Towards Scalable Visual Exploration of Very Large RDF Graphs

Nikos Bikakis[1,2(✉)], John Liagouris[3], Maria Kromida[1], George Papastefanatos[2], and Timos Sellis[4]

[1] NTU Athens, Athens, Greece
bikakis@dblab.ntua.gr
[2] ATHENA Research Center, Athens, Greece
[3] ETH Zürich, Zürich, Switzerland
[4] RMIT University, Melbourne, Australia

Abstract. In this paper, we outline our work on developing a disk-based infrastructure for efficient visualization and graph exploration operations over very large graphs. The proposed platform, called graphVizdb, is based on a novel technique for indexing and storing the graph. Particularly, the graph layout is indexed with a spatial data structure, i.e., an R-tree, and stored in a database. In runtime, user operations are translated into efficient spatial operations (i.e., window queries) in the backend.

Keywords: graphVizdb · Graph data · Disk based visualization tool · RDF graph visualization · Spatial · Visualizing linked data · Partition based graph layout

1 Introduction

Data visualisation provides intuitive ways for information analysis, allowing users to infer correlations and causalities that are not always possible with traditional data mining techniques. The wide availability of vast amounts of graph-structured data, RDF in the case of the Data Web, demands for user-friendly methods and tools for data exploration and knowledge uptake. We consider some core challenges related on the management and visualization of very large RDF graphs; e.g., the Wikidata RDF graph has more than 300M nodes and edges.

First, their size exceeds the capabilities of memory-based layout techniques and libraries, enforcing disk-based implementations. Then, graph rendering is a time consuming process; even drawing a small part of the graph (containing a few hundreds of nodes) requires considerable time when we assume real-time systems. The same holds for graph interaction and navigation. Most operations, such as zoom in/out and move, are not easily implemented to large dense graphs, as their implementations require redrawing and re-layout large parts of them.

Related works in the field handle very large graphs through hierarchical visualization approaches. Although hierarchical approaches provide fancy visualizations with low memory requirements, their applicability is heavily based on the

F. Gandon et al. (Eds.): ESWC 2015, LNCS 9341, pp. 9–13, 2015.
DOI: 10.1007/978-3-319-25639-9_2

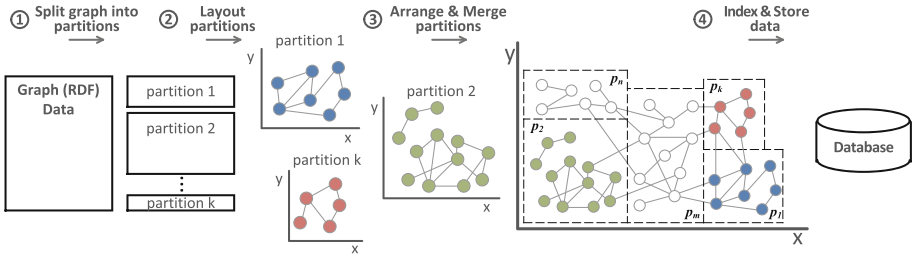

Fig. 1. Preprocessing overview

particular characteristics of the input dataset. In most cases, the hierarchy is constructed by exploiting clustering and partitioning methods [1,4,5,14,19]. In other works, the hierarchy is defined with hub-based [15] and destiny-based [22] techniques. [3] supports ad-hoc hierarchies which are manually defined by the users. Some of these systems offer a disk-based implementation [1,14,19] whereas others keep the whole graph in main memory [3–5,15,22].

In the context of the Web of Data [2,7–9,16,17,20], there is a large number of tools that visualize RDF graphs (adopting a node-link approach); the most notable ones are *ZoomRDF* [21], *Fenfire* [12], *LODWheel* [18], *RelFinder* [13] and *LODeX* [6]. All these tools require the whole graph to be loaded on the UI. Several tools that follow the same non-scalable approach have also been developed in the field of ontology visualization [10,11].

In contrast to all existing works, we introduce a generic platform, called graphVizdb, for scalable graph visualization that do not necessarily depend on the characteristics of the dataset. The efficiency of the proposed platform is based on a novel technique for indexing and storing the graph. The core idea is that in a preprocessing phase, the graph is drawn, using any of the existing graph layout algorithms. After drawing the graph, the coordinates assigned to its nodes (with respect to a Euclidean plane) are indexed with a spatial data structure, i.e., an R-tree, and stored in a database. In runtime, while the user is navigating over the graph, based on the coordinates, specific parts of the graph are retrieved and send to the user.

2 Platform Overview

The graphVizdb platform is built on top of two main concepts: (1) it is based on a "spatial-oriented" approach for graph visualization, similar to approaches followed in browsing maps; and (2) it adopts a disk-based implementation for supporting interaction with the graph, i.e., a database backend is used to index and store graph and visual information.

Partition-Based Graph Layout. Here we outline the partition-based approach adopted by the graphVizdb in order to handle very large graph. Recall that, for graph layout, the graph is drawn once in a preprocessing phase, using

any of the existing graph layout algorithms. However several graph layout algorithms require large amount of memory in order to draw very large graphs. In order to overcome this problem, our partition-based approach (outlined in Fig. 1) is described next.

(1) Initially, the graph (RDF) data is divided into a set of smaller sub-graphs (i.e., partitions) using a graph partitioning algorithm. At the same time, the graph partitioning algorithm tries to minimize the number of edges connecting nodes in different partitions. (2) Then, using a graph layout algorithm, each of the sub-graph resulted from the graph partitioning, is visualized into a Euclidean plane, excluding (i.e., not visualizing) the edges connecting nodes through different partitions (i.e., crossing edges). (3) The visualized partitions are organized and combined into a "global" plane using a greedy algorithm whose goal is twofold. First, it ensures that the distinct sub-graphs do not overlap on the plane, and at the same time it tries to minimize the total length of the crossing edges. (4) Based on the "global" plane, the coordinates for each node and edge are indexed and stored in the database.

Spatial Operations for Graph Exploration. In graphVizdb, most of the user's requests are translated into simple spatial operations evaluated over the database. In this context, *window queries* (i.e., spatial range queries that retrieve the information contained with in a specific spatial region) are the core operation for most user's requests. The user navigates on the graph by moving the viewing window. When the window is moved, its new coordinates with respect to the whole canvas are tracked on the client side, and a window query is sent to the server. The query is evaluated on the server using the R-tree indexes. This way, for each user request, graphVizdb efficiently renders only visible parts of the graph, minimizing in this way both backend-frontend communication cost as well as rendering and layout time. Additionally, more sophisticated operations, e.g., abstraction/enrichment zoom operations are also implemented using spatial operations.

Implementation. We have implemented a graphVizdb prototype[1] which provides interactive visualization over large graphs. The prototype offers three main operations: (1) interactive navigation, (2) multi-level exploration, and (3) keyword search. We use MySQL for data storing and indexing, the Jena framework for RDF data handling, Metis[2] for graph partitioning, and Graphviz[3] for drawing the graph partitions. In the front-end, we use mxgraph[4], a client-side JavaScript visualization library. A video presenting the basic functionality of our prototype is available at: vimeo.com/117547871.

Acknowledgement. This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Pro-

[1] graphvizdb.imis.athena-innovation.gr.
[2] glaros.dtc.umn.edu/gkhome/views/metis.
[3] www.graphviz.org.
[4] www.jgraph.com.

grams “Education and Lifelong Learning” - Funding Program: THALIS and “Competitiveness and Entrepreneurship” (OPCE II) - Funding Program: KRIPIS of the National Strategic Reference Framework (NSRF).

References

1. Abello, J., van Ham, F., Krishnan, N.: ASK-GraphView: a large scale graph visualization system. IEEE Trans. Vis. Comput. Graph. **12**(5), 669–676 (2006)
2. Alonen, M., Kauppinen, T., Suominen, O., Hyvönen, E.: Exploring the linked university data with visualization tools. In: Cimiano, P., Fernández, M., Lopez, V., Schlobach, S., Völker, J. (eds.) ESWC 2013. LNCS, vol. 7955, pp. 204–208. Springer, Heidelberg (2013)
3. Archambault, D., Munzner, T., Auber, D.: GrouseFlocks: steerable exploration of graph hierarchy space. IEEE Trans. Vis. Comput. Graph. **14**(4), 900–913 (2008)
4. Auber, D.: Tulip - a huge graph visualization framework. In: Jünger, M., Mutzel, P. (eds.) Graph Drawing Software, pp. 105–126. Springer, Heidelberg (2004)
5. Bastian, M., Heymann, S., Jacomy, M.: Gephi: an open source software for exploring and manipulating networks. In: ICWSM (2009)
6. Benedetti, F., Po, L., Bergamaschi, S.: A visual summary for linked open data sources. In: ISWC (2014)
7. Bikakis, N., Skourla, M., Papastefanatos, G.: rdf:SynopsViz – a framework for hierarchical linked data visual exploration and analysis. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC Satellite Events 2014. LNCS, vol. 8798, pp. 292–297. Springer, Heidelberg (2014)
8. Brunetti, J.M., Auer, S., García, R., Klímek, J., Necaský, M.: Formal linked data visualization model. In: IIWAS (2013)
9. Dadzie, A., Rowe, M.: Approaches to visualising linked data: a survey. Semant. Web **2**(2), 89–124 (2011)
10. Dudáš, M., Zamazal, O., Svátek, V.: Roadmapping and navigating in the ontology visualization landscape. In: Janowicz, K., Schlobach, S., Lambrix, P., Hyvönen, E. (eds.) EKAW 2014. LNCS, vol. 8876, pp. 137–152. Springer, Heidelberg (2014)
11. Fu, B., Noy, N.F., Storey, M.-A.: Eye tracking the user experience - an evaluation of ontology visualization techniques. Semant. Web J. (2015)
12. Hastrup, T., Cyganiak, R., Bojars, U.: Browsing linked data with fenfire. In: WWW (2008)
13. Heim, P., Lohmann, S., Stegemann, T.: Interactive relationship discovery via the semantic web. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 303–317. Springer, Heidelberg (2010)
14. Rodrigues Jr., J.F., Tong, H., Traina, A.J.M., Faloutsos, C., Leskovec, J.: GMine: a system for scalable, interactive graph visualization and mining. In: VLDB (2006)
15. Lin, Z., Cao, N., Tong, H., Wang, F., Kang, U., Chau, D.H.P.: Demonstrating interactive multi-resolution large graph exploration. In: ICDM (2013)
16. Mazumdar, S., Petrelli, D., Ciravegna, F.: Exploring user and system requirements of linked data visualization through a visual dashboard approach. Semant. Web **5**(3), 203–220 (2014)
17. Mazumdar, S., Petrelli, D., Elbedweihy, K., Lanfranchi, V., Ciravegna, F.: Affective graphs: the visual appeal of linked data. Semant. Web **6**(3), 277–312 (2015)
18. Stuhr, M., Roman, D., Norheim, D.: LODWheel - JavaScript-based visualization of RDF data. In: Workshop on Consuming Linked Data (2011)

19. Tominski, C., Abello, J., Schumann, H.: CGV - an interactive graph visualization system. Comput. Graph. **33**(6), 660–678 (2009)
20. Vocht, L.D., Dimou, A., Breuer, J., Compernolle, M.V., Verborgh, R., Mannens, E., Mechant, P., de Walle, R.V.: A visual exploration workflow as enabler for the exploitation of linked open data. In: IESD (2014)
21. Zhang, K., Wang, H., Tran, D.T., Yu, Y.: ZoomRDF: semantic fisheye zooming on RDF data. In: WWW (2010)
22. Zinsmaier, M., Brandes, U., Deussen, O., Strobelt, H.: Interactive level-of-detail rendering of large graphs. IEEE Trans. Vis. Comput. Graph. **18**(12), 2486–2495 (2012)