Published in final edited form as:
IUI. 2015 ; 2015(Companion): 61–64. doi:10.1145/2732158.2732192.

Interactive Querying over Large Network Data: Scalability, Visualization, and Interaction Design

Robert Pienta,
Georgia Tech

Acar Tamersoy,
Georgia Tech

Hanghang Tong,
Arizona State University

Alex Endert, and
Georgia Tech

Duen Horng Chau
Georgia Tech

Robert Pienta: pientars@gatech.edu; Acar Tamersoy: tamersoy@gatech.edu; Hanghang Tong: htong6@asu.edu; Alex Endert: endert@gatech.edu; Duen Horng Chau: polo@gatech.edu

Abstract

Given the explosive growth of modern graph data, new methods are needed that allow for the querying of complex graph structures without the need of a complicated querying languages; in short, *interactive graph querying* is desirable. We describe our work towards achieving our overall research goal of designing and developing an interactive querying system for large network data. We focus on three critical aspects: scalable data mining algorithms, graph visualization, and interaction design. We have already completed an approximate subgraph matching system called MAGE in our previous work that fulfills the algorithmic foundation allowing us to query a graph with hundreds of millions of edges. Our preliminary work on visual graph querying, Graphite, was the first step in the process to making an interactive graph querying system. We are in the process of designing the graph visualization and robust interaction needed to make truly interactive graph querying a reality.

Author Keywords

Graph Querying and Mining; Visualization; Interaction Design

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

Introduction

With the rise of the World Wide Web came the need for efficient and user friendly methods to search web content. Search engines were made to provide users with a queryable interface to the vast data stored on the Internet. In the same fashion, we hope to construct an interactive, visual, graph querying system for exploring the patterns within large modern network data. Web graphs, social networks, and biological networks are just a few of the many areas producing gigantic graphs.

Much of today's important data can be represented as a network or graph—we will use these terms interchangeably. As more fields begin storing their data as graphs, the size and availability of graphs are growing rapidly. Graphs are a natural way to represent the relationships among connected entities. For example, much of the richness and structure of social ties among people can be captured as a graph, where people are the nodes and their ties are the edges.

In order to ask questions about how entities in a graph are related, we need an approach to querying for both the entities themselves and their relations in a meaningful way. This type of search is often called *graph querying* (visualized in Figure 1 on the following page). There are many domains in which structured queries are directly useful; as in bioinformatics where instances of a particular protein structure are desired; in intelligence where known criminal interactions can be mined; and in computer networks where particular communication patterns may signal unwarranted access. Graph querying captures the rich relationships among multiple entities in an explicit manner, making graph queries both flexible in what they can represent and specific in the detailed structure of relationships.

How do we support such kind of graph query? What kind of tools and techniques do we need? What research problems do we need to address?

Naturally, we need scalable graph querying algorithms; many have been designed to date [4, 3, 2, 7, 9]. However, most works focused on the data mining aspects and algorithmic issues with graph querying, like speed, scalability, and robustness. Usability and interaction design were often a secondary focus, and these systems were rarely evaluated with humans.

This points to the need for intuitive visual user interfaces that helps the user to both specify their queries and visualize the matches, as such interfaces are easier to use and can help the user understand the results [5, 8, 12].

To the best of our knowledge, there has been little work on studying how to design effective interaction techniques for specifying graph queries. For example, how should one design a direct manipulation user interface to facilitate intuitive query construction? Does it suffice to use a basic design, like our preliminary Graphite interface (see Figure 2) [1]? When specifying desired attributes (of nodes or edges) to help narrow the search, how do we determine useful attributes to suggest? There is even less work on investigating how to best visualize the resulting matches for a query; when exact matches do not exist, the visualization will also need to communicate to the user how the approximate results deviate from the original query and to provide intuition behind the differences.

A further need is to evaluate the usability of the visualization and interaction designed, as humans are the eventual users of the system. Do people understand what the visualization mean? Do they find it natural to visually specify their queries as graphs? Thus far, little research has been done to answer these kinds of important questions.

In this paper, we describe our work towards achieving our overall research goal of designing and developing an interactive querying system for large network data. We focus on three critical aspects and contributions:

1. The **algorithmic support** to find the search results; we have successfully developed a scalable querying algorithm called MAGE [9] that supports graphs with millions of nodes and edges.
2. The **graph visualization** of the query results that yield greater understanding and improve query refinements; we have created a basic visualization called Graphite [1], on which we are currently improving.
3. The **interaction design** to strengthen the query generation and refinement processes; we plan to conduct a two-part user study to better understand what kind of queries are common and easy to specify, and how usable and effective our whole system is.

Scenario

The following illustrative scenario demonstrates how interactive graph querying may be used. Consider a data analyst at IMDb (Internet Movie Database) who is interested in finding patterns involving actors, directors and films.

Furthermore imagine a graph demonstrating the structure of cinema, where the people involved with each film (nodes) are connected to their cinematic works (nodes) via edges. According to IMDb's website (http://www.imdb.com), this graph will have over 2.7 million titles, 300 thousand directors, and 2 million actors. We have visualized a small part of this graph in Figure 1.

Looking for a director who directed both a comedy and a cult classic starring a common performer, our analyst needs to have known either a graph querying language or have significant experience using databases. With our prototype we hope to completely avoid these requirements.

First, our analyst draws a node for a director, then two nodes representing two films (see the query in Figure 1). The analyst then draws an edge between the director and each of the two films, denoting that both these films must have this director. Finally, the analyst creates a node for the actor and connects it to both the movies. For any of the nodes or edges, the analyst may specify further desired attributes, like a movie's genre.

At this point, our analyst may sit back while the graph is searched for their pattern. Results are displayed next to the initial query. Two possible visualizations are provided on the right side of Figure 1. Individual matched subgraphs provide the user with a spatial mapping similar to the layout used in the query; however, they do not provide and context about where the results came from in the larger graph. A context-view provides the results embedded in the global graph (or likely a summary of it) suggesting their positions and relation to the overall network. The matches are ordered algorithmically by how far from the query they are (both in attributes and structure) so that the best matches appear first.

At this point the result nodes can be clicked for further details about them, or the original query can be modified if the results were not desirable.

Our Completed Work and Related Work

Algorithmic

To extract interesting patterns from a million node graph (or larger) requires a complicated search that utilizes the graph structure as well as its attributes. The problem of determining if a particular graph is a subgraph of any other graph is computationally challenging problem. Formally known as the subgraph isomorphism problem, this feat is NP-Complete making exact solutions computationally infeasible for even modest sized graphs.

We strive to offer both exact results and near matches based on the query attributes and connectivity structure. Matching and extracting exact patterns based on desired attributes has proven to be a successfully scalable approach [2]. Yet, we must also find inexact matches in the case where the query does not exist in the data. Luckily the subgraphs can be quickly approximated by leveraging attributes in different ways [7, 9, 10]. Systems like OntoSeek, G-Ray, and our previous work MAGE all exhibit the desired behavior for fast approximate graph matching [3, 11, 9].

For our prototype, we have started building our interface atop the MAGE graph querying system. MAGE works by using both attribute and structural data to find a seed node (a key node with structural and attribute properties similar to the query); it expands from the seed to find the other nodes that match parts of the query (if none exist it will approximate), it then approximates the edge structure given the nodes it's already detected and completes the result. For more details on how MAGE functions, we direct the reader to [9]. MAGE offers the underlying mechanism to algorithmically match inputs from the user to real subgraphs in an input graph. MAGE is only a graph querying engine and offers no visual mechanism to input graph queries.

Visualization and Interaction Design

Previous research showed that querying tools greatly benefit from having intuitive visual user interfaces, through which the user can more naturally specify queries and understand the resulting matches [5, 8, 12].

To this end, we have been building on our preliminary visual querying tool called Graphite, a system for visual approximate subgraph matching [1]. Graphite allows a user to draw categorically attributed nodes and the connections between them and offers the results on the right of the query one at a time, as in Figure 2.

Graphite has multiple limitations. For example, it does not support edge attributes, multiple attributes, wildcards, and cannot show the user where the resulting subgraph matches are in the context of the whole graph. For example, are the matches evenly distributed topologically? Or are they all concentrated around a particular location.

Work in Progress

We are addressing the limitations of Graphite's visualization and interaction design by improving the visual and interactive mechanisms for query refinement, result context, and result comparisons through a two-part study.

The first part is a formative investigation that surveys the needs, requirements, and common query patterns of various experts who use graphs or graph-related tools. We will use these data to improve our use cases and overall system design.

Allowing the user to draw their query raises an important question: when a user is given complete freedom to construct their query, how can we demonstrate when their query doesn't exist or has faulty assumptions? For instance, in the IMDb example in Figure 1, a director is never directly connected with themself, actors or other directors. Thus, a query with a director attached directly to an actor will never yield exactly what the user desired as in Figure 3(b). We are investigating visual cues that help users refine their queries in the face of uncertainty.

A data-rich graph may have thousands of types of entities each with hundreds of attributes making the query and results a challenge to visualize. For the query, we have opted to allow the user to choose which attributes to use in their query, for each node and edge (see Figure 3(a)). For the results, we only show the most identifying attribute initially, but allow further attribute details upon mouseover or click. This minimalist view improves visual scalability in complex queries, but still delivers details on demand.

The second part of our study is a pilot human evaluation investigating our prototype graph-search engine. The target population for this human evaluation will be more diverse (in familiarity with graphs and graph querying) than the initial formative study in order to provide realistic usage statistics. Analyzing the quality of our visualization and its interaction design is a challenging problem; we intend to measure several of the low-level tasks for user-graph interactions proposed in [6]. We want to investigate two primary aspects of interactive graph querying: (1) how do people generate queries from ideas or questions and (2) how similar do people find exact and approximate results when offered with and without broader graph context. We will provide subjects with a set of predetermined questions and ask them to formulate analogous queries. As they work we will record the process used and time taken to construct each query. Once the queries have been developed, our prototype will display the search results, with both exact and approximate matches.

The display of the approximate results is essential to a successful query system. Suppose someone forms the query in Figure 3(b) and gets the displayed result, they may be confused as the results look less and less like their initial query. Testing how well people rate a series of approximate matches gives crucial data about; the underlying algorithm, the successful mechanisms of visualizations, and how people think about their results.

Conclusion

We described our work in tackling three critical aspects in achieving our overall research goal of designing and developing an interactive querying system for large network data: (1) the algorithmic support to find the search results; (2) the graph visualizations to allow for intuitive yet detailed query construction; and (3) the interaction design to strengthen the query generation and refinement processes.

We have already fulfilled the first criterion with our previous graph querying system, MAGE. For the latter two, We have prior knowledge and experience gained from other similar works and from our own fundamental work, Graphite. We are well underway developing an interactive query-system that allows a user to search large-scale graph data without the need for programming.

Our proposed study covers several key aspects of an interactive querying system; the use-cases and common design patterns, the process people take to convert an idea into a visual graph query, and how well people track the inequalities between the approximate results and their queries (with and without graph context). This study will provide the community with critical data about how well users interact with and understand graph querying tools.

References

1. Chau, DH.; Faloutsos, C.; Tong, H.; Hong, JI.; Gallagher, B.; Eliassi-Rad, T. ICDM. IEEE; 2008. Graphite: A visual query system for large graphs; p. 963-966.
2. Coffman T, Greenblatt S, Marcus S. Graph-based technologies for intelligence analysis. Commun ACM. 2004; 47(3):45–47.
3. Guarino N, Content-based O, Vetere G, Masolo C. Ontoseek: Content-based access to the web. IEEE Intelligent Systems and Their Applications. 1999; 14(3):70–80.
4. Jin, C.; Bhowmick, SS.; Xiao, X.; Cheng, J.; Choi, B. ACM. New York, NY,USA: 2010. Gblender: Towards blending visual query formulation and query processing in graph databases. SIGMOD '10; p. 111-122.
5. Lee B, Parr CS, Plaisant C, Bederson BB, Veksler VD, Gray WD, Kotfila C. Treeplus: Interactive exploration of networks with enhanced tree layouts. IEEE Transactions on Visualization and Computer Graphics. Nov; 2006 12(6):1414–1426. [PubMed: 17073365]
6. Lee, B.; Plaisant, C.; Parr, CS.; Fekete, JD.; Henry, N. In BELIV 2006, BELIV '06. ACM; New York, NY, USA: 2006. Task taxonomy for graph visualization; p. 1-5.
7. Mongioví M, Natale RD, Giugno R, Pulvirenti A, Ferro A, Sharan R. Sigma: A set-cover-based approach for inexact graph matching. J Bioinformatics and Computational Biology. 2010; 8(2)
8. Perer A, Shneiderman B. Balancing systematic and flexible exploration of social networks. IEEE Transactions on Visualization and Computer Graphics. Sept.2006 12(5):693–700. [PubMed: 17080789]
9. Pienta, R.; Tamersoy, A.; Tong, H.; Chau, DH. Big Data 2014. IEEE; 2014. Mage: Matching approximate patterns in richly-attributed graphs.
10. Tian Y, Patel J. Tale: A tool for approximate large graph matching. ICDE. 2008; 2008:963–972.
11. Tong H, Faloutsos C, Gallagher B, Eliassi-Rad T. Fast best-effort pattern matching in large attributed graphs. KDD. 2007:737–746.

12. Wattenberg, M. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM; New York, NY, USA: 2006. Visual exploration of multivariate graphs; p. 811-819.

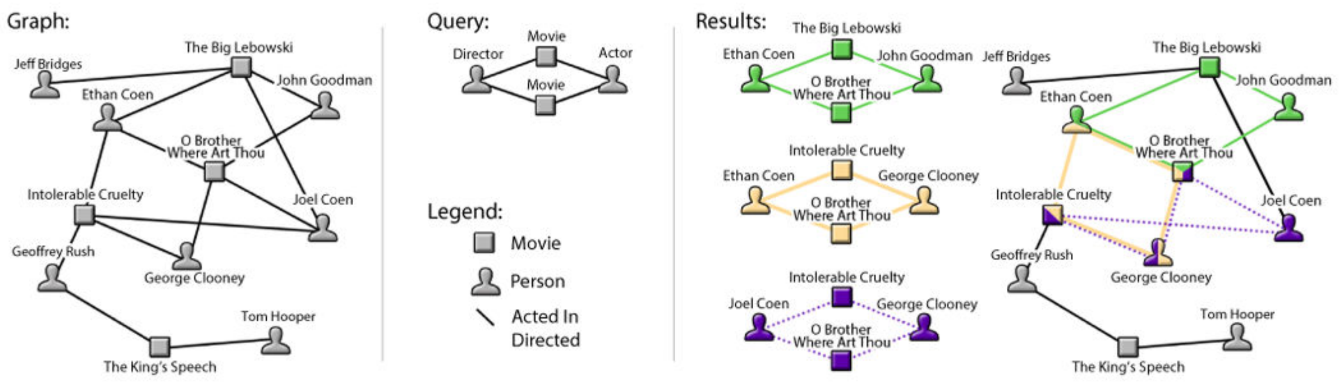

Figure 1.

A portion of a toy graph of movies, actors, and directors (left) with a query (second left) to locate in the graph. The query finds instances of any director who directed two movies starring the same actor. The results are algorithmically detected in the input graph and displayed to the user as separate subgraphs (second right) and as a context view (rightmost).

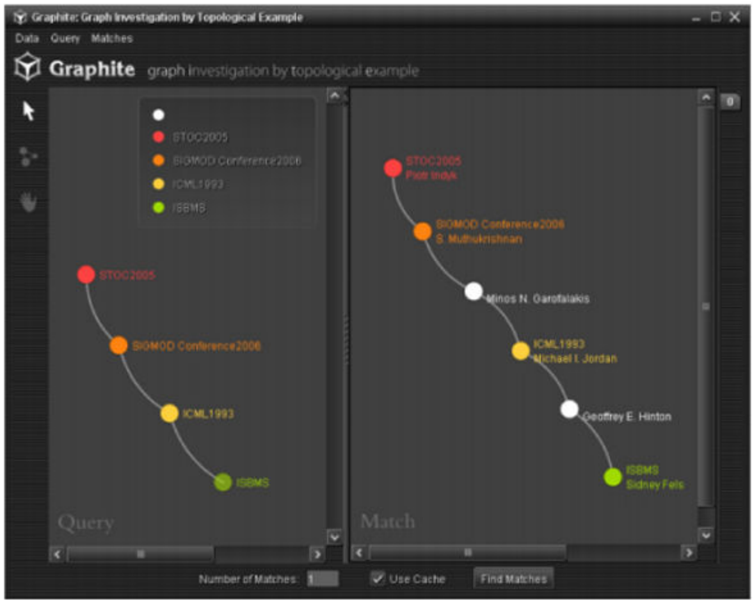

Figure 2.
An image of a query and an approximate result being displayed from the Graphite system, our previous work which strongly impacted the design of our system.

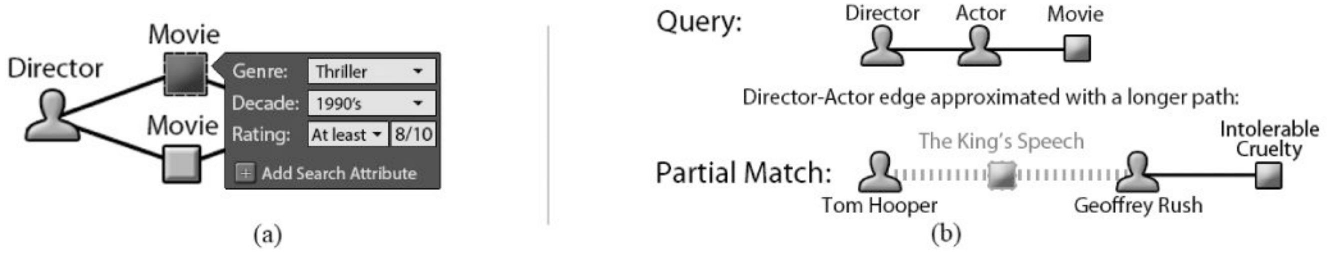

Figure 3.

(a) A prototype for query attribute selection. (b) A query with an approximate match, in dotted gray, that contains an extra movie node, because the structure of our proposed toy graph does not allow direct connections between directors and actors.