International Symposium on Visual Analytics Science and Technology (2010)
J. Kohlhammer and D. Keim (Editors)

Smart Query Definition for Content-Based Search in Large Sets of Graphs

T. von Landesberger[1,2], S. Bremm[1], J. Bernard[1] and T. Schreck[1]

[1] Technische Universität Darmstadt, Germany
[2] Fraunhofer IGD, Darmstadt, Germany

Abstract

Graphs are used in various application areas such as chemical, social or shareholder network analysis. Finding relevant graphs in large graph databases is thereby an important problem. Such search starts with the definition of the query object. Defining the query graph quickly and effectively so that it matches meaningful data in the database is difficult. In this paper, we introduce a system, which guides the user through the process of query graph building. We propose three approaches for graph definition. First, query by example selection starting from an overview of the graph types in the database, second query by sketch combining graph building blocks (i.e., topologic subgraphs) with free graph drawing, and third a combination of both approaches. In all three query definition ways, we support the user with intelligent, data dependent recommendations. It covers the whole spectrum of building parameters such as representative examples, frequent building blocks, or common graph size.

Categories and Subject Descriptors (according to ACM CCS): H.3.1 [Information Systems]: Content Analysis and Indexing—Indexing methods H.3.3 [Information Systems]: Information Search and Retrieval—G.2.2 [Mathematics of Computing]: Graph Theory—H.5.2 [Information Systems]: User Interfaces—Graphical user interfaces (GUI) I.3.4 [Computing Methodologies]: Graphics Utilities—Graphics editors

1. Introduction

Finding interesting objects in large databases is a common problem in many areas such as image retrieval, Internet search, 3D model retrieval, or biochemistry. In all of these examples, graphs are an important data type used for representing structural information and often their visual form is used in the searching.

The search process can roughly be split into three steps: 1) query formulation, 2) identification of most similar objects, and 3) presentation of query results. Usually, the most analytical effort is put into the second step which includes the calculation of a meaningful description of the query object, choosing an effective distance measurement and finding the nearest neighbors. However, the exact definition of the query object is the basis for the search process.

In text searches, the query definition part is simply performed by typing in the search term, and the results are often presented as a list. In other areas, the definition of the query objects is more complex, for example, in the search for components or proteins in biology and pharmacy, in finding similar 3D objects in an engineering environment or the search for music and images. Query definition for graphs usually employs graph editing. In graph editing, a free addition of individual nodes leads to very precise results, however it is very time consuming, especially for large graphs. Template-based systems create graphs by parameter setups, producing common graph types such as trees. This leads to fast results, however is restricted to the pre-defined graph templates.

In this paper, we present an intelligent system, supporting the user in interactively constructing query objects for visual graph search (see Figure 1 for an illustration). The query graphs can 1) be selected from the data space, 2) be built from scratch by drawing or using predefined patterns, or 3) be formed using a combination of these methods. For building a meaningful query object, it is beneficial that the user knows the scope of the underlying data space. For example, when searching for historical events, the users usually know the range of the search parameters such as date span. In contrast, in explorative graph search scenarios, we

DOI: 10.2312/PE/EuroVAST/EuroVAST10/007-012

do not know the level of user familiarity with the data space. This impedes the definition of the query object. To assist the user within the query definition process, we propose to automatically analyze the underlying data space. Based on this analysis, in all three above-mentioned ways of query definition, our approach supports the user by smart recommendations regarding the range of graph building parameters. We offer representative examples from the database, the frequency distribution of graph building blocks, and of basic graph topologic features (e.g., graph size).

This paper is structured as follows. Section 2 reviews briefly related work. Section 3 discusses our approaches for defining useful query objects. Section 4 concludes and outlines interesting future work.

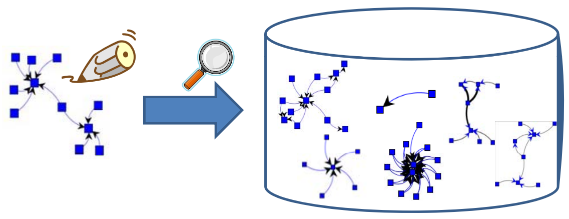


Figure 1: *An illustration of interactive graph search.*

2. Related Work

We briefly recall relevant works from interactive graph specification, retrieval in graph databases, and support for intelligent query formulation.

2.1. Query by Example and Graph Editing

The definition of query objects for search in multimedia databases typically follows two main approaches: Query by example, and query by sketch. In these cases, the user provides either a full example record, or a sketch, outline or other partial representation of the aspects that the search results need to cover. It is particularly important in multimedia retrieval, as there, often no exact search is possible. Some example search systems include [RHM97, FSN*95] for the image domain, and [FMK*03, PKJ*07, BBK*09] for the 3D model domain.

Sketching approaches depend on the considered type of query object. In this paper, we focus on creating graph query objects, i.e., *visual graph editing*. This is closely related to visual graph representation. Graphs are typically represented as either node-link diagrams, adjacency matrices, or a combination of both [vLKS*10]. As graph drawing is not in the focus of the paper, we refer to extensive surveys on visual graph representation [DPS02, HMM00, DBETT99, HJ07, vLKS*10].

Graph editing software tools such as yED [yED] or JUNG [OFS] rely on node-link representation. In both cases, the graphs are composed by successively adding low-level graph elements, i.e., nodes and edges. Adding these items may be cumbersome and time consuming, in particular for creating large graphs. yED also offers to create whole graphs by using pre-defined graph templates including tree and random graphs. The size of the graph can be adjusted by the user. The set of graphs is limited and only whole graphs can be created in this way.

Wong et al. present a system called GreenSketch which allows for fast drawing of combined matrix and node-link graphs representations [WFM*06]. Drawing is performed by editing the adjacency matrix. The resulting graph is visualized both as a matrix, and as a node link diagram. The editing of the adjacency matrix can lead to fast drawing of complex graphs. However, this is a rather complex task and the tool may not be intuitive for non-experts. Moreover, GreenSketch is suited only for undirected graphs.

2.2. Graph Retrieval

The graph retrieval problem is an interesting, yet inherently difficult problem. For exact searches, the mathematical concepts of graph and subgraph isomorphism apply, for which, however, no efficient solutions exist [ZV08, RGW02, YZYH06]. Retrieval based on similarity estimation between graphs is possible by transformation and descriptor-based approaches. Transformation approaches such as the Edit-2 Distance for undirected acyclic graphs [ZWS96] approximate the similarity among graphs by the cost of efficiently transforming one into the other. Descriptor-based approaches such as the graph histogram approach [PM99] or motif-based approach [vLGRS09, vLGS09] capture important attributes of a graph in form of a vector or histogram (descriptor); subsequently, the similarity between graphs is measured by a distance function defined on the descriptors.

2.3. Smart Query Support

A large body of work also exists regarding smart support for the query formulation process. Similarity queries can be supported by relevance feedback strategies that by means of optimization techniques aim to produce more relevant search results for a given user [FBY92]. Recommending and collaborative filtering approaches aim to provide additional relevant results for a query by comparing the query and/or profile of a given user to historic querying profiles or user-provided annotation information [AT05]. In the Visual Analytics context, the concept of intelligent visual querying has been introduced in [HDK*07]. In that work, a user selection of a specific area in a data visualization display triggers an automatic search for similar data portions, which subsequently are displayed to the user.

3. Interactive Graph Query Definition

We propose three ways of defining the query graph, which are described in more detail in the following.

- *Smart choice of data samples:* Using an example graph from the database, based on search for graph attributes and the overview of representative graphs in the database.
- *Graph sketching supported by data-dependent graph building blocks:* Using a graph interactively created (drawn, edited, sketched) by the analyst.
- *Combining smart sketching with data samples:* Using a graph, or a set of example graphs from the database as a basis for further editing of the final query graph.

These approaches support flexible, efficient, and data-oriented query graph definition.

3.1. Smart Choice of Data Samples

In this approach, one interesting graph from the database is used as query input. There are several ways of choosing the query graph. First, the graph can be retrieved from the database by *searching for specific node and edge attributes*. For example, the analyst might be interested in a specific corporation or a chemical structure. The advantage of this approach is the use of a user-specific object of interest as a query input. However, sometimes it can be difficult to select this object from the large set of available objects, in particular, when facing large unknown databases. Therefore, alternatively, the user can use results from a previous graph exploration phase as an input. As graph databases may include a very large number of graphs, their exploration can be very time consuming.

A selection of an appropriately obtained *subset of example graphs from the whole database* helps to get an overview of the types of graphs available. Usually, a random sample of a database subset is used for this purpose. More sophisticated selection strategies have been proposed for sets of numeric values such as systematic or stratified sampling. These approaches are however not directly applicable to graph sets. In order to overcome this drawback, we propose to offer a set of sample graphs obtained from preprocessing of the graph data based on the *Self Organizing Map* (SOM) algorithm [Koh01] as introduced in [vLGS09]. The result of the SOM algorithm depends on the parameter settings of the SOM algorithm and the interactively set similarity function. The result view shows an overview of the graph data space as a grid of prototype graphs (see Figure 2 top). In particular, a grid of objects closest to the cluster centers are presented. One of these objects can be used as a starting point for the graph search (see Figure 2 bottom). The visualization settings (e.g., graph layout) can be adjusted on demand.

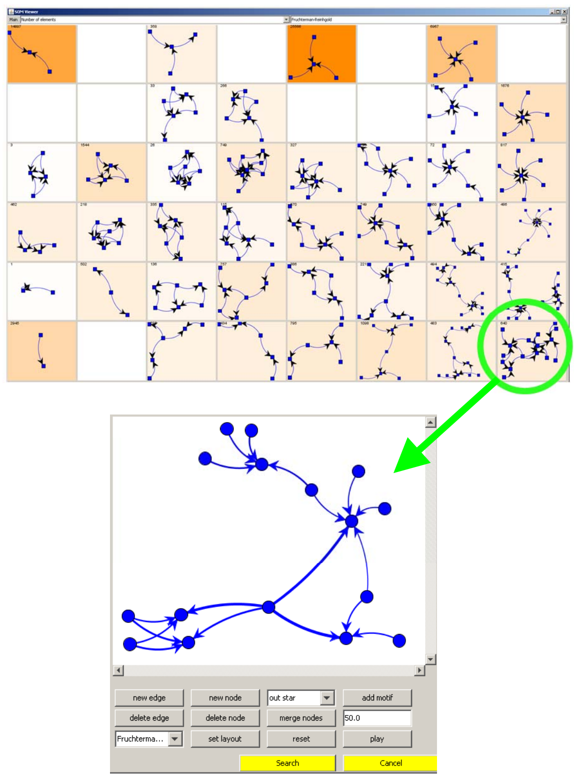


Figure 2: *Using SOM clustering results for defining the query graph. Top: The visualization of SOM clustering results. It shows a grid of prototype graphs from the database. The background color indicates the frequency of the graphs (white to orange: low to high). Bottom: The selected graph query object from the SOM grid (layout differs). This graph is highlighted with green circle in the top figure.*

3.2. Graph Sketching Supported by Data-Dependent Graph Building Blocks

The most intuitive option for defining a query object is creating the object itself. Graph editing usually employs *adding individual nodes and edges* one by one. These graph elements can be additionally detailed by specification of their attributes. This process can be very time consuming, especially when building larger graphs. Therefore, we propose to extend graph editing by adding multiple nodes and edges at once – so called graph building blocks. This idea was inspired by several previous works. Firstly, small building blocks of graphs are called motifs. They have specific functions or are overpresented in graphs. They are often analyzed in biologic applications [Sch08]. Moreover, recently, graph substructures (i.e., motifs) have been used for determining graph similarity [vLGS09]. Second, the yED graph editor [yED] offers the possibility to automatically create a graph (e.g., a tree or random graph of a given size). However, only a small number of graph types are provided, and only whole graphs are created in this way. Third, TopoLayout [AAM07] decomposes graphs into structural forms for choosing best layouts for parts of a larger graph.

For graph creation, we propose the *building blocks* shown in Figure 3. They can be interactively combined and offer the user effective control of the graph editing process and support fast creation of a variety of graphs. All building blocks are parameterizable to enhance these possibilities. Note that we have constructed these blocks for directed graphs; when editing undirected graphs, these blocks can be adapted and extended. Although the query graph construction method presented is very flexible and fast, there is a potential drawback. Specifically, the user gets no feedback about the potential relevance of her query object, until she sees the search results. To help the user building meaningful query graphs, we analyze the underlying data space to present addition guidance information. For building blocks, we color code the frequency of occurrence and show the frequency distribution as bar charts (see Figure 4a). On demand, we outline more details for every motif as diagram or example graph. For stars, these re star-subclasses described by the number of ingoing or outgoing edges.

The following building blocks are currently supported in our system:

- *Out-star:* A node with a certain number of children.
- *In-star:* Similar to out-star, a node with a parameterizable number of parent nodes.
- *Chain:* A set of connected nodes forming a directed path consisting of a certain number of nodes.
- *Complete bipartite:* Two sets of nodes, where each node from the first set is connected to all nodes from the second set. The sizes of the two sets determine the shape of the subgraph.
- *Cycle:* This building block creates a directed cycle with parameterizable number of nodes (a closed path of several nodes). Note that a cycle with two nodes corresponds to the so-called reciprocity motif, and cycle with three nodes to the so-called feed-back motif.
- *Two-way path:* This building block consists of one root node and one leaf node. Between these two nodes, two separate directed paths of a certain length exist. It forms thereby an undirected cycle. The two-way path with three nodes forms a so-called feed-forward motif. With four nodes (two and two edges), a so-called caro motif is defined.
- *Balanced rooted tree:* It is a directed rooted tree of a certain number of levels, whose all nodes (apart from leafs) have a pre-set number of children.

The blocks can be combined either by adding a building block to the existing node or by connecting a block to the graph by a new edge. The building blocks and the resulting graphs can be further edited by adding or deleting individual nodes and edges on demand. The graph layout is user-chosen. This approach allows for creating appropriate and data-oriented query graphs for searching efficiently (see Figure 4).

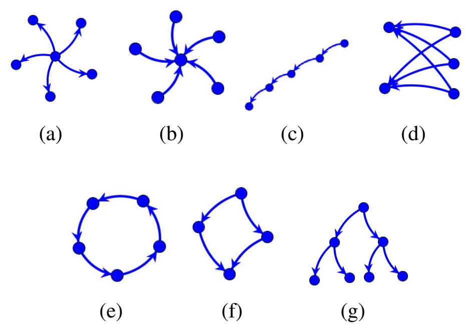


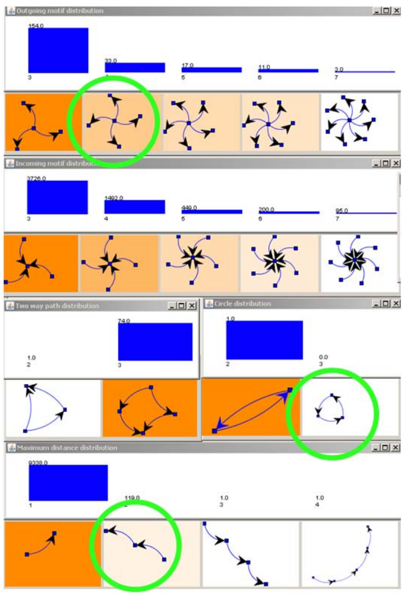
Figure 3: *Building blocks for graph drawing. a) Out-star, b) in-star, c) chain, d) complete bipartite, e) cycle, f) two-way path, g) balanced tree.*

3.3. Combining Smart Sketching With Data Samples

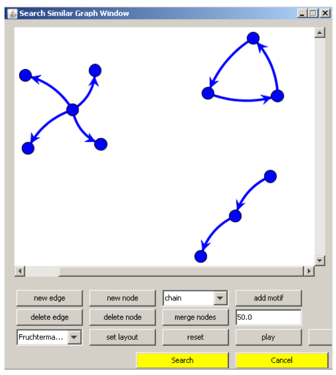
In order to both avoid cumbersome graph drawing when searching in unfamiliar databases, and using fixed graph objects retrieved from the database during the exploratory process, we propose to combine both approaches. To give the analyst the possibility to combine existing objects and modify them on demand enhances the advantages and decreases the disadvantages of the before mentioned methods. The example-based sketching starts from existing objects, which can be modified by adding and deleting of edges and nodes. Available building blocks can be adapted as the user sees fit just as well. Providing powerful, baseline samples, the sketching process can be significantly improved. Bringing all techniques together, the user can quickly define a specific and meaningful query object (see Figure 5).

4. Conclusions and Future Work

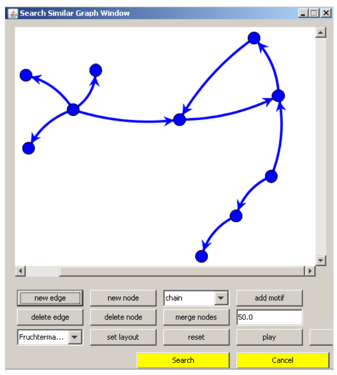
In this paper, we presented three methods for smart definition of a query graph in searching graph databases. The query by data sample and query by sketch techniques are extended and interactively combined so that appropriate query objects can efficiently and effectively be created. Using an overview of the underlying data space, it is easier to formulate meaningful queries. We address this problem by offering user guidance throughout the whole query definition process. We analyze the data space in order to provide important information about key search parameters. Query by example is enhanced by SOM-based graph clustering. Graph sketching is enhanced by offering graph building blocks with respective frequency information regarding their occurrence in the database to be searched. The selected graph examples and graph building blocks can be flexibly combined with


Outgoing motif distribution
154.0
22.0
17.0
11.0
3.0
Incoming motif distribution
3726.0
1492.0
449.0
200.0
95.0
Two way path distribution
74.0
1.0
Circle distribution
1.0
0.0
Maximum distance distribution
9338.0
119.0
1.0
1.0

(a) Distribution of building blocks in the data set

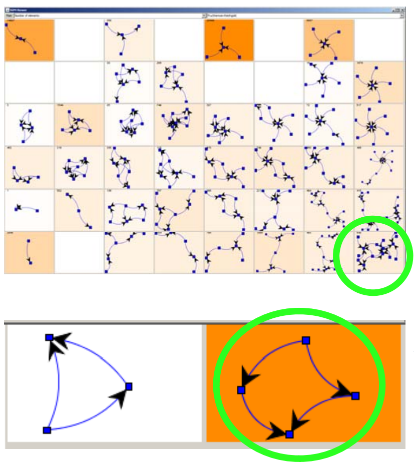


(b) Selection of building blocks

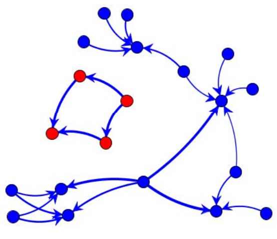


(c) Query graph created by connecting building blocks

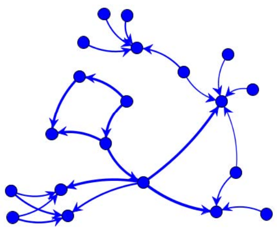
Figure 4: *Smart query object definition supported by graph sketching and data-dependent graph building blocks. a) Distribution of building blocks in data set. Both the bar charts and the background colors of the building blocks represent the frequencies of the corresponding building blocks. b) The selection of data-dependent building blocks for creating the query objects. The building blocks are highlighted in the frequency view with green circles. c) The final query graph created by connecting and adapting the chosen baseline building blocks with sketched edges and merged nodes.*



(a) The choice of graph sample from the SOM result and a selection of graph building blocks



(b) Using graph sample and a building block for graph definition



(c) Query graph created by combining graph sample and a building block

Figure 5: *Combining smart sketching with data samples for leveraging the advantages of both techniques. a) The proposal for graph samples using SOM clustering and graph building blocks with frequency indication. b) The selection of a graph sample and a building block for creating the query graph. The selected sample and the building block are highlighted with green circles in the proposal view. c) The final query graph combining both graph samples and building blocks with sketched edges.*

fully interactive graph editing to leverage advantages of both approaches.

These novel interactive query definition methods can be used for graph search in various application areas such as biology, chemistry, social science, finance, computer networks. It can be used for querying databases of graphs, where the user determines the query object.

In the future, we will improve the visual interface for query definition with more functions, including wildcard queries and node labels, and enhance the visualization of additional data information. We like to expand this system with graph search using feature-based similarity, visual exploration of search results. In this paper, we have introduced the basic concept and presented illustrative results. An evaluation for different use cases needs to be performed in the future.

Acknowledgements

The authors thank R. Rehner, M. Görner, J. Kirchhoff and R. Gutbell for their support. This work was partially supported by the German Federal Ministry of Economics and Technology within the THESEUS project (`http://www.theseus-programm.de/`) and by the German Research Foundation (DFG) within the project Visual Feature Space Analysis as part of the Priority Program on Scalable Visual Analytics (SPP 1335).

References

[AAM07] ARCHAMBAULT D., AUBER D., MUNZNER T.: Topolayout: Multilevel graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (2007), 305–317. 4

[AT05] ADOMAVICIUS G., TUZHILIN E.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering 17* (2005), 734–749. 2

[BBK*09] BERNDT R., BLÜMEL I., KROTTMAIER H., WESSEL R., SCHRECK T.: Demonstration of user interfaces for querying in 3d architectural content in PROBADO3D. In *Proceedings of European Conference on Digital Libraries* (2009). Demonstration Paper. 2

[DBETT99] DI BATTISTA G., EADES P., TAMASSIA R., TOLLIS I. G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999. 2

[DPS02] DÍAZ J., PETIT J., SERNA M.: A survey of graph layout problems. *ACM Comput. Surv. 34*, 3 (2002), 313–356. 2

[FBY92] FRAKES W. B., BAEZA-YATES R. A. (Eds.): *Information Retrieval: Data Structures & Algorithms*. Prentice-Hall, 1992. 2

[FMK*03] FUNKHOUSER T., MIN P., KAZHDAN M., CHEN J., HALDERMAN A., DOBKIN D., JACOBS D.: A search engine for 3D models. *ACM Trans. Graph. 22*, 1 (2003), 83–105. 2

[FSN*95] FLICKNER M., SAWHNEY H., NIBLACK W., ASHLEY J., HUANG Q., DOM B., GORKANI M., HAFNER J., LEE D., PETKOVIC D., STEELE D., YANKER P.: Query by image and video content: The QBIC system. *Computer 28* (1995), 23–32. 2

[HDK*07] HAO M., DAYAL U., KEIM D., MORENT D., SCHNEIDEWIND J.: Intelligent visual analytics queries. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology* (2007). 2

[HJ07] HACHUL S., JÜNGER M.: Large-graph layout algorithms at work: An experimental study. *Journal of Graph Algorithms and Applications 11*, 2 (2007), 234–369. 2

[HMM00] HERMAN I., MELANCON G., MARSHALL M. S.: Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics 6*, 1 (2000), 24–43. 2

[Koh01] KOHONEN T.: *Self-Organizing Maps*, 3rd ed. Springer, Berlin, 2001. 3

[OFS] O'MADADHAIN J., FISHER D., SMYTH P.: Analysis and visualization of network data using (JUNG). *Journal of Statistical Software VV*, 2. 2

[PKJ*07] PU J., KALYANARAMAN Y., JAYANTI S., RAMANI K., PIZLO Z.: Navigation and discovery in 3D CAD repositories. *IEEE Comput. Graph. Appl. 27*, 4 (2007), 38–47. 2

[PM99] PAPADOPOULOS A. N., MANOLOPOULOS Y.: Structure-based similarity search with graph histograms. In *Proceedings of Int. Workshop on Database & Expert Systems Applications* (1999), IEEE Computer Society, p. 174. 2

[RGW02] RAYMOND J., GARDINER E., WILLETT P.: RASCAL: calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal 45*, 6 (2002), 631–644. 2

[RHM97] RUI Y., HUANG T. S., MEHROTRA S.: Content-based image retrieval with relevance feedback in mars. In *Proceedings of IEEE Int. Conf. on Image Processing* (1997), pp. 815–818. 2

[Sch08] SCHWÖBBERMEYER H.: *Analysis of Biological Networks*. Wiley Series on Bioinformatics, Computational Techniques and Engineering. Wiley, 2008, ch. 5, pp. 85 – 112. 3

[vLGRS09] VON LANDESBERGER T., GÖRNER M., REHNER R., SCHRECK T.: A system for interactive visual analysis of large graphs using motifs in graph editing and aggregation. In *Proceedings of Vision Modeling Visualization Workshop* (2009). 2

[vLGS09] VON LANDESBERGER T., GOERNER M., SCHRECK T.: Visual analysis of graphs with multiple connected components. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology* (2009). 2, 3

[vLKS*10] VON LANDESBERGER T., KUIJPER A., SCHRECK T., KOHLHAMMER J., VAN WIJK J., FEKETE J.-D., FELLNER D.: Visual analysis of large graphs. In *Proceedings of EuroGraphics: State of the Art Report* (2010). 2

[WFM*06] WONG P. C., FOOTE H., MACKEY P., PERRINE K., JR. G. C.: Generating graphs for visual analytics through interactive sketching. *IEEE Transactions on Visualization and Computer Graphics 12* (2006), 1386–1398. 2

[yED] yED. `http://www.yworks.com/en/products_yed_about.html`. 2, 3

[YZYH06] YAN X., ZHU F., YU P. S., HAN J.: Feature-based similarity search in graph structures. *ACM Trans. Database Syst. 31*, 4 (2006), 1418–1453. 2

[ZV08] ZAGER L. A., VERGHESE G. C.: Graph similarity scoring and matching. *Applied Mathematics Letters 21*, 1 (2008), 86 – 94. 2

[ZWS96] ZHANG K., WANG J., SHASHA D.: On the editing distance between undirected acyclic graphs. *Int. Journal of Foundations of Computer Science 7*, 1 (1996), 43–57. 2