TOURVIZ: Interactive Visualization of Connection Pathways in Large Graphs

Duen Horng (Polo) Chau
Carnegie Mellon University
dchau@cs.cmu.edu

Leman Akoglu
Carnegie Mellon University
lakoglu@cs.cmu.edu

Jilles Vreeken
University of Antwerp
jilles.vreeken@ua.ac.be

Hanghang Tong
IBM T.J. Watson Research
htong@us.ibm.com

Christos Faloutsos
Carnegie Mellon University
christos@cs.cmu.edu

ABSTRACT

We present TOURVIZ, a system that helps its users to interactively visualize and make sense in large network datasets. In particular, it takes as input a set of nodes the user specifies as of interest and presents the user with a visualization of connection subgraphs around these input nodes. Each connection subgraph contains good pathways that highlight succinct connections among a 'close-by' group of input nodes. TOURVIZ combines visualization with rich user interaction to engage and help the user to further understand the relations among the nodes of interest, by exploring their neighborhood on demand as well as modifying the set of interest nodes.

We demonstrate TOURVIZ's usage and benefits using the DBLP graph, consisting of authors and their co-authorship relations, while our system is designed generally to work with any kind of graph data. We will invite the audience to experiment with our system and comment on its usability, usefulness, and how our system can help with their research and improve the understanding of data in other domains.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]; H.5.2 [**Information Interfaces and Presentation**]

General Terms

Algorithms, Design, Human Factors

Keywords

Sensemaking, Large networks, Connection subgraphs

1. INTRODUCTION

Finding associations among a set of objects is an important problem in many domains ranging from biology (gene/protein interactions), security (criminal/terrorist interactions), immunology (patient interactions), and so on. Often, these objects are connected with certain type of relations within a large network. In such a network setting, the task of finding associations among objects can be thought of finding succinct connection pathways among these objects. While these connection pathways reveal how the set of objects associate with one another in the network, the additional nodes (connectors) revealed on these pathways provide useful information about other nodes that might be of interest in understanding these associations.

The above setting finds numerous applications in diverse domains. For example, given a gene interaction network, an experiment may reveal for particular conditions a number of genes to be up- (or down-) regulated, and a biologist might be interested in finding associations between these genes. This would give him/her a good summary of possible pathways the genes may be involved in. Another example relates to security; given a set of suspicious people detected by an anomaly detection algorithm, an analyst could use our system to group them to reveal associations; by finding how as well as through whom else they are connected in the network. Moreover, given an event affecting a set of people connected via a network, e.g. people in a geographical region affected by a certain disease, an immunolog can use our system to figure out the pathways through which the disease spread among these people, potentially discovering other affected people on the connection pathways. More applications can be given, however, the main point

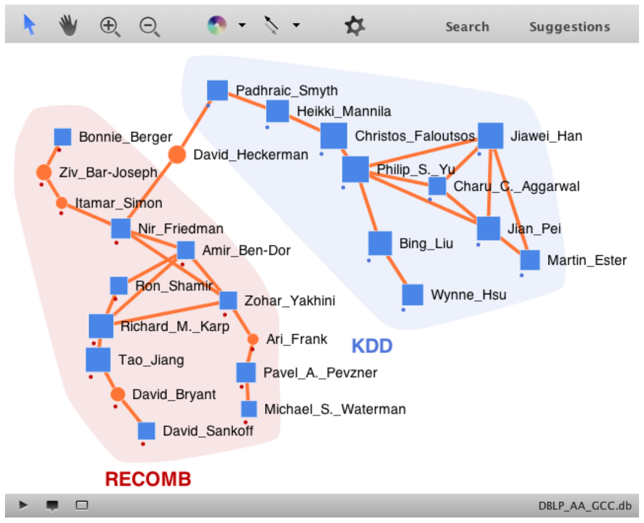

Figure 1: Screenshot of TOURVIZ showing a user exploring the connection subgraphs among a set of authors of interest (square nodes) in KDD and RECOMB (computational biology). Edges are co-authorship relations.

is that the system we shortly introduce is quite general and can be used in various settings independent of a specific domain.

We developed TOURVIZ (Figure1), an interactive visualization system, for a domain analyst to explore and understand the associations among a set of objects (i.e. nodes) in a network. The analyst selects a set of nodes that s/he is interested in within the network and specifies them as 'marked'. Next, within TOURVIZ an efficient algorithm finds succinct connection subgraphs among the marked nodes. Intuitively, 'close-by' (highly associated nodes) in the network can be connected by simple paths, while 'far away' (not-so-highly associated) nodes are hard to link together. Therefore, the algorithm returns possibly multiple connection subgraphs for multiple groups of 'close-by' nodes. TOURVIZ offers visualization features to show the analyst these groups and their associated connection subgraphs found by our algorithm. Our system also embodies interaction features for the analyst to update the nodes of interest on demand.

We summarize our main contributions as follows:

- We build a new system called TOURVIZ to provide users with the understanding of how a set of nodes in a network are associated, through visualizing good connections and connectors among them. TOURVIZ integrates novel algorithms we developed [1] for finding succinct connection pathways among groups of highly associated nodes.
- TOURVIZ integrates our novel algorithm into an interactive environment that allows users to easily specify nodes of interest, reveal connections among them through visualization, and iteratively refine their specifications to improve their understanding.

2. DEMONSTRATING TOURVIZ

We will demonstrate TOURVIZ's usage, user interaction and algorithm through scenarios on understanding connections among researchers in various computer science fields in the DBLP[1] coauthorship graph, which contains about 329K authors (nodes) and 1094K coauthorship relations (edges).

Scenario. Here, we illustrate one example scenario, where our user uses TOURVIZ to explore and understand the connections between several researchers in KDD (data mining, machine learning) and RECOMB (computational biology). This scenario will touch upon major features of TOURVIZ.

Our user begins by searching for authors that he is familiar with, using TOURVIZ's search feature (Figure 2). Authors whose names contains the search text show up as a list. They can be sorted alphabetically, or by various metrics that TOURVIZ has pre-computed for the graph (the first time the graph is loaded), e.g., PageRank score, degree, etc. Once a match is found, our user drags the name into the visualization, which turns into a circle, its size proportional to its coauthor count (i.e., node degree).

Our user proceeds to drag in more authors he is familiar with, in the domains of KDD and RECOMB. Although the user is familiar with these researchers' work, s/he does not recall how they have been collaborating. In particular, s/he is interested in using TOURVIZ to figure out the researchers who have been working at the intersection of these two domains.

To help keep track of the two groups of researchers, our user groups them into KDD and RECOMB using TOURVIZ's *grouping* feature (Figure 3). Each group is assigned a color, and its nodes enclosed by a convex hull.

These are all the nodes that our user wants to create a connection subgraph for (Figure 3). To mark these nodes as inputs, our user

[1]http://dblp.uni-trier.de/

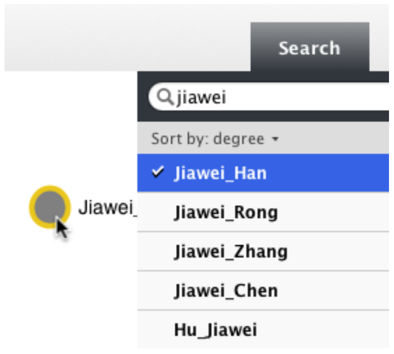

Figure 2: User searching for "jiawei". Matching authors shown in a list, sorted by the authors' coauthor counts (i.e., node *degrees* in the graph. Dragging the name of "Jiawei Han" into the visualization turns it into a circle; the name becomes the node label. Node size is scaled by degree. The yellow halo around the node indicates it is a match.

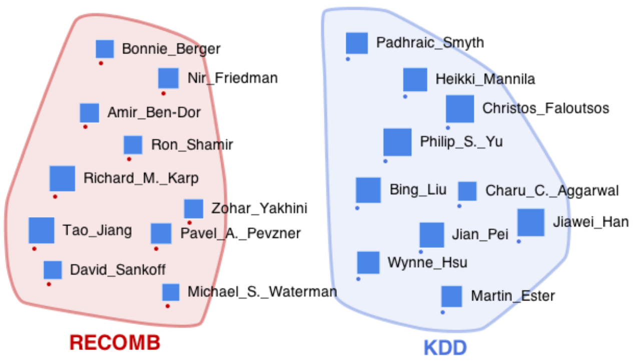

Figure 3: Our user has dragged multiple authors into the visualization, and grouped them into KDD and RECOMB using TOURVIZ's *grouping* feature. Each group is enclosed by a convex hull. Group names are provided by the user. Blue square nodes are the *marked* input nodes to TOURVIZ.

changes their shapes into squares (node colors do not matter). Next, they are given as input to TOURVIZ's algorithm that finds the best subgraph. We will explain how the algorithm works in more detail in Section 3. Simply put, it finds a simple subgraph spanning all the input nodes that also has as few edges and as few additional intermediary nodes as possible.

Figure 4 shows the result of the algorithm —a tree whose edges are shown as thick orange lines, connecting the *marked* nodes with a few *intermediary* nodes (orange circles). Thin edges are other relations among all the nodes being displayed, but not part of the tree. Most interesting to our user is the discovery of David Heckerman; a prolific researcher who have been publishing with authors who frequent KDD and RECOMB. Additional discoveries include the intermediary nodes, that is the authors such as Ziv Bar-Joseph and Ari Frank, who are good connectors among the authors that our user is interested in observing the relations.

Here, since the whole tree has few nodes, our user can rearrange the nodes manually to create a visually pleasing layout. Optionally, he can also use TOURVIZ's built-in force-directed *layout* feature to automatically layout the nodes, which often helps reduce node and edge overlap.

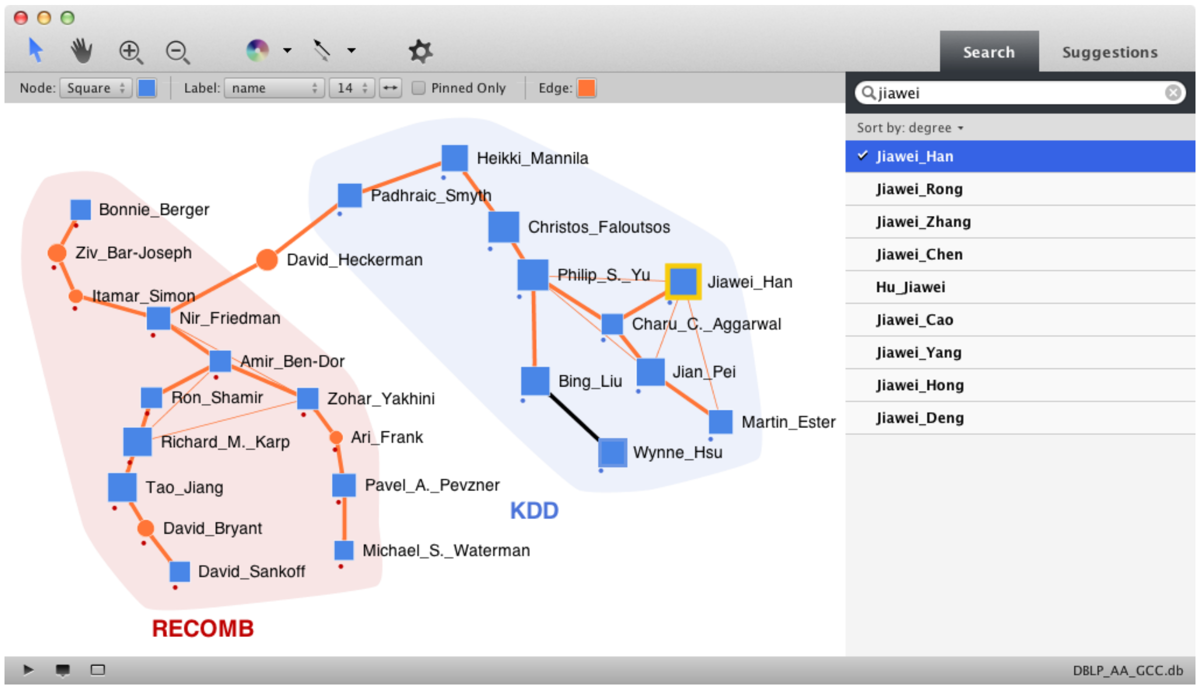

Figure 4: Screenshot of TOURVIZ showing our user exploring the connection subgraph among a set of authors of interest (blue, square nodes) in KDD and RECOMB (computational biology). Node shape is set using the first drop-down menu on the left; these nodes serve as input to the algorithm that finds the best tree (thick, orange edges) that connects them; other edges are shown as thin lines. Edges are co-authorship relations. Node size is scaled by the node's degree. TOURVIZ comes with useful visualization features: pan and zoom tool on the first row, color tools for nodes and edges, node label tool for specifying font size and node attribute to show (node "name" is chosen here). Our user can search for authors using the *Search Panel* on the right.

Our user can interactively add or remove nodes (marked or unmarked), and refresh the visualization with an updated connection subgraph by invoking the algorithm. Nodes that were previously in the visualization will maintain their positions, to help our user preserve his mental model about the connections.

Engaging Our Audience. We will invite our audience to try out TOURVIZ with their own set of authors, and collect their feedback on TOURVIZ's usability and usefulness. Since we created TOURVIZ to be a general tool, we will be particularly interested in discussing with our audience how TOURVIZ may help them with research and data visualization in their domains.

3. TECHNICAL DETAILS

In this section, we first provide details about how the algorithm in our system works, and then give information about implementation and integration with the visualization component.

3.1 Method Details

Idea of encoding. Our method addresses the problem of succinctly describing a given set of marked nodes in a graph. We formulate this problem using an encoding scheme, which involves a sender and a receiver. In this scheme both the sender and the receiver know the graph structure $G = (V, E)$, while only the sender knows the set of marked nodes. The goal of the sender, then, is to transmit to the receiver the information of which nodes are marked, using *as few bits as possible*.

Intuitively, the sender could use fewer bits to encode 'close-by' marked nodes; by following a (short) path from one to another. For each such path, the sender encodes the starting node using $\log |V|$ bits. The following nodes on the path are described by encoding the particular neighbor of the previous node i to go to the next node $i+1$, which requires $\log d_i < \log |V|$ bits since the degree of each node in the graph is bounded by $|V| - 1$. A *simple* connection subgraph for a group of 'close-by' nodes is the union of such paths.

Notice that the simplest subgraph is in fact a *tree* since it requires fewer bits to directly refer to a node that is already encoded, compared to following a path to it. In addition, low degree nodes are preferred over hub-like node on the paths, since it takes fewer bits to encode where to go next for such nodes. Intuitively, hub-like nodes are avoided in connecting the marked nodes since practically they connect everything in the graph and thus do not constitute interesting candidates as connectors. All in all, we find one simple tree for every group of 'close-by' marked nodes. Two groups are separated if no path of cost less than $\log |V|$ bits exists inbetween.

Steiner tree problem. Simplicity of a connection tree is determined by the number of nodes the sender visits in this encoding, how many unmarked nodes s/he visits, and in particular how easily per visited node s/he can identify which edge to follow next. We show that the problem of finding the simplest tree with the minimum encoding is NP-hard, with a reduction from the Directed Steiner tree problem: Given a directed weighted graph $G = (V, E)$ and a subset of (marked) nodes M, find the minimum cost arborescence that spans all (marked) nodes in M. The cost of the tree is the sum of the weights (encoding costs) of its edges. The tree may include (unmarked) nodes that are not in M, which are referred to as Steiner nodes.

Fast heuristic methods. As the directed Steiner tree problem is NP-hard, we develop four fast heuristic approximation algorithms for large graphs. (1) The first and simplest method only considers the connected components induced on the marked nodes. (2)

The second method first builds the transitive closure graph of the marked nodes in which each pair of marked nodes are connected by an edge of weight equal to the minimum cost path inbetween them (no edge if the path length is greater than $\log |V|$). Next, it runs a minimum arborescence algorithm on this small graph of M nodes. Finally each edge in the returned tree(s) is expanded to include the original nodes on the path represented by that edge. (3) The third method builds the minimum depth-1 tree(s) in which the root marked node is connected to each other marked node with the shortest path length. (4) Finally, the fourth method builds on the depth-1 tree(s) to build larger depth trees, by finding intermediate nodes that decrease the total cost. Our method returns the best tree among found by various heuristics that has the minimum cost.

3.2 System Details

The visualization component of TOURVIZ system is written in Java 1.7. built on top of the open-source JUNG network visualization library [8] and with design based on the APOLO [2] system. TOURVIZ stores the graph in a SQLite[2] embedded database, for its cross-platform portability. The graph database's schema was designed independently from the TOURVIZ system, so that different graphs that follow the schema can be readily used. TOURVIZ takes advantage of built-in features from SQLite, such as its full-text search capability to quickly locate nodes whose attribute values match users' search text (Figure 2).

When the user interactively specifies the set of marked nodes for which the connection subgraphs are to be found, the IDs of these nodes are written to a temporary file. This file as well as the directory for the network data are input to our algorithm which is implemented in Matlab 7.10. The algorithm then outputs the best connection tree(s) as well as small subgraphs around the seed nodes (candidate graph). TOURVIZ shows the candidate graph and highlights (in bold) the edges that correspond to the best tree edges.

4. RELATED WORK

TOURVIZ builds on a large body of research aimed at understanding and supporting how people can gain insights through network visualization [7]. It provides several features to help people specify the nodes of interest in a network and visualize connection subgraphs in context of local neighborhoods around these nodes that are not too overwhelming [2].

Much work has been done on finding connection subgraphs, introduced by [4], and extended by [9] and [11]. Related work include expanding communities around a given set of seed nodes [10, 12] such that the modularity or conductance of the subgraph containing the seed nodes is optimized. [13] finds the most 'centered' node that has strong connections (good paths) to most or all of the seed nodes. [6] finds the subgraph up to a certain size which retains most of the proximity between seed nodes. Proximity and connection subgraphs are also exploited for visualization and summarization [3, 5].

Few tools have integrated connection subgraphs algorithms to interactively help people make sense of associations among nodes in a graph, and they often only support some of the sensemaking features offered by TOURVIZ, e.g., showing (multiple) connection subgraphs only among highly associated *groups* of nodes, rather than a single large subgraph including all nodes at once.

5. CONCLUSIONS

We presented TOURVIZ, an interactive system to help people understand connections between sets of nodes in large graphs. Our system integrates 1) novel algorithms that find the best subgraphs for succinctly connecting these nodes; and 2) visualization and interaction features that help people interactively and visually explore such subgraphs.

We demonstrated TOURVIZ's usage and benefits using a DBLP co-authorship graph, which consists of 329K authors (nodes) and 1094K co-authorship relations (edges). We invite our audience to try TOURVIZ, comment on its usability and usefulness, and discuss how TOURVIZ may help with their work as well as data analytics in other domains.

[2]www.sqlite.org

6. ACKNOWLEDGEMENT

Research was sponsored by the National Science Foundation under Grant No. IIS1017415 and the Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053. It is continuing through participation in the Anomaly Detection at Multiple Scales (ADAMS) program sponsored by the U.S. Defense Advanced Research Projects Agency (DARPA) under Agreements No. W911NF-11-C-0200 and W911NF-11-C-0088. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, of the National Science Foundation, of the U.S. Government, or any other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. This work is also partially supported by an IBM Faculty Award.

Jilles Vreeken is supported by a Post-Doctoral Fellowship of the Research Foundation – Flanders (FWO).

7. REFERENCES

[1] L. Akoglu, J. Vreeken, H. Tong, D. H. Chau, and C. Faloutsos. Islands and bridges: Making sense of marked nodes in large graphs. Technical Report CMU-CS-12-124, Carnegie Mellon University, 2012.

[2] D. Chau, A. Kittur, J. I. Hong, and C. Faloutsos. Apolo: Making Sense of Large Network Data by Combining Rich User Interaction and Machine Learning. In *CHI*, 2011.

[3] D. H. Chau, C. Faloutsos, H. Tong, J. I. Hong, B. Gallagher, and T. Eliassi-Rad. Graphite: A visual query system for large graphs. In *ICDM Workshops*, pages 963–966, 2008.

[4] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *KDD*, 2004.

[5] J. F. R. Jr., H. Tong, A. J. M. Traina, C. Faloutsos, and J. Leskovec. Gmine: A system for scalable, interactive graph visualization and mining. In *VLDB*, pages 1195–1198, 2006.

[6] Y. Koren, S. C. North, and C. Volinsky. Measuring and extracting proximity in networks. In *KDD*, 2006.

[7] B. Kules. From keyword search to exploration: How result visualization aids discovery on the web.

[8] J. O'Madadhain, D. Fisher, P. Smyth, S. White, and Y. Boey. Analysis and visualization of network data using JUNG. *Journal of Statistical Software*, 10:1–35, 2005.

[9] C. Ramakrishnan, W. H. Milnor, M. Perry, and A. P. Sheth. Discovering informative connection subgraphs in multi-relational graphs. 7(2):56–63, 2005.

[10] J. Riedy, D. A. Bader, K. Jiang, P. Pande, , and R. Sharma. Detecting communities from given seeds in social networks. *Technical Report GT-CSE-11-01*, 2011.

[11] P. Sevon and L. Eronen. Subgraph queries by context-free grammars. *J. Integrative Bioinformatics*, 5(2), 2008.

[12] D. A. Spielman and S.-H. Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *CoRR*, abs/0809.3232, 2008.

[13] H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In *KDD*, pages 404–413, 2006.