Visually-supported graph traversals for exploratory analysis

Albert Amor-Amorós, Paolo Federico, and Silvia Miksch *

ABSTRACT

Many real-world problems appearing in diverse application domains involve large multivariate interrelated data. For this reason, graph-based data models have gained popularity in recent years. Graph traversal is a powerful computational paradigm addressing the challenges of graph data management; yet, its complexity and specificity might hinder its use for interactive data exploration by non-expert users in absence of appropriate interfaces. We have designed and implemented a system for visually-supported graph traversal, featuring (1) a graphical block metaphor for traversal formulation and execution, and (2) data probes providing relevant visual feedback about the results. The proposed approach aims at enhancing the usability of graph querying and retrieval techniques, in order to assist users with gaining and interpreting insights during exploratory analysis.

Keywords: Graph traversal, visual querying, large graph exploration, graph visualization, graph querying, exploratory analysis.

1 INTRODUCTION

Graph-based data management techniques have become increasingly popular over recent years, due to their expressiveness for modelling and manipulating interrelated data, i.e., they facilitate the design of logical schemas in a close match with conceptual schemas. In a graph model, data is represented as a collection of relations (graph edges) between pairs of entities from a given set (graph vertices), typically with elements of both types having a number of attributes. In combination with the large scale characteristic of modern applications, the inherent complexity of this form of data can turn the analytic process into a very challenging problem. Improving the scalability of existing visualization techniques in order to provide better overviews of large graphs has been one of the major topics in graph visualization research. However, providing a complete overview for a large multivariate graph is often unfeasible, and sometimes also unnecessary: many scenarios only require local exploration, since the analytic focus starts on a fraction of the data, and follows specific paths over the graph structure on the basis of intermediate findings [6].

In order to address the specific needs of this form of analysis, we have designed and implemented an interactive system that supports the exploration of large graphs through the specification of graph traversal workflows in a visually-supported graphical environment.

2 PROBLEM DESCRIPTION

Visual systems designed to support exploration on large multivariate graphs have started to appear in recent years [3, 6]. Search frequently constitutes the initial step in these approaches, and, once a starting point has been chosen, expansion around a node of interest is typically the main supported interaction, enabling the user to incorporate into the visual representation hidden parts of the graph. These approaches usually feature an intuitive interface that combines a well-known visual metaphor (e.g., node-link diagrams) with a simple interaction model (e.g., clicking on a node to retrieve the nodes adjacent to it). The simplicity and intuitiveness of these approaches clearly constitute strong points in simple cases, but they can also turn into important limiting factors in more complex analytical scenarios. For example, these visualization and interaction models tend to focus on individual targets instead of classes of equivalent elements; for this reason, the simultaneous manipulation of multiple elements (e.g., selecting the set of adjacent elements, given a set of nodes) is generally not supported. Furthermore, the user has little means to specify interesting paths on the basis of their individual characteristics or intermediate computational results.

These enhanced analytic capabilities, such as a rich set of operations and a higher degree of user-control, are the main advantages of graph querying tools [2, 5]. Such systems are the preferred choice amongst advanced users, but they present a number of usability issues that make them less suitable for non-experts. Graph querying tools are usually operated within a command-line interface, requiring the user to enter every request in the form of text-based statements expressed in a formal language. In order to use this kind of interface, a user must be familiar with the grammar of its query language, i.e., what are the elements that compose it, and how are they combined into coherent statements. Accordingly, one of the fundamental challenges is enabling users to formulate requests in a more natural and effective way. Additionally, these tools tend to provide limited feedback, forcing the user to rely on mental models and a priori knowledge. This is particularly problematic in large scale scenarios with high data complexity, where a data consumption approach based exclusively on retrieval and display actions is not feasible. In order to successfully steer the exploration process, users need frequent and accurate feedback regarding both the characteristics of the selected data subset, and the outcomes of the operations performed on it.

3 VISUALLY-SUPPORTED GRAPH TRAVERSALS

We have designed and implemented a system for the exploration of large multivariate graphs that combines the full analytic power of a graph querying tool, with the enhanced usability and cognitive amplification effect of graphical user interfaces and data visualization. We have grounded this approach in the so-called *graph traversal pattern*, a computational paradigm in which graph problems are represented as traversals, i.e., "visiting elements (i.e., vertices and edges) in a graph in some algorithmic fashion" [4]. In the following we describe the two main components of the system, which are displayed side-by-side forming the user interface (see Figure 1): the graphical query composition component, and the visual data probing component.

Graphical query composition In practice, a graph traversal specification consists of a series of steps that represent low-level data manipulation operations. Examples of such operations are: moving the selection focus to the set of edges incoming to a given set of vertices, or filtering out elements with values outside of predefined range for a given attribute. A graph traversal specification defines a data transformation pipeline; accordingly, graph traversal languages typically embrace a dataflow paradigm, in which the output of one step constitutes the input for the next one. Our design communicates this intuitive behaviour with a graphical metaphor in which individual steps are represented as blocks, that can be attached to each other by means of drag-and-drop interactions in order to specify a pipeline.

*Albert Amor-Amorós, Paolo Federico, and Silvia Miksch are with the Institute of Software Technology and Interactive Systems, Vienna University of Technology. E-mail: {amor,federico,miksch}@ifs.tuwien.ac.at

Figure 1: The two main components of the interface: on the left side, the traversal composition canvas; on the right side, the visual probing component.

Blocks representing operations that require some kind of user input in the form of parameter specification can be enhanced with selection controls appropriate to the nature and size of the corresponding parameter space. Some examples of these are bargrams, lists, auto completing text boxes, or range sliders. New steps are added to the pipeline by choosing from a list of operations that are applicable in the given context. In some cases, a description of a graph traversal in terms of individual low-level operations might be too specific, and, therefore, the user might be interested in abstracting some of these details. This can be achieved by defining a composite step that combines the functionality of a particular sequence of operations in a single step. In our approach, we enable the user to perform this abstraction by collapsing a particular sequence of selected blocks into a single one that represents the respective composite operation. The reverse transformation, i.e., the expansion of a composite block into its constituents, might also be of interest to the user, e.g., for examining the internal structure of the composite block, or changing some of its parameters. Moreover, we have also introduced a persistence mechanism that enables the user to name and store composite blocks for the sake of reusability, facilitating the automation of parts of the analytic workflow in future sessions. Finally, we have also incorporated a parser that automatically translates textual statements into its respective graphical representation, for its study or use in further elaboration.

Visual data probes Our system provides up-to-date feedback in response to any changes on the pipeline, i.e., the addition, removal, or modification of a transformation step by the user. In order to address the challenges associated with the high complexity and large scales of the data, we have introduced the concept of data probes, which provide the user with multiple alternative methods for consuming output results. These methods take into account relevant aspects in a particular context, providing information summaries on the basis of aggregates, metrics, or more complex analytic models. For example, visualizing an output consisting of several thousand graph vertices with multiple attributes might be not just challenging, but also particularly ineffective if the user is only interested in the distribution of values for some attribute or metric. We have designed an extensible probing system that supports the definition of new data probes with minimal effort. In our current solution, the set of active probes has to be configured a priori, by linking context conditions, reduction mechanisms, and visual representations within a configuration file. However, we are currently working on the design of a graphical interface that enables the definition of new probes by the user within the analysis session. We provide a basic set of visual representations that can be used in the probes, including bar charts, node-link diagrams, and several other, as well as a tabular view. One of the advantages of relying on an explicit representation of the data transformation workflow is the availability of provenance information. We have enabled probing of results to be perfomed retrospectively at any step of the workflow (and not just to the output of the last operation), by simply pointing to the corresponding block. Such a representation of the workflow might lead the user to perform changes to previous steps in the pipeline, e.g., the modification of some parameter, or the insertion of a new block. Such upstream changes will, in general, affect the data flow through the pipeline; in particular, they might load to an empty output from one block, resulting in an empty input for one or more subsequent ones. In order to assist the user in identifying and assessing this sort of issues, we have introduced small visual cues on each of the blocks indicating their execution state by means of color with a traffic-light metaphor.

Implementation We have implemented this design in a software prototype with a client-server architecture built on the foundations of the Apache TinkerPop framework [1]; its cornerstone, the Gremlin language and traversal engine, constitutes the *de facto* standard amongst graph traversal languages. On the client side, we have built an AngularJS web application following an MVC pattern: the graphical block composition subsystem acts as controller for defining a query model, which in turn results in updates of the view provided by the probing subsystem. We have used an adapted version of the Blockly library for building the graphical query composition subsystem, as well as d3.js for the visualizations used in the probing subsystem. On the server side, a standard Gremlin Server instance running a compatible graph database backend (e.g., Neo4j). Finally, client-server communication relies on the gremlin-javascript library, which provides synchronous and asynchronous connectivity.

4 Conclusion

The scale and complexity of large multivariate graph data hinder the application of top-down analytic techniques for exploratory analysis. Approaches following an exploration paradigm based on search and directed expansion through the graph have proven effective in different scenarios. We have designed and implemented a visually-supported environment for graph traversal that combines graphical composition using a drag-and-drop block metaphor, with a visual data probing mechanism for consuming data at large scales and different output contexts. In future work, we aim at achieving a tighter integration of the traversal composition canvas and the probing component, as well as providing a complete provenance description of the traversal process by including a representation of the history of block modification and removal actions.

Acknowledgements

The authors wish to thank Johannes Mauerer, Lukas Mayr, Andreas Roschal, and Xiashuo Lin, who contributed parts of the implementation, as well as their partners within the research project EXPAND (EXploratory Visualization of PAtent Network Dynamics), funded by the Austrian Research Promotion Agency (FFG), grant number 835937.

References

[1] Apache Tinkerpop. `http://tinkerpop.apache.org/`. Accessed August 5, 2016.

[2] Cypher Query Language. `http://neo4j.com/developer/cypher/`. Accessed August 5, 2016.

[3] Linkurious. `http://linkurio.us/`. Accessed August 5, 2016.

[4] M. Rodriguez and P. Neubauer. *Graph Data Management: Techniques and Applications*, chapter The Graph Traversal Pattern. IGI Global, 2011.

[5] M. A. Rodriguez. The Gremlin graph traversal machine and language (invited talk). In *Proceedings of the 15th Symposium on Database Programming Languages*, pages 1–10. ACM, 2015.

[6] F. van Ham and A. Perer. "Search, show context, expand on demand": Supporting large graph exploration with degree-of-interest. *IEEE TVCG*, 15(6):953–960, Nov 2009.