Visual Querying of Large Multilayer Graphs

Erick Cuenca
LIRMM, Univ. Montpellier, CNRS
Montpellier, France
erick.cuenca@lirmm.fr

Arnaud Sallaberry
LIRMM, Univ. Paul-Valéry Montpellier 3, Univ.
Montpellier, CNRS
Montpellier, France
arnaud.sallaberry@lirmm.fr

Dino Ienco
IRSTEA, UMR TETIS
LIRMM, Univ. Montpellier
Montpellier, France
dino.ienco@irstea.fr

Pascal Poncelet
LIRMM, Univ. Montpellier, CNRS
Montpellier, France
pascal.poncelet@lirmm.fr

ABSTRACT

Many real world data can be represented by a network with a set of nodes linked each other by multiple relations. Such a rich graph is called multilayer graph. In this demo, we present a tool for Visual Querying of Large Multilayer Graphs that allows to visually draw the query, retrieve result patterns and finally navigate and browse the results considering the original multilayer graph database. Our approach does not only provide a graphical user interface for the graph engine but the query processing is fully integrated.

CCS CONCEPTS

• **Human-centered computing** → **Visualization**; • **Information systems** → *Data management systems*;

KEYWORDS

visual graph querying, subgraph results, multilayer graphs

ACM Reference Format:
Erick Cuenca, Arnaud Sallaberry, Dino Ienco, and Pascal Poncelet. 2018. Visual Querying of Large Multilayer Graphs. In *SSDBM '18: 30th International Conference on Scientific and Statistical Database Management, July 9–11, 2018, Bozen-Bolzano, Italy.* , 4 pages. https://doi.org/10.1145/3221269.3223027

1 INTRODUCTION

Many real world data can be represented by a network with a set of nodes linked each other by multiple relations. Such a rich graph is called multilayer graph and allows the definition of multiple types of relationships (edge) between vertices [1]. Examples of multilayer graphs are: social networks spanning over the same set of people, but with different life aspects (e.g. social relationships - Facebook, Twitter, work relationships - LinkedIn, etc.); bibliographic networks where nodes are authors and the edge type spans over the set of conferences or journals [1]; protein-protein interaction networks where each layer represents a different pathway on which proteins interact [12] and RDF knowledge graph where the same subject/object node pair is connected by different predicates [8].

Query efficiently this kind of data is only the first step to support end user analysis [6]. In order to facilitate such analysis, several visual query and exploration systems for graph databases exist in literature [9, 11] but, to the best of our knowledge, none of them can deal with neither multilayer networks nor the exploration of the results on the original graph database.

In this demo paper, we present a new efficient visual tool to query multilayer graphs and explore the obtained result patterns through navigation on the original graph database. Our system not only allows to visually draw the multilayer query graph and browse the corresponding result patterns but, it also supports the interaction between the user and the underlying query process. The user can stop and resume the graph engine with the possibility to inspect partial collected results. Finally, the results can be analyzed at different levels of detail: globally, projecting the set of results on the original graph database and locally, focusing on a portion of the database inspecting particular nodes involved in the query result.

2 A MULTILAYER GRAPH VISUALIZATION TOOL

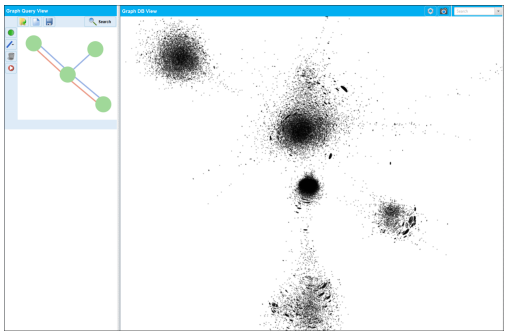
Our approach has three main functionalities: i) a graph visualization component to show the whole graph database; ii) an interface to graphically specify multilayer query graphs and iii) an interactive system to browse and visualize query result patterns at different levels of detail. Furthermore, our tool not only provides the visualization functionalities but it also tightly integrates the multilayer graph engine [6] that processes the query and retrieves the result patterns from the multilayer graph database.

2.1 Graph DB Visualization

With the aim to produce a suitable visualization of the multilayer graph database, we adopt a force direct layout where edges act as springs and nodes as repulsive forces [7]. Working with large-graph entails some difficulties: high cluttering, scalability and reasonable time interaction. In order to deal with all these issues, we leverage a multilevel technique named Multipole Multilevel Method FM3 [4]. FM3 is a well known approach in the graph visualization field that allows to manage large graphs with a good trade off between time

SSDBM '18, July 9–11, 2018, Bozen-Bolzano, Italy

ACM ISBN 978-1-4503-6505-5/18/07.
https://doi.org/10.1145/3221269.3223027



(a)

(b)

Figure 1: Visualization examples of our approach. (a) The multilayer graph visualization of the graph database (b) The Heat Map visualization showing the retrieved result patterns on the multilayer graph.

and visualization performances. An example of graph layout visualization obtained by our visual tool is presented in Figure 1a. The input graph is a biological graph database [1] composed of 38 936 nodes, 310 664 edges and 7 edge types (the layers of the multilayer graph). We note that such visualization has the advantage to distinguish groups of nodes that clearly form communities. In this particular example we clearly identify five different node groups.

2.2 Querying Multilayer Graph DB

With the aim to query the graph database, we conceive an intuitive and simple graphic interface to visually draw the multilayer graph query. This interface allows the user to define the query graph structure (nodes, edge types, data-attributes) via different interaction buttons. Figure 2 depicts the query graph interface with an example of multilayer graph query. We can note that the interface contains two tool bars. The left sided toolbar includes different actions to build the query graph. From top to bottom, the actions are: add a node, add an edge of a specific layer, delete a node or an edge and arrange the layout of the query graph by applying a force directed algorithm. The user can draw edges of several types (layers) distinguishing them by different colours. The user can specify a node attribute by right-clicking on the desired node then, selecting a value from the displayed pop-up list (Figure 2.a). The upper central toolbar supplies standard features such as: load, create, and save a query graph we have defined. This toolbar also contains the *Search* button that runs the multilayer graph query engine. Since a multilayer graph involves different relationships between nodes, standard query engines such as TurboISO [5] or VF3 [2] are not adapted for this kind of task. To deal with this issue we integrate in our approach a recent multilayer query engine named SuMGra [6]. This query engine exploits specialised indexing techniques in order to speed up the backtracking algorithm that is commonly employed to deal with the subgraph isomorphism problem [5].

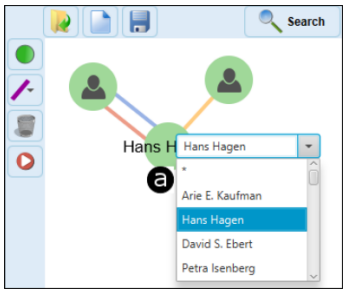


Figure 2: The multilayer graph query interface.

2.3 Browsing Result Patterns

Once the *SuMGra* query engine starts to produce results, they can be visually analyzed through our approach. Since the number of result patterns can be large, visualizing them is challenging. To overcome this issue, we have two different levels of analysis: i) a global level in which we analyze query results considering the multilayer graph database and ii) a local level in which we can inspect particular nodes involved in the resulting patterns.

Considering the global level, we employ an Heat Map [10] visual representation that dynamically illustrates the portion of the graph database in which results are found. However, the high abstraction level supplied by the heat map makes it difficult to know the exact number/distribution of pattern nodes that are found in a region. To deal with this drawback, our approach allows to change the levels of detail focusing (and defocusing) on some particular node. The user can i) select a set of nodes (involved in the result patterns) ii) visualize the list of associated result patterns and iii) get a condensed (visual) representation of them.

2.3.1 Heat Map. Figure 1b shows the Heat Map visualization of the results induced by the query graph of Figure 2 on the data graph of Figure 1a. The density distribution is represented by color gradient. Such gradient varies between red (high dense area) to blue (low dense area). In the example depicted in Figure 1, we can highlight that the Graph Visualization produces different node communities since it groups the nodes in different visual clusters, and the query results are not equally distributed among these groups. As interesting point, thank to this visualization, we can note a high density (high concentration of pattern results visually depicted in red) in the center of the image while a low concentration of the query graph appears elsewhere. Our Heat Map mechanism can be employed to visually understand if a multilayer query graph is specific to a certain part of the graph database or not.

Our approach allows the user to stop and resume the underlying query process.This can be particularly useful if the number of results grows quickly. In that scenario, the process can be stopped at any moment, the results are fetched considering the current state of the query engine and the corresponding *Heat Map* visualization is produced or updated. Finally, the user can restart the query processing that will continue from the point it was stopped.

2.3.2 Node Oriented Result Patterns List. After the *Heat Map* visualization is produced, the user can zoom on a particular area of the graph database to select a set of nodes that are involved in the result set. Once a set of nodes is selected, a list of *Result Patterns* can be visualized on the right side of our tool. Considering Figure 3, we can see on the right of this image a portion of the pattern list. Logically, each result pattern has the same structure of the query and it involves the previously selected nodes.

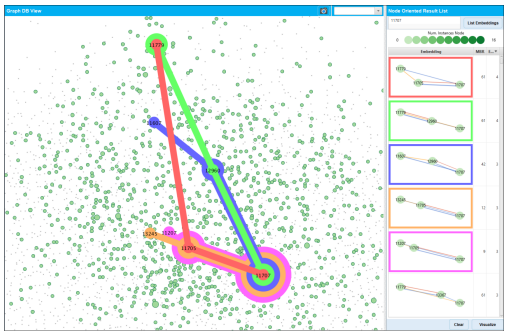


Figure 3: Selection of five result patterns (on the right) and the related Kelp-like diagram (on the left). The color links the result patterns and the diagrams.

2.3.3 Node Oriented Result Patterns Visualization. From the *Node Oriented Result Patterns List*, the user can choose up to five different result patterns that we can visually summarize via a Kelp-like diagram [3]. Figure 3 represents such condensed result patterns visualization. The visual summary involves all the nodes covered by the set of chosen result patterns and the edges between them. For each chosen result pattern (on the right of the figure) a minimum spanning tree is obtained (on the left of the figure) with the purpose to visually underline nodes belonging to the same query result. More specifically, the different colors make a visual link between the selected *Result Patterns* (on the right) and the diagram visualization (on the left).

2.4 Implementation and Equipment

Our approach is developed using Java technologies (JavaFX, Processing[1]) for interface and interaction among the different components. We use the FM3 implementation supplied by the Open Graph Drawing Framework (OGDF)[2] for graph layout placement. The graph database engine [6] is also implemented in Java and integrated in our visual tool. Demonstration scenario is carried out on a Work Station with an Intel Xeon processor at 3.00 GHz with 16 GB of RAM memory with a 64 bits Windows OS.

[1]Processing Website: https://processing.org/
[2]OGDF Website: http://www.ogdf.net/

3 DEMONSTRATION SCENARIO

The following scenario shows the use of our approach and it describes what we will present to the audience. A demonstration video is available here https://goo.gl/Hy3k4q.

3.1 Dataset

For our scenario, we consider a real co-authorship network obtained from the *DBLP*[3] bibliographic database. The network is composed by 37 878 nodes (the authors) and 129 983 edges (co-authorship). The multilayer network has 19 layers (edge types) each of them corresponding to a different venue (journal or conference) on which two authors have published. Two nodes are linked together if the corresponding authors have co-authored at least one paper together in that specific venue. The different venues are chosen considering the Visualization and the Data Mining fields of research.

3.2 Visual Analysis

Figure 4 depicts both: the *DBLP* multilayer graph (on the right) and the query graph (on the left) we have used for the case study. Considering the multilayer graph database, nodes (authors) are shown as black circles with equal sizes and edges are not depicted to avoid cluttering. We can note that the chosen directed force layout algorithm allows to visually recognize two dense areas. It is seems logical since the *DBLP* dataset is composed by two different communities (Visualization and Data Mining).

Figure 4: Visualization of the *DBLP* dataset. On the left, the graph query. On the right, the multilayer graph database including a Heat Map visualization showing dense areas w.r.t. the submitted query.

Figure 5 shows the graphical user interface we conceived to draw the multilayer graph query. Considering our scenario, the objective of this query is to retrieve groups of authors (three authors) that collaborate together and one of them (the central node of the query) has publications in both fields: he/she published a *TVCG* paper with one author and he/she also published an *ICDM* and a *KDD* article with another author. More precisely, we explicitly state that the author corresponding to the central node has published both an *ICDM* and a *KDD* article with the same co-author. We can observe that the multilayer graph structure allows us to easily specify multiple constraints between the same pair of nodes.

[3]DBLP Website: http://dblp.uni-trier.de/

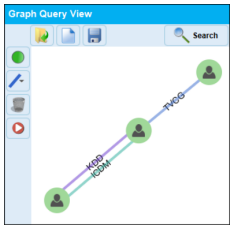


Figure 5: The multilayer graph query we used in Figure 4.

Once the query is executed, our visual tool retrieves the result patterns and start to produce the Heat Map visualization in order to locate the obtained results (Figure 4). As we describe before, the heat map visualization is smoothly updated since the user can stop and resume the process at any time. The heat map visualization shows to the user dense areas where the result patterns are found. In order to navigate and explore the results obtained by the graph engine, we can focus on the densest area of the heat map (the red areas). Figure 6 shows the result of a zoom operation on one of these dense areas. We can observe that, once we change the level of detail, node labels appear. We also see that nodes have different size. The size of a node is proportional to the number of result patterns it belongs to. This means that bigger nodes correspond to the authors that occur in more query answers. In this example, we focus on the author *Shixia Liu*, since this node occurs in many result patterns (we can see that the size of the corresponding nodes is bigger w.r.t. its neighbours). Once the node is selected, we are able to retrieve the list of result patterns it belongs to. This list is drawn in the right side of Figure 6. We remind that each graph that belongs to the pattern list has the same topological structure than the input query.

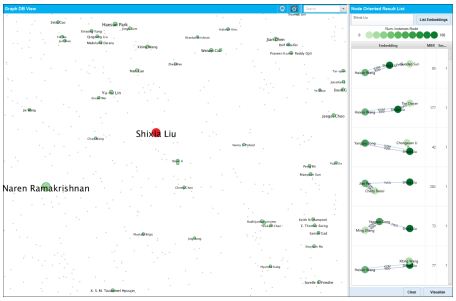


Figure 6: Focus on a portion of the graph guided by the Heat Map visualization.

From the *Node Oriented Result Patterns List*, we select 4 result patterns in order to convey them using a Kelp-like diagram visualization. Figure 7 shows the Kelp-like diagram associated to the selection. Every selected result is bordered by a different color. The same color is used to depict the Kelp-like diagram on the multilayer graph database (on the right of Figure 7). This kind of visualization allows to easily browse and explore the set of nodes associated to the result patterns. We can observe, for example, that the node labeled *Yang Giu Song* (arrow 1) and the node labeled *Haixun Wang* (arrow 2) do not belong to the same set of result patterns since they are involved in different answers (they did not share any common color). Having in mind the query structure, in this particular case we can also note that the authors *Xizhou Zhu* (arrow 3) and *Panpan Xu* (arrow 4) are researchers that make a bridge between the two communities since they are matched by the central node of the query. Finally, the Kelp-diagram also helps to understand the number of result patterns a node participate to (regarding the selected subset). This can be easily deduced by counting the number of different colored circles around it.

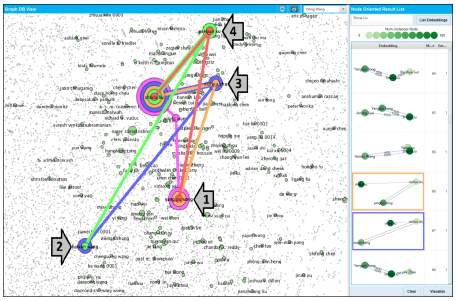


Figure 7: The Kelp-like diagram associated to the selected pattern results from the right list.

4 CONCLUSION

In this demo paper, we have presented a multilayer graph visualization tool that allows to query multilayer graphs and explore the obtained result patterns through navigation on the original database. The results can be analyzed at different levels of detail and the conceived visual components support the user to focus on particular interesting areas of the multilayer graph. The proposed system also enables the interaction between the user and the query process. The user can stop and resume the graph engine with the possibility to inspect partial collected results. We will invite our audience to test our approach and experiment their own queries.

REFERENCES

[1] F. Bonchi, A. Gionis, F. Gullo, and A. Ukkonen. 2014. Distance oracles in edge-labeled graphs. In *EDBT*. 547–558.
[2] V. Carletti, P. Foggia, A. Saggese, and M. Vento. 2017. Introducing VF3: A New Algorithm for Subgraph Isomorphism. In *GbRPR*. 128–139.
[3] Kasper D., M. J van Kreveld, B. Speckmann, and M. A Westenberg. 2012. Kelp diagrams: Point set membership visualization. In *Computer Graphics Forum*, Vol. 31. 875–884.
[4] S. Hachul and M. Jünger. 2004. Drawing large graphs with a potential-field-based multilevel algorithm.. In *Graph Drawing*. 285–295.
[5] W.-S. Han, J. Lee, and J.-H. Lee. 2013. Turbo$_{iso}$: towards ultrafast and robust subgraph isomorphism search in large graph databases. In *SIGMOD*. 337–348.
[6] V. Ingalalli, D. Ienco, and R. Poncelet. 2016. SuMGra: Querying Multigraphs via Efficient Indexing. In *Proceedings of the 27th. International Conference on Database and Expert Systems Applications*. Springer, Porto, Portugal, 1–15.
[7] M. Kaufmann and D. Wagner. 2003. *Drawing graphs: methods and models*. Springer.
[8] L. Libkin, J. Reutter, and D. Vrgoč. 2013. Trial for RDF: adapting graph query languages for RDF data. In *PODS*. ACM, 201–212.
[9] Robert P., Fred H., Alex E., Acar T., Kevin R., Chris G., Shamkant N., and Duen Horng C. 2018. VIGOR: Interactive Visual Exploration of Graph Query Results. *TVCG* 24, 1 (2018), 215–225.
[10] A. Perrot, R. Bourqui, N. Hanusse, F. Lalanne, and D. Auber. 2015. Large interactive visualization of density functions on big data infrastructure. In *LDAV*. 99–106.
[11] P. Yi, B. Choi, S. S. Bhowmick, and J. Xu. 2017. AutoG: a visual query autocompletion framework for graph databases. *VLDB J.* 26, 3 (2017), 347–372.
[12] A. Zhang. 2009. Protein Interaction Networks: Computational Analysis. (2009).