Eurographics Conference on Visualization (EuroVis) 2016
K.-L. Ma, G. Santucci, and J. van Wijk
(Guest Editors)

Volume 35 (2016), Number 3

Pathfinder: Visual Analysis of Paths in Graphs

C. Partl[1], S. Gratzl[2], M. Streit[2], A. M. Wassermann[3], H. Pfister[4], D. Schmalstieg[1], and A.Lex[5]

[1]Graz University of Technology, Austria
[2]Johannes Kepler University Linz, Austria
[3]Pfizer, USA
[4]Harvard University, USA
[5]University of Utah, USA

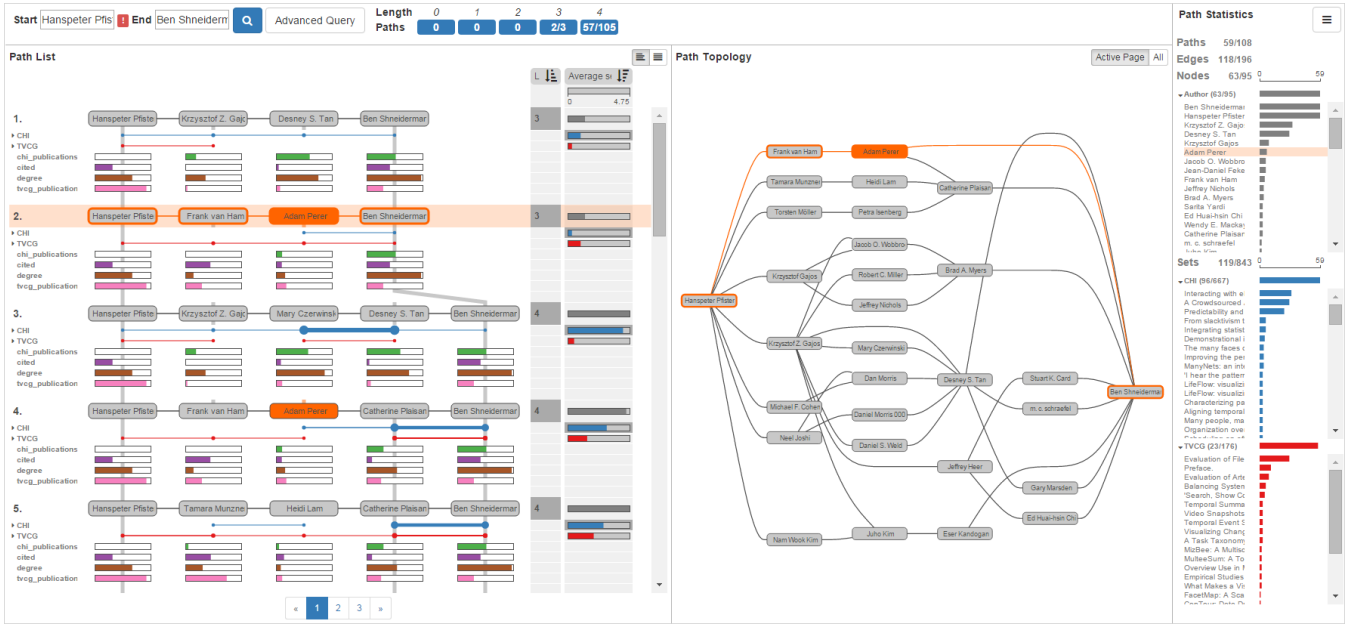


Figure 1: *Pathfinder visualizes multiple paths of a coauthor graph connecting Hanspeter Pfister and Ben Shneiderman. The paths are shown in a ranked list together with associated sets and attributes on the left (path list view). To its right, a node-link diagram shows the topology of the paths (path topology view). The path statistics view on the far right shows an overview of the properties of the paths.*

Abstract

The analysis of paths in graphs is highly relevant in many domains. Typically, path-related tasks are performed in node-link layouts. Unfortunately, graph layouts often do not scale to the size of many real world networks. Also, many networks are multivariate, i.e., contain rich attribute sets associated with the nodes and edges. These attributes are often critical in judging paths, but directly visualizing attributes in a graph layout exacerbates the scalability problem. In this paper, we present visual analysis solutions dedicated to path-related tasks in large and highly multivariate graphs. We show that by focusing on paths, we can address the scalability problem of multivariate graph visualization, equipping analysts with a powerful tool to explore large graphs. We introduce Pathfinder, a technique that provides visual methods to query paths, while considering various constraints. The resulting set of paths is visualized in both a ranked list and as a node-link diagram. For the paths in the list, we display rich attribute data associated with nodes and edges, and the node-link diagram provides topological context. The paths can be ranked based on topological properties, such as path length or average node degree, and scores derived from attribute data. Pathfinder is designed to scale to graphs with tens of thousands of nodes and edges by employing strategies such as incremental query results. We demonstrate Pathfinder's fitness for use in scenarios with data from a coauthor network and biological pathways.

Categories and Subject Descriptors (according to ACM CCS):
H.5.2 [Information Systems]: Information Interfaces and Presentation—User Interfaces–Graphical user interfaces

1. Introduction

Graphs capture relationships between items, for example, friendships between people in social networks, interactions of genes in biological networks, or researchers coauthoring scientific papers in collaboration networks. Graph analysis and visualization have always been important for scientific discovery and decision-making, but their role and ubiquity have increased in the last decade. It is now common to encounter networks that cannot be sensibly drawn due to both computational and perceptual constraints. The analysis of graphs of nontrivial size depends on a combination of algorithmic, statistical, and visual approaches [vLKS*11]. Also, interaction, for example, through queries, plays a critical role in tackling scalability problems.

Graphs are also increasingly associated with rich node and edge attributes. In many cases, only the combined analysis of attributes and topology can lead to meaningful insights. However, the visualization and analysis of these rich attributes present additional challenges [KPW14], as there is a trade-off between optimizing a layout for conveying topology and attributes. As more data is collected and the graphs become bigger, scalable methods to extract knowledge and reason about them become more important. Many tasks on such large graphs cannot be addressed by showing all nodes and links, even if a layout could be drawn.

An important class of tasks for graph analysis is concerned with paths. Learning about how two suspects are connected in a criminal case or understanding why two genes are co-regulated are examples of important domain tasks that can be abstracted to path analysis tasks. For small networks, these path analysis tasks can be solved by visually finding paths, for example, in a node-link layout. In larger networks, however, queries are essential to enable these tasks. In this paper, we introduce methods to comprehensively address path analysis tasks.

Our primary contribution is Pathfinder, a technique for the visual analysis of paths queried from large and multivariate networks. We introduce methods to (a) interactively query for paths and dynamically refine queries, (b) visualize the resulting paths and their relationships, (c) investigate the attributes associated with the paths, and (d) rank and compare paths. We also have developed a list of requirements for path analysis that we use to justify and evaluate our design and analyze related techniques.

We realize and test our technique in a prototypical implementation. We pay particular attention to scalability, so that we can handle tens of thousands of nodes and edges interactively. We employ strategies to deliver fast and progressive results [FPDs12], i.e., paths are added to the visualization as soon as they are found. These immediate results can be used to evaluate and refine queries, even before a complete and comprehensive answer is available. A demo version of Pathfinder is available at `http://demo.caleydo.org/pathfinder/`.

We show in a use case that our methods and the Pathfinder tool can help answer important questions about gene regulatory networks, when analyzing the networks in the context of rich experimental measurements.

2. Tasks and Requirements

We introduce a set of requirements for a query-based path visualization technique. In our requirement analysis, we focus on static networks that are highly multivariate, i.e., the nodes are associated with rich attributes and additional sets. Our assumptions on datasets are described in detail in Section 4. We use these requirements to evaluate the related work, to analyze our Pathfinder technique, and to identify areas of future work. Our choice of requirements is based on discussions with potential end-users, our analysis of the literature on path visualization, and the task taxonomy for general graph visualization by Lee et al. [LPP*06b]. Lee at al. distinguish topology-based tasks and attribute-based task, as well as browsing and overview tasks. The browsing tasks are related to topology (*follow path* and *revisit*) and the overview task is concerned with analyzing general properties of the graph, such as estimating the overall size. For our requirements, we assume an underlying fundamental task of exploring paths. Some of the tasks introduced by Lee et al., especially regarding connectivity (identifying clusters, connected components, bridges, and articulation points), cannot be addressed with a pure path-based approach. Hence, we envision Pathfinder to be part of a larger graph-visualization system to address those tasks in the future.

R I: Query for paths. Users should be able to easily query for paths that adhere to some criteria. A simple query searches for the paths connecting two nodes. Other criteria, such as querying based on sets (find short paths that connect node A with any node in set S) or topological restrictions (find short paths from A to C that do not go through B), must also be supported. It should be easy to refine an existing query, as analysts often can identify restrictions once they see results matching their initial query.

R II: Visualize attributes. Many networks contain rich and heterogeneous attributes for nodes and edges. Understanding these attributes is often critical for judging paths. In a gene regulatory network, for example, low values for associated experimental data can tell analysts that the path is inactive for the given samples.

R III: Visualize group structures in paths. Group structures, such as set relationships and clusters, provide additional information about relationships between nodes. For example, they are important when judging the relevance of an edge in a path. If two connected genes occur in many pathways, for example, it is likely that their relationship is important.

R IV: Rank paths. A common goal of users querying for paths is to find "good" paths, according to some criteria. These criteria sometimes are as simple as finding the shortest path, but can also involve a more intricate combination of topological features and attributes. A path visualization technique should allow its users to dynamically define these criteria and rank paths according to them.

R V: Visualize topology context. The relevance of a path for an analysis can be influenced by its surrounding topology properties. For example, in a gene-regulatory network, feedback cycles are common, and it is important to know whether a path is involved in one of these cycles.

R VI: Compare paths. Comparing paths is important to evaluate similarity or dissimilarity between paths. As for path ranking, comparison can be based on many criteria, such as shared nodes, common set relationships, or similar attribute values. These similarities

and differences are often not easy to spot, so a path visualization technique should make this comparison easier.

R VII: Group paths. It is common that many paths match a specific query, but multiple paths will be only slight variations of each other. Ideally, a system should convey which groups of paths are similar and provide an overview of the main path classes, as this can reveal important paths that do not rank at the very top based on other criteria.

In addition to these specific requirements, a path visualization system must be scalable, as the costs of using and learning a sophisticated path visualization technique mainly pay off for large graphs. Finally, the system also should follow best practices of conventional graph visualization, such as overview (e.g., graph statistics) and details on demand (e.g., full information about sets or nodes).

3. Related Work

The two principal ways to visualize graphs are node-link diagrams and matrices. Node-link diagrams are generally regarded to be advantageous for topology-based tasks, whereas matrix diagrams are considered to be beneficial for tasks on the edge sets, e.g., when visualizing dense networks with edge weights.

Visualizing all nodes and edges, in either node-link or matrix layouts, does not scale to large graphs. There are two strategies to address scale: top-down and bottom-up approaches. **Top-down** approaches start by showing an overview using aggregation and sampling methods, and support drilling down into regions of interest. Luger et al. [LSG*15], for example, employ hierarchical and motif-based aggregation in data provenance graphs. In **bottom-up** approaches, the exploration starts with a small subset of a graph that can then be expanded. For example, in Treeplus [LPP*06a] a graph can be continuously expanded in a flexible tree layout, starting from a seed node. Ham and Perer [vHP09], Abello et al. [AHSS13], and Luger et al. [LSG*15] use more sophisticated methods to select and expand graph subsets based on degree of interest functions.

Our approach belongs to the bottom-up approaches. We start by querying a large network for paths and visualize the result set, which can then be either further expanded or filtered. Most graph libraries and databases support path-queries. The GUESS language by Adar [Ada06], Orion by Heer and Perer [HP14], or the Neo4j database are prominent examples of systems that support such queries. Here, however, we focus on the visual analysis of paths. Figure 2 shows four ways of visualizing paths in networks, where (a)-(c) are variations of a node-link diagram, and (d) makes use of the matrix layout.

Path highlighting in node-link diagrams, as shown in Figure 2 (a), is supported by many graph visualization frameworks and their plugins. Typically, these tools support highlighting a shortest path between two nodes. Examples are the two popular graph visualization tools Cytoscape [SOR*11] and Gephi [BHJ09]. Several tools also show multiple short paths in a node-link diagram, such as TimeArcTrees [GBD09], which presents the shortest paths for each view in a small multiples arrangement of networks, or enRoute [PLS*13], which highlights all paths between two nodes with bubble sets on top of a network.

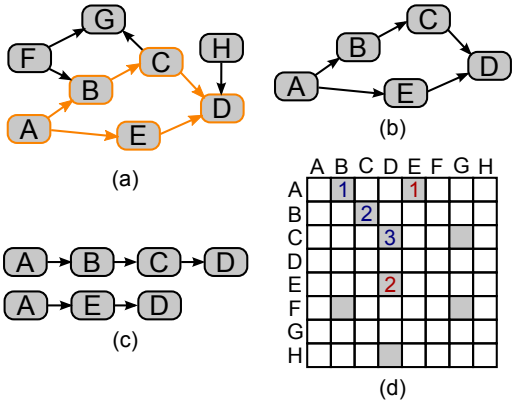


Figure 2: *Four ways of visualizing paths in a graph, connecting the nodes A and D: (a) highlighting in a node-link diagram, (b) drawing only the subset of the graph connecting node A and D, (c) drawing a path list, (d) enumerating edges in a matrix.*

These approaches are very limited in terms of scalability, but they support topology-based tasks (R V, R VI) for smaller networks. They typically do not support rich queries (R I), but rather rely on manually selecting two nodes, for which the path is calculated. General purpose node-link diagrams are also not ideal for visualizing rich attributes [PLS*13] (R II) and have limited scalability with respect to visualizing group structures [VBW15] (R III).

Rendering paths as node-link diagrams, as shown in Figure 2 (b), is similar to the approach described above, with the critical difference that *only* nodes and edges of the paths are rendered. As a consequence, this approach scales to much larger networks, yet makes judging the topology context (R V) harder.

Most examples given in this and the next section are for semantic web networks, which capture relationships between many types of entities. A typical example is to search for relationships between people, e.g., to visualize how Albert Einstein and Kurt Gödel are connected. These networks are highly heterogeneous, containing a multitude of node and edge types. For such networks, attribute and set relationships are of only limited interest, and consequently, few of the techniques support them (R II, R III).

RelFinder [HHL*09] is an example of such a technique for semantic web networks. It supports both queries and dynamic query refinement (R I). RelFinder, however, has no means to rank paths (R IV) and does not provide dedicated methods to compare paths (R VI). Tekusova and Kohlhammer [TK08] show paths connecting two nodes (companies in that case), but in contrast to many other approaches, they also show important nodes that are not part of the paths (R V).

Path lists (Figure 2 (c)) are also mostly based on queries, but instead of visualizing each node only once in a node-link diagram, each path is visualized in a separate row as part of a list. Nodes that occur in multiple paths are duplicated. Two examples that use this list-based approach to visualize semantic web data and also support rankings of paths (R IV) are the work by Aleman et al. [AMHWA*05] and SemRank [AMS05]. The former ranks path as a combination of static attributes, such as path length and trust, whereas the latter computes a semantic score. Neither of those

rankings is interactive, and neither of the tools supports requirements beyond ranking and querying. Two semantic web tools support aggregation of paths into classes (R VII). Explass [CZQ14] aggregates on a single level, whereas RelClus [ZCQ13] uses hierarchical aggregation. Both tools also support the refinement of queries (R I) based on facets, but do not address other requirements.

enRoute [PLS*13] uses a hybrid path list/path highlighting approach. In addition to visualizing the path in the node-link layout, users can select a single path, which is shown in a linear layout in a separate view for the purpose of multivariate data visualization (R II). Entourage [LPK*13], an extension of enRoute, also shows basic set relationships. enRoute does not support advanced queries (R I) and is limited to the analysis of single paths. However, we adopt its approach to attribute visualization and extend it to the general case of multiple paths.

Matrices without extensions are considered ill-suited for topology-based tasks and, consequently, are rarely used to visualize paths. There is, however, an extension by Henry and Fekete, Matlink [HF07], that supplements a matrix with explicit links between the rows and columns, respectively. MatLink also automatically calculates the shortest paths between two selected nodes. The paper compares path finding tasks in MatLink with a node-link diagram and a matrix that uses highlighting for shortest paths as shown in Figure 2 (d). The authors found that users performed best on shortest path tasks with MatLink. Shen and Ma introduce an augmentation for matrices that visualizes multiple paths as links on top of a matrix [SM07], with the goal of combining the benefits of matrix layouts with good path-finding performance. However, their approach works well for only a limited number of paths.

The specific examples discussed here do not support complex queries (R I), but matrices could be used in combination with queries, just as node-link diagrams. However, they are ill-suited for a larger number of attribute rich paths (R II).

4. Dataset Characteristics and Examples

We consider static graphs where both nodes and edges can be associated with attributes. These attributes can be sets, categories, or numerical data. There can be different types of nodes and edges. For example, a network can contain nodes of types "places" and "people". These types commonly define what attributes are associated with them. In this example, all "places" could be associated with coordinates, whereas all "people" have an age.

Node and edge attributes can be very rich. In many biological networks, nodes of a particular type can be associated with rows or columns of large matrices, containing, for example, gene expression data. These matrices typically contain samples taken from patients or cell lines. The samples can be grouped to compare, for example, different types of cell lines.

Sets can be defined on the level of both nodes or edges, but they are more commonly associated with nodes. We distinguish sets from categories, because they explicitly define additional relationships within a network.

A path through a graph connects two nodes by a sequence of edges and nodes. The length of a path is defined by the number of hops between its start and end nodes. Many applications require identifying short (or cheap) paths between two nodes. The shortest paths connecting two nodes can be computed with, for example, breadth-first search (BFS) or, when considering edge weights, with Dijkstra's algorithm. In practice, however, the shortest path might not be the "best" path, as other factors such as node or edge attributes or contextual knowledge play a role in identifying an important path. In many cases, defining a cost-function to find the cheapest path is not straightforward and requires human judgment. Thus, exploring a number of "candidate" short paths is important. Such paths can be computed with *k*-shortest path algorithms.

In this paper, we use two networks to illustrate Pathfinder. The first is a coauthor network that we created from extracting all ACM CHI and IEEE TVCG papers from the DBLP computer science library (`http://dblp.uni-trier.de/`). We treat authors as nodes and add an edge if two authors have coauthored a publication. We also consider individual papers as sets, i.e., a paper is a set that contains all its authors. As attributes in this network, we provide node degrees, paper counts for each venue, and the number of citations to the visualization papers, which we extracted from the Visualization Publication Dataset [IHK*15]. This network contains about 34k nodes, 45k edges, and 13k sets.

The second network is extracted from the KEGG pathway database. Nodes in KEGG are primarily of two types: genes (proteins) and metabolites (intermediate chemical compounds). Edges describe various kinds of interactions. Genes, for example, can work in signaling cascades that influence each other based on activation levels (gene expression), which in turn can trigger a process, such as cell death. The most important nodes and edges associated with a specific function or disease are summarized in pathways. We treat these pathways as sets of nodes. We supplement this network data with gene expression and copy number data from the CCLE dataset [Bc12]. This dataset contains values for both gene expression and copy numbers for most genes (nodes) for 479 cell lines taken from 23 cancer tissues (originating, e.g., from lung, stomach, thyroid). Gene expression measures how active a gene is, whereas copy number measures how many copies of a gene a specific sample has. Typically, a healthy sample has two copies of each gene, but deletions (0 or 1 copies) or amplifications are possible. In total, this network contains about 11k nodes, 71k edges, and 300 sets (pathways).

5. Method

Our goal was to develop a method for path analysis that is very scalable and addresses the requirements introduced in Section 2. The key to achieve scalability is **path queries**: we introduce a rich query interface that can cover a wide range of questions relevant to path analysis (R I). The visual representation of the graph is manifested in the **path list view**, the **path topology view**, and the **path statistics view**. These views are part of a multiple coordinated view setup that supports linking and brushing of paths and their elements. The path list is ideally suited to visualize attributes and sets (R II, R III) due to the linear layout of the individual paths, which allows us to juxtapose the nodes and edges with plots of the attributes. It is also a perfect match to dynamically rank paths and explore those

Figure 3: *The advanced interface showing a query where the start node is either Hanspeter Pfister or Jarke van Wijk, the paths must not contain Jean-Daniel Fekete, and the last node is Ben Shneiderman or an author of the NodeTrix paper.*

rankings [GLG*13] (R IV). Additionally, the path list can be used to compare paths (R VI), especially with respect to attributes and sets. The shortcomings of the path list are addressed by the path topology view, which provides the topological context (R V) and covers the aspects of comparison with respect to topology (R VI). Also, the topology view is beneficial for an overview and for initial orientation. In the following sections, we introduce the various components of Pathfinder in detail.

5.1. Path Queries

As queries are at the heart of our technique, we introduce a sophisticated graphical interface for querying, paired with choices of algorithms, technologies, and concepts that make the analysis and querying process efficient. The simplest queries are for individual nodes and their neighbors, or for the paths connecting two nodes. In fact, our collaborators mainly used queries of this simple type. To make this simple use case as smooth as possible our default query interface consists of two fields only: one for the start, and one for the end node.

Once such a query is triggered, we run a k-shortest path algorithm based on breadth-first search (BFS) to retrieve the shortest paths between two nodes. While BFS cannot consider edge weights which, for example, Dijkstra's algorithm can, BFS is faster in practice and response time is critical for interactive analysis systems. BFS has a runtime complexity of $O(|N|+|E|)$, whereas Dijkstra's algorithm has a complexity of $O(|N|log(|N|)+|E|)$, where $|N|$ denotes the number of nodes and $|E|$ the number of edges. Also, we can consider weights later in the analysis process through interactive rankings on the resulting set of paths.

Another method to improve scalability is incremental results. Incremental and approximate results have been shown to accelerate and open up the query process [FPDs12]. Consequently, we visualize each path and all its attributes, as soon as it becomes available, even if the search process is still ongoing.

We retrieve new paths until we reach a fixed threshold of k paths. Once we have passed this number, we continue to fetch all paths that are of equal length l to the last path fetched. Using this approach, we can guarantee that we consider all paths of length l in the subsequent analysis. However, in rare cases, the resulting number of paths can be very large. To avoid excessive computation, we also define a maximum threshold on the number of paths, which is significantly larger than k. The fields to the right of the query interface (see Figure 1 top) give an overview of how many paths of which length are in the current result set.

We also support queries between sets of start and end nodes, which can be useful, for example, to find out how an author is connected to the authors of a paper of interest. In addition to the simple query interface, Pathfinder also provides an advanced interface for specifying more complex queries that consider topological restrictions, logical combinations of nodes, and node and edge properties. Topological restrictions can be used to consider, for example, only paths that go through a certain node, or contain a sequence of certain nodes. Logical restrictions can combine nodes and sets through Boolean operations (*AND*, *OR*, *NOT*). Also, Pathfinder can treat set relationships as edges, if desired. Figure 3 shows a complex query example for a coauthor network, with an *OR* combination at both the start and the end node and a topological restriction to exclude all paths that go through the node between them.

Queries can be expressed either in the query interface or by interacting with elements in any of the views. For example, a user can exclude all paths through a certain node via its context menu in the path list view, which simplifies query refinement (R I).

Any change to the query acts as an immediate filter for the current set of paths, which guarantees rapid feedback. This is often sufficient for queries that restrict the result set, but queries that expand or change the result set have to be run against the whole network to produce reliable results.

5.2. Path List View

Pathfinder's main view displays all paths that match a query in a ranked and paginated list of paths (see Figure 1). Each path is displayed as a sequence of nodes and edges. For example, Figure 4 shows the path from *Jean-Daniel Fekete* to *Ben Shneiderman* in the coauthor network. A key benefit of this linear layout is that it allows us to easily show attributes associated with the nodes and edges (R II) and sets connecting the nodes below the path (R III). Compared to nonlinear layouts, this arrangements allows us to display multiple plots for each node. Attributes and sets can be hidden on demand, which allows analysts to focus on the paths exclusively.

Set-memberships of a node are indicated by a circle below the node. If two successive nodes are within the same set, the circles are connected by a line. As shown in red in Figure 4, tracing set

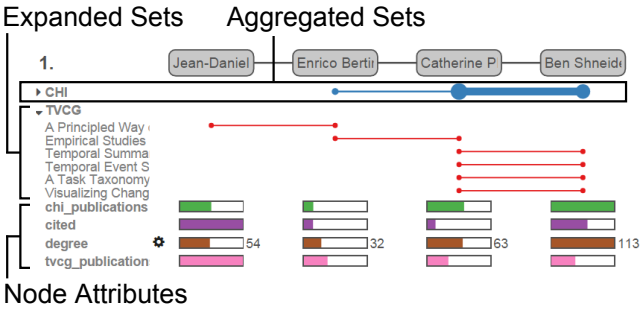


Figure 4: *A path from the coauthor network connecting Jean-Daniel Fekete and Ben Shneiderman. The coauthored papers are treated as sets and visualized below the path. The CHI papers are aggregated—only the strength of the connection can be estimated. We see that Catherine Plaisant is highly connected with Ben Shneiderman. The TVCG papers show details: We can see who has coauthored which papers. Below the sets, numerical attributes, such as the number of publications in each venue, are shown for each node.*

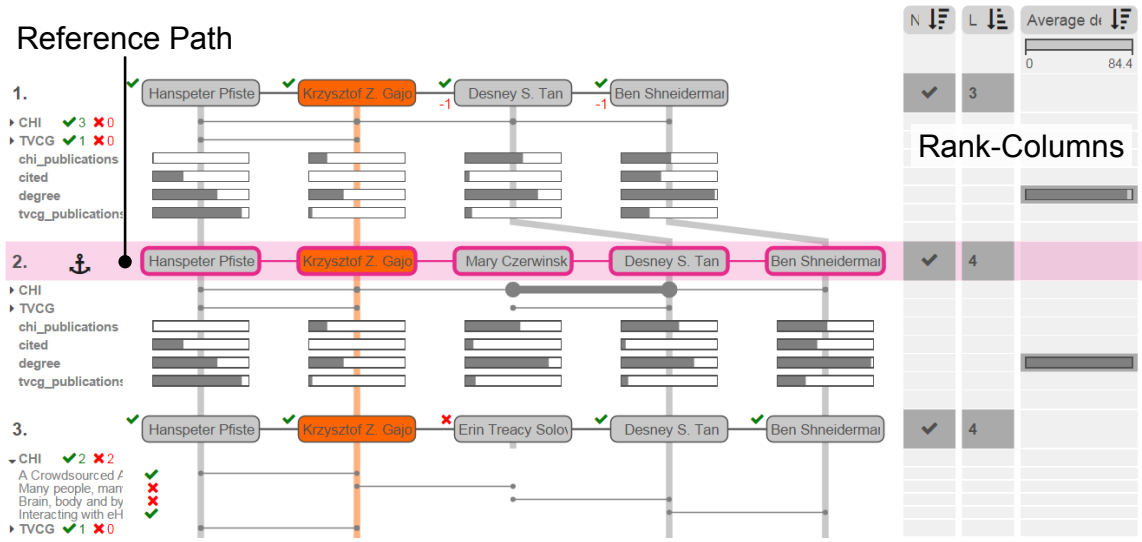


Figure 5: *Path comparison and ranking. The second path (violet) is the reference path. Icons next to the nodes and sets in all paths indicate whether they are shared with the reference path. For example, the third path shares two CHI papers, one TVCG paper, and four authors. The rank columns show different scores by which the paths are ranked. Their order indicates score priority: First, paths are ranked by whether they contain Krzysztof Z. Gajos, second, by length, and third, by the average node degree.*

relationships, such as common publications of authors in a coauthor network, is easy. To save space, the sets can be aggregated by type. These aggregated relationships are scaled according to the number of represented sets: Thicker lines and larger circles represent more sets. Figure 4, for example, shows that *Enrico Bertini* coauthored only a few CHI publications (blue) with *Catherine Plaisant*, but she coauthored many papers with *Ben Shneiderman*. By default, we show only sets that connect nodes; however, all available sets can be shown on demand.

When visualizing attributes, we can show either bar charts (Figure 4) or box plots (Figure 8), but other representations, depending on the data type, the number, and the structure of the associated data, are equally conceivable. Attributes can be aggregated hierarchically. The box plots in Figure 8, for example, summarize hundreds of mRNA and copy number data values.

By default, we assign colors to set types and attribute types, in order to facilitate identification across different views. However, coloring can be disabled on demand.

Taken together, the visual encodings that are made possible by the linear representation allow us to efficiently visualize group structures and attributes in graphs, which are both considered challenging at scale [VBW15, KPW14].

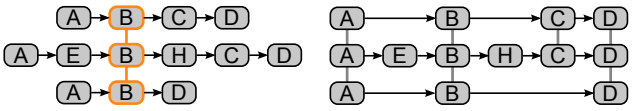
Path Ranking. A core aspect of our method is the dynamic ranking of paths (R IV). In concert with query refinement, dynamic ranking allows us to utilize human intelligence, contextual knowledge, and individual judgment for path analysis. It enables us to consider paths based not only on their topological properties or simple attributes, such as edge weights, but based on complex relationships of attributes, topology, and group structures.

By default, paths are ranked by their length, i.e., number of edges, but other rankings based on node attributes (e.g., minimum, maximum, average value across a path), the number of connecting sets, etc., are easily accessible. In addition, we empower users to write custom scoring functions using a script editor, which enables complex, targeted rankings as they are often desirable in advanced analysis cases.

Pathfinder also supports multiple scores for each path. Each score is represented by a rank-column displayed on the right of the paths. Figure 1, for example, shows two scores: The first score is the length of the path (represented by numbers) and the second score is based on set connection strength (average number of connecting sets, represented by bars). The second score is used to break ties. In this example, the ranking is first driven by the length, then by the connection strength score. An arbitrary number of scores is possible. In the example shown in Figure 5, paths are ranked on top if they contain the selected node *Krzysztof Z. Gajos*, have a short length, and a high average node degree.

Path Comparison Being able to compare and judge the similarity of paths is an important task in path analysis (R VI). The path list view provides a variety of methods that enable path comparison. Nodes shared between adjacent paths are connected with a gray line in the background. When a node is contained in a nonadjacent path, we draw a line-stub pointing in its direction. We also support different modes for node alignment to make shared nodes evident. By default, paths are drawn as compactly as possible, from left to right. On demand, paths can be aligned based on a selected pivot node or based on an optimized global alignment, as illustrated in Figure 6, which makes it easy to spot re-occurring nodes. A drawback of these layouts is that they are less compact than the default, left-aligned layout.

Pathfinder also introduces methods to compare a selected reference path to other paths. Figure 5 shows an example where the second path is the reference path; all other paths show icons next to nodes and sets indicating whether they are shared or not. Shared nodes also indicate their position relative to the same node in the



(a) Align pivot node (b) Align all

Figure 6: *Node-alignments that support path comparison. (a) All paths are aligned around a selected pivot node. (b) A layout that strives to put re-occurring nodes at the same horizontal position.*

reference path. To make these differences and similarities pop out, we disable the coloring of sets and attributes when a reference path is selected. The reference path can also be considered in path ranking: Instead of absolute scores, each score can be computed as a difference to the reference path. As a result, the reference path will be displayed first, followed by paths with decreasing similarity.

5.3. Path Topology View

The path topology view complements the path list view by showing the topology of the nodes in the paths, which addresses the requirement of judging the topology and its context (R V), and providing an overview. By default, we use a layered layout that makes it easy to trace paths from their start at the left to their end at the right (see Figure 1). When working with many paths, the topology view suffers from scalability issues. Zooming and panning helps to reveal details, but we can also show only the paths on the active page of the path list. This addresses the scalability issues and still shows the topology for the top-ranked paths.

To judge the path topology in the context of attributes, a user can select individual node attributes to be mapped onto the nodes. This can be helpful for identifying the relationship between attributes and topology, which is important, for example, when investigating gene regulatory networks.

Some usage scenarios demand contextual topological information that goes beyond the topology of the path result set (R V). Pathfinder provides two ways to add additional topological information: First, the links that connect a node to other nodes in the path result-set, but are not covered by the paths, can be shown on demand. Second, the user may add all neighbors of a selected node, even if they are not in the set of paths (see Figure 7). This can be done repeatedly with neighbors to enable a simple node-by-node graph exploration, which is especially desirable when only querying for start nodes. To better support this approach to graph exploration, Pathfinder also provides a force-directed layout.

5.4. Path Statistics View

The path statistics view provides basic statistics and gives an overview of the most important nodes and sets. For paths, nodes, edges, and sets, it displays the total numbers in the result-set, and the numbers considering active filters. To make the most important nodes and sets apparent, we show them in ranked lists. The nodes and sets are ranked by the number of paths of which they are part of, which is displayed in bar charts. Details about all these entities can be accessed by following links to external web-sites (if available), which can be accessed using the context menu.

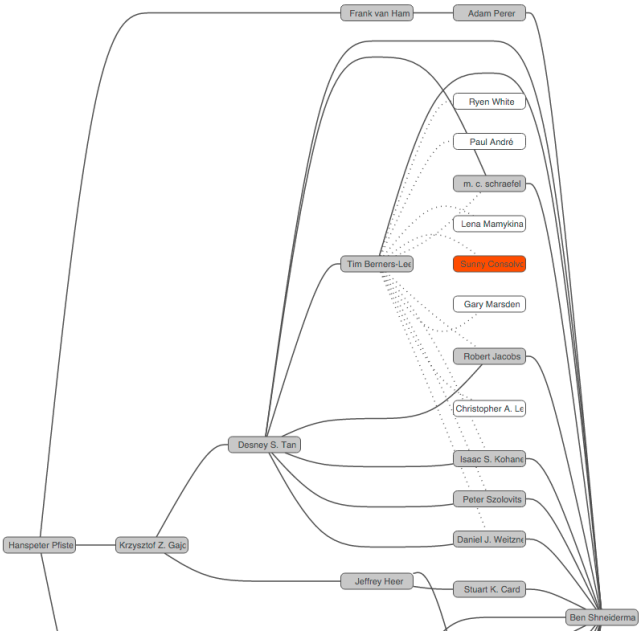


Figure 7: *The path topology view showing paths that connect Hanspeter Pfister and Ben Shneiderman. All neighbors of Tim Berners-Lee were added and are shown in white, indicating that they are not part of any path. Links to those nodes are stippled to distinguish them from links that occur in the path list.*

Figure 1 shows an example where *Hanspeter Pfister* and *Ben Shneiderman* are ranked at the top, which is not surprising, as they are the start and end nodes, respectively, and thus occur in all paths. However, the highly ranked authors *Krzysztof Z. Gajos* and *Desney S. Tan* also play a key role in connecting these authors, as they are part of about half of the paths.

6. Implementation

Pathfinder is implemented as a client-server web-application using Caleydo Web [GGL*15]. The front-end is implemented with HTML, JavaScript, and TypeScript. We use D3 (`http://d3js.org`) for the visual components. The layered graph layout in the topology view is computed using the dagre library [Pet16]. Dagre attempts to balance computation performance and quality of results by using a combination of different algorithms. On the back-end, we use a Python web-server and a Neo4j graph database (`http://neo4j.org`) to store the graph data. We use a custom Neo4j plugin written in Java for the k-shortest path search. It uses the built-in Neo4j BFS algorithms, streams intermediate results via WebSocket, and computes virtual edges for set relationships on-the-fly. The source code of Pathfinder is freely available at `https://github.com/Caleydo/pathfinder`.

7. Use Case: Pathway Analysis

In addition to the examples given on the coauthor network throughout this paper, we demonstrate Pathfinder's value for an intricate analysis in a use case about biological pathways. This use case was conducted by a coauthor of this paper, who is a chemical biologist

and a researcher at a large pharmaceutical company. We provide detailed figures documenting the analysis process in the supplementary material. Pathfinder was developed with constant feedback from this researcher and other domain experts.

Our collaborator used Pathfinder to analyze biological signaling cascades driving cell proliferation and cancer formation. These signaling cascades consist of proteins that are encoded by genes. For simplicity, we use the terms protein and gene interchangeably. A well-known signaling cascade in biology is the so-called *ERK-MAPK pathway* [RD07], which transduces signals from the cell membrane to the cell nucleus. In the nucleus, these signals influence gene expression (i.e., the activity level of the genes) and can induce changes in the cell leading to cell division. Uncontrolled cell division, in turn, causes tumors. The gene *RAS* is the starting point of the *ERK-MAPK pathway*; *RAS* is attached to the cell membrane, where it is switched on by incoming signals. The end point of the signaling pathway is the gene *ERK*, which activates proteins that bind to the DNA and change gene expression. Accordingly, our collaborator queried for paths that connect *KRAS* (one representative member of the RAS gene family) and *MAPK3* (a member of the *ERK* family) in Pathfinder. She ranked the 149 paths that were returned by their average set connection strength (see supplementary Figure SP1; the first three paths are also visible in Figure 8). She chose this score because she was interested in paths that occur

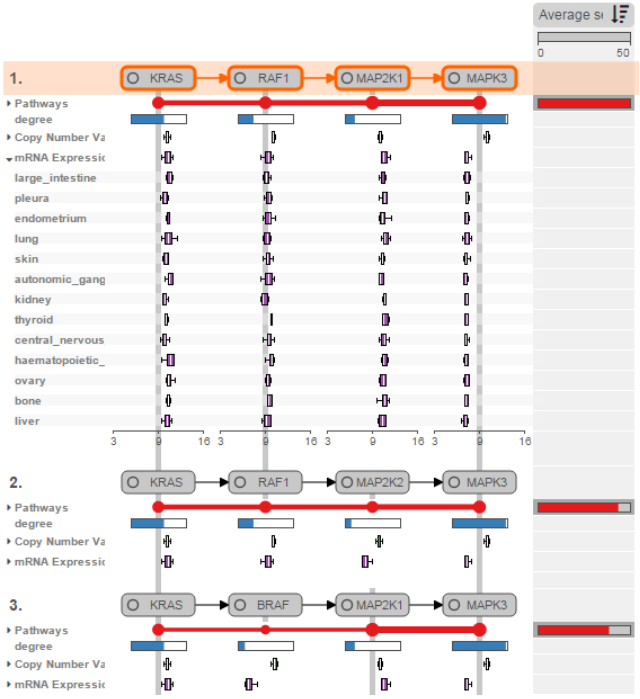


Figure 8: *The query result in Pathfinder for paths connecting KRAS and MAPK3. The paths are ranked by set connection strength, which places the path KRAS-RAF1-MAP2K1-MAPK3 on top (notice the thick lines for the aggregated pathways). This path corresponds to the ERK-MAPK signaling cascade. Associated copy number and mRNA expression data are shown as box plots. The expression dataset is expanded to investigate the expression across different tissue types in detail. The box plots show that the four genes are expressed in all displayed tissues, which emphasizes the importance of this path.*

in many KEGG pathway maps. The top-ranked path was *KRAS-RAF1-MAP2K1-MAPK3*, which corresponds exactly to the *ERK-MAPK pathway*. Thus, Pathfinder was able to identify correctly the most important communication path between RAS and ERK. An analysis of the pathways from which these connections were drawn (see supplementary Figure SP2) revealed that this path is present in many cancer pathways: *Colorectal cancer*, *pancreatic cancer*, *glioma*, *prostate cancer*, and *non-small cell lung cancer*, to name just a few. Similarly, the importance of this path was emphasized by the gene expression box plots showing data from cancer cell lines. The box plots revealed that these four genes are ubiquitously expressed across diverse tissues (see Figure 8), which indicates that this signaling cascade is active in many different cell types.

Another important pathway involved in the formation of cancer is the so-called *mTOR pathway* [LS12]. Our collaborator used Pathfinder to detect cross-talk between these two pathways, i.e., she explored how one signal transduction pathway could affect the other. For this purpose, she defined an advanced query, searching for connections between the four genes mentioned above and the five genes *MTOR, AKT1, TSC1, TSC2,* and *MLST8*, which are part of the *mTOR pathway*. She again ranked the paths by their average connection strength (see supplementary Figure SP4). The resulting path topology view highlighted that the *ERK-MAPK pathway* can modulate the *mTOR pathway* through the gene *PI3K*, which in turn modulates *AKT1* (note that there are different subtypes for this gene, which are all displayed in the view). Furthermore, the top-ranked path revealed a second route for regulation of the *mTOR pathway* by the *ERK-MAPK pathway*: *MAPK3* can modulate *TSC2*. To learn more about this regulation, our collaborator chose the path *MAP2K1-MAPK3-TSC2* as reference path and sorted all other paths by their similarity to the reference path (see supplementary Figure SP5). This ranking revealed an additional, indirect way for *MAPK3* to regulate *TSC2*: via *RPS6K*. Our collaborator was excited that Pathfinder was able to demonstrate the complexity of biological pathway regulation: The *ERK-MAPK pathway* can either modulate the *mTOR pathway* through *PIK3CA* or through *TSC2*. Understanding cross-talk between signaling cascades is important for the development of cancer therapeutics because they can contribute to drug resistance.

In summary, the feedback from our collaborator was very positive. She claimed that the analysis of multiple paths across pathways would have been difficult or even impossible with standard tools such as the KEGG web interface, especially when also considering experimental data.

8. Discussion

When developing Pathfinder, we set out to address two major goals: scalability with respect to the number of nodes and edges, and scalability with respect to associated sets and attributes. We deal with them in a number of ways. Foremost, the query-based approach reduces the complexity of analyzing the whole graph to analyzing resulting paths. Of equal importance is the list-based path view, which enables us to show ranked paths and rich attribute and set data that can be aggregated on demand. The computationally expensive and time-consuming path query process is mitigated by showing intermediate results, i.e., a path matching the query is

shown as soon as it is found. As the resulting paths can be numerous, easily accessible methods to refine the query help narrow down the result-set, and path ranking helps to quickly identify relevant paths. The ranked list representation of paths scales very well to large numbers of paths. The pagination of the list helps to improve rendering performance and also defines meaningful subsets of the ranked paths to be explored in the topology view.

Besides scalability, a major strength of our approach is that it can be extended and customized to comply with requirements in different use cases without changing the overall workings of the technique. Complex scores can easily be added to support customized path rankings, and the sequential layout of nodes makes adding visualizations of attribute data easy.

In our design we employ the most effective visual channels for the most important aspects of the data. For example, position, the strongest visual variable, is used to encode path rank, and connectedness is used to encode relationships. Also, all attribute visualizations use a position/size encoding (bar charts, box plots); color is used only as a redundant channel or for highlighting.

In the path list view some information is duplicated— for example, the start and end nodes, and their associated attributes, occur in every path. Although this wastes space, we argue that the benefits of the linear representations (easy to rank, excellent for attribute visualization) outweigh the cost of non-optimal space usage.

Our fundamental approach of transforming the graphs into a list that can be ranked and queried is similar to UpSet [LGS*14], where set intersections are displayed as a list that also can be ranked and queried. However, the underlying data type, the types of queries, and the visualization of associated data are different.

Limitations We distinguish limitations of the technique from limitations of our implementation. We argue that our technique addresses tasks related to paths and attributes well. A wide range of other tasks discussed by Lee et al. [LPP*06b] can be addressed by using the topology view and its capability to dynamically extend the network (see Figure 7). Specifically, the topology view allows us to also address most adjacency, accessibility, common connection, and connectivity tasks, with the exceptions of finding clusters, connected components, bridges, and articulation points, i.e., tasks that are related to an overview of the graph. To address these tasks, we envision integrating Pathfinder with a general purpose, query-based graph visualization system such as Orion [HP14].

A limitation of the current implementation is the expressiveness of the query interface. For example, attributes of nodes and edges cannot be considered in the query. Also, the visual and interaction design of the advanced query interface could be improved. Another limitation is that we currently do not visualize edge attributes and edge types. The display of both, however, is straightforward, as they can be shown in the same way as node attributes: by adding dedicated rows below the paths in the path list view. Finally, our topology view can contain avoidable edge crossings, which could be eliminated by improving the layout algorithm used.

Scalability We tested Pathfinder for graphs with more than 30k nodes and more than 70k edges and found no significant scalability problems. Queries on our example datasets typically return results within a few seconds. Query performance is influenced by the server's computing power, but also by caching: Paths through frequently used nodes are retrieved faster. Pathfinder can handle at least 600 paths without problems, with the exception of the overview mode of the topology view, which can become cluttered, as any node-link diagram. The maximum length of paths that can be conveniently shown depends on both screen resolution and the node size configured by the user. On full-HD displays, paths of length 6-12 can be shown conveniently, but longer paths may require scrolling. However, we found that the paths we consider are rarely longer than 6-8 hops. The number of paths that can be shown at the same time in the list view largely depends on the amount of displayed attributes and sets. When all attributes and sets are hidden, up to 20 paths fit on a typical full-HD display. Especially combined with path ranking, which puts the most relevant paths for a specific question at the top of the list, we argue that this number accommodates most tasks.

9. Conclusion and Future Work

In this work, we presented Pathfinder, a technique for the visual analysis of paths in large multivariate graphs. Our query-based approach allows users to search for paths between a specified start and end. The immediate display of intermediate results allows for early query refinements and speeds up the analysis process. Paths can be judged and ranked holistically, taking topology, attributes, and grouping structures into account. We showcased Pathfinder in context of a coauthor graph and a biological network. However, we are confident that our technique can be useful for the analysis of networks from other domains, such as social or computer networks.

Our technique addresses all requirements save one: the exploration of path classes (R VII). Aggregating similar paths and showing their basic structure could give analysts a better overview of the variation of paths. Combined with revealing details on demand, this could be an alternative approach to ranking for tackling large lists of paths. Aggregations could also be driven by a user-defined combination of properties, such as topology, attributes, and sets. We believe this to be a fruitful direction for future research.

Acknowledgements This work was co-funded by the European Union's Seventh Framework Programme (600641, GoSmart: A Generic Open-end Simulation Environment for Minimally Invasive Cancer Treatment), the Austrian Research Promotion Agency (840232), the Austrian Science Fund (P27975-NBL), and the US National Institutes of Health (U01 CA198935).

References

[Ada06] ADAR E.: GUESS: A Language and Interface for Graph Exploration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2006), CHI '06, ACM, pp. 791–800. doi:10.1145/1124772.1124889. 3

[AHSS13] ABELLO J., HADLAK S., SCHUMANN H., SCHULZ H.: A Modular Degree-of-Interest Specification for the Visual Analysis of Large Dynamic Networks. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13) 20*, 3 (2013), 337–350. doi:10.1109/TVCG.2013.109. 3

[AMHWA*05] ALEMAN-MEZA B., HALASCHEK-WEINER C., ARPINAR I., RAMAKRISHNAN C., SHETH A.: Ranking complex

relationships on the semantic Web. *IEEE Internet Computing 9*, 3 (2005), 37–44. doi:10.1109/MIC.2005.63. 3

[AMS05] ANYANWU K., MADUKO A., SHETH A.: SemRank: Ranking Complex Relationship Search Results on the Semantic Web. In *Proceedings of the 14th International Conference on World Wide Web* (2005), WWW '05, ACM, pp. 117–127. doi:10.1145/1060745.1060766. 3

[Bc12] BARRETINA J., COLLEAGUES: The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature 483*, 7391 (2012), 603–607. doi:10.1038/nature11003. 4

[BHJ09] BASTIAN M., HEYMANN S., JACOMY M.: Gephi: An Open Source Software for Exploring and Manipulating Networks. In *Third International AAAI Conference on Weblogs and Social Media* (2009). URL: http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154. 3

[CZQ14] CHENG G., ZHANG Y., QU Y.: Explass: Exploring Associations Between Entities via Top-K Ontological Patterns and Facets. In *Proceedings of the 13th International Semantic Web Conference - Part II* (2014), ISWC '14, Springer-Verlag New York, Inc., pp. 422–437. doi:10.1007/978-3-319-11915-1_27. 4

[FPDs12] FISHER D., POPOV I., DRUCKER S., SCHRAEFEL M.: Trust Me, I'M Partially Right: Incremental Visualization Lets Analysts Explore Large Datasets Faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2012), CHI '12, ACM, pp. 1673–1682. doi:10.1145/2207676.2208294. 2, 5

[GBD09] GREILICH M., BURCH M., DIEHL S.: Visualizing the Evolution of Compound Digraphs with TimeArcTrees. *Computer Graphics Forum 28*, 3 (2009), 975–982. doi:10.1111/j.1467-8659.2009.01451.x. 3

[GGL*15] GRATZL S., GEHLENBORG N., LEX A., STROBELT H., PARTL C., STREIT M.: Caleydo Web: An Integrated Visual Analysis Platform for Biomedical Data. In *Poster Compendium of the IEEE Conference on Information Visualization (InfoVis '15)* (2015), IEEE. 7

[GLG*13] GRATZL S., LEX A., GEHLENBORG N., PFISTER H., STREIT M.: LineUp: Visual Analysis of Multi-Attribute Rankings. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13) 19*, 12 (2013), 2277–2286. doi:10.1109/TVCG.2013.173. 5

[HF07] HENRY N., FEKETE J.-D.: MatLink: Enhanced Matrix Visualization for Analyzing Social Networks. In *Human-Computer Interaction – INTERACT 2007*, Baranauskas C., Palanque P., Abascal J., Barbosa S. D. J., (Eds.), no. 4663 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 288–302. URL: http://link.springer.com/chapter/10.1007/978-3-540-74800-7_24. 4

[HHL*09] HEIM P., HELLMANN S., LEHMANN J., LOHMANN S., STEGEMANN T.: RelFinder: Revealing Relationships in RDF Knowledge Bases. In *Semantic Multimedia*, Chua T.-S., Kompatsiaris Y., Mérialdo B., Haas W., Thallinger G., Bailer W., (Eds.), no. 5887 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 182–187. URL: http://link.springer.com/chapter/10.1007/978-3-642-10543-2_21. 3

[HP14] HEER J., PERER A.: Orion: A system for modeling, transformation and visualization of multidimensional heterogeneous networks. *Information Visualization 13*, 2 (2014), 111–133. doi:10.1177/1473871612462152. 3, 9

[IHK*15] ISENBERG P., HEIMERL F., KOCH S., ISENBERG T., XU P., STOLPER C., SEDLMAIR M., CHEN J., MÖLLER T., STASKO J.: Visualization Publication Dataset, 2015. URL: http://vispubdata.org/. 4

[KPW14] KERREN A., PURCHASE H. C., WARD M. (Eds.): *Multivariate Network Visualization*. No. 8380 in Lecture notes in computer science. Springer, 2014. 2, 6

[LGS*14] LEX A., GEHLENBORG N., STROBELT H., VUILLEMOT R., PFISTER H.: UpSet: Visualization of Intersecting Sets. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '14) 20*, 12 (2014), 1983–1992. doi:10.1109/TVCG.2014.2346248. 9

[LPK*13] LEX A., PARTL C., KALKOFEN D., STREIT M., GRATZL S., WASSERMANN A. M., SCHMALSTIEG D., PFISTER H.: Entourage: Visualizing Relationships between Biological Pathways using Contextual Subsets. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13) 19*, 12 (2013), 2536–2545. doi:10.1109/TVCG.2013.154. 4

[LPP*06a] LEE B., PARR C., PLAISANT C., BEDERSON B., VEKSLER V., GRAY W., KOTFILA C.: TreePlus: Interactive Exploration of Networks with Enhanced Tree Layouts. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (2006), 1414–1426. doi:10.1109/TVCG.2006.106. 3

[LPP*06b] LEE B., PLAISANT C., PARR C. S., FEKETE J.-D., HENRY N.: Task Taxonomy for Graph Visualization. In *Proceedings of the AVI Workshop on BEyond time and errors: novel evaluation methods for information visualization (BELIV '06)* (2006), pp. 1–5. doi:10.1145/1168149.1168168. 2, 9

[LS12] LAPLANTE M., SABATINI D. M.: mTOR signaling in growth control and disease. *Cell 149*, 2 (2012), 274–293. doi:10.1016/j.cell.2012.03.017. 8

[LSG*15] LUGER S., STITZ H., GRATZL S., GEHLENBORG N., STREIT M.: Interactive Visualization of Provenance Graphs for Reproducible Biomedical Research. In *Poster Compendium of the IEEE Conference on Information Visualization (InfoVis '15)* (2015), IEEE. 3

[Pet16] PETTITT C.: dagre - Graph layout for JavaScript, Mar. 2016. URL: https://github.com/cpettitt/dagre. 7

[PLS*13] PARTL C., LEX A., STREIT M., KALKOFEN D., KASHOFER K., SCHMALSTIEG D.: enRoute: Dynamic Path Extraction from Biological Pathway Maps for Exploring Heterogeneous Experimental Datasets. *BMC Bioinformatics 14*, Suppl 19 (2013), S3. doi:10.1186/1471-2105-14-S19-S3. 3, 4

[RD07] ROBERTS P. J., DER C. J.: Targeting the Raf-MEK-ERK mitogen-activated protein kinase cascade for the treatment of cancer. *Oncogene 26*, 22 (2007), 3291–3310. doi:10.1038/sj.onc.1210422. 8

[SM07] SHEN Z., MA K.-L.: Path Visualization for Adjacency Matrices. In *Eurographics/ IEEE-VGTC Symposium on Visualization* (2007), The Eurographics Association. doi:10.2312/VisSym/EuroVis07/083-090. 4

[SOR*11] SMOOT M. E., ONO K., RUSCHEINSKI J., WANG P.-L., IDEKER T.: Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics 27*, 3 (2011), 431–432. doi:10.1093/bioinformatics/btq675. 3

[TK08] TEKUŠOVÁ T., KOHLHAMMER J.: Visual analysis and exploration of complex corporate shareholder networks. In *Proceedings of the SPIE Conference on Visualization and Data Analysis (VDA '08)* (2008), vol. 6809, pp. 68090F–68090F–10. doi:10.1117/12.761555. 3

[VBW15] VEHLOW, CORINNA, BECK, FABIAN, WEISKOPF, DANIEL: The State of the Art in Visualizing Group Structures in Graphs. In *Eurographics Conference on Visualization (EuroVis) - STARs* (2015). doi:10.2312/eurovisstar.20151110. 3, 6

[vHP09] VAN HAM F., PERER A.: "Search, Show Context, Expand on Demand": Supporting Large Graph Exploration with Degree-of-Interest. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '09) 15*, 6 (2009), 953–960. doi:10.1109/TVCG.2009.108. 3

[vLKS*11] VON LANDESBERGER T., KUIJPER A., SCHRECK T., KOHLHAMMER J., VAN WIJK J., FEKETE J.-D., FELLNER D.: Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. *Computer Graphics Forum 30*, 6 (2011), 1719–1749. doi:10.1111/j.1467-8659.2011.01898.x. 2

[ZCQ13] ZHANG Y., CHENG G., QU Y.: Towards Exploratory Relationship Search: A Clustering-Based Approach. In *Semantic Technology*, Kim W., Ding Y., Kim H.-G., (Eds.), no. 8388 in Lecture Notes in Computer Science. Springer International Publishing, Nov. 2013, pp. 277–293. doi:10.1007/978-3-319-06826-8_21. 4