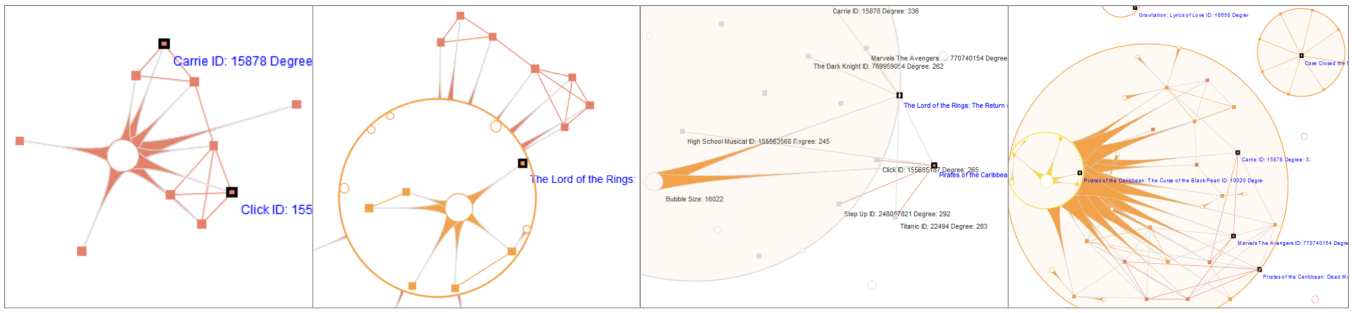
Interactive Multi-resolution Exploration of Million Node Graphs

Zhiyuan Lin, Georgia Tech
Cao Nan, IBM
Hanghang Tong, City College of New York
Wang Fei, IBM
U Kang, KAIST
Duen Horng Chau, Georgia Tech

zlin48@gatech.edu, nancao@us.ibm.com, tong@cs.ccny.cuny.edu, fwang@us.ibm.com, ukang@cs.kaist.ac.kr, polo@gatech.edu



(a) Graph overview (b) Preview a bubble (c) Fade unselected nodes (d) Nested previewed and broken-down bubbles

Figure 1: (a) Overview of the top 10 movies and top 10 components of the Rotten Tomatoes *related movies* graph. Node are movies; an edge connects two movies if they are similar. (b) *Previewing* a bubble shows a few movies and components within in; the user can expand the bubble further to show more nodes. (c) A *broken down* bubble showing all edges within a bubble and across bubble boundary; here, only edges connected to a selected movie is shown. (d) Bubble preview works across levels.

Abstract

We are working on a scalable, interactive graph visualization system to support multi-resolution exploration of million-node graphs in real time. By adapting a state-of-the-art graph algorithm, our prototype system generates a multi-resolution view of graphs with up to 68 million edges under a few seconds. We are experimenting with interaction techniques that help users interactively explore this overview and drill down into details. While many visualization systems for million-node graphs require dedicated servers to process the graphs, our prototype runs on a commodity laptop computer. We aim to handle graphs that are at least an order of magnitude (100M edges) larger than what current systems can support.

Keywords: graph visualization, multi-resolution, hubs and spokes

1 Introduction

Given a large graph with million or billion nodes and edges, how to visualize it? Showing every single node and edge will not work, due to limited screen size. Furthermore, this may not be the right approach, since the visual complexity will be overwhelming. Recent research [1, 2, 4] investigated how to create overviews of large graphs, visualize those views, and allow users to interactively drill down. However, they often only work for graphs with well-defined hierarchies (e.g., graphs that are *trees*), or need to first transform the graphs into hierarchies.

How do we visualize more general kinds of graphs without depending on or assuming any hierarchical structures? Can we visualize such graphs at scale, say with 100 million nodes and edges (order of magnitude larger than what current systems support)? Can we support all these by using one commodity computer, without requiring the graph to fit in the main memory [2], or a dedicated client-server architecture [1]? These are the foci of our investigation. To summarize, we aim to make the following contributions:

- We present a prototype system that supports multi-resolution exploration of large graphs with up to *68 million* edges.
- We adapts a fast, state-of-the-art graph algorithm, called *Slash & Burn* (Section 3) that can create a multi-resolution graph overview under a few seconds.
- We present techniques that help users interactively explore the multi-resolution view via *prioritized visualization* (Section 3), which helps the user determine what to visualize and explore.

2 Scenario

We introduce some of the visualization and interaction features of our prototype in a brief scenario. Alice is a data scientist at Rotten Tomatoes (RT), who wants to better understand what pairs of movies are considered similar by RT users. She has constructed a graph from her data of *related movies*, where each node in the graph is a movie, and an edge connects two movies if a user has suggested that those movies are similar (we crawled this graph from Rotten Tomatoes). There are roughly 200,000 nodes and 150,000 edges.

Our tool first presents an overview to Alice (Fig. 1a). It shows the top 10 movies (called *bridges*) with the highest degrees (they have the highest number of related movies). It also shows the top 10 components (called *bubbles*) which have the highest number of movies within them. This first overview gives Alice some idea about what the most popular movies are, how they are connected among themselves, and to the rest of the movies ("hidden" within the components/bubbles). Among the top 10 movies, Alice sees some popular ones, such as *Titanic* and the *Dark Knight*. She is intrigued that all of them are connected to the large component at the center (with many movies in it), even for horror movie *Carrie* and comedy movie *Click*. She decides to check out what are inside the bubble.

By pressing the "+" key on her keyboard, Alice is able to see (preview) the contents of the component, while keeping the nodes

and edges outside the component in place. At first, only a few movies and (sub)components are shown. Alice wants to see a few more, so she pressed "+" a few more times; more nodes and components show up in the big component (Fig. 1b) .

Alice sees that *Lord of the Rings* is in this component and she remembers a few horror scenes in this chivalric movie. This seems to be a good clue for tracing the hidden relationship between the movies *Carrie* and *Click*. She double clicks the previewed components to *break it down*, which reveals all edges between the nodes inside and outside of the component.

Now the screen is cluttered with too many edges. To reduce visual complexity, Alice selects *The Lord of the Rings*, and invoke an action (not shown in figure) to fade away the unselected nodes (shown in gray color). Now, she sees only *The Lord of the Rings* and the names of movies directly connected to it. It turns out *Pirates of the Caribbean* is one of those neighbors and she cannot help but smile about Jack Sparrow's witty humor. Then she holds down the *shift* key and select *Pirates of the Caribbean* too. This reveals that *Carrie* is similar to *The Lord of the Rings*; *The Lord of the Rings* and *Pirates of the Caribbean* share some characteristics; and finally, humor connects *Pirates of the Caribbean* and *Click* (Fig. 1c) .

Now Alice moves on to other movies. She interactively previews and breaks down a few more bubbles, at different parts of the graph and at different levels of abstraction (Fig. 1d).

3 Design Rationale and Algorithm Overview

3.1 Algorithm to generate graph overview

The general design principle that drives our design is **to prioritize what to visualize**, since the graph is large, but our screen real estate is limited. While this idea is implicit in recent works (e.g., [1, 5]), we use it as our first principle to guide our design.

In the visualization community, one common approach is to transform the graph into a tree (or tree-like structure), which is simpler to visualize and interact with, since nodes will then have well-defined parent-child relationships. In the visualization, we can then treat a parent node as the high-level representation of its children.

However, this approach unavoidably changes the semantics of the original graph (e.g., from graph with cycles into a tree). Can we generate an explorable overview for more general kinds of graphs without such transformation?

We surveyed data mining literature for solutions, and identified a state-of-the-art graph algorithm, called Slash & Burn [3], that offers such capability. Its design is based on the observation that most real-world graphs have power law degree distributions; such a graph have few hub nodes with very high degrees, while the majority of the nodes have low degrees. This means if we remove these highest degree hub nodes (e.g., top 25) and their edges, we will *shatter* the graph into smaller components, each being a *meta node* that may contain many nodes and edges—we call these nodes "bubbles", and visually we treat them as a higher-level representation of the nodes within it.

Due to its power law degree distribution, a graph can be quickly shattered by iteratively applying the above algorithm to components at each round. (All components eventually contain few number of nodes, say 50.) Here, we could only give a high-level description of the Slash & Burn algorithm due to limited space. We refer the readers to [3] for details.

A desirable effect of adapting this algorithm is that now we can rank both the nodes and components by their "importance". In our description above, we defined the "top" nodes as those having the highest degrees; but we can flexibly use "highest PageRank scores", or something else, instead. Similarly, components may be ranked by the number of nodes that they contain, or by other statistics.

3.2 Interaction technique to "preview" component

It is not sufficient to only create an overview visualization. We also need to provide interaction techniques for the user to explore and drill down. But, a component can contain tens of thousands of nodes and edges (as in the largest connected component of a million-node graph). We would not want to show all of them.

Fortunately, as we described above, nodes within a component are ranked (e.g., by node degrees). This inspires us to design an interaction technique that allow the user to choose how many nodes and edges they may want to visualize, based on how large they expanded a bubble (component), as if *previewing* or taking a glimpse into the contents of a component. The user can preview multiple bubbles that at different levels at the same time (Fig. 1d).

3.3 Scaling to large graphs

Current visualization systems often require dedicated servers to process their graphs (even for million-node scale), to run algorithms on them, and to compute their layouts [5, 1]. Is this always necessary? This question drives us to explore how much we can do with a single commodity machine. With our prototype, we are able to create a multi-resolution views for graphs from up to 68M edge graph, in under 5 seconds. These views can be interactively explored, in real time. A main technique that we use to scale to such large graphs, is to avoid storing the graph in memory; specifically, we keep the graph's "edge list" on the hard drive, which would otherwise take up a lot of memory. We are now testing our prototype on even larger graphs, with 100 million edges and more.

4 Conclusions and Work in Progress

We are designing and creating a scalable, interactive graph visualization system to support multi-resolution exploration of million-node graphs in real time. Thus far, our prototype can handle graphs up to 68 million edges on a laptop computer. We are experimenting with various visualization and interaction techniques to allow users to fluidly navigate and explore and drill down in the graphs.

We plan to conduct lab studies to evaluate those techniques, and work with security experts at Symantec to test our system in the real world. They will use our tool to help analyze large graphs of network traffic and email communications to spot suspicious activities. We believe our tool can help them develop overviews of their data and allow them to rapidly delve into details.

References

[1] J. Abello, F. Van Ham, and N. Krishnan. Ask-graphview: A large scale graph visualization system. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):669–676, 2006.

[2] D. Archambault, T. Munzner, and D. Auber. Grouse: Feature-based, steerable graph hierarchy exploration. In *Proceedings of the 9th Joint Eurographics/IEEE VGTC conference on Visualization*, pages 67–74. Eurographics Association, 2007.

[3] U. Kang and C. Faloutsos. Beyond'caveman communities': Hubs and spokes for graph compression and mining. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 300–309. IEEE, 2011.

[4] C. Tominski, J. Abello, and H. Schumann. Cgvan interactive graph visualization system. *Computers & Graphics*, 33(6):660–678, 2009.

[5] F. Van Ham and A. Perer. search, show context, expand on demand: Supporting large graph exploration with degree-of-interest. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):953–960, 2009.