Visual Comparison of Document Collections Using Multi-Layered Graphs

Mountaz Hascoët, Pierre Dragicevic

Visual Comparison of Document Collections Using Multi-Layered Graphs

Mountaz Hascoët[2] and Pierre Dragicevic[1]

[1] LIRMM, CNRS UMR 5506, Univ. Montpellier II,
161, rue Ada 34392 Montpellier Cedex, France
[2] INRIA, Bât 490, UPS, 91405 Orsay Cedex, France
mountaz@lirmm.fr,dragice@lri.fr

Abstract. In this paper, we propose a general visual and interactive tool named Donatien that supports the comparison of collections of documents like sets of documents evolving over time or query results coming from different sources. Our approach builds on previous work from different areas of research and conciliates visualization, interaction and optimization approaches. Sets of documents are represented by graphs where documents are nodes, and edges between nodes are relationships between documents. Our contribution is to combine (1) a layer mechanism that makes possible the superposition of different layers representing different sets of documents, (2) original deterministic layout strategies that facilitate comparisons across layers, and (3) an interaction model based on drag-and-drop that lets users manually adjust matchings between documents. Our approach is used in a real case study based on data extracted from the InfoVis contest benchmark. The data used contains approximately 600 research papers with approximately 8000 citation links between documents covering almost 10 years of the InfoVis conference. Different subsets, corresponding to different years, are extracted from the data and compared with our approach in order to visually analyze the evolution of papers and topics over years.

Keywords: Visual Analysis, Graph Matching, Multi-Layer Comparison, Deterministic Layout of Graph

1 Introduction

With the advent of digital libraries and the huge amounts of documents now available electronically, the comparison of collections of interconnected documents can be useful in many situations. Comparing sets of documents resulting from different search engines for the same query might be an alternative to recall and precision to evaluate the quality of a search engine at a user level. Comparing the evolution of collections such as conferences or journal series over time can be used to better make sense of a given research field.

Collections of documents can be represented in many ways and compared at different levels of detail. For example, they can be compared based on their textual content [8, 22] or based on metadata. In this paper we focus on comparing collections of documents based on their relationships with shared entities such as authors and keywords, and represented as graphs evolving over time. We take as a case study a collection of research papers collected in 2004 [16] to serve as the benchmark for a challenge in an international information visualization conference. This collection covers 10 years of research in information visualization. Research papers are connected to keywords, authors, and sources (venues or journals).

This paper starts with a quick review of related work in graph comparison in section 2. Section 3 describes our multi-layer approach enabling the conciliation of multiple layout strategies as well as an original multi-scale approach to the visual comparison of graphs. This approach is made possible thanks to semi-automatic matching. Section 4 further describes the main principles underlying our approach to semi-automatic graph matching. Section 5 is used to illustrate our approach for the comparison of collections of documents extracted from the Information Visualization Benchmark [27].

2 Background

Since our document collections are graphs, we have to deal with the problem of comparing graphs. Graph comparison is a challenging task that concerns a wide range of domains and that can be subdivided into two subproblems: graph matching and visual graph comparison. The problem of graph matching is to find the mappings between nodes from two different graphs, since testing for equality of single node attributes is often not enough (e.g., authors can appear with different names or conference names can evolve over time). Graph theorists have been interested in automatically matching graphs based on their structure alone [10], a specific instance of this problem being the graph isomorphism problem, known as computationally challenging [19]. Matching techniques that incorporate semantic information present in node and edge attributes have also been studied in domains such as knowledge engineering [15] and databases [28].

The problem of visual graph comparison is to present collections of matched graphs to end users in a way that allows for easy comparison. A range of techniques have been proposed in Infovis and can be categorized under three main approaches: (a) side-by-side views [24, 21, 1], (b) superimposed or merged views [13, 1] and (c) animations [30, 11]. These approaches are often complemented with techniques for highlighting the matches or differences between the graphs. Those include visual links [9, 21], color coding schemes [24, 14, 23] and brushing and linking interactions [21]. A domain closely related to graph comparison is the visualization of time-varying graphs, for which similar techniques have been devised [5, 7, 13]. In addition, some work in the graph drawing community has focused on providing stable layout algorithms to allow for animation [11].

For the purpose of comparing series of graphs that represent collections of interconnected documents, we use a multi-layer approach that contrasts with approaches (a), (b) and (c). Layers are a construct commonly used in applications for authoring graphics, animations or videos and are hence familiar to most computer users [17, 18]. Layers can also be seen as a particular case of visual planes, a concept that has been introduced by Collins and Carpendale [9] for comparing visualizations. But in contrast with the use cases explored by Collins and Carpendale, our layers are 2-D visual planes that are stacked and translucent. For the purpose of comparing graphs, this has the advantage of combining the benefits of animation, i.e., the ability to preserve the user's mental map of related visualisations [14] with those of general visual planes, i.e., the ability to see several related visualisations at the same time and tune them separately.

As we further show in this article, another advantage of using layers for comparing graphs is that they provide basic support for the three main approaches (a), (b) and (c) known in the domain. Although some systems provide support for more than one technique, they rarely attempt to unify them within a single coherent visualization and interaction paradigm. Exceptions are systems that let users rotate 3-D stacked views to support both side-by-side and superimposed comparisons [13, 6, 14] and systems like Gevol that unify color coding and animation by displaying the union graph and having node colors change over time [7]. The most notable exception is Brandes et al.'s system that combines (a), (b) and (c) by drawing graphs on 3D layers whose opacity can vary to show graph evolution in a way similar to the Flash "onion-peel" effect [5]. But layers are only used as a visualization metaphor and the system does not exploit the full potential of layers in terms of interaction, as layers cannot be manipulated and tuned individually. Supporting this is likely to be more difficult with 3-D layers than with 2-D ones [29].

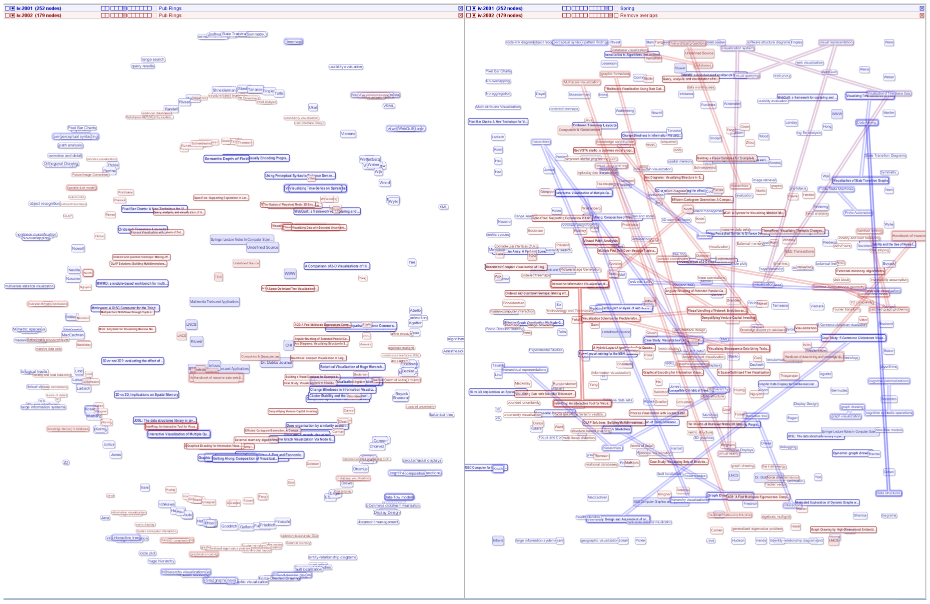

Fig. 1. Using side-by-side approach for the customization of fitness function

Finally, another issue in the domain of graph comparison is that traditionally, the graph matching problem is addressed separately from the visual graph comparison problem. As a consequence, graph matching systems have no or poor GUI support. This is despite the fact that the user's expertise is ultimately needed to validate, guide or tune the matching process, and therefore a visual graph comparison interface is almost always needed to check the results. Conversely, when using a visual graph comparison interface users might see matching mistakes and might need to re-run matching algorithms with different parameters or fix the matches manually. However, most visual graph comparison systems do not or poorly support these tasks. For example, the system from [1] uses a similarity matrix as an input to the visualization but does not explain where this similarity matrix comes from. There has been little research on how to integrate graph matching and visual comparison tasks, except in the field of ontology alignment [23, 20] where integrated systems have been proposed but are typically limited to manual node adjustment.

3 A Multi-Layer, Multi-Scale and Multi-Layout Interaction Model

Layers in Donatien share the main characteristics of layers commonly found in graphical authoring applications: they act as semi-transparent sheets that are stacked on top of each other. Each layer is reified in the form of a title bar that can serve to make layers visible or invisible on request and to rearrange their display order. Layers can also be panned, individually or together. Since layers in Donatien are designed for graph comparison purposes, they also have some specificities that distinguish them from traditional layers.

Each layer contains a document graph and is assigned a layout. The layout can be dynamically changed for each layer through the title bar. Donatien currently supports common graph layout algorithms through the JUNG library as well as the linlog layout [26] and Dwyer's node overlap removal algorithm [12]. In addition to the animations that are intrinsic to incremental layout algorithms, node motions are smoothly animated to clarify how layouts operate in case iterations are slow and/or involve node jumps (e.g., Kamada-Kawai). Layout algorithms can be started and stopped at will and their initial node positioning step has been removed (random positioning is provided as a separate layout) so that users are able to combine them in sequence (e.g., performing a linlog followed by a spring layout to separate nodes, then removing remaining overlaps).

Both layouts and graphs can be shared by different layers, making a variety of comparisons possible. For example, one can choose to visualize the same set of data using different layouts in order to explore the differences between the different layouts in terms of visual appearance and computation times. The same layout can also be used for different graphs and these graphs can be animated using a crossing-based technique (Figure 3). Finally, layouts can be adjusted manually by dragging nodes.

Second, a geometrical transformation is also associated to each layer, and layers can be zoomed in or out in addition to standard pan operations. This effectively combines the layer paradigm with the zoomable user interface (ZUI) paradigm [3, 4]. One direct benefit of this compared to traditional ZUIs is that users can simultaneously control different levels of zoom to match their foci of interest. As an example, suppose a user wants an overview of a specific document collection first with other documents at a lower context at a lower zoom level, and then wants to zoom all collections to get more details on common items. Using zoomable layers, the user would just have to zoom-out layers underneath and zoom-in one layer of interest, and then further zoom-in all layers simultaneously. Furthermore, we support three types of zoom (through keyboard modifiers): (a) the standard homogeneous zoom, (b) a zoom that only affects object size (i.e., node label size and link thickness) and (c) a zoom that only affects object distances. Being able to control object size and object distance separately is important while navigating node-link diagrams, as users may want to spread nodes apart while keeping labels visible, increase the size of all labels or get an overview of a graph by shrinking everything.

Third, layers can be put side-by-side so that both side-by-side and superimposed comparisons are possible. This is done by combining a tiling window manager [25] with a drag-and-drop technique that allows to move individual layers from a window to another. A similar drag-and-drop technique where tabs are used instead of layers has been introduced in [2] and is now implemented in applications such as Google Chrome. Another way of performing side-by-side comparisons is by simply shifting individual layers using Ctrl-drag (i.e., panning individual layers). This technique is well-suited to temporary side-by-side comparisons while tiling is more suited to longer-term comparisons and allows to organize layers in groups. These groups will also be used for editing and applying graph matchings, as will be described in the next section.

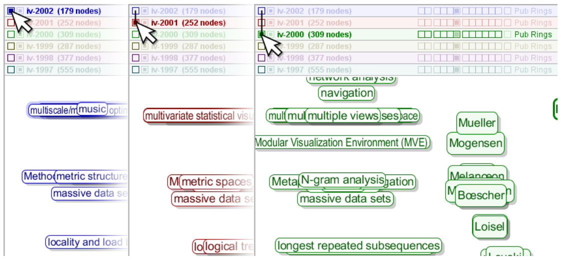

Fig. 2. Crossing based interaction to rapidly flip through layers

4 Node Signature Layouts for Visual Graph Matching and Comparison

Fitness functions play a key role in the graph comparison problem, both from the perspective of graph matching and of visual graph comparison. Algorithms based on optimization work by minimizing fitness functions. Those include most graph layout algorithms (where energy depends on how legibility criteria such as inter-node distances and edge crossings are met) as well as graph matching algorithms (where energy depends on the similarity between pairs of matched nodes). In both areas however, it usually remains opaque to the end-user. The underlying assumption is that fitness functions are too complex and end users should be spared the details of their computation. We propose an alternative approach for graph comparison that (1) offers users additional graph layout algorithms that are not based on optimization and (2) lets them choose amongst simple fitness functions for graph matching, preview them and fix them by hand.

Our approach is based on graph layouts that use node signatures as coordinates systems. We call a *node signature* a property of a node that is preserved after the nodes have been reordered. Node signatures can be semantic (based on attributes) or structural (for example node degree, i.e, the number of connected edges). When they are structural, node signatures are equivalent to the concept of *node invariant* from graph theory [19]. Node invariants are used as an initial step in graph isomorphism algorithms to prune the search space: for example, a node of a given degree in a graph cannot match a node of different degree in an isomorphic graph.

One simple example of a layout based on node signatures is a layout that positions nodes such as their x-coordinate is their degree and their y-coordinate is their clustering coefficient (i.e., the density of edges in their neighborhood). Other x- or y- dimensions can be used such as numerical node attributes or the hash value of string attributes. Such layouts based on node signatures have at least two potential benefits.

First, these layouts are *deterministic*, i.e., they yield the same node positions for the same graphs, even after their nodes have been reordered. Most of them are also *stable*, i.e., they yield only slight changes after small modifications of the graph. These layout schemes contrast significantly with the vast majority of layout algorithms such as force-directed ones that are based on optimization. With most layout algorithms, the stability of node positions across several runs is not guaranteed after the nodes have been reordered or the graph has been slightly modified. Furthermore, most of them also depend on an initial placement of nodes that is arbitrary and often chosed to be random. However, they try to acheive an optimal legibility of graphs. In contrast, layouts based on node signatures do not try to produce legible layouts or support traditional node-link visualization tasks such as following links, but their determinism and stability makes them good candidates for rapidly comparing graphs: the graphs just have to be superimposed to be compared. In the example above, two nodes that are the same in two isomorphic graphs will necessarily have the same position, so two isomorphic graphs will necessarily have the same visual representation, i.e., the representation itself is a *graph invariant* [19]. This invariant is naturally minimalistic and too inaccurate for establishing graph isomorphism, but it will often allow users to detect non-isomorphism and further see how much two graphs differ and how.

A second potential benefit of layouts based on node signatures is that they can be treated as visual representations of simple fitness functions for the graph matching problem. As will be illustrated in the case study, Donatien can intepret pairs of layers as matching functions: the similarity between nodes is computed from their Euclidian distance and two nodes will be matched if and only if they are superimposed across the two layers. Since users can move individual nodes, they can easily fix mistakes: for example they can group nodes in the same cluster to indicate they are similar or drag-and-drop a node on top of another to indicate that they are identical. Previous systems support similar drag-and-drop techniques but they either require users to perform the whole matching manually or fix the results of matching algorithms that are ran separately and that they often do not understand. Instead, using our approach, users can initially chose among existing fitness functions that they can preview and fully understand, then adjust them in-place.

In the current revision of Donatien, we implemented three layouts based on simple node signatures that support different matching heuristics. The two first algorithms are general purpose and the third is application-dependent. The first layout is based on labels and uses a monotonic hash function (i.e., it preserves lexicographic ordering) and string size to layout nodes. The second layout is based on structural properties of nodes and has been described above as an example. The third layout, we called ring-based layout, use concentric rings for different types of nodes. In our dataset where nodes can represent documents, keywords, authors or collections, each type of node is laid out on a distinct ring. On a ring, nodes are placed according to the monotonic hash function mentioned above. This allows to compare nodes based on their type and label. Even though the algorithm was specifically developed for this dataset, it can be generalized to other data collections containing different types of nodes.

These are simple examples but one can think of more sophisticated node signatures that use more accurate structural invariants, more attributes or even a combination of both. Good node signatures however strongly depend on the dataset and must ultimately match the comparison needs of experts. In this area, Donatien has an open architecture that makes it possible to integrate new layouts as new comparison needs arise. One limitation to our approach is that most fitness functions used in actual matching algorithms cannot be easily expressed by a 2-D layout. However, our exploration suggests the possibility that the graph matching problem could benefit from having its sophisticated fitness functions replaced with simple 2-D based fitness functions that can be easily understood and corrected by experts.

5 Case study: visual comparison of collection of documents extracted from the infovis benchmark

Subgraphs corresponding to different years were extracted from the infovis benchmarks [16]. The first step of the analysis consists in displaying them, each year appearing on a different layer. A crossing-based interaction allows to rapidly flip through layers as displayed in Figure 3 and see the evolution of topics, authors and papers in the collection and their relative amount. Invariants or frequent repetitions are visually noticeable. For example, keywords such as *massive datasets* are recurrent topics and can be detected by flipping through the layers.

Further, suppose the expert focuses on year 2001 and 2002. He uses a side-by-side approach to customize the fitness function. On the left side of the display, depicted in Figure 1 left, nodes with similar labels are automatically superimposed by the layout based on the node signature described in the previous section. The expert further displays the two collections for years 2001 and 2002 using one of his favourite layout algorithm on the right side of his display as illustrated on the right side of Figure 1. Double line edges are used to provide immediate feedback to the expert to indicate which nodes match according to the nodes that are perfectly aligned on the left side of his display. The expert can then further select a layout that automatically aligns the layers on the right side of his display so that all matching nodes are superimposed. The expert can further zoom-in or out all layers at once or separately to discover the details and context surrounding matching nodes as illustrated in Figure 3.

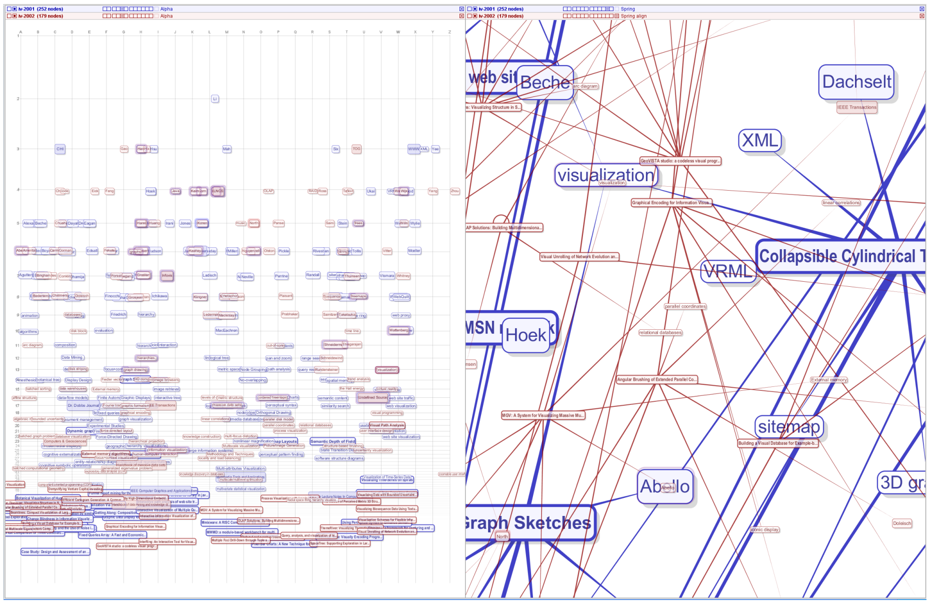

Fig. 3. Label-based layout (left) and layers with multiple zoom factors (right).

6 Conclusion

In this paper we have proposed a new interaction and visual model for the comparison of collection of documents represented as graphs. Our interaction and visualization model is based on multiple layer displays handling different layouts and different levels of zoom. An original multi-scale navigation model emerges from the combination of zooming in and out different superimposed layers at various scales simultaneously. Deterministic layout are proposed as a mean to ensure predictive and stable layout. Combined with a crossing based interaction model, deterministic layouts facilitate flip through layers to best perceive similarities and differences amongst layers. Deterministic layouts can further be used to preview and adjust fitness functions of graph matching. Our initial informal experiment with the comparison of various subsets of documents suggests that the approach offers a lot of benefits. Future work includes experimenting the approach with real users engaged in comparison tasks and design additional deterministic layout algorithms as new needs arise.

References

1. Andrews, K., Wohlfahrt, M., Wurzinger, G.: Visual graph comparison. In: IV. pp. 62–67 (2009)
2. Beaudouin-lafon, M., Mackay, W.E., al.: Cpn/tools: A post-wimp interface for editing and simulating coloured petri nets. In: ICATPN 2001. pp. 71–80. Springer-Verlag (2001)
3. Bederson, B.B., Grosjean, J., Meyer, J.: Toolkit design for interactive structured graphics. IEEE Trans. Software Eng. 30(8), 535–546 (2004)
4. Bederson, B.B., Meyer, J., Good, L.: Jazz: an extensible zoomable user interface graphics toolkit in java. In: UIST. pp. 171–180 (2000)
5. Brandes, U., Corman, S.R.: Visual unrolling of network evolution and the analysis of dynamic discourse. In: IEEE Symposium on Information Visualization. pp. 145–151 (2002)
6. Brandes, U., Dwyer, T., Schreiber, F.: Visual understanding of metabolic pathways across organisms using layout in two and a half dimensions. Journal of Integrative Bioinformatics 1 (2004)
7. Collberg, C., Kobourov, S., Nagra, J., Pitts, J., Wampler, K.: A system for graph-based visualization of the evolution of software. In: Software Visualization (2003)
8. Collins, C.: Docuburst: Document content visualization using language structure. In: Proc. of IEEE Symp. on Information Visualization (InfoVis 2006), Poster Session, Proceedings Compendium. pp. 112–113. IEEE Press, Baltimore, USA (2006)
9. Collins, C., Carpendale, M.S.T.: Vislink: Revealing relationships amongst visualizations. IEEE Transactions on Visualization and Computer Graphics 13, 1192–1199 (2006)
10. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. International Journal of Pattern Recognition and Artificial Intelligence 18, 265–298 (2004)
11. Diehl, S., Gorg, C.: Graphs, they are changing dynamic graph drawing for a sequence of graphs. In: Symposium on Graph Drawing (2002)

12. Dwyer, T., Marriott, K., Stuckey, P.J.: Fast node overlap removal - correction. In: Graph Drawing. pp. 446–447 (2006)
13. Erten, C., Harding, P.J., Kobourov, S.G., Wampler, K., Yee, G.V.: Graphael: Graph animations with evolving layouts. In: Symposium on Graph Drawing. pp. 98–110 (2003)
14. Erten, C., Kobourov, S.G., Le, V., Navabi, A.: Simultaneous graph drawing: Layout algorithms and visualization schemes. In: Symposium on Graph Drawing. pp. 437–449 (2006)
15. Euzenat, J., Valtchev, P.: Similarity-based ontology alignment in OWL-lite. In: de Mántaras, R.L., Saitta, L. (eds.) Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-04). pp. 333–337. IOS Press (2004)
16. Fekete, J.D., Grinstein, G., Plaisant, C.: Ieee infovis 2004 contest, the history of infovis (2006), `www.cs.umd.edu/hcil/iv04contest`, [Online; accessed 16-June-2011]
17. Fekete, J.D.: A multi-layer graphic model for building interactive graphical applications. In: Proceedings of the conference on Graphics interface '92. pp. 294–300. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1992)
18. Fekete, J.D., Beaudouin-Lafon, M.: Using the multi-layer model for building interactive graphical applications. In: Proceedings of UIST'96. pp. 109–118. ACM, New York, NY, USA (1996)
19. Fortin, S.: The graph isomorphism problem. Tech. rep., University of Alberta (1996)
20. Granitzer, M., Sabol, V., Onn, K.W., Lukose, D., Tochtermann, K.: Ontology alignmentï£¡a survey with focus on visually supported semi-automatic techniques. Future Internet 2(3), 238–258 (2010), `http://www.mdpi.com/1999-5903/2/3/238/`
21. Holten, D., Wijk, J.J.V.: Visual comparison of hierarchically organized data. Computer Graphics Forum 27, 759–766 (2008)
22. Keim, D.A., Oelke, D., Rohrdantz, C.: Analyzing document collections via context-aware term extraction. In: Horacek, H., Métais, E., Muñoz, R., Wolska, M. (eds.) NLDB. Lecture Notes in Computer Science, vol. 5723, pp. 154–168. Springer (2009)
23. Lanzenberger, M., Sampson, J.: Alviz - a tool for visual ontology alignment. In: International Conference on Information Visualisation. pp. 430–440 (2006)
24. Munzner, T., Guimbretière, F., Tasiran, S., Zhang, L., Zhou, Y.: Treejuxtaposer: scalable tree comparison using focus+context with guaranteed visibility. ACM Trans. Graph. 22, 453–462 (July 2003)
25. Myers, B.A.: Window interfaces: A taxonomy of window manager user interfaces. IEEE Computer Graphics and applications pp. 65–84 (1988)
26. Noack, A.: Energy models for graph clustering. J. Graph Algorithms Appl. 11(2), 453–480 (2007)
27. Plaisant, C., Fekete, J.D., Grinstein, G.: Promoting insight-based evaluation of visualizations: From contest to benchmark repository. IEEE Transactions on Visualization and Computer Graphics 14, 120–134 (January 2008), `http://dx.doi.org/10.1109/TVCG.2007.70412`
28. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal 10, 334–350 (December 2001)
29. Shneiderman, B.: Why not make interfaces better than 3d reality? IEEE Computer Graphics and Applications 23(6), 12–15 (2003)
30. ping Yee, K., Fisher, D.: Animated exploration of dynamic graphs with radial layout. In: in INFOVIS. pp. 43–50 (2001)