

ChainGraph: A New Approach to Visualize Shared Properties in Resource Collections

Philipp Heim

(University of Stuttgart, Visualization and Interactive Systems, Germany
philipp.heim@vis.uni-stuttgart.de)

Steffen Lohmann

(University of Duisburg-Essen, Interactive Systems and Interaction Design, Germany
steffen.lohmann@uni-due.de)

Abstract: Common graph visualizations tend to produce edge crossings and overlaps when used to display resource collections that are highly interrelated via shared properties. This hampers visual exploration and understanding of relationships between resources and can negatively affect information and knowledge management. In this paper, we present a new approach that visualizes resources and their shared properties in chains to prevent dense graphs and to better support the exploration of relationships. We explain the basic idea, describe an appropriate algorithm and discuss optimization issues. Furthermore, we report on a comparative evaluation showing that this kind of graph visualization supports particularly the visual tracking of relationships and the identification of commonalities between resources.

Key Words: graph visualization, resource collections, information management, shared properties, relationship discovery, interrelated resources, chain arrangement, visual exploration

Category: H.3.3, H.5, M.7

1 Introduction

Many activities in information and knowledge management are concerned with discovering, understanding and exploring relationships within resource collections [Marchionini 2006]. Often, relationships are based on shared properties, for instance, e-mails might be linked by same subject headings or sender addresses, books might have the same author or publisher. However, finding and following these relations can be difficult for system users, mostly due to the constraints of the presentation forms that are used to display the resources and their properties. For instance, tables, matrices, and spreadsheets are highly efficient for many sorting, searching and filtering purposes but are poor in visualizing relationships along multiple dimensions. Alternative presentation forms are needed, particularly for the visualization of relationships that result from shared properties.

1.1 Using Graphs to Visualize Shared Properties

Graphs are highly appropriate to visualize multiple relationships within a limited set of resources [Mutton and Golbeck 2003]. A common way of presenting resources and

their properties as nodes connected by edges is shown in Fig. 1a: All resources that share a certain property are linked to it and are therefore also indirectly related to each other via this property (e.g., resources A, B, C, and D are all connected via property 3 in Fig. 1a). This kind of graph visualization is well-known and widely used for showing different types of relationships in a variety of fields. However, when used to visualize resources that are highly interrelated via shared properties, this visualization type has two main limitations regarding readability and usability:

- A high number of shared properties leads to many n-ary relations, resulting in a rather dense graph.
- A dense graph increases the probability of crossing and partially overlapping edges and nodes and can thus hamper visual tracking of relations or even lead to misinterpretations [Frasincar et al. 2006].

In order to overcome these limitations we developed an advanced graph visualization – the *ChainGraph* – that particularly supports the exploration of relationships and commonalities between resources by arranging their shared properties in chains (see Fig. 1b). Basically, it visualizes indirect, n-ary relations between resources as one-to-one relations and thus resolves high densities and edge crossings. Shared properties can easily be identified and interrelated resources can be explored by following the chains.

In the following, we present this approach in more detail. First, we explain the general idea, describe a generation algorithm and give an example in Section 2. In Section 3, we report on results of a user study where we compared the ChainGraph approach with a common graph visualization, in particular with respect to their ability to support the exploration of shared property relations in highly interconnected resource collections.

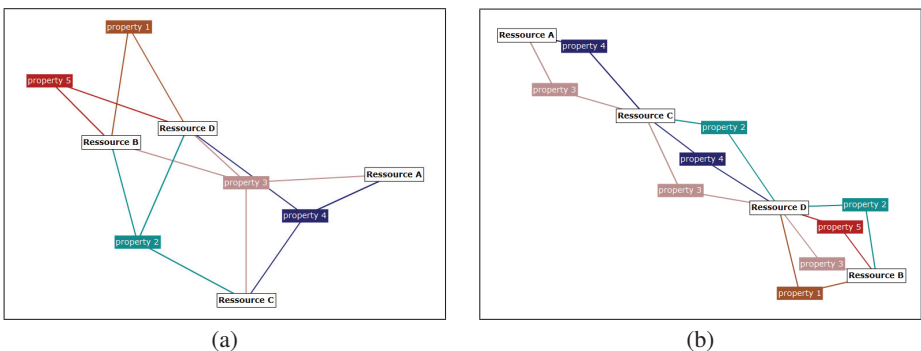


Figure 1: Visualizing resources and shared properties with (a) a common graph and (b) the ChainGraph in a force-directed layout [Fruchterman and Reingold 1991]

2 The ChainGraph Approach

The basic idea of the ChainGraph approach is best described by comparing it with a common way of visualizing resource collections in a graph as it is shown in Fig. 1a. In common graph visualizations, each resource and each property is typically represented by exactly one node. If a property is shared by many resources, the force-directed layout [Fruchterman and Reingold 1991] arranges the resource nodes radially around the property node (cp. property 3 in Fig. 1a). In the ChainGraph, by contrast, property nodes are connected with only two resource nodes at most. This is realized by multiplying property nodes in the visualization: Every property that is shared by more than two resources is represented by several nodes arranged in a chain that connects the resource nodes (cp. property 3 in Fig. 1b). Although this multiplication increases the total number of nodes and edges in the graph, it can significantly reduce the graph's density and the number of crossing and overlapping edges compared to common visualizations. As a consequence, connections within a chain must be interpreted in a transitive manner, i.e., all resources along one chain are related to each other via the same property, independently of their order.

However, in opposite to common graph visualizations, the order in which resources are added has a strong impact on the layout of the resulting ChainGraph. This is best illustrated by a small example.

2.1 Example

Suppose we have a set of resources, each of them with certain properties, as it is shown in Table 1. Furthermore, suppose that these resources should be visualized in a ChainGraph with shared properties arranged as chains.

	Property p1	Property p2	Property p3	Property p4	Property p5
Resource A			+	+	
Resource B	+	+	+		+
Resource C		+	+	+	
Resource D	+	+	+	+	+

Table 1: Resources and shared properties

Fig. 2 shows two alternative ways of adding resources to generate a ChainGraph. The resulting visualizations both display the data that is given in Table 1, however, the left layout is rather hard to read whereas the right one shows an optimal ChainGraph. Being aware of these differences, we developed an algorithm that generates ChainGraphs with optimal layouts for given sets of resources and shared properties.

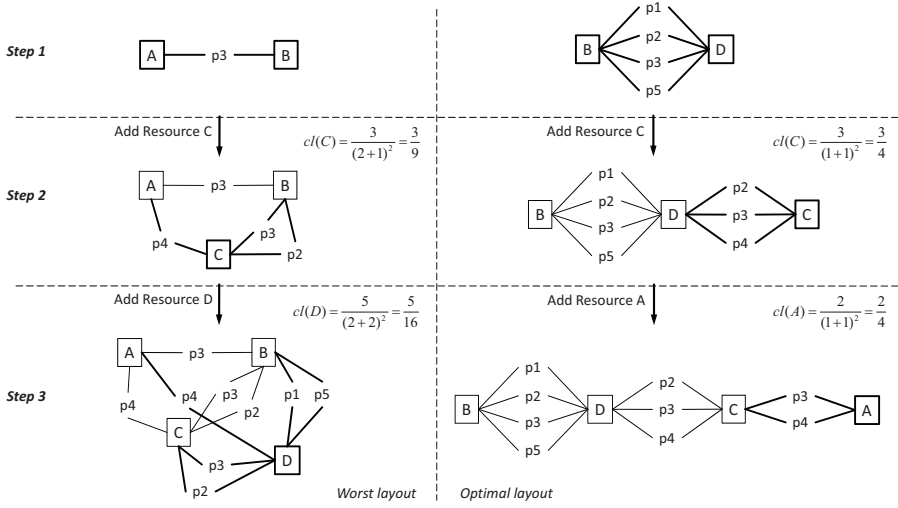


Figure 2: The order in which resources are added to generate a ChainGraph highly affects the quality of the resulting layout.

2.2 Algorithm

The algorithm first defines what data is given and what should be computed (see Algorithm 1). Note that the ChainGraph is described as a set of chains connecting the resources. Thus, every property is represented by a chain, independently of the number of resources that shares a property.

Algorithm 1 Definitions

- 1: $R = \{r_1, \dots, r_l\}$ // given set of resources
 - 2: $P = \{p_1, \dots, p_m\}$ // given set of properties
 - 3: $f : R \rightarrow \mathcal{P}(P)$ // given mapping: every resource has certain properties
 - 4: // ChainGraph definition:
 - 5: $C = \{(e_1, \dots, e_n) \in R^n \mid n \in \{1, \dots, l\}\}$ // set of chains connecting the resources
 - 6: $g : C \leftrightarrow P$ // bijective: there is one chain for each property
 - 7: $addAtFront : (x, (e_1, \dots, e_n)) \rightarrow (x, e_1, \dots, e_n)$
 - 8: $addAtEnd : ((e_1, \dots, e_n), x) \rightarrow (e_1, \dots, e_n, x)$
-

Starting from a set of resources R and their properties $f(R)$, Algorithm 2 adds the resources step by step to the chains. In every step, two decisions need to be made: which resource to add next and where to add it. Generally, we want resources that share many properties (i.e., have many commonalities) to be placed as close as possible to each

other. This is already relevant when drawing the first pair of resources. Consequently, we start with the two resources that share most properties (see Algorithm 2, line 1 and 2, or Fig. 2, step 1).

Algorithm 2 Adding resources to the chains

Require: $R, f(R), C$ // a set of resources and their properties, plus the empty chains

- 1: $InitialPair = \{a, b \in R \mid \forall c, d \in R : (|f(a) \cap f(b)| \geq |f(c) \cap f(d)|)\}$
- 2: $addToChains(InitialPair, C)$ // pair of resources with most shared properties
- 3: $UnboundR = R - InitialPair$ // set of resources that is not bound to chains yet
- 4: **while** $Unbound \neq \emptyset$ **do**
- 5: $next = null$
- 6: $maxCLevel = -1$ // initial max constraint level
- 7: **for all** $x \in Unbound$ **do**
- 8: **if** $constraintLevel(C, x) > maxCLevel$ **then**
- 9: $next = x$ // new most constrained resource
- 10: $maxCLevel = constraintLevel(C, x)$
- 11: **end if**
- 12: **end for**
- 13: $addToChains(next, C)$ // add most constrained resource
- 14: $Unbound = Unbound - next$
- 15: **end while**

Besides the number of shared properties, some additional issues need to be considered when selecting the resources that should be added next. With the *constraintLevel* we defined a heuristic value that selects resources in an order that leads to an optimal ChainGraph layout. Given a set of chains C , the *constraintLevel* of a certain resource x is computed by the following formula:

$$constraintLevel(C, x) = \frac{numSharedProps(C, x)}{(\minNumConnectedRes(C, x) + |Alternatives(C, x)|)^2}$$

The formula weights the selection priority of all resources that have not already been added to the graph based on the following three heuristic strategies:

1. *numSharedProps*: If a resource shares many properties with the already drawn resources it also produces many new edges (i.e., many shared properties = higher constraint level).
2. *minNumConnectedRes*: If a resource would get connected to only a few of the already drawn resources it would produce a more parallel arrangement of the chains (i.e., few connected resources = higher constraint level).

3. *Alternatives*: If there are only few alternatives to add a resource in a way that it gets connected to the found minimal number of already drawn resources (*minNumConnectedRes*) it should be added as early as possible (i.e., few connection alternatives = higher constraint level).

In each generation step, the *constraintLevel* is calculated for all resources that have not yet been added to the graph (see Algorithm 2, lines 4-12). The resource with the highest *constraintLevel* is added next (line 13)¹.

3 Evaluation

In a first evaluation, we tested if system users understand the general visualization approach of the ChainGraph and if they benefit from it when exploring resource collections that contain shared properties. We performed a comparative user study where we visualized resources and shared properties both with the ChainGraph and a common graph visualization (as shown in Fig. 1). Since we were particularly interested in how well the graphs support the visual tracking of shared properties and the identification of commonalities we defined the following three user tasks:

1. Find the pair of resources that shares most properties.
2. Find all properties that are shared by a given pair of resources.
3. Find all properties that are shared by a given triple of resources.

Twelve participants, mainly students, took part in the study, with an average age of 29 (ranging from 22 to 47). All subjects had experiences with graph visualizations, by a general familiarity of 7.7 (median of 8.5) on a scale of 1 to 10. We presented the two types of graphs along with the three tasks to all participants. Each graph type and each task were shortly introduced and explained by an example. To control learning effects, we interchanged the presentation order of the two graph types and randomly assigned the participants to one of the settings. We kept the number of resources and properties constant (six each) but varied the number of interrelations (i.e., the ratio of shared properties). Overall, we thus applied a 2x3x3 within-subject design with variables *graph type*, *task* and *shared properties ratio*. The participants had to complete an evaluation sheet after solving all three tasks with one of the two graph types. They were additionally asked to compare both types in a final questionnaire.

3.1 Results

Overall, the ChainGraph performed very well in the user study. Nine of the twelve participants preferred using the ChainGraph to solve the tasks of the study. It also reached

¹ The resource is added in a way that it gets connected to the found minimal number of already drawn resources (*minNumConnectedRes*).

slightly better results in the evaluation sheets: Fig. 3a shows the user ratings on the four dimensions *attractiveness*, *control*, *understandability* and *effectiveness* that were generated from the items of the evaluation sheet (higher value = better rating).

Furthermore, the participants quickly understood the ChainGraph layout and did not report serious difficulties in using it to accomplish the tasks; the colored edges proved to be helpful in following the chains (cp. Fig. 1b). Although the compactness of the common graph was considered positive, the study participants complained about the high number of crossings edges and overlapping nodes in this type of visualization (cp. Fig. 1a). Regarding the average time that was needed to accomplish the tasks, the ChainGraph performed significantly better in the first task (see Fig. 3b). This indicates that the specific ChainGraph layout assists particularly in the identification of commonalities. Of course, the results are limited to the tasks of the user study and need further validation in real use cases with other user groups.

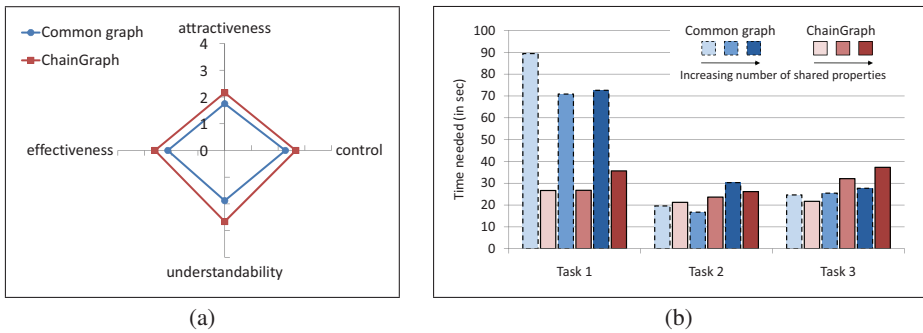


Figure 3: Results from the comparative study: a) user ratings for both graph types on the four evaluation dimensions, b) time needed to accomplish the tasks

4 Discussion

With the ChainGraph we introduced a new visualization approach for shared properties in resource collections. Since the ChainGraph represents shared properties by multiple nodes it avoids an agglomeration of resource nodes around property nodes and thus reduces the graph's general density. This decreases the probability for edge crossings and overlaps and therefore fosters readability and usability of the graph visualization.

4.1 Limitations

Depending on the distribution of properties, the ChainGraph approach might provide only limited support or no benefits at all. For instance, the ChainGraph visualization

would be identical to a common graph visualization in distributions where all properties are shared by exactly and only two resources; albeit such distributions are very unusual in real application scenarios. Another case are resource collections with no shared properties: The ChainGraph approach supports the exploration, discovery, and understanding of relationships and commonalities based on shared properties but offers no specific visualization for properties that are not shared at all. A general limitation of the ChainGraph is its relatively large size that results from the multiplication of property nodes. We developed the ChainGraph for the visualization of a limited set of resources of interest but not as a visualization for the whole structure of large resource collections.

4.2 Related Work

A lot of research has been conducted with the aim of drawing more understandable and readable graphs [Di Battista et al. 1994]. The reduction of edge crossings is one major goal in many of the proposed optimizations. Several algorithms have been developed that tackle this problem with various heuristics (e.g., by simulated annealing [Davidson and Harel 1989], by multidimensional scaling [Kruskal and Seery 1980] or by a combination of heuristics [Tunkelang 1992]). However, none of these optimization approaches applies mechanisms that multiply nodes or arrange nodes in chains to improve relationship discovery and exploration. With the ChainGraph we thus proposed a visualization approach that is of particular interest to the field of information and knowledge management. It might provide a valuable alternative to common presentation forms for resource collections when relationships and commonalities between the resources are in the user's interest.

4.3 Future Work

In this paper, we focused on the description and evaluation of a static ChainGraph visualization. We have not discussed benefits resulting from an interactive implementation where certain chains of properties can be shown or hidden and gradually extended by the user in order to follow some "information scent" [Pirolli et al. 2003]. A closer look at the information seeking strategies that users apply when exploring resource collections with the ChainGraph remains a topic for future work.

Acknowledgments

We thank Lena Tetzlaff who assisted with the comparative user study. Parts of this work were funded by the BMBF under project grant 01-ISF-02-C.

References

- [Davidson and Harel 1989] Davidson, R., Harel, D.: "Drawing Graphs Nicely Using Simulated Annealing"; Technical Report CS 89-13, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot (1989).

- [Di Battista et al. 1994] Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: “Algorithms for Drawing Graphs: An Annotated Bibliography”; *Computational Geometry* (1994), 235-282.
- [Frasincar et al. 2006] Frasincar, F., Telea, A., Houben, G.-J.: “Adapting Graph Visualization Techniques for the Visualization of RDF Data”; *Visualizing the Semantic Web* (2005), Springer, 154-171.
- [Fruchterman and Reingold 1991] Fruchterman, T., Reingold, E.: “Graph Drawing by Force-Directed Placement”; In: *Softw. Pract. Exper.*, John Wiley & Sons (1991), 1129-1164.
- [Kruskal and Seery 1980] Kruskal, J.B., Seery, J.B.: “Designing Network Diagrams”; *Proc. 1st General Conference on Social Graphics*, Washington, D.C. (1980), 22-50.
- [Marchionini 2006] Marchionini, G.: “Exploratory Search: From Finding to Understanding”; *Commun. ACM*, 49, 4 (2006), 41-46.
- [Mutton and Golbeck 2003] Mutton, P., Golbeck, J.: “Visualization of Semantic Metadata and Ontologies”; *Proc. 7th Int. Conference on Information Visualization*, IEEE (2003), 300-305.
- [Pirolli et al. 2003] Pirolli, P., Card, S., Van Der Wege, M.: “The Effects of Information Scent on Visual Search in the Hyperbolic Tree Browser”; *ACM Trans. Comput.-Hum. Interact.*, 10, 1 (2003), 20-53
- [Tunkelang 1992] Tunkelang, D.: “An Aesthetic Layout Algorithm for Undirected Graphs”; M.S. Thesis, Department of Electrical Engineering and Computer Science, MIT (1992).