Visualizing Large Dynamic Digraphs

Michael Burch

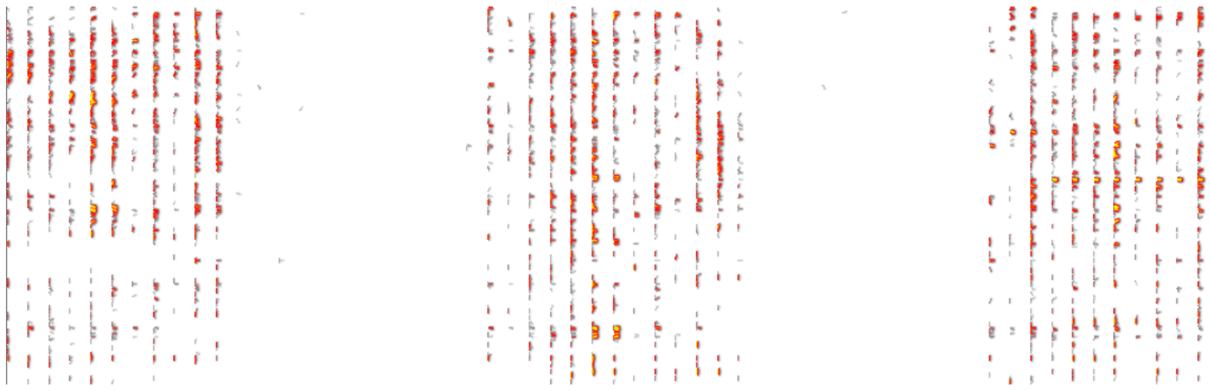

Fig. 1. Visualization for a time-varying social network acquired during 3 days of the Hypertext 2009 conference on a per hour basis: 113 of vertices, 20,818 edges, and 57 time steps are displayed.

Abstract—In this paper we investigate the problem of visually representing large dynamic directed graphs with many vertices, edges, and time steps. With this work we do not primarily focus on graph details but more on achieving an overview about long graph sequences with the major focus to be scalable in vertex, edge, and time dimensions. To reach this goal we first map each graph to a bipartite layout with vertices in the same order for each graph supporting a preservation of the viewers' mental map. A sequence of graphs is placed in a left-to-right reading direction. To further reduce link crossings we draw partial links with user-definable lengths and finally apply edge splatting as a concept to emphasize graph structures by color coding the generated density fields. Time-varying visual patterns can be recognized by inspecting the changes in the color coding in certain regions in the display. We illustrate the usefulness of the approach in two case studies investigating call graph sequences changing during software development with 21 releases which is a rather short graph sequence but contains several thousand vertices and edges. Visual scalability in the time dimension is shown in a data set with more than 1,000 graphs from a dynamic social network data set consisting of face-to-face conversation relations acquired during the Hypertext 2009 conference recorded by RFID badges.

Index Terms—Dynamic graph visualization, partial links, edge splatting

1 Introduction

Dynamic graphs appear in many fields of application, for example call graphs in software engineering, metabolic networks in bioinformatics, face-to-face contacts in social networking, or co-author relationships in digital bibliographies. The visualization of such data remains challenging due to the many data dimensions to be visually encoded. Graph structures are composed of vertices with labels and attributes, edges with weights and directions, and the time dimension which can be inherent in all of the single graph features.

Traditional dynamic graph visualization approaches either deal with the time dimension by applying a time-to-time mapping (animation) or by a time-to-space mapping (static display) as surveyed in the work of Beck et al. [2]. Graph animation typically uses node-link diagrams which are then smoothly animated from one diagram to the next by using as little layout changes as possible and achieve a dynamic stability during the animation process. The preservation of the mental map [1] is a very important aspect in this visualization strategy to keep the dynamic graph readable and explorable for the human viewer.

In our work we rely on a static representation (apart from interaction techniques) which can be used to display long graph sequences consisting of single static directed graphs. Such a representation can serve as a good overview about long graph evolution and relies on the efficient pattern recognition of the human's visual system [16].

For each graph in the time-to-space mapping we exploit the traditional concept of node-link diagrams introduced in early work of Euler [7]. Although node-link diagrams are intuitive and useful for solving path-related tasks [9], they lead to visual clutter [13], in particular when the graphs become large and dense. Due to the fact that we use a bipartite graph layout which maps graph vertices to one-dimensional vertical lines, the visual clutter phenomenon is getting worse. In our approach we mitigate this situation by using the concept of edge splatting [4] of partially drawn links [5] which has benefits concerning two aspects: It first improves visual scalability since the reduced link lengths allow to show more graphs on the same display space and second the color coded peaks of the density fields still give a hint about the target of a directed graph edge. The focus of this concept is not on seeing and identifying graph details but more on the exploration of longer graph sequences with respect to structural changes.

We apply our interactive tool to a dynamic graph dataset from the field of software engineering with dynamic call relations consisting of thousands of vertices and edges but only 21 time steps. Moreover, to test the visual scalability issue in the time dimension, we show a social network evolution describing face-to-face contacts during the 2009 Hypertext conference with more than 1,000 time steps. In both scenarios we can easily see that the displayed dynamic graph still reveals interesting time-varying patterns while simultaneously scaling to long graph sequences that may also contain dense graphs.

2 Related Work

Various techniques have been designed for visualizing dynamic graphs as surveyed by Beck et al. [2]. Two general trends can be found, i.e., time-to-time mappings which make use of graph animation and, on the other hand, time-to-space mappings which are static displays of dynamic graph data.

Static diagrams have benefits when graphs have to be compared since such a comparison can be done directly by inspecting the graphs on screen. This is difficult for animated graphs since there, only one graph at a time is shown and is smoothly animated into the next one in the sequence [6, 8]. The preservation of the mental map [1] which is typically achieved by having a high degree of dynamic stability is one of the key concepts to obtain a better graph animation for solving graph exploration tasks [12]. But displaying longer graph sequences with the goal to derive time-varying patterns is a challenging, i.e., even impossible task when graph animation is applied.

We base our approach on static small multiples with graph vertices aligned to horizontal lines. This is similar to the approach of Burch et al. [4] which was inspired by the visual concept in parallel coordinates plots [11], i.e., small vertical stripes which make longer graph sequences visually scalable. Burch et al., instead, use complete links which have a higher probability to cause link crossings making a diagram unreadable which is mitigated by applying edge splatting to generate density fields for the link coverage information similar to the approach used for generating continuous parallel coordinates plots [10].

In this paper we argue that we do not need the complete link in a node-link diagram to show connectedness of objects as evaluated by Burch et al. [5] and mathematically modeled as partial edge drawing by Bruckdorfer et al. [3]. In our approach we use those partially drawn links which make the diagrams more visually scalable in the time dimension since they do not require the full link length and consequently the full display space to show an individual graph. But to be still able to identify hot spots and edge directions, i.e., graph patterns, we also apply edge splatting to make them more readable, in particular, when the graphs are dense and time-varying. This edge splatting concept applied to partial edges allows to display longer graph sequences while they still allow to identify changing graph patterns.

3 Dynamic Graph Visualization

In many applications graph data has a dynamic nature and consists of very many timesteps. Moreover, if the graphs to be visualized are large and dense, i.e., contain many edges compared to the vertices, visualization can result in hairball-like representations when node-link diagrams are used or, on the other hand, layout algorithms are very time-consuming leading to non-interactive visualizations.

In our work we try to give an overview about long, large, and dense graph sequences with the goal to support an analyst to see time-varying visual patterns, to filter the data for specific patterns, and finally get insights in the dynamic graph which would otherwise be hard to find. We base our work on the Visual Information Seeking Mantra by Shneiderman [14] which states overview first, zoom and filter, then details on demand.

3.1 Data Model and Layout

In the context of this paper a directed graph is modeled as

$$G = (V, E)$$

where V denotes the set of vertices and

$$E \subseteq V \times V$$

expresses the set of directed edges. An edge

$$(u, v) \in E$$

expresses that the edge starts at vertex $u \in V$ and ends at vertex $v \in V$.

A weighted graph is one with quantitative values attached to each edge, i.e., a function

$$f : E \longrightarrow \mathbb{R}$$

maps a value to each of the edges.

A dynamic graph is modeled as a sequence of $n \in \mathbb{N}$ single graphs

$$\Gamma := (G_1, \ldots, G_n).$$

Each single graph might be attached by a time stamp generating the order of the graphs.

We define a layout of a graph as a function

$$L : V \longrightarrow \mathbb{N} \times \mathbb{N}$$

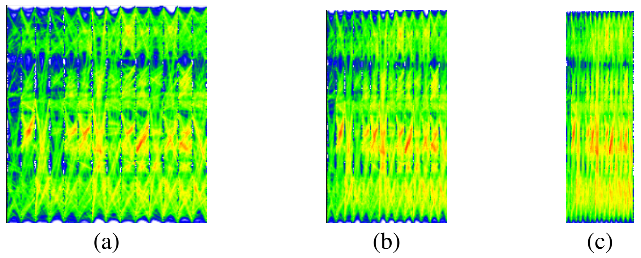

Fig. 2. Rendering a sequence of splatted graphs on changing display space: 14 graphs in a bipartite layout on (a) full screen, (b) 50 percent screen, and (c) 25 percent screen. 50 percent partial links are used to reduce the amount of display space to render the graphs. Time-varying patterns can be detected in all of the representations.

which maps graph vertices to two-dimensional positions on the display. In our work we use a bipartite layout, i.e., at first the vertex set is copied. The original vertices are mapped equidistantly to one-dimensional vertical lines where each vertex of the copied set is mapped to the same vertical position of its corresponding original vertex but with a certain offset. Links are drawn in-between from left-to-right. A dynamic graph is mapped to a sequence of such vertical stripes. This concept is already successfully applied in the work of Burch et al. [4].

In this work we are merely interested in visualizing longer graph sequences in which the individual graphs which can be large, dense, directed, and weighted. This means our intent is to give an overview about time-varying visual patterns in the evolution of large dynamic digraphs which can then be further analyzed by applying interaction techniques such as filtering and details on demand. We argue that in many dynamic graph visualization scenarios an overview is missing. The viewer definitely needs a starting point for further explorations, i.e, most of the data should be displayed at once [15].

3.2 Graph Sequence Representation

A dynamic graph, i.e., a sequence of individual graphs, is mapped in a left-to-right reading direction with the individual graphs mapped to small vertical stripes. The vertex positions on the vertical lines remain the same since we are focusing on the preservation of the mental map which is crucial in our approach to derive time-varying graph patterns.

As a novelty to the traditional edge splatting technique proposed by Burch et al. we apply the concept of partially drawn links allowing to win some display space for showing more graphs from the complete graph sequence. The partially drawn links do not show the complete link information but they still give a hint about the starting point of an edge and the direction in which it is pointing. Computing a density field from all those partial links and use color coding to visually encode the hot spots in the diagram supports to give an overview about the evolution of graph data.

Figure 2 shows the scalability of the technique in the time dimension. The graph sequence in (a) is displayed at a smaller display area in (b) and (c). The time-varying visual patterns are still recognizable. Edges are drawn partially but still 50 percent link lengths are used. In the case studies in Section 4 we use even shorter partial links which allow to display many more graphs next to each other while the changing graph structures can still be visually explored.

3.3 Interaction Techniques

Our visualization tool supports several interaction techniques. The most important ones are listed below:

- **Partial link length:** The link lengths can be varied by changing a corresponding percentage value. Each link is displayed in the requested length while it is simultaneously splatted to achieve a useful visualization technique for the graph evolution.

- **Graph aggregation:** Graphs can be aggregated on different levels of time granularity. Time intervals can be selected and the

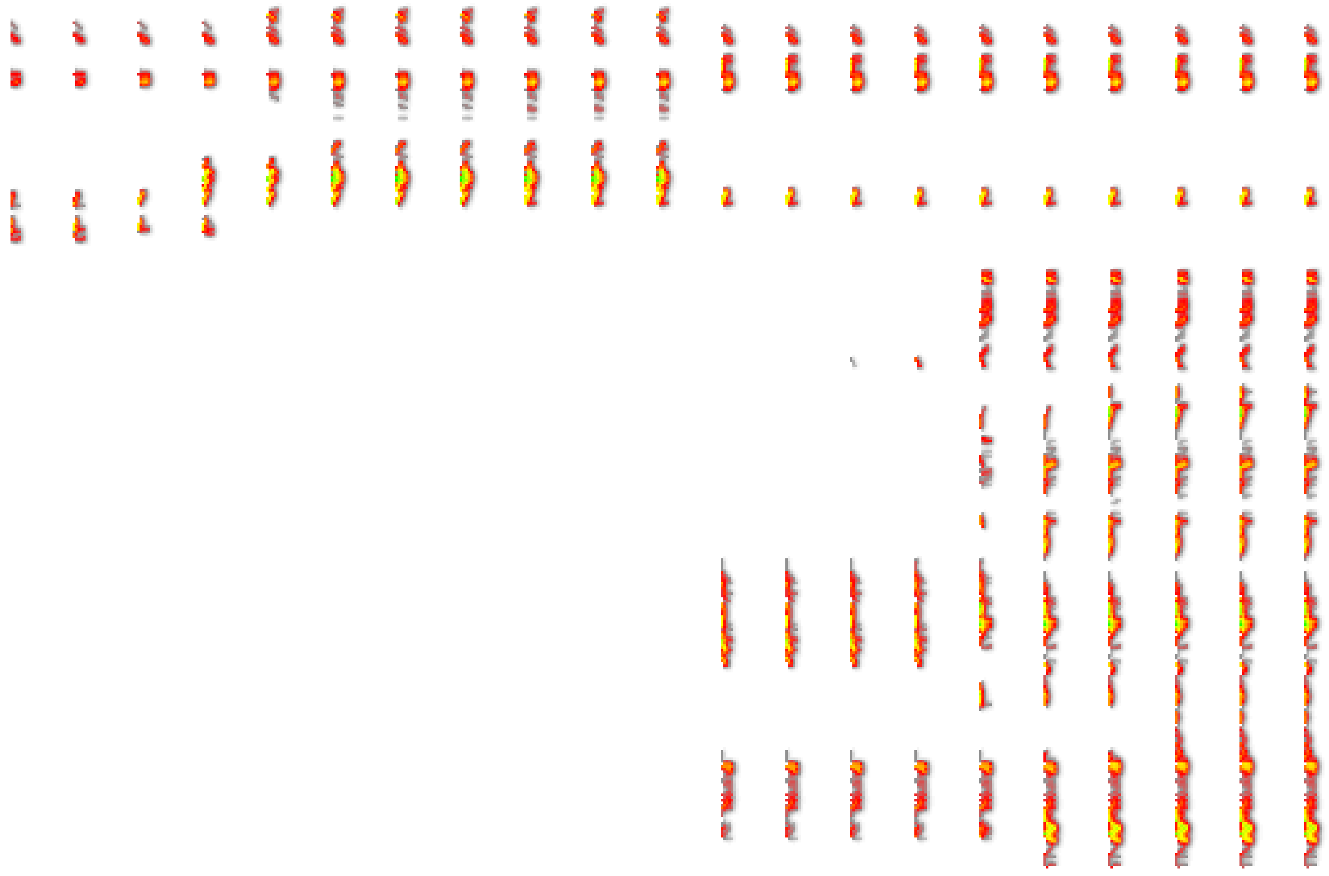

Fig. 3. The JUnit open source software project with its 21 releases can be explored on a call graph perspective. We can see the call graph changes although the links are only drawn partially. The graph contains 2,817 vertices (program methods) and 15,339 directed edges (method calls).

contained graphs can be merged into one. A similar approach works for the set of vertices. Neighbored ones on the vertical lines can be aggregated into individual merged nodes. The edges are summed up accordingly.

- **Color coding:** The applied color coding plays also a crucial role in this visualization technique since the generated hot spots in the splatted graphs can make smaller value difference disappear. Consequently, a logarithmic color coding might be applied.

- **Details-on-demand:** Clicking on an individual graph can show it in a different view and in a different layout. This allows the analyst to explore the relational data for details. Also textual information in the form of text labels are difficult to be displayed but in a detail view such additional information might be displayable at least to some extent.

- **Filtering:** The graph sequence can be filtered for vertices, edges, or weights. The dynamic graphs can also be filtered for time intervals which might also be aggregated allowing to display the remaining graphs to larger regions.

4 Case Studies

We applied our dynamic graph visualization with splatted partial links to two scenarios from software development and from social networking. These scenarios are different since the dynamic call graph only consists of 21 time steps whereas the dynamic social network is a longer graph sequence with more than 1,000 graphs. Visually representing such a long sequence of relational information on one screen is challenging.

4.1 Dynamic Call Relations

The dynamic call relations already contain some kind of structure given by the hierarchical organization of the software functions/methods which build the vertices of the graphs. Mapping the vertices in a hierarchy-preserving linear order reduces the amount of link crossings (if the software is well-structured).

By preserving this hierarchical organization we obtain the diagram shown in Figure 3 which shows the 21 releases of the JUnit open source software project. In this figure we can directly see that the graph structure is changing over time which is due to changes during the software development. Source code and functionality is added introducing many more links. But several links are also disappearing during the evolution meaning that the functionality is either put to different modules or that it is completely removed from the project. Understanding and analyzing such details during software development requires to have a look inside the source code.

4.2 Dynamic Social Network

The actors in social networks are typically unstructured, i.e., every person is initially treated in a similar way. Hierarchical clustering can be used to derive a structure among the people which can be used to order the vertices of the underlying graph. If this procedure was not applied the resulting visualization would suffer from vast amounts of visual clutter. For the dynamic social network dataset we first compute an adjacency matrix for all conference attendees, use the face-to-face contacts as relational data, and finally apply a hierarchical clustering on the time-aggregated adjacency matrix.

The face-to-face contacts recorded during a conference at 20 seconds intervals were leading to a long sequence of social networks. Using the formerly computed hierachical information to map the vertices to the one-dimensional lines gives some structure among the dynamic

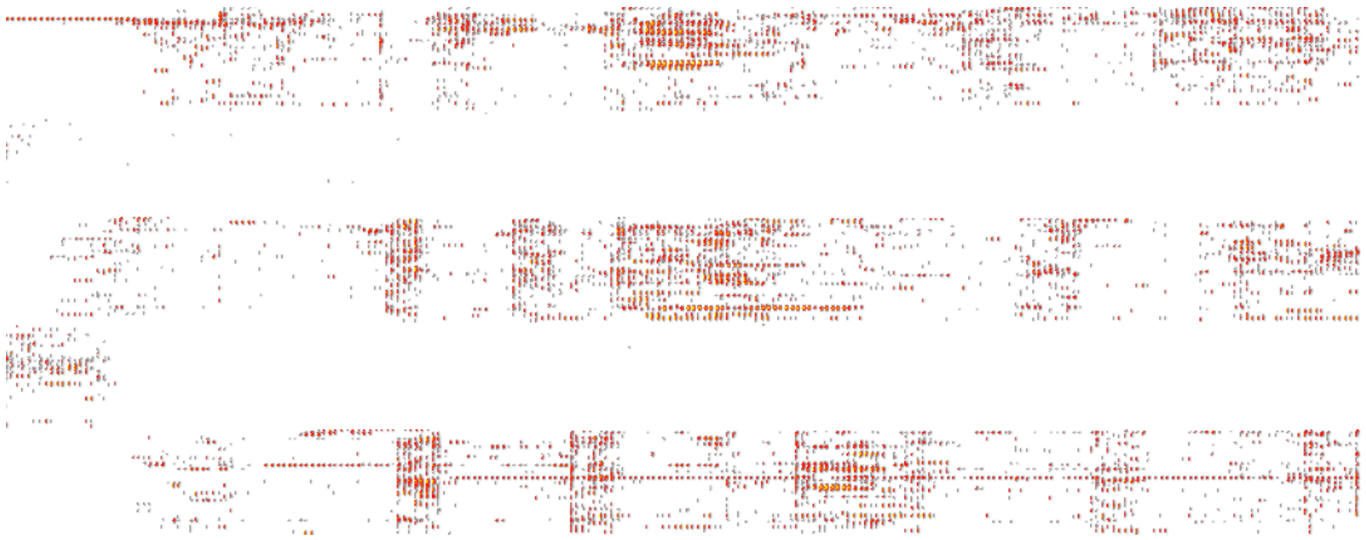

Fig. 4. Face-to-face contacts during three conference days displayed on a per 3 minutes basis. The original data from Figure 1 with 113 vertices (conference attendees) and 20,818 edges is now displayed on a more fine granular time scale with more than 1,000 graphs. We can see that all graphs are displayable on screen without scrolling interaction, Rapid Serial Visual Presentaton (RSVP), or graph animation in general.

graph data. If we are intending to analyze this time-varying network we first need an overview about the dynamic patterns. The dynamic data might be aggregated to hourly intervals shown in the diagram in Figure 1. In this visualization we can see that there are three time clusters being the conference days. People are attending the conference and are talking to each other. The empty gaps in the display visually encode the face-to-face contacts over night. In these time periods people are not talking to each other. The hierarchical clustering unhides some smaller groups of people which are talking to each other more frequently than others and also aover longer time periods.

The same data is shown in Figure 4 on a per 3 minute basis. Here, we can easily detect the daily patterns again but now we can also derive more fine-granular time-varying graph patterns of more than 1,000 social networks. In the more fine granular data representation time clusters can be found also during the conference days. These refer to the coffee breaks during the conference where only small groups of conference attendees are talking to each other.

5 Conclusion and Future Work

In this paper we illustrated how long graph sequences can be displayed with the goal to give an overview about time-varying patterns. To achieve a more scalable dynamic graph visualization we applied the concept of partially drawn links which are still useful to show the rough directions of the graph edges. To obtain visual graph structures those partial links are splatted which results in time-varying density fields. The densities are color coded leading to a sequence of small vertical stripes with visual patterns. Consequently, the approach can easily be used to understand an evolving graph focusing on the extraction of time-varying graph patterns. Simple interaction techniques are integrated such as the adjustment of the partial link lengths and the applied color coding.

For future work we plan to test our technique with even longer graph sequences from other fields of application. A user study should be conducted with the goal to find out if people are able to explore dynamic graphs by using this visualization technique.

References

[1] D. Archambault, H. C. Purchase, and B. 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.

[2] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. The state of the art in visualizing dynamic graphs. In *EuroVis State-of-the-Art Reports*, EuroVis STAR, 2014.

[3] T. Bruckdorfer, S. Cornelsen, C. Gutwenger, M. Kaufmann, F. Montecchiani, M. Nöllenburg, and A. Wolff. Progress on partial edge drawings. In *Proceedings of International Symposium on Graph Drawing*, pages 67–78, 2012.

[4] M. Burch, C. Vehlow, F. Beck, S. Diehl, and D. Weiskopf. Parallel edge splatting for scalable dynamic graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2344–2353, 2011.

[5] M. Burch, C. Vehlow, N. Konevtsova, and D. Weiskopf. Evaluating partially drawn links for directed graph edges. In *Proc. of Graph Drawing*, pages 226–237, 2011.

[6] S. Diehl and C. Görg. Graphs, they are changing. In *Proceedings of 10th International Symposium on Graph Drawing*, pages 23–30, 2002.

[7] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Petropolitanae*, 8:128–140, 1741.

[8] Y. Frishman and A. Tal. Dynamic drawing of clustered graphs. In *Proceedings of 10th IEEE Symposium on Information Visualization*, pages 191–198, 2004.

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

[10] J. Heinrich and D. Weiskopf. Continuous parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1531–1538, 2009.

[11] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *IEEE Visualization*, pages 361–378, 1990.

[12] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proc. of the AVI Workshop on BEyond time and errors: novel evaluation methods for information visualization*, pages 1–5, 2006.

[13] R. Rosenholtz, Y. Li, J. Mansfield, and Z. Jin. Feature congestion: a measure of display clutter. In *Proceedings of Conference on Human Factors in Computing Systems*, pages 761–770, 2005.

[14] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of Visual Languages*, pages 336–343, 1996.

[15] E. R. Tufte. *The visual display of quantitative information*. Graphics Press, 1992.

[16] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2004.