Eurographics/ IEEE-VGTC Symposium on Visualization 2009
H.-C. Hege, I. Hotz, and T. Munzner
(Guest Editors)

Volume 28 (2009), Number 3

Visualizing the Evolution of Compound Digraphs with TimeArcTrees

M. Greilich[1], M. Burch[1] and S. Diehl[1]
[1]University of Trier, Germany

Abstract

Compound digraphs are a widely used model in computer science. In many application domains these models evolve over time. Only few approaches to visualize such dynamic compound digraphs exist and mostly use animation to show the dynamics. In this paper we present a new visualization tool called TimeArcTrees that visualizes weighted, dynamic compound digraphs by drawing a sequence of node-link diagrams in a single view. Compactness is achieved by aligning the nodes of a graph vertically. Edge crossings are reduced by drawing upward and downward edges separately as colored arcs. Horizontal alignment of the instances of the same node in different graphs facilitates comparison of the graphs in the sequence. Many interaction techniques allow to explore the given graphs. Smooth animation supports the user to better track the transitions between views and to preserve his or her mental map. We illustrate the usefulness of the tool by looking at the particular problem of how shortest paths evolve over time. To this end, we applied the system to an evolving graph representing the German Autobahn and its traffic jams.

Categories and Subject Descriptors (according to ACM CCS): Data Structures [E.1]: Graphs and Networks—

1. Introduction

Graphs are a mathematical method to express relations or dependencies between a set of objects. Compound graphs additionally encapsulate a hierarchical order on the graph nodes. Many layout algorithms have been developed to better understand interesting structures or substructures of a given compound graph by producing node-link diagrams. In many cases these algorithms can only deal with static graphs that do not change their structure over time. Only few algorithms can deal with changing relations [PB08].

Information hierarchies occur in many application domains such as the hierarchical organization of companies, news topics and subtopics, file/directory systems, products and product groups of a department store, or phylogenetic trees in biology. The evolution of dependencies in such information hierarchies can be modeled by sequences of compound digraphs with edge weights.

While there has been a lot of work on visualizing information hierarchies [RT81, YFDH01, JS91, AH98, SZ00], only few researchers have developed methods to visualize dependencies between elements in the hierarchy [NSC05, FWD*03, ZMC05, Hol06, PvW06], see Section 4 for a more detailed discussion.

The TimeArcTrees approach shows a sequence of compound digraphs in a traditional node-link representation. Based on the order of the nodes in the hierarchy the nodes of the graph are placed on a vertical line from top to bottom for each graph of the sequence. The hierarchy is represented as a node-link diagram on the left hand side of the whole view and can be expanded or collapsed to an interactively selectable level. The visualization of the graph sequence automatically adapts to the current expansion level of the hierarchy. The frontier of the current hierarchy tree forms the nodes of the graph sequence. By frontier we mean the currently visible leaf nodes of the hierarchy which are either real leaf nodes or collapsed nodes of the underlying hierarchy. The mental map is preserved by smooth animation making the transitions between different views easier to understand.

The graphs of the sequence are drawn from left to right as separate node-link diagrams. In these diagrams we use the following conventions. First, all nodes of a graph are aligned vertically above each other. Edges are not drawn as straight lines with bends, but as arcs to make it easier for the human eye to follow a path. Furthermore, we show three kinds of edges. For *upward edges* the starting node is located below the target node in the visualization, whereas the remaining edges are either *downward edges* or *self-edges*. Upward edges as well as self-edges are shown on the left hand side of the corresponding graph. All other edges are visualized on the right hand side of each graph. This layout reduces visual clutter but does not avoid it completely.

Moreover edges can be weighted. These weights are indicated by color using one of many color scales provided by the system.

We have developed several interaction techniques that allow the users to explore the data. On top of each graph one can find a time slider that is used to aggregate a graph subsequence, for example. These and other features will be explained in more detail later on.

The rest of this paper is organized as follows: In Section 2 we introduce the different constituents of the TimeArcTrees visualization by means of examples. In Section 3 we show how TimeArcTrees can be used to explore shortest paths in a sequence of graphs. Related work is discussed in Section 4. Finally, Section 5 gives some conclusions and possible directions for future work.

2. TimeArcTrees – step by step

We illustrate our visualization technique by means of a small example. Assume the graph sequence $S = (G_1, \ldots, G_k)$ that consists of k graphs $G_i = (V, E_i)$, $1 \leq i \leq k$ with V the set of nodes and $E_i \subseteq V \times V \times \mathbb{N}$ the set of directed weighted edges between pairs of nodes. In the context of this paper we denote an edge $(u, v, w) \in E_i$ of graph G_i with

$$u \underset{G_i}{\overset{w}{\rightarrow}} v$$

We call w the edge weight. A path

$$u \underset{G_i}{\overset{c}{\rightarrow}}{}^{*} v$$

is a possibly empty sequence of edges such that

$$u \underset{G_i}{\overset{w_1}{\rightarrow}} v_1 \underset{G_i}{\overset{w_2}{\rightarrow}} \ldots \underset{G_i}{\overset{w_{n-1}}{\rightarrow}} v_{n-1} \underset{G_i}{\overset{w_n}{\rightarrow}} v$$

and $c = \sum_{j=1}^{n} w_j$ are the accumulated costs along the path.

A cycle or loop is a nonempty path

$$u \underset{G_i}{\overset{c}{\rightarrow}}{}^{*} u$$

.

In the following example we assume that the nodes of the graph sequence correspond to IP-addresses and the edge weights represent network delays. The graph can be obtained by merging the information of multiple invocations of the `traceroute` program. Table 1 shows the nodes of the graph sequence with a short labeling, namely $A, \ldots, F$, the corresponding IP-addresses, and the WWW domains.

Node	IP-address	Domain
A	136.199.55.209	www.st.uni-trier.de
B	136.199.199.105	www.uni-trier.de
C	136.199.55.175	dbis.uni-trier.de
D	134.96.7.179	www.uni-sb.de
E	134.93.178.2	www.uni-mainz.de
F	131.246.120.51	www.uni-kaiserslautern.de

Table 1: *Short notation for IP-addresses and their WWW domains.*

2.1. First Step - A Single Graph

We start with this small graph example in Figure 1. The graph consists of 6 nodes and 10 weighted directed edges that are annotated with the corresponding edge weights. Edges are visualized as straight lines pointing from the start node to the destination node.

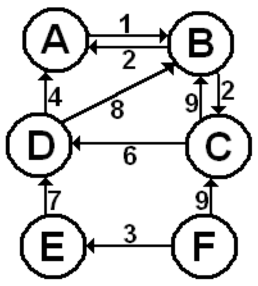


Figure 1: *Node-link diagram of a directed graph with weighted edges. Each edge is annotated with its weight.*

Figure 2 depicts the same graph using the TimeArcTrees visualization approach. The graph nodes are vertically aligned and edges are visualized as colored arcs. The more reddish the color of the edge, the higher is its weight. In contrast, green indicates low edge weights.

Upward edges and self-edges are always drawn on the left hand side of the graph, downward edges are placed on the right hand side of the graph. Furthermore, the horizontal distance of an edge from the vertical axis formed by the nodes increases with the number of nodes which are located between the start and the target node of the edge. With this edge placement we intend to reduce edge crossings and visual clutter. For the same reason, all upward edges that are ending in the same node share a single arrow head, as shown in Figure 3. The same holds for all downward edges.

Edge crossings can be further reduced by reordering the nodes. Finding the right node order can be reduced to

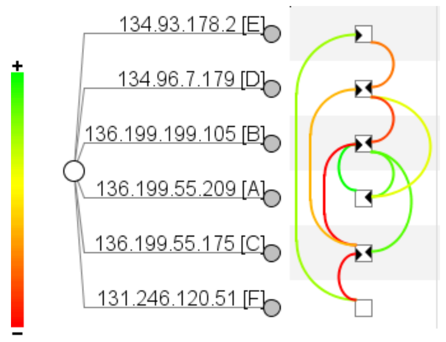


Figure 2: *Node-link diagram of a directed weighted graph in a TimeArcTrees representation. The weighted edges are represented by colored arcs.*

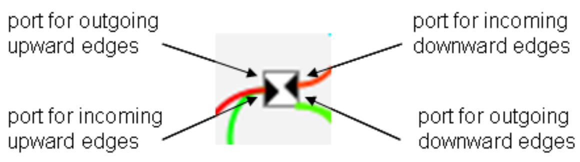

port for outgoing upward edges
port for incoming downward edges
port for incoming upward edges
port for outgoing downward edges

Figure 3: *Incoming and outgoing edge ports*

the *NP*-complete problem of Optimal Linear Arrangement (OLA) [GJ79]. As a consequence, we first reorder the leaf nodes to reduce the edge crossings between sibling nodes, then we recurse to the parent nodes and reorder these to further reduce the edge crossings.

2.2. Second Step - Hierarchy Levels

As we have to deal with compound digraphs we have to visualize the hierarchical ordering of the graph nodes as well. Figure 4 shows how the information hierarchy could be represented. We also use a traditional node-link diagram that is located on the left side of the whole graph view to give an overview of the hierarchical information of the compound digraph. Subtrees can be collapsed into a single node or expanded. The size of a collapsed subtree is indicated by the color of its root node. Large subtrees are represented by white hierarchy nodes, whereas small subtrees are visualized as dark blue nodes.

The hierarchy can be transformed by clicking on a node. If the subtree of the node is expanded, it will be collapsed, and vice versa. The graph is automatically updated to show only nodes of the current frontier of the hierarchy. Smooth animations to get from the old view to the new one help the user to keep track of what is going on and to preserve his or her mental map of the graphs.

2.3. Third Step - Aggregation of Edges

If the hierarchy is transformed by clicking on an intermediate node, the graph is adjusted as shown in Figure 5. The corresponding nodes of a collapsed subtree are replaced by a single node and all edges to and from nodes within the subtree are shown as edges to and from the new node. If there are several edges to or from the same node, they are only represented by a single edge. The weight of this edge is either computed as the sum or average of the individual edge weights depending on the user's preferences.

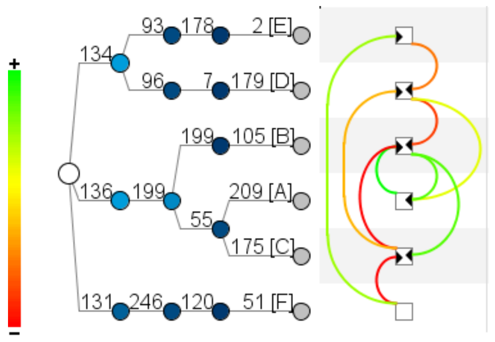


Figure 4: *A graph in a TimeArcTrees representation: The hierarchy is shown on the left hand side. In this example, the hierarchy is completely expanded.*

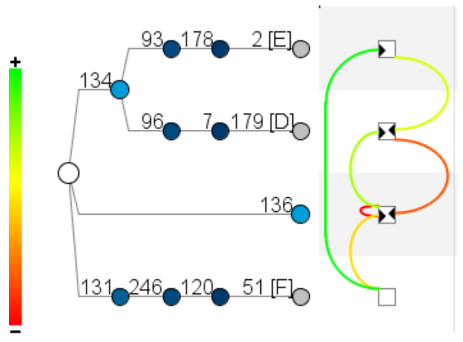


Figure 5: *Parts of the hierarchy can be expanded or collapsed to an interactively selectable level. In this example, the complete subtree with root 136 is collapsed and all incoming, outgoing and inner edges of this subtree are aggregated.*

For example, the edge from D to 136 in Figure 5 aggregates the edge from D to B and the edge from D to A in Figure 4 and its color indicates the sum of the individual edge weights.

2.4. Fourth Step - Graph Sequence

The TimeArcTrees tool is at its best if we have to deal with sequences of graphs. Dependencies and relations are changing over time. The graph structure could be a completely new one or just subgraphs are evolving. The node-link diagrams in Figure 6 show a sequence of small graphs. For example, the edge $D \rightarrow B$ in the first graph disappears in the second graph and reappears in the third one, but with a smaller weight than before. The node *E* in the last graph has neither incoming nor outgoing edges and thus is not shown at all.

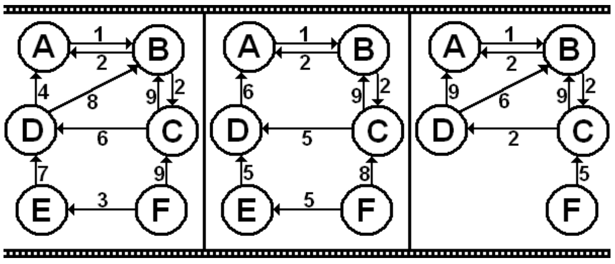


Figure 6: *A sequence of graphs in a traditional node-link representation.*

The same scenario as in Figure 6 is represented in Figure 7, but here we use our new TimeArcTrees approach. Edges are colored and the hierarchy is added on the left side of the graph sequence view.

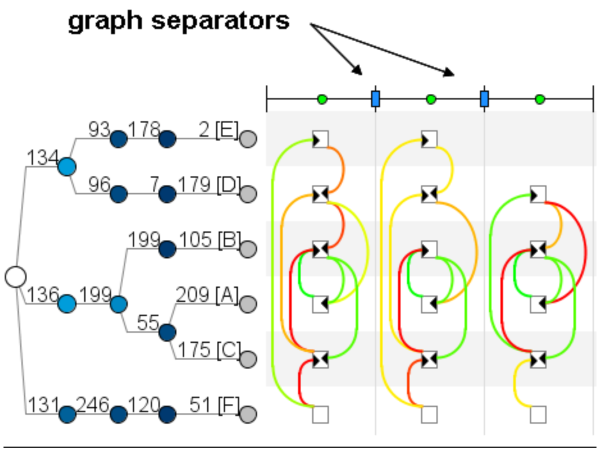


Figure 7: *The TimeArcTrees visualization for a sequence of three graphs with the information hierarchy. Graph separators are interactive widgets that can be used to change the horizontal space for each graph, as well as to aggregate several subsequent graphs.*

2.5. Fifth Step - Aggregation of Graphs over Time

Subsequent graphs can be aggregated into a single graph. In the example shown in Figure 8 the second and third graph of Figure 7 are aggregated. To indicate this, there is no graph separator between the two bullets representing the two original graphs.

In the aggregated graph the edge from E to D stems from the second graph and the edge from D to B from the third graph. The color of the edge from F to C represents the average of the weights of the same edge in both original graphs. Alternatively, the TimeArcTrees tool can use the sum of the

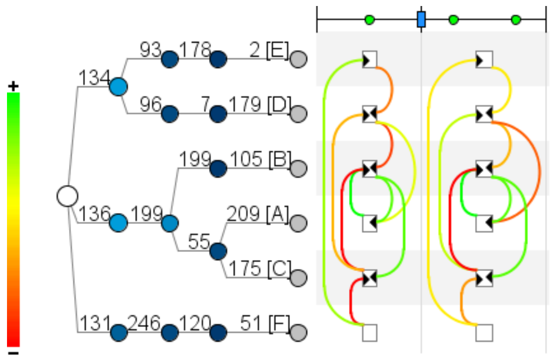


Figure 8: *The second and third graph are aggregated to a single graph. The graph separator between the two original graphs was removed.*

weights for aggregated edges. It goes without saying that TimeArcTrees also allows to expand aggregated graphs by reinserting graph separators.

2.6. Interactive Features

The TimeArcTrees tool has many additional interactive features that help the user to explore the given graphs. In this section we will explain some of them in more detail.

Expanding/Collapsing of the Hierarchy
The information hierarchy can be expanded or collapsed by clicking on the corresponding node. This has the positive effect that subgraphs can be combined to one single node and the graphs become much smaller.

Graph Aggregation
Subsequent graphs can be aggregated by dissolving the graph separator between two or more subsequent graphs. The edge weights of the aggregated edges are either computed as the sum or the average of the weights of the individual edges.

Node Activation and Deactivation
Nodes in the hierarchy can be activated and deactivated. If a node is deactivated only those edges are visible that are not incoming or outgoing edges of this deactivated node.

Filtering Functions
In order to reduce visual clutter, edges can be filtered out by applying a threshold to their edge weights. In other words, if the weight of an edge is below a given threshold, it is not drawn on the screen.

Detail on Demand
Detailed information can be obtained in form of a tooltip when moving the mouse cursor over a node or edge position.

Color Coding
The tool provides a set of predefined color scales from which the user can select an adequate one.

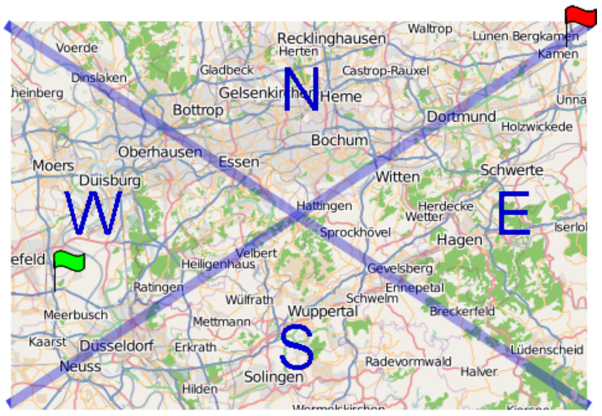
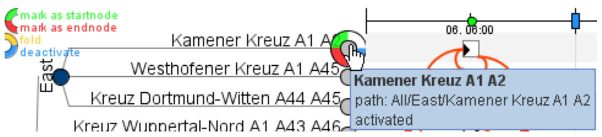


Figure 9: *Excerpt of the German Autobahn map: The map is divided into four regions. The start node for our shortest path example is indicated by a green flag. The target node by a red one.*

3. Shortest Paths

So far, we have shown how TimeArcTrees can be used to explore a sequence of graphs. In addition, the tool is able to compute shortest paths using the Bellmann-Ford algorithm. TimeArcTrees can also visualize some other graph properties like maximum flows that we do not further discuss in this paper. To this end, the user has to select start and target nodes as shown in Figure 10. By clicking at the green circle sector the node is marked as the start node, whereas by clicking at the red circle sector it is marked as the target node.



Kamener Kreuz A1 A2
Westhofener Kreuz A1 A45
Kreuz Dortmund-Witten A44 A45
Kreuz Wuppertal-Nord A1 A43 A46

Kamener Kreuz A1 A2
path: All/East/Kamener Kreuz A1 A2
activated

Figure 10: *The start node for shortest path visualization is selected by clicking at the green circle sector of a node. The target node can be selected with the red circle sector.*

A shortest path in a graph G from node u to node v is the path $u \overset{c}{\underset{G}{\rightarrow}}{}^{*} v$ with the minimal costs c. To visualize a shortest path, only those edges along the path are drawn in color, while the remaining edges are drawn in light grey. At each node along the path, the accumulated cost relative to the cost of the longest shortest path are indicated by a circular bar, as shown in Figure 11. For each graph of the sequence we may have a different shortest path. Of these shortest paths, the longest shortest path is the one with the highest cost.

To illustrate our approach, we look at the question of how the fastest way to get home on the German Autobahn changes during the day. Or more precisely, how to get from the junction "Kreuz Meerbusch" to the junction "Kamener

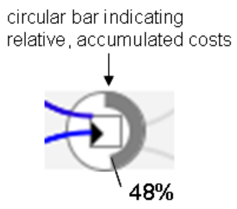


Figure 11: *The costs accumulated along the shortest path from the start node up to this node are represented by a circular bar.*

Kreuz". Figure 9 shows an excerpt of a German Autobahn map that we will consider in the following. We divided the map into four regions: South, North, West, and East.

From this map we derived an Autobahn graph consisting of 27 nodes and 78 edges, in which the nodes represent the junctions and the edge weights represent the time to get by car from one junction to another. These times change during the day due to traffic jams. In the example, we used real data recorded on a typical day.

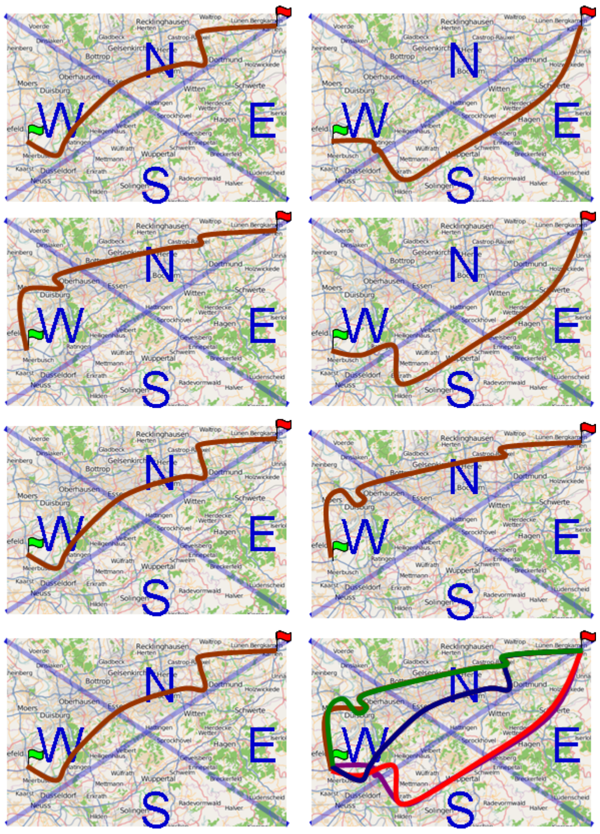


Figure 12: *A sequence of maps showing the shortest paths at different points of time. In the lower right map all paths are shown at once.*

In Figure 12 the shortest paths are drawn into the map to preserve geographic information. For each of seven different times of day a separate map is shown. Finally, the map at the lower right shows all shortest paths in a single map. Visualizing the paths using separate maps requires a lot of screen space, showing the maps one after another using animation leads to high cognitive load, and integrating the paths into a single map produces a lot of visual clutter and makes it hard to show additional information. By visualizing the Autobahn graph instead of the map, we lose the geographic information, but it allows us to apply the TimeArcTrees approach to explore the data in a space efficient and highly interactive way.

In the example in Figure 13 the shortest paths are shown for 8 different points of time of the same day. Actually, the graphs for 12:00 and 14:00 are aggregated in this example to save screen space, and because there was no difference between the shortest paths in both graphs. As can be seen by the closed circular bar around the target node in the graph for 16:00, this graph contains the longest shortest path. When moving the cursor onto the circular bar, the tooltip shows that this path takes 102 minutes. In contrast, the shortest shortest path only takes 62 minutes. Surprisingly, the longest shortest path and the shortest shortest path are along the same nodes. Whereas, most of the other shortest paths have different intermediate nodes. Thus, in this case we found out that it does not pay off to use alternative routes during the rush hour because these are also jammed.

Although we found the Autobahn example very intuitive, there is a risk that by removing the spatial information and aligning all junctions vertically, the user might misinterpret the length and direction of edges as the geographical distance and orientation.

4. Related Work

Many approaches for visualizing dependencies in information hierarchies encode dependencies between objects as directed or undirected edges in node-link diagrams. In particular, specialized graph-drawing algorithms for compound digraphs have been developed [FT04, SM91, BM99]. In these representations the appearance of edges, for example, their color, shape, orientation, thickness or connection, can represent the weights of edges. There are several approaches that extend Treemaps [JS91] to also show different kinds of graphs on top of the treemap [FWD*03, ZMC05, BD06]. For example, ArcTrees [NSC05] is an interactive visualization tool for hierarchical and non-hierarchical relations. It extends the hierarchical view of the Treemap approach with arc diagrams [Wat02] to present relations. Hierarchical Edge Bundles [Hol06] show relations by bundled edges between the nodes of a radial icicle view, treemap or balloon layout. None of these approaches is able to show the evolution of the graphs.

Several researchers have developed animated node-link representations of dynamic compound digraphs [FT07, PB08]. In contrast, the TimeRadarTrees approach [BD08] visualizes a sequence of weighted, compound digraphs in a single view. This radial and space-filling visualization technique uses colored circle sectors to represent weighted edges. TimeArcTrees can be seen as an extension of the Timeline Trees approach [BBD08] which shows transitions instead of digraphs, i.e., no edges are drawn, instead the participation in a transition is indicated by color. The TimeArcTrees tool provides many of the interaction features of the Timeline Trees tool like collapsing and expanding of tree nodes, but extends these by graph aggregation, node activation and filtering functions, see Sections 2.5 and 2.6.

Discussion While one can certainly think of possible extensions of the approaches discussed above, to the best of our knowledge none of the existing approaches has so far been extended and applied to visualize dynamic, weighted compound digraphs in a single, static image using a node-link representation. Timeline Trees [BBD08] and TimeRadarTrees [BD08] provide a compact, crossing free representation of dynamic graphs, but it is still difficult for the human eye to follow paths in these representations. TimeArcTrees are a compromise. They are a relatively compact representation of dynamic compound digraphs which allows the human eye to follow paths more easily.

As can be seen in Figure 13 up to 10 graphs with about 30 expanded or collapsed nodes and about 100 edges per graph can be displayed simultaneously without a significant loss of readability on a computer screen (1680*x*1080 resolution), i.e., a total of 30.000 edges. For the horizontal dimension there is a tradeoff between the number of edges per graph and the number of graphs which can be readably shown at the same time. Thus, the interaction features provided by the tool are very crucial for inspecting larger data sets.

5. Conclusions and Future Work

In this paper we have presented the TimeArcTrees visualization tool for exploring dynamic, weighted compound digraphs. To illustrate the usefulness of our tool and some of its interaction techniques, we focussed on how shortest paths evolve in these dynamic graphs. To this end, we applied it to a sequence of graphs representing the German Autobahn and its traffic jams at different times of the day. The compact and clear representation facilitated comparison of the graphs. The horizontal alignment of the instances of the same node in different graphs helped us to detect where the shortest paths diverged.

As part of our future work we plan to improve the system by adding an interactive street map and allowing "linking and brushing" in both the TimeArcTrees and the street map view. Furthermore, we intend to use the system to explore the evolution of other graph properties like maximum flow, strongly connected components and the like. Finally,

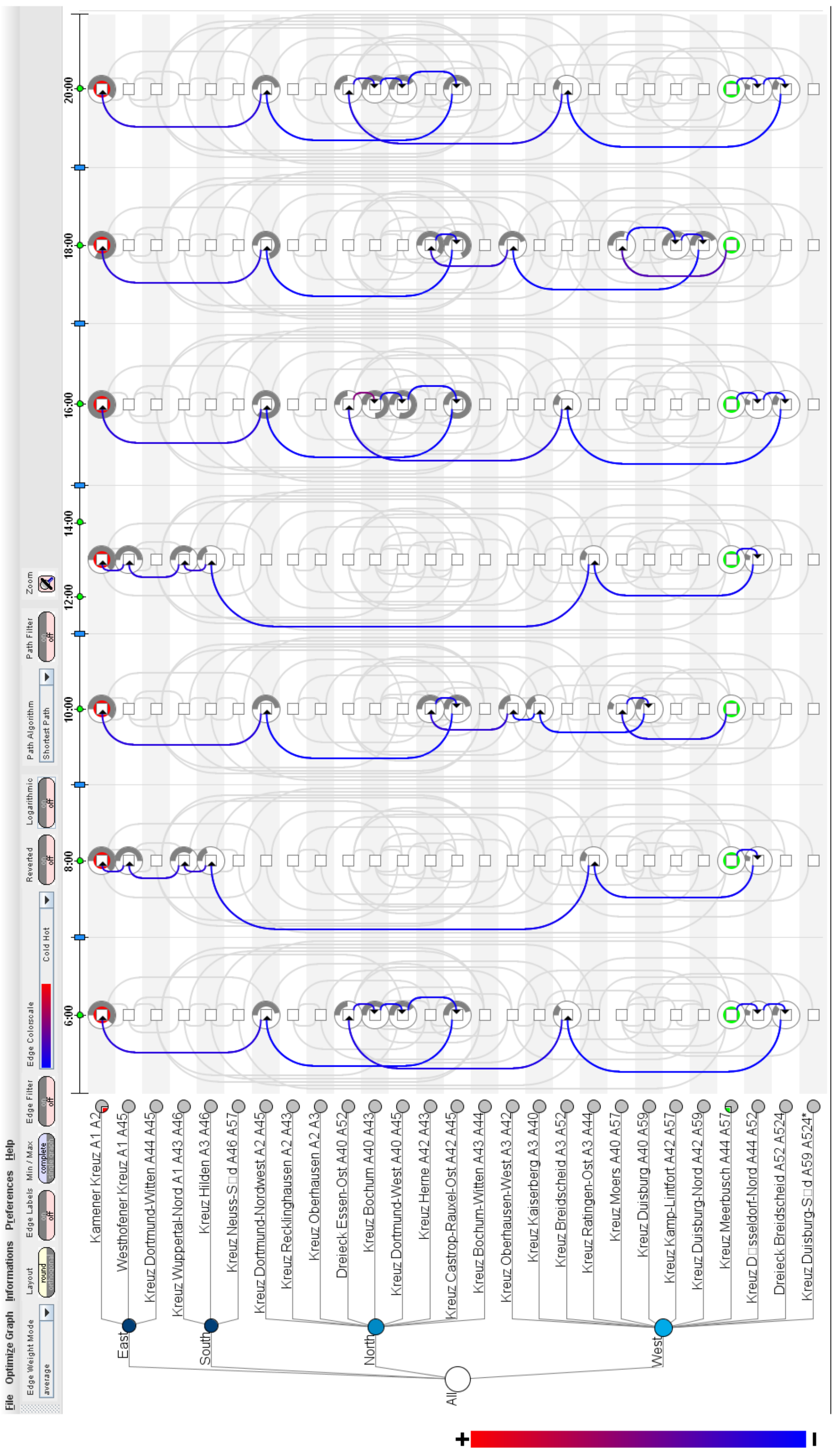


Figure 13: *Screenshot of the TimeArcTrees tool showing the shortest paths between the nodes "Kreuz Meerbusch" and "Kamener Kreuz" for different points of time. The graph in the middle is actually an aggregation of two graphs. The edges are colored with a blue to red color scale.*

we want to perform a user study to compare it with other existing approaches with respect to different tasks like counting and correlation problems [BBBD08], and in particular, tasks related to the evolution of shortest paths. To this end, we will have to extend the other tools to also compute and visualize shortest paths.

References

[AH98] ANDREWS K., HEIDEGGER H.: Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs (late breaking hot topic paper). In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'98)* (Research Triangle Park, NC, 1998), pp. 9–12. 1

[BBBD08] BURCH M., BOTT F., BECK F., DIEHL S.: Cartesian vs. radial – a comparative evaluation of two visualization tools. In *Proceedings of 4th International Symposium on Visual Computing (ISVC08), Las Vegas, Nevada* (2008). 8

[BBD08] BURCH M., BECK F., DIEHL S.: Timeline trees: Visualizing sequences of transactions in information hierarchies. In *Proceedings of 9th International Working Conference on Advanced Visual Interfaces (AVI2008)* (Naples, Italy, 2008). 6

[BD06] BURCH M., DIEHL S.: Trees in a treemap. In *Proceedings of 13th Conference on Visualization and Data Analysis (VDA 2006)* (San Jose, California, 2006). 6

[BD08] BURCH M., DIEHL S.: Timeradartrees: Visualizing dynamic compound digraphs. In *Proceedings of Tenth Joint Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis'2008), Eindhoven, The Netherlands* (2008), Eurographics Association. 6

[BM99] BERTAULT F., MILLER M.: An Algorithm for Drawing Compound Graphs. In *Proceedings of 7th International Symposium on Graphdrawing, GD* (1999), vol. 1731 of *LNCS*, Springer, pp. 197–204. 6

[FT04] FRISHMAN Y., TAL A.: Dynamic Drawing of Clustered Graphs. In *Proceedings of 10th IEEE Symposium on Information Visualization, INFOVIS* (2004), IEEE Computer Society, pp. 191–198. 6

[FT07] FRISHMAN Y., TAL A.: Online dynamic graph drawing. In *Proceedings of Nineth Joint Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis'2007), Norrköping, Sweden* (2007), Eurographics Association. 6

[FWD*03] FEKETE J.-D., WAND D., DANG N., ARIS A., PLAISANT C.: Overlaying graph links on treemaps. In *Poster Compendium of the IEEE Symposium on Information Visualization (INFOVIS'03), Los Alamitos, CA, USA* (2003), IEEE. 1, 6

[GJ79] GAREY M. R., JOHNSON D. S.: *Computers and Intractability A guide to the Theory of NP-Completeness*. W.H Freeman Publ., 1979. 3

[Hol06] HOLTEN D.: Hierarchical edge bundles: Visualizing of adjacency relations in hierarchical data. In *Proceedings of the IEEE Transactions on Visualization and Computer Graphics, 12, 741-748* (2006), IEEE. 1, 6

[JS91] JOHNSON B., SHNEIDERMAN B.: Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of IEEE Visualization Conference* (San Diego, CA, 1991), pp. 284–291. 1, 6

[NSC05] NEUMANN P., SCHLECHTWEG S., CARPENDALE S.: Arctrees: Visualizing relations in hierarchical data. In *Data Visualization, Eurographics/IEEE VGTC Symposium on Visualization* (Aire-la-Ville, Switzerland, 2005), Brodlie K. W., Duke D. J., Joy K. I., (Eds.). 1, 6

[PB08] POHL M., BIRKE P.: Interactive exploration of large dynamic networks. In *Proceedings of International Conference on Visual Information Systems (Visual 2008), Salerno, Italy* (2008), Springer Verlag. 1, 6

[PvW06] PRETORIUS A., VAN WIJK J.: Visual analysis of multivariate state transition graphs. In *Proceedings of IEEE Symposium on Information Visualization (InfoVis 2006), IEEE CS Press* (2006). 1

[RT81] REINGOLD E. M., TILFORD J. S.: Tidier drawing of trees. *IEEE Transactions on Software Engineering 7*, 2 (1981), 223–228. 1

[SM91] SUGIYAMA K., MISUE K.: Visualization of Structural Information: Automatic Drawing of Compound Digraphs. *IEEE Transactions on Systems, Man and Cybernetics 21*, 4 (1991), 876–892. 6

[SZ00] STASKO J., ZHANG E.: Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Proceedings of the Symposium on Information Visualization (InfoVis'00), Salt Lake City, UT* (2000), IEEE Computer Society Press, pp. 57–65. 1

[Wat02] WATTENBERG M.: Arc diagrams: Visualizing structure in strings. In *Proceedings of IEEE Symposium on Information Visualization* (2002). 6

[YFDH01] YEE K.-P., FISHER D., DHAMIJA R., HEARST M.: Animated exploration of dynamic graphs with radial layout. In *Proceedings of the IEEE Symposium on Information Visualization, San Diego, CA, USA* (2001). 1

[ZMC05] ZHAO S., MCGUFFIN M. J., CHIGNELL M. H.: Elastic hierarchies: Combining treemaps and node-link diagrams. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'05), Minneapolis, MN, USA* (2005), IEEE. 1, 6