TimeSpiderTrees: A Novel Visual Metaphor for Dynamic Compound Graphs

Michael Burch
VISUS, University of Stuttgart, Germany
michael.burch@visus.uni-stuttgart.de

Michael Fritz Fabian Beck Stephan Diehl
University of Trier, Germany
{s4mifrit,beck,diehl}@uni-trier.de

Abstract

Graphs are a mathematical method to model relations between objects. The most common metaphor to visualize graphs is the node-link technique, which typically suffers from visual clutter caused by many edge crossings. Much research has been done on the development of sophisticated algorithms aimed at enhancing the layout with respect to edge crossings and a series of other aesthetic criteria. In this paper we propose a novel visual metaphor, called TimeSpiderTrees, which is based on a radial layout. In our technique, relations are visually indicated by orientation instead of connectedness to circumvent the problem of edge crossings. The strength of this novel visualization technique lies in the visual encoding of time-series relational data in a single view without animation, which helps to preserve the mental map and hence to reduce cognitive efforts.

1 Introduction

Node-link diagrams are the most common approach to visually encoding relational information. They follow the very intuitive metaphor of visual objects that are connected by lines. Mainly because nodes or edges overlap, it becomes challenging to create readable layouts of larger or denser graphs. The graph drawing community addresses this problem by formulating aesthetic criteria to find readable graph layouts. Certainly, the most prominent criterion is the number of edge crossings—minimizing these crossings promises a good layout. Other criteria are, for instance, reducing the number of edge bends or lengths of edges. Much research has been done on developing sophisticated layout algorithms that generate aesthetic layouts trading off these criteria [3].

Many real world applications deal with dynamic graphs—graphs that are changing over time. Usually, in node-link diagrams this evolution of relational data is represented by animation. The major problem of animated visualizations, however, is the user's effort to preserve his abstract mental image of the graph, called the mental map, from step to step of the animation. Specialized dynamic graph layout algorithms [17, 6, 9] try to tackle the problem and reduce this cognitive effort. But they still cannot support fast comparisons of arbitrary points in time, follow abrupt changes, or capture trends over long periods.

Representing time on an axis, and not as an animation, leads to a single static integrated view providing a considerably enhanced overview. So far, only few works have followed this approach: TimeArcTrees [11] implements a straightforward adaptation of node-link diagrams to the integrated approach of representing dynamic graphs. It effectively increases the overview, but at the cost of an increased edge crossing problem. In contrast, TimeRadarTrees [5] builds upon a matrix representation of a graph in a radial layout. Here, the less intuitive metaphor and the multiple representation of objects impairs the readability.

Having in mind the effective visualization of dynamic graphs, in this paper we develop a new metaphor for graph visualizations starting from the node-link paradigm. We circumvent both the problem of edge crossings and the problem of multiple representations without neglecting overview. Our novel technique was inspired by using orientation instead of connectedness to express the relation of two objects. This leads to visualizations that often look like spiderwebs, hence the name TimeSpiderTrees.

The rest of this paper is structured as follows. Section 2 introduces our novel visualization technique TimeSpiderTrees. We document the general usefulness of the approach by visualizing real world datasets from different domains (Section 3). A systematic qualitative comparison to directly related visualizations in Section 4 reveals its main characteristics and completes the assessment. Finally, Section 5 discusses further related work and Section 6 concludes the paper.

2 The TimeSpiderTrees Visualization

In this work we are looking at relational data that is changing over time. Each relation may have a certain

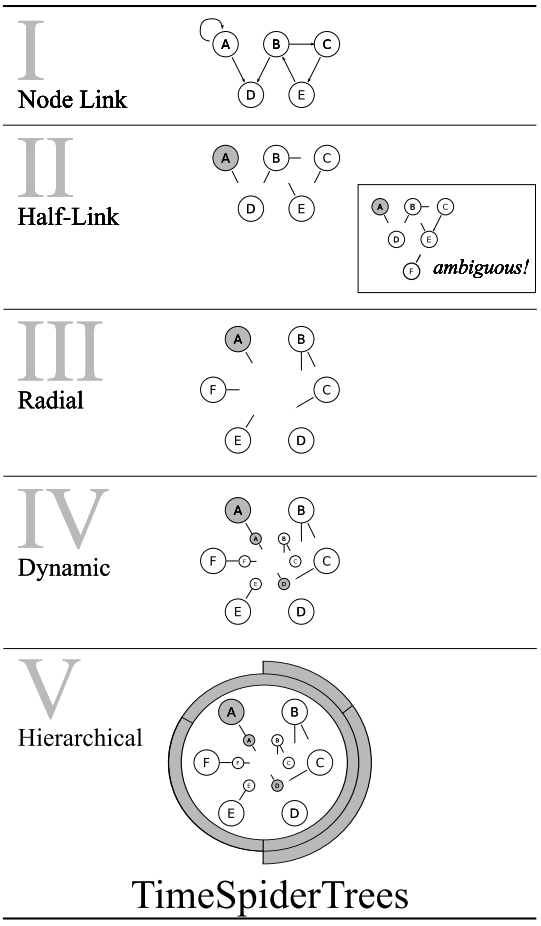

Figure 1. TimeSpiderTrees Step by Step

strength, which, of course, could also depend on time. An optional hierarchical structure of the considered objects (i.e., an additional tree data structure on the nodes) helps to reasonably group the objects. In the terminology of graph theory, such a dataset forms a weighted dynamic compound graph.

The hierarchy might be already given by a semantic hierarchical organization of the objects (e.g., cities may be hierarchical organized into regions, authors into working groups and institutions, etc.). If there is no hierarchy given, it can be easily created by applying a hierarchical graph clustering algorithm.

In the following we will describe TimeSpiderTrees step by step, starting at a conventional node-link diagram and transforming it into a TimeSpiderTrees visualization. Each step is illustrated in Figure 1 for a small sample graph.

Step I: Node-Link The starting point of the transformation process is a conventional node-link directed graph that consists of five nodes. The graph contains a cycle including nodes B, C, and E. Moreover, node A has a self-edge.

Step II: Half-Link Becker et al. [2] introduced a new visual representation of edges: Links do not have to connect the starting node with the target node by a complete line—a half or partly drawn line is enough. The human eye is still able to approximately follow these half-links. We avoid visual clutter that is caused by edge crossings at the cost of making following edges more difficult.

The approach is working, at least to some extent: In this small example, we are still able to easily follow edges, for example, detecting the cycle for nodes B, C, and E. Nevertheless, the approach often causes ambiguous situations: For instance, an additional node F as depicted in Figure 1 II might be connected to node C, to node E, or even to both. Larger graphs would intensify this problem.

Since partly drawn circular self-links look strange, we think that using the fill color of the respective node to represent self-edges instead is more readable.

Step III: Radial Placing the nodes on a circle and spacing them evenly is often a good starting point to lay out general node-link diagrams. It is especially well suited when using half-links because links cannot overlap nodes, and thus, the target node can be unambiguously identified by the orientation of the link. If we guarantee that each link does not exceed an imaginary area that exclusively belongs to its source node like a Voronoi region, we can even avoid all edge crossings—Figure 1 III shows the modified version of our running example. We can check the resolved ambiguity: The link starting at node F points to node C, but not to node E.

Step IV: Dynamic It is questionable—however not totally unrealistic—whether the approach as presented so far has general advantages over conventional node-link diagrams for static graphs. But we argue that the major strength of the idea lies in the representation of dynamic graphs—relational data that changes over time.

As already discussed, animating node-link diagrams provides unsatisfying results with respect to effectively analyzing heavy changes and trends, mainly caused by a high cognitive effort and a lack of overview. In contrast, the proposed metaphor of half-links does not require animation. It already leaves a blank area at the circle center which can be recursively filled with subsequent graphs (Figure 1 IV).

Step V: Hierarchical The difference between a *graph* and a *compound graph* is an additional hierarchy on the nodes. The main advantage of adding such a hierarchy is, on the one hand, that the hierarchy induces a meaningful order on the circularly laid out nodes and, on the other hand, that it allows the observer to easily focus on certain parts of the graph by interactively collapsing parts of the hierarchy. It thus promises to improve the readability as well as the

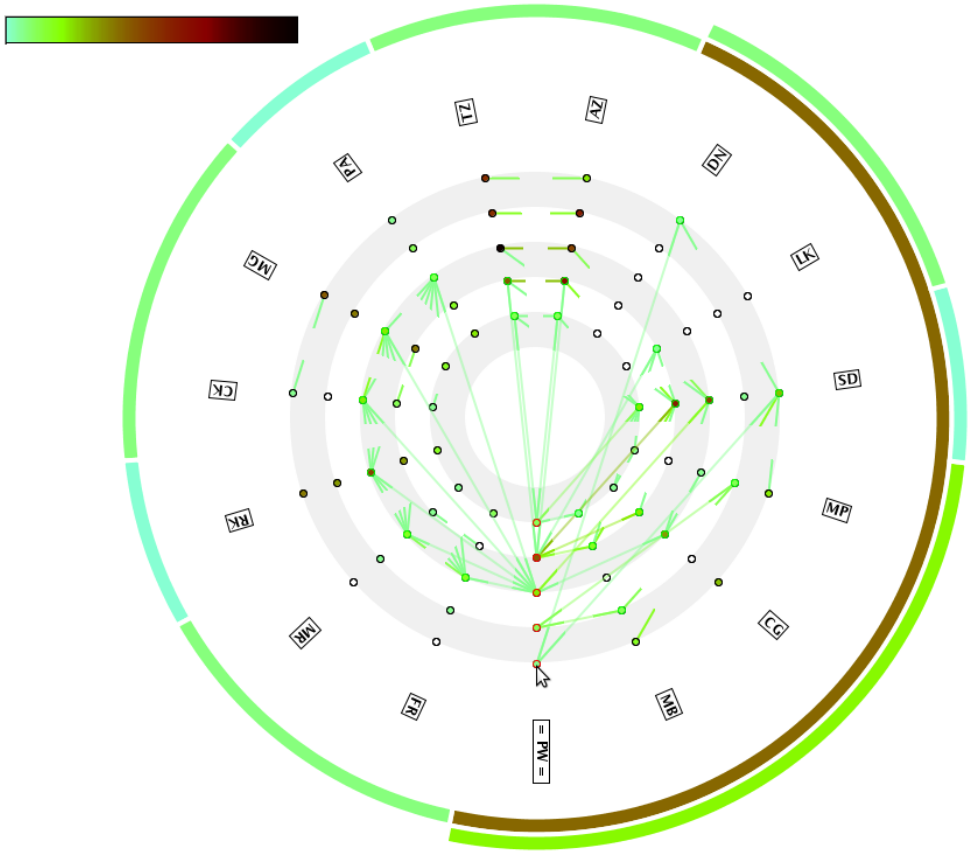

Figure 2. A Dynamic Co-Author Graph (focused on author *PW*).

scalability of the visualization. If no hierarchy is available, a hierarchical clustering algorithm is able to generate one automatically.

To represent the hierarchical information, we prefer a radial layered icicle plot [20, 21] that encloses the radial diagram. It neither disturbs the graph representation nor consumes much space. Finally, Figure 1 V sketches the representation that we call TimeSpiderTrees.

Interaction We implemented a prototype of the TimeSpiderTrees approach. Our first experiences using this prototype revealed a deficiency of the approach: It is often hard to determine to which target an edge is pointing. To overcome this problem, we introduce some interaction techniques.

In our prototype implementation, focusing on a node in a particular graph, the associated outgoing edges will be completely drawn: A full link again connects the nodes. Colored border lines of the target nodes support the visual perception of these links. This crucial feature is activated just by moving the mouse over a node. Moreover, we also allow a node to be highlighted in all graphs. Outgoing links will be drawn completely in every graph like concurrently focusing the node over the time axis (Figure 2). This improves the observation of changes in time with respect to the selected node.

These two focusing techniques address the outgoing edges of a node. But the incoming edges are often interesting as well. Pressing the *Ctrl* button on the keyboard switches the focusing mode from outgoing to incoming edges: Edges that target at the focused node will be completed now.

3 Applications

In the following sections we show the usefulness of our TimeSpiderTrees visualization technique by applying it to example datasets from two different real world applications. In these applications TimeSpiderTrees enabled us to detect, for instance, trends and periodic events, outliers—with respect to nodes, edges, or even entire graphs—and dense clusters.

3.1 Bibliometrics: Co-Author Graphs

Bibliometrics explore and benchmark fields of research. In the application at hand, we look at co-author graphs,

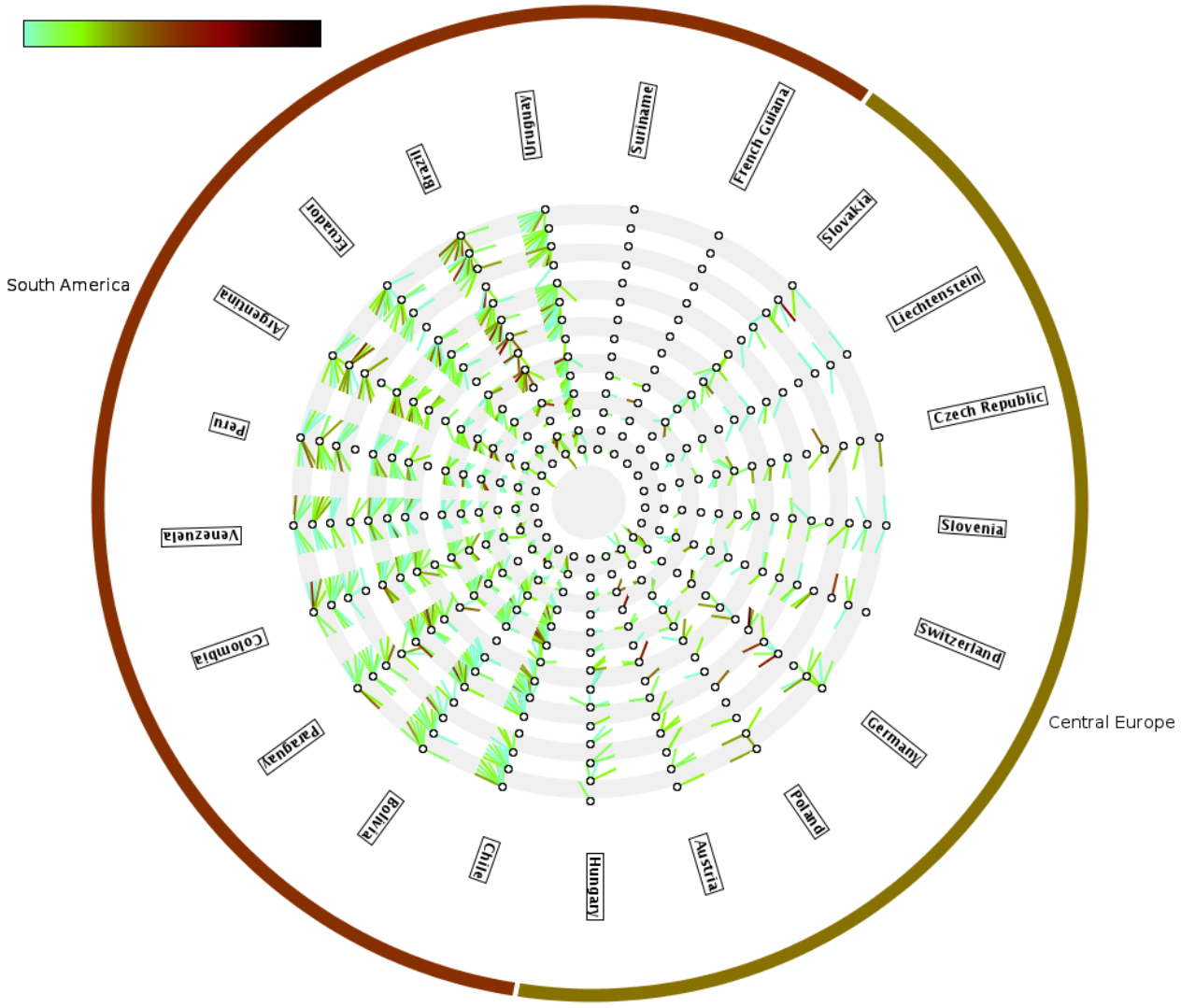

Figure 3. International Soccer Results of South America and Central Europe (1992 to 2005)

which could provide information on, for example, relations between authors, working groups, author roles, or publication quantities. A co-author graph consists of author nodes, which are related by joint publications. We aggregate the publications of a year to one graph and relate two authors A and B with extent n if they published n joint works in the respective year. We use colors to encode these edge weights.

In particular, our example presents a real world co-author graph of a single author extracted from DBLP, a bibliographic database on major computer science journals and proceedings. We use abbreviated author names to simplify the example. Figure 2 shows the co-author graph for *PW* in the years 2004 (inner circle) until 2008 (outer circle). The hierarchy subdivides the authors into working groups and, for the working group of *PW*, further into roles.

PW published together with six other members of his working group and eight researchers from five different other working groups. While he frequently collaborated with the members of his own group in different combinations, the collaboration with six of the eight other researchers is limited to a single publication—a contribution to a seminar proceedings in 2006. With exception of this publication, the other working groups do not further publish together. The connection between the group of *PW* and the group of *TZ* and *AZ* seems to be a bit stronger because they cooperated twice and with fewer participants.

The visualization also hints at the different roles of the authors. The total numbers of publications, modeled as the strengths of the self-edges and thus encoded by the node colors, suggest that *RK*, *MG*, *TZ*, *AZ*, and *SD* play leading

roles and might be the heads of the working groups. In contrast, *DN* and *LK* each just published once.

3.2 Sports Visualization: Soccer Results

As the second application, we chose soccer results, an example from the area of sports visualization. Figure 3 visualizes the goals scored in international matches from 1992 to 2005 restricted to South America and Central Europe: Each graph represents the results of a year, and within each graph, each relation from a national team A to a national team B sums up all goals scored by A against B in the respective year. Accordingly, the relation only exists if the two teams played against each other in this year. The visualization provides an overview of 21 teams over 14 years, which amounts to 1300 individual relations.

Perhaps most obvious is the significantly different density of the two clusters of the graph, the South American teams and the European teams. The South American sub-graph—with exception of the small states of Suriname and French Guiana—is very dense while the European sub-graph is relatively sparse. The small number of states in South America in relation to the large number of states in Europe explains why the same South American teams have to play against each other more frequently.

We are able to detect temporal patterns like the periodic density gap in the South American sub-graph every four years—the years of the World Cup where few intra-continental matches take place.

High numbers of goals, indicated by dark colors, highlight good teams: In South America, Brazil and Argentina are the leading teams while in Central Europe Germany and the Czech Republic perform well. Analyzing trends with respect to these high scores, for instance, Austria seems to be quite successful in the nineties and declines since the year 2000. A glance at the official world ranking confirms this descent.

Searching for anomalies, for example, we find the highest sum of goals in a relation from Germany to Liechtenstein (a friendly game in 1996, which Germany won 9:1). Another outlier is Venezuela, which never played against a Central European team in the time period shown. Interactively focusing this node over all graphs validates this observation.

4 Aesthetic Dimensions Analysis

Beck et al. [1] propose a set of aesthetic dimensions for dynamic graph visualizations. The aesthetic dimensions provide a framework to systematically compare the readability of TimeSpiderTrees to related visualizations in a qualitative assessment. This evaluation, however, depends on the subjective ratings of the authors.

Apart from animated node-link diagrams, the number of different approaches to visualizing dynamic graphs is small. The following list gives a brief overview of dynamic graph visualization approaches. Figure 4 shows the co-author dataset from Figure 2 in each of three visualizations. These techniques provide a reference to examine the strengths and weaknesses of the presented TimeSpiderTrees visualization approach.

Animated Node-Link (ANL) *An animated dynamic graph visualization based on a node-link representation (Figure 4 (top), [17, 6, 9]).*

Animated node-link diagrams are the straightforward solution to represent dynamic graphs. State-of-the-art visualizations do not only optimize the layout of a single graph, but consider the whole sequence of graphs. For Figure 4 (top) we used a foresighted algorithm [6] based on a force-directed layout.

TimeRadarTrees (TRT) *An integrated dynamic graph visualization based on a combined matrix-list representation (Figure 4 (left), [5]).*

TimeRadarTrees uses a radial layout where nodes are represented by circle sectors of the inner circle. The representation depicts each graph from the sequence of graphs as a ring of the inner circle. Incoming edges are colored sectors in the inner circle. Outgoing edges are colored sectors in the outer circles in the same context as the associated incoming edge.

TimeArcTrees (TAT) *An integrated dynamic graph visualization based on a node-link representation (Figure 4 (right), [11]).*

TimeArcTrees draws a sequence of node-link diagrams from left to right such that each node is placed in a particular row.

The following paragraphs discuss TimeSpiderTrees (TST) in the terminology of the proposed aesthetic dimensions. Abbreviations in brackets reference the particular dimension in the original framework [1]: General Aesthetic Criteria (GAC), Dynamic Aesthetic Criteria (DAC), and Scalability Criteria (SC). Figure 5 lists all aesthetic dimensions and summarizes the results in a parallel coordinates plot.

4.1 Static Aspects

Neither the visual representations of edges nor of nodes overlap each other in TimeSpiderTrees. In contrast, overlapping edges (edge crossings) produce visual clutter (GAC1) in usual node-link representations. Following edges in TimeSpiderTrees is difficult: The target node could be mistaken for another (spatial alias: GAC2) because the

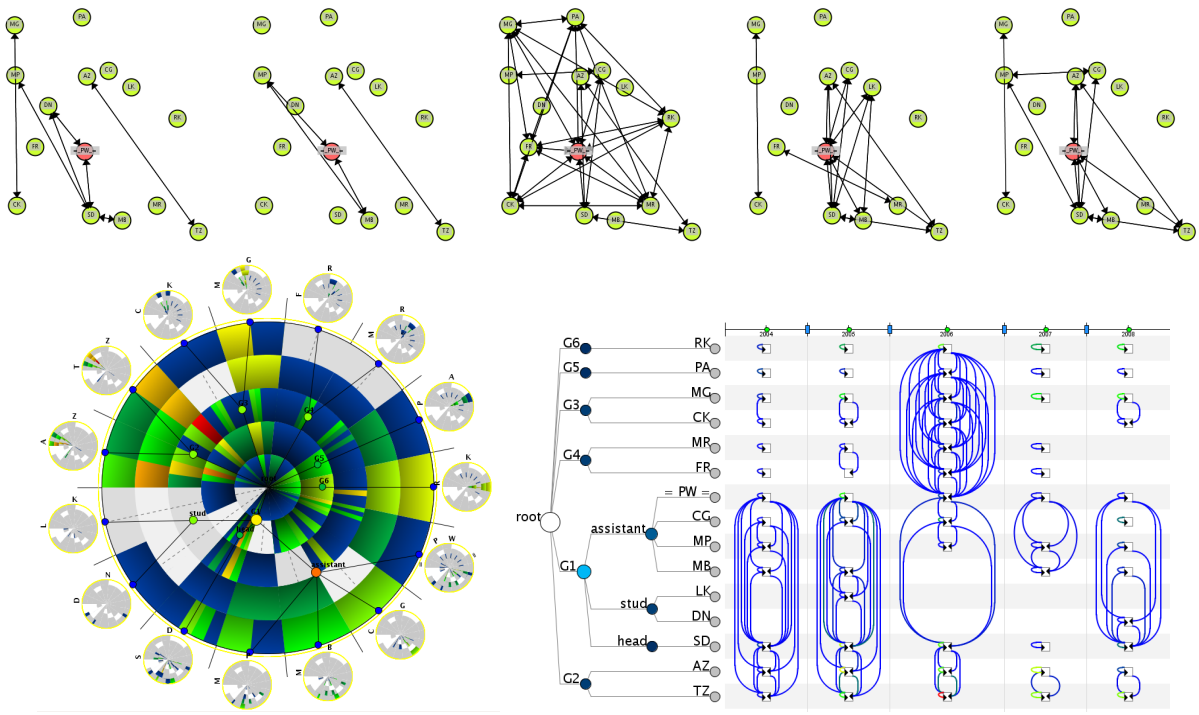

Figure 4. Co-Author Graph of PW (compare to Figure 2) as an Animated Node-Link Diagram (top), TimeRadarTrees (left), and TimeArcTrees (right).

edge representation just shows the direction but does not directly connect the nodes. Interaction techniques, however, alleviate the problem. TimeRadarTrees shows similar characteristics for these two criteria, visual clutter and spatial aliases, but additionally needs multiple visual representations of edges (GAC3). In TimeSpiderTrees as well as in every node-link approach, only one visual element represents each edge. Moreover, TimeSpiderTrees uses the available screen space very efficiently like TimeRadarTrees—it is compact (GAC4).

4.2 Dynamic Aspects

The user is only able to follow changes in an animated graph representation when his abstract mental representation (mental map) of the graph is preserved (DAC1). In visualizations that show the whole time span in a single integrated image like TimeSpiderTrees, TimeRadarTrees, and TimeArcTrees, a mental map is implicitly preserved by the static node layout. Also the cognitive load comparing subsequent graphs (DAC2) is lower in these integrated visualizations because the user does not have to remember information about longer time spans. Furthermore, the risk of mistaking one element for the other in different graphs (temporal alias: DAC3) is lower. While these negative effects for animated graphs are most significant for heavily changing graphs (as you can observe in Figure 4 (top)), they are much less grave for slowly evolving graphs.

4.3 Scalability

Similar to TimeRadarTrees, in TimeSpiderTrees the scalability in number of nodes (SC1) is limited by the circle circumference. Even less space for each node is available in TimeArcTrees while animated node-link diagrams provide a much better scalability in number of nodes. Nevertheless, the density of the graph (scalability in number of edges: SC2) is no problem for TimeSpiderTrees (compare to Figure 3) because similar to a graph adjacency matrix, space is reserved for each potential edge, regardless if it exists or not.

Although, in theory, animated node-link diagrams are able to show an unlimited number of graphs, when taking readability into account, the scalability in number of graphs (SC3) is better for an integrated visualization like TimeSpiderTrees. It is almost impossible to compare non-subsequent graphs in an animation. In contrast, an integrated representation allows arbitrary graphs to be compared without having to memorize much information.

All the visualizations considered have to struggle with scalability problems. While the number of nodes is most

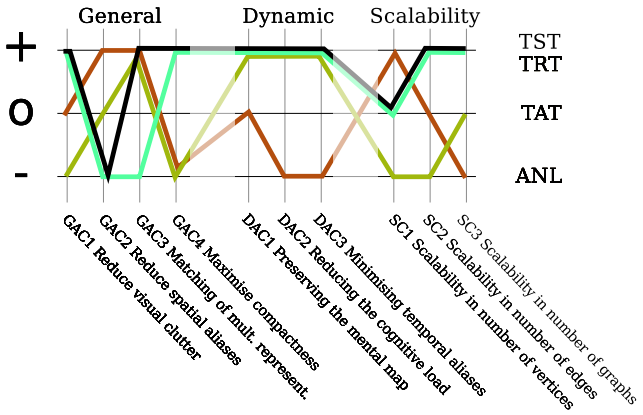

Figure 5. Analysis Summary

crucial for TimeSpiderTrees, animated node-links have problems with dense graphs and overview in time.

4.4 Hierarchy

An additional hierarchy that groups the nodes of the dynamic graph helps to semantically arrange the graphs. In TimeSpiderTrees the hierarchy induces a linear ordering of the nodes: Grouped nodes are placed next to each other. We can easily check if these neighboring nodes show similar characteristics in the dynamic graph. Moreover, the hierarchy provides a natural backbone to explore the graphs on different levels of detail. Interactively expanding and collapsing the hierarchy largely improves the scalability as the number of nodes increases.

In TimeSpiderTrees, changes in the hierarchy are simulated by copying the nodes to their new positions. Thus, TimeSpiderTrees can only cope with small changes in the hierarchy. An animation is more flexible: nested boxes usually represent the hierarchical structure (e.g., [4, 18]). Changing parts of the hierarchies can be treated like changing parts of the graphs.

4.5 Summary

In contrast to animated node-link diagrams, we consider TimeSpiderTrees appropriate for analyzing significantly changing dynamic graphs, providing an overview on the whole dataset, exploring dense dynamic graphs, and identifying temporal patterns and trends in dynamic graphs. Consequently, we rate TimeSpiderTrees to be less suitable for analyzing slowly evolving dynamic graphs, visualizing significant changes in the hierarchy, and exploring single graphs of the sequence in detail.

5 Related Work

In addition to related work with respect to dynamic graphs, we look at some of the work related to visual clutter in graphs and the visualization of compound graphs. For details on other radial visualization techniques, especially ring-based ones, we refer to a recent survey by Draper et al. [7].

5.1 Visual Clutter in Graphs

Graphs are widely used to model relations among objects. A widespread visual metaphor is the representation of this relational data as node-link diagrams. To reduce visual clutter that is caused by many edge crossings, many sophisticated graph-drawing algorithms have been developed with respect to a set of aesthetic criteria. Some of those criteria cannot be fulfilled simultaneously because of trade-offs. For example, it is not possible in general to reduce the number of edge crossings and to reduce the length of edges.

Drawing half-links—the approach that we use to reduce the visual clutter—was already proposed by Becker et al. [2] and implemented in their SeeNet tool. Among other techniques like focusing, filtering, and highlighting, they use this approach to effectively visualize time-dependent geospatial relational data in an animated node-link diagram. The fixed node positions given by their geographical positions, on the one hand, circumvent the problem of finding a good node layout, but on the other hand, lead to largely ambiguous half-link representations of edges.

Another means for reducing visual clutter in node-link diagrams was introduced by Holten and van Wijk [12, 13]. The authors developed a force-directed edge bundling approach for graphs. Recently, Jianu et al. [14] alleviate edge crossings by coloring crossing edges in colors which are as different as possible. For radial graph layouts, Gansner and Koren [10] present different techniques to reduce the number of edge crossings. Their algorithm based on edge-length minimization could be also integrated into TimeSpiderTrees if we did not apply a hierarchy.

To totally circumvent the edge crossing problem, matrix representations of graphs might be a solution. Keller et al. [15] point out that matrices have many benefits over traditional node-link diagrams when visualizing very dense graphs and that visual clutter is reduced to a minimum. Nevertheless, a matrix-based representation is problematic when it comes to visualizing sequences of graphs in a single view.

5.2 Compound Graph Visualization

Georg Sander [19] developed a method to automatically lay out compound graphs. In this approach the hierarchy

is expressed by nested boxes and the relations by links that point between these boxes. Fekete et al. [8] introduced a similar approach and overlaid curved graph links on treemaps. The ArcTrees [16] technique uses a one-dimensional treemap and represents links as arcs that are drawn above the treemap view and differ in height with respect to their inter-hierarchical relation. But also the hierarchical edge bundling approach by Holten [12], discussed previously, visualizes compound graphs. The approach is very helpful to explore the overall structure of the compound graph. Following a single edge from the start to the target node, however, is difficult. Interactive features are absolutely necessary to solve path-related tasks.

6 Conclusion

We presented TimeSpiderTrees, a novel visual metaphor for dynamic weighted compound graphs. The visualization is based on a radial node-link approach, but indicates edges by directed, partly drawn lines instead of complete links between the nodes. This totally avoids edge crossings, handles dense graphs efficiently, and allows the entire sequence of graphs to be drawn in a single integrated diagram—interaction techniques allow particular edges to be followed. Our case study shows that TimeSpiderTrees is a suitable visualization for analyzing dynamic graphs. Furthermore, the aesthetic dimension analysis especially proposes to use TimeSpiderTrees to analyze heavily changing graphs, dense graphs, as well as temporal trends and patterns.

References

[1] F. Beck, M. Burch, and S. Diehl. Towards an Aesthetic Dimensions Framework for Dynamic Graph Visualisations. *International Conference on Information Visualisation*, pages 592–597, 2009.

[2] R. Becker, S. G. Eick, and A. R. Wilks. Visualizing Network Data. *IEEE Transactions on Visualization and Computer Graphics*, 1:16–28, 1995.

[3] C. Bennett, J. Ryall, L. Spalteholz, and A. Gooch. The Aesthetics of Graph Visualization. In *Proceedings of Computational Aesthetics in Graphics, Visualization, and Imaging*, pages 57–64, 2007.

[4] R. Brockenauer and S. Cornelsen. Drawing Clusters and Hierarchies. In M. Kaufmann and D. Wagner, editors, *Drawing Graphs*, volume 2025 of *LNCS*, pages 193–227. Springer, 2001.

[5] M. Burch and S. Diehl. TimeRadarTrees: Visualizing Dynamic Compound Digraphs. In *Proceedings of Tenth Joint Eurographics/IEEE-VGTC Symposium on Visualization*, pages 823–830, 2008.

[6] S. Diehl and C. Görg. Graphs, They Are Changing. In *Revised Papers from the 10th International Symposium on Graph Drawing*, pages 23–30. Springer, 2002.

[7] G. M. Draper, Y. Livnat, and R. F. Riesenfeld. A Survey of Radial Methods for Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):759–776, 2009.

[8] J.-D. Fekete, D. Wang, N. Dang, A. Aris, and C. Plaisant. Overlaying Graph Links on Treemaps. In *Poster Compendium of the IEEE Symposium on Information Visualization (INFOVIS'03)*, pages 82–83, 2003.

[9] Y. Frishman and A. Tal. Dynamic Drawing of Clustered Graphs. In *Proceedings of 10th IEEE Symposium on Information Visualization*, pages 191–198. IEEE Computer Society, 2004.

[10] E. Gansner and Y. Koren. Improved circular layouts. In M. Kaufmann and D. Wagner, editors, *Graph Drawing*, volume 4372 of *Lecture Notes in Computer Science*, chapter 37, pages 386–398. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[11] M. Greilich, M. Burch, and S. Diehl. Visualizing the Evolution of Compound Digraphs with TimeArcTrees. In *Proceedings of the Eleventh Joint Eurographics/IEEE-VGTC Symposium on Visualization, Berlin, Germany*, pages 975–982, 2009.

[12] D. Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[13] D. Holten and J. J. van Wijk. Force-Directed Edge Bundling for Graph Visualization. In *Proceedings of the 11th Eurographics/IEEE-VGTC Symposium on Visualization*, pages 983–990, 2009.

[14] R. Jianu, A. Rusu, A. J. Fabian, and D. H. Laidlaw. A Coloring Solution to the Edge Crossing Problem. In *Proceedings of the 2009 13th International Conference Information Visualisation*, pages 691–696. IEEE Computer Society, 2009.

[15] R. Keller, C. M. Eckert, and P. J. Clarkson. Matrices or Node-Link Diagrams: Which Visual Representation is Better for Visualising Connectivity Models? *Proceedings of Information Visualization*, 5(1):62–76, 2006.

[16] P. Neumann, S. Schlechtweg, and S. Carpendale. ArcTrees: Visualizing Relations in Hierarchical Data. In *Joint Eurographics/IEEE VGTC Symposium on Visualization*, pages 53–60. Eurographics Association, 2005.

[17] S. North. Incremental Layout in DynaDAG. In *Proceedings of 4th International Symposium on Graph Drawing*, volume 1190, pages 409–418. Springer, 1996.

[18] M. Pohl and P. Birke. Interactive Exploration of Large Dynamic Networks. In M. Sebillo, G. Vitiello, and G. Schaefer, editors, *VISUAL*, volume 5188 of *Lecture Notes in Computer Science*, pages 56–67. Springer, 2008.

[19] G. Sander. Layout of Compound Directed Graphs. Technical report, Universität des Saarlandes, FB 14 Informatik, 1996.

[20] J. Stasko and E. Zhang. Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations. In *Proceedings of the IEEE Symposium on Information Visualization 2000*, page 57. IEEE Computer Society, 2000.

[21] J. Yang, M. O. Ward, E. A. Rundensteiner, and A. Patro. InterRing: A Visual Interface for Navigating and Manipulating Hierarchies. *Information Visualization*, 2(1):16–30, 2003.