Hartmut Ziegler. Challenges in visual data analysis. In *Proceedings of Information Visualisation*, pages 9–16, 2006. doi:10.1109/IV.2006.31.
- [KNC⁺11] Udayan Khurana, Viet-An Nguyen, Hsueh-Chien Cheng, Jae wook Ahn, Xi Chen, and Ben Shneiderman. Visual analysis of temporal trends in social networks using edge color coding and metric timelines. In *Proceedings of the IEEE International Conference on Privacy, Security, Risk and Trust and on Social Computing*, pages 549–554, 2011. doi:10.1109/PASSAT/SocialCom.2011.212.
- [Kra03] Menno-Jan Kraak. The space-time cube revisited from a geovisualization perspective. In *Proceedings of International Cartographic Conference*, pages 1988–1995, 2003. url: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.9231&rep=rep1&type=pdf>.
- [LA93] Ying K. Leung and Mark D. Apperley. E³: Towards the metrickation of graphical presentation techniques for large data sets. In *Proceedings of International Conference on Human-Computer Interaction*, pages 125–140, 1993. doi:10.1007/3-540-57433-6_44.
- [LBA10] Antoine Lambert, Romain Bourqui, and David Auber. 3D edge bundling for geographical data visualization. In *Proceedings of the International Conference on the Information Visualisation*, pages 329–335, 2010. doi:10.1109/IV.2010.53.
- [LBE04] Anatole Lécuyer, Jean-Marie Burkhardt, and Laurent Etienne. Feeling bumps and holes without a haptic interface: The perception of pseudo-haptic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 239–246, 2004. doi:10.1145/985692.985723.
- [LD11] David Lloyd and Jason Dykes. Human-centered approaches in geovisualization design: Investigating multiple methods through a long-term case study. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2498–2507, 2011. doi:10.1109/TVCG.2011.209.
- [LDR⁺10] Chen Li, Marco Donizelli, Nicolas Rodriguez, Harish Dharuri, Lukas Endler, Vijayalakshmi Chelliah, Lu Li, Enuo He, Arnaud Henry, Melanie I. Stefan, Jacky L. Snoep, Michael Hucka, Nicolas Le Novère, and Camille Laibe. BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Systems Biology*, 4(92):1–14, 2010. doi:10.1186/1752-0509-4-92.
- [Ley02] Michael Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *Proceedings of the International Symposium on String Processing and Information Retrieval*, pages 1–10, 2002. doi:10.1007/3-540-45735-6_1.
- [Ley09] Michael Ley. DBLP – Some lessons learned. *Proceedings of the VLDB Endowment*, 2(2):1493–1500, 2009. doi:10.14778/1687553.1687577.
- [LF08] Hao R. Lu and James Fogarty. Cascaded treemaps: Examining the visibility and stability of structure in treemaps. In *Proceedings of the Graphics Interface Conference*, pages 259–266, 2008. url: <http://dl.acm.org/citation.cfm?id=1375714.1375758>.
- [LLCM12] Sheng-Jie Luo, Chun-Liang Liu, Bing-Yu Chen, and Kwan-Liu Ma. Ambiguity-free edge-bundling for interactive graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(5):810–821, 2012. doi:10.1109/TVCG.2011.104.

- [Lop13] Thomas Lopatic. Olsrd link quality extensions. <http://www.olsr.org/docs/README-Link-Quality.html>. Accessed: 2013-03-25.
- [LPP⁺06] Bonghin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In *Proceedings of the Workshop BEyond time and errors: novel evaluation methods for Information Visualization*, pages 1–5, 2006. doi:10.1145/1168149.1168168.
- [LS08a] Martin Luboschik and Heidrun Schumann. Discovering the covered: Ghost-views in information visualization. In *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 113–118, 2008. url: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.212.7888>.
- [LS08b] Martin Luboschik and Heidrun Schumann. Illustrative halos in information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 384–387, 2008. doi:10.1145/1385569.1385639.
- [LSC08] Martin Luboschik, Heidrun Schumann, and Hilko Cords. Particle-based labeling: Fast point-feature labeling without obscuring other visual features. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1237–1244, 2008. doi:10.1109/TVCG.2008.152.
- [LSM⁺12] Martin Luboschik, Hans-Jörg Schulz, Carsten Maus, Adelinde M. Uhrmacher, and Heidrun Schumann. Heterogeneity-based guidance for exploring multiscale data in systems biology. In *Proceedings of the 2nd IEEE Symposium on Biological Data Visualization*, 2012. doi:10.1109/BioVis.2012.6378590.
- [LWZZ09] Guiling Li, Yuanzhen Wang, Liping Zhang, and Xiaolian Zhu. Similarity measure for time series based on piecewise linear approximation. In *Proceedings of the International Conference on Wireless Communications Signal Processing*, pages 1–4, 2009. doi:10.1109/WCSP.2009.5371709.
- [LY07] Chun-Cheng Lin and Hsu-Chun Yen. On balloon drawings of rooted trees. *Journal of Graph Algorithms and Applications*, 11(2):431–452, 2007. doi:10.1007/11618058_26.
- [Mac86] Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986. doi:10.1145/22949.22950.
- [MDH⁺03] Alan MacEachren, Xiping Dai, Frank Hardisty, Diansheng Guo, and Gene Lengerich. Exploring high-D spaces with multiform matrices and small multiples. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 31–38, 2003. doi:10.1109/INFVIS.2003.1249006.
- [ME09] Bryan McDonnel and Niklas Elmqvist. Towards utilizing GPUs in information visualization: A model and implementation of image-space operations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1105–1112, 2009. doi:10.1109/TVCG.2009.191.
- [MJ09] Michael J. McGuffin and Igor Jurisica. Interaction techniques for selecting and manipulating subgraphs in network visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):937–944, 2009. doi:10.1109/TVCG.2009.151.
- [MK97] Alan M. MacEachren and Menno-Jan Kraak. Exploratory cartographic visualization: Advancing the agenda. *Computers & Geosciences*, 23(4):335–343, 1997. doi:10.1016/S0098-3004(97)00018-6.

- [MMKN08] Peter McLachlan, Tamara Munzner, Eleftherios Koutsofios, and Stephen North. LiveRAC: Interactive visual exploration of system management time-series data. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 1483–1492, 2008. doi:10.1145/1357054.1357286.
- [MPG⁺04] Steffen Mader, Christian Peter, Roland Göcke, Randolph Schultz, Jörg Voskamp, and Bodo Urban. A freely configurable, multi-modal sensor system for affective computing. In *Affective Dialogue Systems*, pages 313–318, 2004. doi:10.1007/978-3-540-24842-2_34.
- [MSDK12] Thorsten May, Martin Steiger, James Davey, and Jörn Kohlhammer. Using signposts for navigation in large graphs. *Computer Graphics Forum*, 31(3pt2):985–994, 2012. doi:10.1111/j.1467-8659.2012.03091.x.
- [MTJ12] Hemant Makwana, Sanjay Tanwani, and Suresh Jain. Axes re-ordering in parallel coordinate for pattern optimization. *International Journal of Computer Applications*, 40(13):43–48, 2012. doi:10.5120/5044-7370.
- [NEH13] Quan Nguyen, Peter Eades, and Seok-Hee Hong. StreamEB: Stream edge bundling. In *Proceedings of Graph Drawing*, pages 400–413, 2013. doi:10.1007/978-3-642-36763-2_36.
- [New04] Mark E. J. Newman. Coauthorship networks and patterns of scientific collaboration. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5200–5205, 2004. url: http://www.pnas.org/content/101/suppl_1/5200.abstract.
- [NH03] Quang Vinh Nguyen and Mao Lin Huang. Space-optimized tree: A connection+enclosure approach for the visualization of large hierarchies. *Information Visualization*, 2(1):3–15, 2003. doi:10.1057/palgrave.ivs.9500031.
- [Noi95] Emanuel G. Noik. Encoding presentation emphasis algorithms for graphs. In *Proceedings of Graph Drawing*, pages 428–435, 1995. doi:10.1007/3-540-58950-3_396.
- [NSP03] Mario A. Nascimento, Jörg Sander, and Jeffrey Pound. Analysis of SIGMOD’s co-authorship graph. *SIGMOD Record*, 32(3):8–10, 2003. doi:10.1145/945721.945722.
- [OFS⁺05] Joshua O’Madadhain, Danyel Fisher, Padhraic Smyth, Scott White, and Yan-Biao Boey. Analysis and visualization of network data using JUNG. http://jung.sourceforge.net/doc/JUNG_journal.pdf. Accessed: 2013-05-07.
- [ONG⁺07] Benoit Otjacques, Monique Noirhomme, Xavier Gobert, Pierre Collin, and Fernand Feltz. Visualizing the activity of a web-based collaborative platform. In *Proceedings of the International Conference on Information Visualisation*, pages 251–256, 2007. doi:10.1109/IV.2007.137.
- [Osa01] Noritaka Osawa. A multiple-focus graph browsing technique using heat models and force-directed layout. In *Proceedings of the International Conference on Information Visualisation*, pages 277–283, 2001. doi:10.1109/IV.2001.942071.
- [OW00] Chris Olston and Allison Woodruff. Getting portals to behave. In *Proceedings of Symposium on Information Visualization*, pages 15–25, 2000. doi:10.1109/INFVIS.2000.885087.
- [PD08] Mathias Pohl and Stephan Diehl. What dynamic network metrics can tell us about developer roles. In *Proceedings of the Cooperative and Human Aspects of Software Engineering*, pages 81–84, 2008. doi:10.1145/1370114.1370135.

- [Pha90] Binh Pham. Spline-based color sequences for univariate, bivariate and trivariate mapping. In *Proceedings of the Conference on Visualization*, pages 202–208, 1990. doi:[10.1109/VISUAL.1990.146383](https://doi.org/10.1109/VISUAL.1990.146383).
- [PHG07] Helen Purchase, Eve Hoggan, and Carsten Görg. How important is the “mental map”? - An empirical investigation of a dynamic graph layout algorithm. In *Proceedings of Graph Drawing*, pages 184–195, 2007. doi:[10.1007/978-3-540-70904-6_19](https://doi.org/10.1007/978-3-540-70904-6_19).
- [PIB⁺11] Robert Patro, Cheuk Yiu Ip, Sujal Bista, Samuel S. Cho, Devarajan Thirumalai, and Amitabh Varshney. MDMap: A system for data-driven layout and exploration of molecular dynamics simulations. In *Proceedings of the IEEE Symposium on Biological Data Visualization*, pages 111–118, 2011. doi:[10.1109/BioVis.2011.6094055](https://doi.org/10.1109/BioVis.2011.6094055).
- [PMD12] Bruno Pinaud, Guy Melançon, and Jonathan Dubois. PORGY: A visual graph rewriting environment for complex systems. *Computer Graphics Forum*, 31(3pt4):1265–1274, 2012. doi:[10.1111/j.1467-8659.2012.03119.x](https://doi.org/10.1111/j.1467-8659.2012.03119.x).
- [PS12] Adam Perer and Jimeng Sun. MatrixFlow: Temporal network visual analytics to track symptom evolution during disease progression. In *AMIA Annual Symposium Proceedings*, volume 2012, pages 716–725, 2012. url: <http://www.ncbi.nlm.nih.gov/pubmed/23304345>.
- [PvH12] Adam Perer and Frank van Ham. Integrating querying and browsing in partial graph visualization. Technical Report 12-01, IBM Research, 2012. url: <http://perer.org/papers/adamPerer-DOIGraphs-IBM2011.pdf>.
- [PvW08] A. Johannes Pretorius and Jarke J. van Wijk. Visual inspection of multivariate graphs. In *Proceedings of the Joint Eurographics / IEEE - VGTC Conference on Visualization*, pages 967–974, 2008. doi:[10.1111/j.1467-8659.2008.01231.x](https://doi.org/10.1111/j.1467-8659.2008.01231.x).
- [PWR04] Wei Peng, Matthew O. Ward, and Elke A. Rundensteiner. Clutter reduction in multi-dimensional data visualization using dimension reordering. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 89–96, 2004. doi:[10.1109/INFVIS.2004.15](https://doi.org/10.1109/INFVIS.2004.15).
- [PXYH05] Doantam Phan, Ling Xiao, Ron Yeh, and Pat Hanrahan. Flow map layout. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 219–224, 2005. doi:[10.1109/INFVIS.2005.1532150](https://doi.org/10.1109/INFVIS.2005.1532150).
- [RB10] Martin Rosvall and Carl T. Bergstrom. Mapping change in large networks. *PLoS ONE*, 5(1):e8694, 2010. doi:[10.1371/journal.pone.0008694](https://doi.org/10.1371/journal.pone.0008694).
- [RG93] Jun Rekimoto and Mark Green. The information cube: Using transparency in 3D information visualization. In *Proceedings of the Workshop on Information Technologies and Systems*, pages 125–132, 1993. url: <http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.4003>.
- [RHF05] Patrick Riehmann, Manfred Hanfler, and Bernd Fröhlich. Interactive Sankey diagrams. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 233–240, 2005. doi:[10.1109/INFVIS.2005.1532152](https://doi.org/10.1109/INFVIS.2005.1532152).
- [RM13] Sébastien Rufiange and Michael J. McGuffin. DiffAni: Visualizing dynamic graphs with a hybrid of difference maps and animation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2556–2565, 2013. doi:[10.1109/TVCG.2013.149](https://doi.org/10.1109/TVCG.2013.149).
- [RMF09] Sébastien Rufiange, Michael J. McGuffin, and Christopher Fuhrman. 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*, pages 51–54, 2009. doi:[10.1145/1629826.1629834](https://doi.org/10.1145/1629826.1629834).
- [RMFS⁺11] Eduarda Mendes Rodrigues, Natasa Milic-Frayling, Marc Smith, Ben Shneiderman, and Derek Hansen. Group-in-a-box layout for multi-faceted analysis of communities. In *Proceedings of the IEEE International Conference on Privacy, Security, Risk and Trust and on Social Computing*, pages 354–361, 2011. doi:[10.1109/PASSAT/SocialCom.2011.139](https://doi.org/10.1109/PASSAT/SocialCom.2011.139).
- [RO86] Philip Robertson and John O'Callaghan. The generation of color sequences for univariate and bivariate mapping. *IEEE Computer Graphics and Applications*, 6(2):24–32, 1986. doi:[10.1109/MCG.1986.276688](https://doi.org/10.1109/MCG.1986.276688).
- [RPD09] Florian Reitz, Mathias Pohl, and Stephan Diehl. Focused animation of dynamic compound graphs. In *Proceedings of the International Conference on Information Visualisation*, pages 679–684, 2009. doi:[10.1109/IV.2009.24](https://doi.org/10.1109/IV.2009.24).
- [RT]⁺11] Khairi Reda, Chayant Tantipathananandh, Andrew Johnson, Jason Leigh, and Tanya Berger-Wolf. Visualizing the evolution of community structures in dynamic social networks. In *Proceedings of the Eurographics / IEEE - VGTC Conference on Visualization*, pages 1061–1070, 2011. doi:[10.1111/j.1467-8659.2011.01955.x](https://doi.org/10.1111/j.1467-8659.2011.01955.x).
- [RUK⁺10] Markus Rohrschneider, Alexander Ullrich, Andreas Kerren, Peter F. Stadler, and Gerek Scheuermann. Visual network analysis of dynamic metabolic pathways. In *Proceedings of the International Conference on Advances in Visual Computing - Volume Part I*, pages 316–327, 2010. doi:[10.1007/978-3-642-17289-2_31](https://doi.org/10.1007/978-3-642-17289-2_31).
- [RZT⁺09] Lei Ren, Lin Zhang, Dongxing Teng, Guozhong Dai, and Qian Li. DOI-Wave: A focus+context interaction technique for networks based on attention-reactive interface. In Mao Lin Huang, Quang Vinh Nguyen, and Kang Zhang, editors, *Visual Information Communication*, chapter 5, pages 85–94. Springer, 2009. doi:[10.1007/978-1-4419-0312-9_5](https://doi.org/10.1007/978-1-4419-0312-9_5).
- [SA06] Ben Shneiderman and Aleks Aris. Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):733–740, 2006. doi:[10.1109/TVCG.2006.166](https://doi.org/10.1109/TVCG.2006.166).
- [SAM13] Hans-Jörg Schulz, Zabedul Akbar, and Frank Maurer. A generative layout approach for rooted tree drawings. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 225–232, 2013. doi:[10.1109/PacificVis.2013.6596149](https://doi.org/10.1109/PacificVis.2013.6596149).
- [Sch10] Hans-Jörg Schulz. *Explorative Graph Visualization*. PhD thesis, University of Rostock, 2010. url: http://rosdok.uni-rostock.de/metadata/rosdok_disshab_000000000457.
- [Sch11] Hans-Jörg Schulz. Treevis.net: A tree visualization reference. *IEEE Computer Graphics and Applications*, 31(6):11–15, 2011. doi:[10.1109/MCG.2011.103](https://doi.org/10.1109/MCG.2011.103).
- [SDMW09] Falk Schreiber, Tim Dwyer, Kim Marriott, and Michael Wybrow. A generic algorithm for layout of biological networks. *BMC Bioinformatics*, 10(1):1–12, 2009. doi:[10.1186/1471-2105-10-375](https://doi.org/10.1186/1471-2105-10-375).
- [SDW08] Aidan Slingsby, Jason Dykes, and Jo Wood. Using treemaps for variable selection in spatio-temporal visualization. *Information Visualization*, 7(3):210–224, 2008. doi:[10.1057/palgrave.ivs.9500185](https://doi.org/10.1057/palgrave.ivs.9500185).

- [SF12] Andre Spritzer and Carla M. D. S. Freitas. Design and evaluation of MagnetViz—A graph visualization tool. *IEEE Transactions on Visualization and Computer Graphics*, 18(5):822–835, 2012. doi:10.1109/TVCG.2011.106.
- [SGL08] John Stasko, Carsten Görg, and Zhicheng Liu. Jigsaw: Supporting investigative analysis through interactive visualization. *Information Visualization*, 7(2):118–132, 2008. doi:10.1057/palgrave.ivs.9500180.
- [SHQ08] Ross Shannon, Thomas Holland, and Aaron Quigley. Multivariate graph drawing using parallel coordinate visualisations. Technical Report UCD-CSI-2008-06, University College Dublin, 2008. url: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.178.2648>.
- [SJS08] Hans-Jörg Schulz, Mathias John, Andrea Unger, and Heidrun Schumann. Visual analysis of bipartite biological networks. In *Proceedings of the Eurographics Workshop on Visual Computing for Biomedicine*, pages 135–142, 2008. doi:10.2312/VCBM/VCBM08/135–142.
- [SK03] Heidrun Schumann and Matthias Kreuseler. Fokus-und-Kontext-Darstellung im geographischen Kontext. In *Visualisierung und Erschliessung von Geodaten: Seminar GEOVIS*, volume 7, page 25, 2003. url: http://www.informatik.uni-rostock.de/~schumann/papers/2002+/GeoVis_schumann.pdf.
- [SKK06] Christof Rezk Salama, Maik Keller, and Peter Kohlmann. High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1021–1028, 2006. doi:10.1109/TVCG.2006.148.
- [SKW⁺07] Markus Schedl, Peter Knees, Gerhard Widmer, Klaus Seyerlehner, and Tim Pohle. Browsing the web using stacked three-dimensional sunbursts to visualize term co-occurrences and multimedia content. In *Poster Compendium of the IEEE Conference on Information Visualization*, pages 2–3, 2007. url: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.211.3470>.
- [SL10] Frank Steinbrückner and Claus Lewerentz. Representing development history in software cities. In *Proceedings of the International Symposium on Software Visualization*, pages 193–202, 2010. doi:10.1145/1879211.1879239.
- [SLN05] Purvi Saraiya, Peter Lee, and Chris North. Visualization of graphs with associated timeseries data. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 225–232, 2005. doi:10.1109/INFVIS.2005.1532151.
- [SNHS13] Hans-Jörg Schulz, Thomas Nocke, Magnus Heitzler, and Heidrun Schumann. A design space of visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2366–2375, 2013. doi:10.1109/TVCG.2013.120.
- [SSSM11] Samuel Silva, Beatriz Sousa Santos, and Joaquim Madeira. Technical section: Using color in visualization: A survey. *Computers & Graphics*, 35(2):320–333, 2011. doi:10.1016/j.cag.2010.11.015.
- [ST11] Heidrun Schumann and Christian Tominski. Analytical, visual, and interactive concepts for geo-visual analytics. *Journal of Visual Languages & Computing*, 22(4):257–267, 2011. doi:10.1016/j.jvlc.2011.03.002.
- [STH02] Chris Stolte, Diane Tang, and Pat Hanrahan. Query, analysis, and visualization of hierarchically structured data using Polaris. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 112–122, 2002. doi:10.1145/775047.775064.

- [SWS10] Klaus Stein, Rene Wegener, and Christoph Schlieder. Pixel-oriented visualization of change in social networks. In *Proceedings of the Conference on Advances in Social Networks Analysis and Mining*, pages 233–240, 2010. doi:[10.1109/ASONAM.2010.18](https://doi.org/10.1109/ASONAM.2010.18).
- [SWW11] Lei Shi, Chen Wang, and Zhen Wen. Dynamic network visualization in 1.5D. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 179–186, 2011. doi:[10.1109/PACIFICVIS.2011.5742388](https://doi.org/10.1109/PACIFICVIS.2011.5742388).
- [SZ00] John Stasko and Eugene Zhang. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 57–65, 2000. doi:[10.1109/INFVIS.2000.885091](https://doi.org/10.1109/INFVIS.2000.885091).
- [SZG⁺96] Doug Schaffer, Zhengping Zuo, Saul Greenberg, Lyn Bartram, John Dill, Shelli Dubs, and Mark Roseman. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Transactions on Computer-Human Interaction*, 3(2):162–188, 1996. doi:[10.1145/230562.230577](https://doi.org/10.1145/230562.230577).
- [TA08] Alexandru Telea and David Auber. Code flows: Visualizing structural evolution of source code. *Computer Graphics Forum*, 27(3):831–838, 2008. doi:[10.1111/j.1467-8659.2008.01214.x](https://doi.org/10.1111/j.1467-8659.2008.01214.x).
- [TAS09] Christian Tominski, James Abello, and Heidrun Schumann. CGV – An interactive graph visualization system. *Computers & Graphics*, 33(6):660–678, 2009. doi:[10.1016/j.cag.2009.06.002](https://doi.org/10.1016/j.cag.2009.06.002).
- [Tel06] Alexandru Telea. Combining extended table lens and treemap techniques for visualizing tabular data. In *Proceedings of the Joint Eurographics / IEEE VGTC Conference on Visualization*, pages 51–58, 2006. doi:[10.2312/VisSym/EuroVis06/051-058](https://doi.org/10.2312/VisSym/EuroVis06/051-058).
- [TFS08a] Conrad Thiede, Georg Fuchs, and Heidrun Schumann. Smart lenses. In *Proceedings of Symposium on Smart Graphics*, pages 178–189, 2008. doi:[10.1007/978-3-540-85412-8_16](https://doi.org/10.1007/978-3-540-85412-8_16).
- [TFS08b] Christian Tominski, Georg Fuchs, and Heidrun Schumann. Task-driven color coding. In *Proceedings of the International Conference on Information Visualisation*, pages 373–380, 2008. doi:[10.1109/IV.2008.24](https://doi.org/10.1109/IV.2008.24).
- [TJ92] David Turo and Brian Johnson. Improving the visualization of hierarchies with treemaps: Design issues and experimentation. In *Proceedings of the IEEE Conference on Visualization*, pages 124–131, 1992. doi:[10.1109/VISUAL.1992.235217](https://doi.org/10.1109/VISUAL.1992.235217).
- [TM02] Soon Tee Teoh and Kwan-Liu Ma. RINGS: A technique for visualizing large hierarchies. In *Proceedings of Graph Drawing*, pages 268–275, 2002. doi:[10.1007/3-540-36151-0_25](https://doi.org/10.1007/3-540-36151-0_25).
- [TM04] Melanie Tory and Torsten Möller. Human factors in visualization research. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):72–84, 2004. doi:[10.1109/TVCG.2004.1260759](https://doi.org/10.1109/TVCG.2004.1260759).
- [TON03] Yoichi Tanaka, Yoshihiro Okada, and Koichi Nijima. Treecube: Visualization tool for browsing 3D multimedia data. In *Proceedings of the International Conference on Information Visualisation*, pages 427–432, 2003. doi:[10.1109/IV.2003.1218020](https://doi.org/10.1109/IV.2003.1218020).
- [TS07] Ying Tu and Han-Wei Shen. Visualizing changes of hierarchical data using treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1286–1293, 2007. doi:[10.1109/TVCG.2007.70529](https://doi.org/10.1109/TVCG.2007.70529).

- [TS08] Ying Tu and Han-Wei Shen. Balloon focus: A seamless multi-focus+context method for treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1157–1164, 2008. doi:[10.1109/TVCG.2008.114](https://doi.org/10.1109/TVCG.2008.114).
- [TSAA12] Christian Tominski, Heidrun Schumann, Gennady Andrienko, and Natalia Andrienko. Stacking-based visualization of trajectory attribute data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2565–2574, 2012. doi:[10.1109/TVCG.2012.265](https://doi.org/10.1109/TVCG.2012.265).
- [TWS05] Christian Tominski, Petra Schulze-Wollgast, and Heidrun Schumann. 3D information visualization for time dependent data on maps. In *Proceedings of the International Conference on Information Visualisation*, pages 175–181, 2005. doi:[10.1109/IV.2005.3](https://doi.org/10.1109/IV.2005.3).
- [TvW02] Alexandru Telea and Jarke J. van Wijk. An augmented fast marching method for computing skeletons and centerlines. In *Proceedings of the Symposium on Data Visualisation*, pages 251–260, 2002. url: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.163.9385>.
- [US09] Andrea Unger and Heidrun Schumann. Visual support for the understanding of simulation processes. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 57–64, 2009. doi:[10.1109/PACIFICVIS.2009.4906838](https://doi.org/10.1109/PACIFICVIS.2009.4906838).
- [vdEHBvW13] Stef van den Elzen, Danny Holten, Jorik Blaas, and Jarke J. van Wijk. Dynamic network visualization with extended massive sequence views. *IEEE Transactions on Visualization and Computer Graphics*, 2013. PrePrints. doi:[10.1109/TVCG.2013.263](https://doi.org/10.1109/TVCG.2013.263).
- [vHP09] Frank van Ham and Adam Perer. “Search, show context, expand on demand”: Supporting large graph exploration with degree-of-interest. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):953–960, 2009. doi:[10.1109/TVCG.2009.108](https://doi.org/10.1109/TVCG.2009.108).
- [vHSD09] Frank van Ham, Hans-Jörg Schulz, and Joan M. Dimicco. Honeycomb: Visual analysis of large scale social networks. In *Proceedings of the International Conference on Human-Computer Interaction: Part II*, pages 429–442, 2009. doi:[10.1007/978-3-642-03658-3_47](https://doi.org/10.1007/978-3-642-03658-3_47).
- [vHvW02] Frank van Ham and Jarke J. van Wijk. Beamtrees: Compact visualization of large hierarchies. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 93–100, 2002. doi:[10.1109/INFVIS.2002.1173153](https://doi.org/10.1109/INFVIS.2002.1173153).
- [vLdL03] Robert van Liere and Wim de Leeuw. GraphSplatting: Visualizing graphs as continuous fields. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):206–212, 2003. doi:[10.1109/TVCG.2003.1196007](https://doi.org/10.1109/TVCG.2003.1196007).
- [vLKS⁺11] Tatiana von Landesberger, Arjan Kuijper, Tobias Schreck, Jörn Kohlhammer, Jarke J. van Wijk, Jean-Daniel Fekete, and Dieter W. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011. doi:[10.1111/j.1467-8659.2011.01898.x](https://doi.org/10.1111/j.1467-8659.2011.01898.x).
- [VMCJ10] Christophe Viau, Michael J. McGuffin, Yves Chiricota, and Igor Jurisica. The FlowVizMenu and parallel scatterplot matrix: Hybrid multidimensional visualizations for network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1100–1108, 2010. doi:[10.1109/TVCG.2010.205](https://doi.org/10.1109/TVCG.2010.205).
- [vWvdW99] Jarke J. van Wijk and Huub van de Wetering. Cushion treemaps: Visualization of

- hierarchical information. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 73–78, 1999. doi:[10.1109/INFVIS.1999.801860](https://doi.org/10.1109/INFVIS.1999.801860).
- [VWvH⁺07] Fernanda B. Viégas, Martin Wattenberg, Frank van Ham, Jesse Kriss, and Matt McKeon. Many eyes: A site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, 2007. doi:[10.1109/TVCG.2007.70577](https://doi.org/10.1109/TVCG.2007.70577).
- [Wat05] Martin Wattenberg. A note on space-filling visualizations and space-filling curves. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 181–185, 2005. doi:[10.1109/INFVIS.2005.1532145](https://doi.org/10.1109/INFVIS.2005.1532145).
- [Wat06] Martin Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 811–819, 2006. doi:[10.1145/1124772.1124891](https://doi.org/10.1145/1124772.1124891).
- [WD08] Jo Wood and Jason Dykes. Spatially ordered treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14:1348–1355, 2008. doi:[10.1109/TVCG.2008.165](https://doi.org/10.1109/TVCG.2008.165).
- [Wet03] Kai Wetzel. Pebbles – using circular treemaps to visualize disk usage. 2003. url: <http://lip.sourceforge.net/ctreemap.html>.
- [WF94] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press, 1994. url: <http://www.worldcat.org/oclc/818663583>.
- [WH11] Hadley Wickham and Heike Hofmann. Product plots. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2223–2230, 2011. doi:[10.1109/TVCG.2011.227](https://doi.org/10.1109/TVCG.2011.227).
- [WL05] T. Warren Liao. Clustering of time series data-A survey. *Pattern Recognition*, 38(11):1857–1874, 2005. doi:[10.1016/j.patcog.2005.01.025](https://doi.org/10.1016/j.patcog.2005.01.025).
- [WL07] Richard Wettel and Michele Lanza. Visualizing software systems as cities. In *Proceedings of the IEEE International Workshop on Visualizing Software for Understanding and Analysis*, pages 92–99, 2007. doi:[10.1109/VISSOF.2007.4290706](https://doi.org/10.1109/VISSOF.2007.4290706).
- [WM09] Till Wollenberg and Thomas Mundt. Interference aware route optimization with predicted network conditions. In *Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks Workshops*, pages 1–7, 2009. doi:[10.1109/WOWMOM.2009.5282427](https://doi.org/10.1109/WOWMOM.2009.5282427).
- [WOA⁺01] Allison Woodruff, Chris Olston, Alexander Aiken, Michael Chu, Vuk Ercegovic, Mark Lin, Mybrid Spalding, and Michael Stonebraker. DataSplash: A direct manipulation environment for programming semantic zoom visualizations of tabular data. *Journal of Visual Languages & Computing*, 12(5):551–571, 2001. doi:[10.1006/jvlc.2001.0219](https://doi.org/10.1006/jvlc.2001.0219).
- [Wol12] Till Wollenberg. Performance measurement study in a wireless olsr-etx mesh network. In *Proceedings of the Wireless Days*, pages 1–7, 2012. doi:[10.1109/WD.2012.6402882](https://doi.org/10.1109/WD.2012.6402882).
- [WS98] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684):440–442, 1998. doi:[10.1038/30918](https://doi.org/10.1038/30918).
- [WT06] Yingxin Wu and Masahiro Takatsuka. Visualizing multivariate network on the surface of a sphere. In *Proceedings of the Asia-Pacific Symposium on Information Visualisation - Volume 60*, pages 77–83, 2006. url: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.4846>.

- [WTC08] Han-Ming Wu, ShengLi Tzeng, and Chun-houh Chen. Matrix visualization. *Handbook of Data Visualization*, 3:681–708, 2008. doi:[10.1007/978-3-540-33037-0_26](https://doi.org/10.1007/978-3-540-33037-0_26).
- [WWDW06] Weixin Wang, Hui Wang, Guozhong Dai, and Hongan Wang. Visualization of large hierarchical data by circle packing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 517–520, 2006. doi:[10.1145/1124772.1124851](https://doi.org/10.1145/1124772.1124851).
- [XCHT07] Kai Xu, Andrew Cunningham, Seok-Hee Hong, and Bruce H. Thomas. Graph-Scape: Integrated multivariate network visualization. In *Proceedings of the International Asia-Pacific Symposium on Visualization*, pages 33–40, 2007. doi:[10.1109/APVIS.2007.329306](https://doi.org/10.1109/APVIS.2007.329306).
- [XH08] Zengwang Xu and Robert Harriss. Exploring the structure of the U.S. intercity passenger air transportation network: A weighted complex network approach. *GeoJournal*, 73(2):87–102, 2008. doi:[10.1007/s10708-008-9173-5](https://doi.org/10.1007/s10708-008-9173-5).
- [XWI05] Rui Xu and Donald Wunsch II. Survey of clustering algorithms. *Transactions on Neural Networks*, 16(3):645–678, 2005. doi:[10.1109/TNN.2005.845141](https://doi.org/10.1109/TNN.2005.845141).
- [YEL10] Ji Soo Yi, Niklas Elmqvist, and Seungyoon Lee. TimeMatrix: Analyzing temporal social networks using interactive matrix-based visualizations. *International Journal of Human-Computer Interaction*, 26(11 & 12):1031–1051, 2010. doi:[10.1080/10447318.2010.516722](https://doi.org/10.1080/10447318.2010.516722).
- [ZBDS12] Michael Zinsmaier, Ulrik Brandes, Oliver Deussen, and Hendrik Strobelt. Interactive level-of-detail rendering of large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2486–2495, 2012. doi:[10.1109/TVCG.2012.238](https://doi.org/10.1109/TVCG.2012.238).
- [ZJGK10] Hartmut Ziegler, Marco Jenny, Tino Gruse, and Daniel A. Keim. Visual market sector analysis for financial time series data. In *IEEE Symposium on Visual Analytics Science and Technology*, pages 83–90, 2010. doi:[10.1109/VAST.2010.5652530](https://doi.org/10.1109/VAST.2010.5652530).
- [ZJK12] Björn Zimmer, Ilir Jusufi, and Andreas Kerren. Analyzing multiple network centralities with ViNCent. In *Proceedings of SIGRAD Conference on Interactive Visual Analysis of Data*, pages 87–90, 2012. url: <http://urn.kb.se/resolve?urn=urn:nbn:se:lnu:diva-22586>.
- [ZMC05] Shengdong Zhao, Michael J. McGuffin, and Mark H. Chignell. Elastic hierarchies: Combining treemaps and node-link diagrams. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 57–64, 2005. doi:[10.1109/INFVIS.2005.1532129](https://doi.org/10.1109/INFVIS.2005.1532129).

Own Publications

- [AHSS14] James Abello, Steffen Hadlak, Heidrun Schumann, and Hans-Jörg Schulz. A modular degree-of-interest specification for the visual analysis of large dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):337–350, 2014. doi:[10.1109/TVCG.2013.109](https://doi.org/10.1109/TVCG.2013.109).
- [HSCW13] Steffen Hadlak, Heidrun Schumann, Clemens H. Cap, and Till Wollenberg. Supporting the visual analysis of dynamic networks by clustering associated temporal attributes. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2267–2276, 2013. doi:[10.1109/TVCG.2013.198](https://doi.org/10.1109/TVCG.2013.198).
- [HSS11] Steffen Hadlak, Hans-Jörg Schulz, and Heidrun Schumann. In situ exploration of large dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2334–2343, 2011. doi:[10.1109/TVCG.2011.213](https://doi.org/10.1109/TVCG.2011.213).
- [HTSS10] Steffen Hadlak, Christian Tominski, Hans-Jörg Schulz, and Heidrun Schumann. Visualization of attributed hierarchical structures in a spatio-temporal context. *International Journal of Geographical Information Science*, 24(10):1497–1513, 2010. doi:[10.1080/13658816.2010.510840](https://doi.org/10.1080/13658816.2010.510840).
- [SHS11a] Hans-Jörg Schulz, Steffen Hadlak, and Heidrun Schumann. The design space of implicit hierarchy visualization : A survey. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):393 – 411, 2011. doi:[10.1109/TVCG.2010.79](https://doi.org/10.1109/TVCG.2010.79).
- [SHS11b] Hans-Jörg Schulz, Steffen Hadlak, and Heidrun Schumann. Point-based visualization for large hierarchies. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):598 – 611, 2011. doi:[10.1109/TVCG.2010.89](https://doi.org/10.1109/TVCG.2010.89).
- [SHS13] Hans-Jörg Schulz, Steffen Hadlak, and Heidrun Schumann. A visualization approach for cross-level exploration of spatiotemporal data. In *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*, pages 2:1–2:8, 2013. doi:[10.1145/2494188.2494199](https://doi.org/10.1145/2494188.2494199).

Thesis Statements

1. The visual analysis of graphs is an important endeavor in many fields. It often includes a very diverse set of aspects such as the graph structure, associated attributes as well as a spatial and temporal context to be considered. The size of these graphs, be it a high number of nodes, edges or time points, further complicates their analysis.
2. The literature provides a wide variety of very different visualization techniques to deal with the diversity and the size of these graphs. Each visualization provides its very own focus on the data and in this way supports different analysis goals. Yet, still not all aspects have been sufficiently addressed in their entirety.
3. An analysis session often consists of multiple analysis steps, each with a possibly rather different focus on the data. Hence, for its facilitation more than a single visualization is needed. Thus to support a free and flexible visual analysis these visualizations have to be adequately combined.
4. To deal with the **diversity** of the graphs in its entirety more general approaches are necessary that do not only provide a single, limited but a multitude of similar, yet differently suited visualizations. These approaches can better adapt to various circumstances considering the different graph aspects.
 - a) One basis for a more general approach is the utilization of a family of visualizations all following a similar algorithm. An example is the family of point-based tree layouts that are based on the same layout algorithm but use different layout parameters. An appropriate visualization can be chosen for a given hierarchy by selecting the right parameter presets.
 - b) Alternatively, a visualization design space can be derived by identifying common design decisions. As shown for a design space of implicit tree visualizations, it allows the creation of completely new visualizations. But it also enables the adaptation of a given visualization to a particular hierarchy by modifying some of the design decisions.
5. For providing a **scalable** visual analysis approach that is able to cope with the increasing graph size generally abstraction and selection techniques are used to reduce the amount of data prior to their visualization. While there are various reduction techniques for static graphs, their application for dynamic graphs is very limited. Hence, novel techniques especially for the abstraction and selection of dynamic graphs are needed.

- a) For abstracting a dynamic graph, either the structural or the temporal aspect can be reduced while utilizing the information contained in the respectively other aspect. In this way, temporal relationships between nodes and edges can be identified on the one hand by clustering the graph's structure on the basis of the temporal development of the individual nodes and edges. And on the other hand, recurring temporal patterns can be determined by clustering the graph's dynamics concerning the configuration of the graph structure for all time points.
 - b) To select interesting subsets within a dynamic graph, the definition of a user's degree-of-interest can be generalized to dynamic graphs. Therefore, common terms of existing DoI functions for static graphs can be modularized and additional components can be devised to address the graph's dynamics. Such a modular DoI function definition then allows the reproduction of existing DoI functions but also the adaptation of a given function to address different analysis goals.
6. Enabling a thorough analysis demands for a **flexible** switch between visualizations as the user usually cannot decide for a single visualization once and for all. For such a flexible switch a tight linking and synchronization of the different visualizations is necessary to maintain the user's mental map of the graph. While there are common strategies such as multiple coordinated views to combine visualizations, especially concerning the graph structure their linking is still an open research question.
- a) A flexible and in situ switch of a visualization can be provided by extending a portal-based approach. These extensions specifically target the maintenance of important connections between the base visualization and embedded visualizations. In this way, the user can locally adapt a given visualization to his needs while supporting his mental map.
 - b) For an even tighter linking of visualizations it is not only necessary to maintain important connections between them but also to reflect what data actually exists and is shown in those visualizations. Here, an abstract visualization explicitly reflecting all aspects of the graph and their interconnections serves as a compact overview of the graph as well as a synchronization mechanism for multiple detail views.
7. All of the novel visualization approaches have been applied to real world use cases concerning for instance internet classifications, communication networks or train connection networks. Their application allows not only the investigation of their practicability but also the determination of interesting insights.

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Promotionsarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Ich versichere weiterhin, dass ich bisher weder die vorliegende Dissertation noch Teile von ihr als Prüfungsarbeit oder zum Zweck der Promotion eingereicht bzw. verwendet habe.

Rostock, 24. Juni 2014

Steffen Hadlak