Mining Billion-Scale Graphs: Patterns and Algorithms

Christos Faloutsos
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213 USA
christos@cs.cmu.edu

U Kang
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213 USA
ukang@cs.cmu.edu

ABSTRACT

Graphs are everywhere: social networks, the World Wide Web, biological networks, and many more. The sizes of graphs are growing at unprecedented rate, spanning millions and billions of nodes and edges. What are the patterns in large graphs, spanning Giga, Tera, and heading toward Peta bytes? What are the best tools, and how can they help us solve graph mining problems? How do we scale up algorithms for handling graphs with billions of nodes and edges? These are exactly the goals of this tutorial. We start with the patterns in real-world static, weighted, and dynamic graphs. Then we describe important tools for large graph mining, including singular value decomposition, and HADOOP. Finally, we conclude with the design and the implementation of scalable graph mining algorithms on HADOOP.

This tutorial is complementary to the related tutorial "Managing and Mining Large Graphs: Systems and Implementations".

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Design, Experimentation, Algorithms

Keywords

Graph Mining, MapReduce, Hadoop

1. INTRODUCTION

Graphs are everywhere: social networks, computer networks, mobile call networks, the World Wide Web [2], protein interaction networks, and many more. The lower cost of disk storage, the success of social networking websites and web 2.0 applications, and the high availability of data sources lead to graphs being generated at unprecedented size. They are now measured in Terabytes or even Petabytes, with millions and billions of nodes and edges.

Finding patterns on large graphs have a lot of applications including cyber security [16], social network analysis (Facebook, Twitter) [9], and fraud detection [4], among others. This tutorial exactly addresses the problem of finding patterns on large graphs. Specifically, we address the following questions: What are the patterns in large graphs? What are the useful tools for large graph mining? How to scale up graph mining algorithms for large graphs? We provide patterns, tools, and scalable algorithms for mining large graphs.

We start with the patterns in real-world static, weighted, and dynamic graphs. Then we describe important tools for large graph mining, including singular value decomposition and HADOOP. Finally, we conclude with the design and the implementation of scalable graph mining algorithms on HADOOP.

2. TUTORIAL OUTLINE

Our tutorial consists of three parts: patterns, tool, and scalable algorithms.

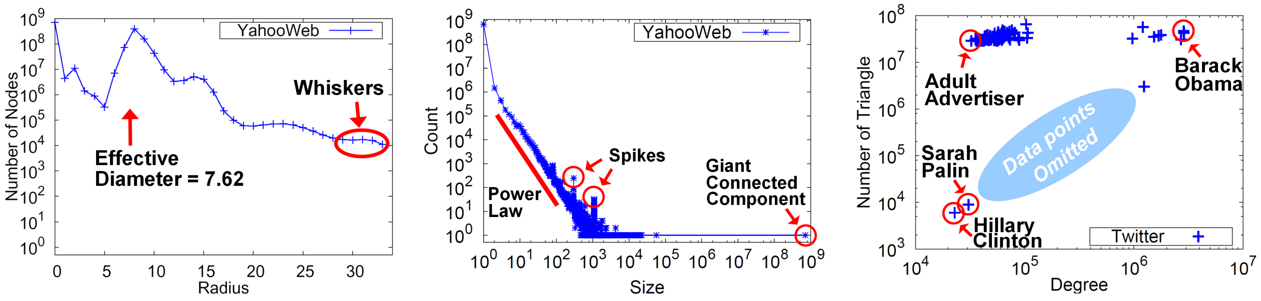
- Patterns in large static, weighted, and dynamic graphs.
- Tools for large graph mining, including singular value decomposition and HADOOP.
- Scalable algorithms for large graph mining on HADOOP, including structure analysis, eigensolver, storage/indexing, and graph layout/compression.

2.1 Patterns in Large Graphs

Real graphs obey several patterns, clearly deviating from random graphs (the so-called 'Erdős-Rényi' graphs). What are the distinguishing characteristics of real graphs? What rules and patterns hold for them? When can we say that two different graphs are *similar* to each other? To answer these questions, we need to have some basic set of graph attributes; these would be our vocabulary in which we can discuss different graph types. We describe three types of patterns: static, weighted, and dynamic graphs.

Static Graphs. Real graphs have relatively small diameter, with the majority of nodes have small radii while few nodes (marked 'whiskers') have large radii, as shown in Figure 1 (a). It means that the distances of nodes in real graphs are much smaller than expected. Another pattern is the power laws in real graphs. The degree distribution follow a power-law in the form of $f(d) \propto d^{\gamma}$, where d is the degree and $f(d)$ is the number of nodes with such degree, with the exponent $\gamma < 0$ [3, 6]. The number of

SIGMOD'12, May 20–24, 2012, Scottsdale, Arizona, USA.
Copyright 2012 ACM 978-1-4503-1247-9/12/05 ...$10.00.



(a) Radius plot of YahooWeb

(b) Connected components of YahooWeb

(c) Triangle counts in Twitter

Figure 1: Patterns and anomalies in large graphs. (a) Radius plot of YahooWeb graph, a Web snapshot at year 2002 containing 1.6 billion pages and 6.6 billion edges. Notice the effective diameter is surprisingly small, and the few whiskers which have large radii. (b) Connected components size distribution of the YahooWeb. Notice the two anomalous spikes which deviate significantly from the constant-slope power law line. (c) The degree vs. participating triangles of some 'celebrities' in Twitter who follows whom snapshot at year 2009. Also shown are accounts of adult site advertisers which have smaller degree, but belong to an abnormally large number of triangles. The reason of the large number of triangles is that adult accounts are often from the same provider, and they follow each other to form a clique, to possibly boost their rankings or popularity.

participating triangles [20], eigenvalues of the graph adjacency matrix [19], and the sizes of non giant connected components [11] follow the power law, too. An application of these features is the anomaly detection on graphs. For example, the connected components size distribution plot in Figure 1 (b) shows two anomalous spikes which deviate from the constant-slope tails. Another example is the number of participating triangles vs. degree plot in Figure 1 (c) which clearly shows anomalous adult advertisers which contain much larger number of triangles compared to some U.S. politicians.

Weighted Graphs. The total weight of the edges W_n attached to each node and the number of such edges, that is, the degree d_n, follow a power-law in the form of $W_n \propto d_n^\theta$, with $\theta > 1$ [17].

Dynamic Graphs. Patterns in dynamic graphs are observed in time evolving graphs. The (effective) diameter of graphs shrink over time [15]. Also, the number of nodes $N(t)$ and the number of edges $E(t)$ at time t follow a power-law in the form of $E(t) \propto N(t)^\alpha$, with $\alpha > 1$, over time [15]. Finally, the sizes of connected components show interesting pattern: while the giant connected component keeps growing, the secondary and tertiary connected components tend to remain constant in size with small oscillations [17].

2.2 Tools for Large Graph Mining

What tools can we use for large graph mining? We describe the two of the most important tools, namely singular value decomposition and HADOOP, a large scale data processing platform.

Singular Value Decomposition. Singular Value Decomposition (SVD) is an important tool for finding concepts and reducing dimensions in graphs as well as matrices [7, 1]. We describe the motivation, definition, and various applications including text clustering, compression, rule discovery, steady-state probabilities computation, and solving linear systems.

Hadoop. Large scale graph mining poses challenges in dealing with massive amount of data: they exceed memory and even disks of a single machine. A promising alternative for large graph mining is MAPREDUCE [5], a parallel programming framework for processing web-scale data, and its open-source version HADOOP. MAPREDUCE has two advantages. First, the data distribution, replication, fault-tolerance, and load balancing are handled automatically. Second, it uses the familiar concept of functional programming: the programmer needs to define only two functions, a *map* and a *reduce*. We describe the preliminaries on HADOOP for large graph mining.

2.3 Scalable Algorithms for Large Graph Mining

How to scale up the algorithms for mining very large graphs which do not fit in memory, or disks of a single machine? How to use parallelism? We describe how to design and implement such algorithms on HADOOP. The algorithms are general and cover diverse cases including the structural analysis, eigensolver, storage/indexing, and compression.

Structure Analysis. How can we find connected components, diameter, PageRank, and node proximities of very large graphs quickly? Furthermore, how can we design a general primitive which can be applied to many different algorithms? We describe `GIM-V` (Generalized `I`terative `M`atrix-`V`ector multiplication) [11, 14], an important primitive which unifies many seemingly different algorithms including connected components, diameter [12, 13], PageRank, and node proximities. We also describe how to develop fast algorithms for `GIM-V` on MAPREDUCE framework.

Eigensolver. Given a billion-scale graph, how can we find near-cliques, the count of triangles, and related graph properties? All of them can be found quickly if we have the first several eigenvalues and eigenvectors of the adjacency matrix of the graph [20, 18]. Despite their importance, however, existing eigensolvers do not scale well. We introduce HEIGEN [9], an eigensolver for billion-scale, sparse matrices. We describe the design decisions and fast algorithms that enable the scalable billion-scale eigensolver.

Storage and Indexing. How to store and index graph edge files so that graph mining queries can be answered quickly? Graph storage and indexing are important especially for targeted graph mining queries whose answers require the access to only parts of the graph. Examples of targeted queries include k-step in/out-neighbors, and egonet queries. We describe how to store and index the nonzero elements in the adjacency matrix of the graphs to quickly answer graph mining queries [10].

Graph Layout and Compression. Given a real world graph, how should we lay-out its edges? How can we compress it? These questions are closely related, and the typical approach so far is to find clique-like communities, like the 'cavemen graph', and compress them. We show that the block-diagonal mental image of the 'cavemen graph' is the wrong paradigm, in full agreement with earlier results that real world graphs have no good cuts. We describe the recent development for graph compression, called SlashBurn method [8] which has several advantages: (a) it avoids the 'no good cuts' problem, (b) it gives better compression, and (c) it leads to faster execution times for matrix-vector operations, which are the back-bone of most graph processing tools [11, 14].

3. REFERENCES

[1] M. W. Berry. Large scale singular value computations. *International Journal of Supercomputer Applications*, 1992.

[2] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks 33*, 2000.

[3] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A recursive model for graph mining. *SIAM Int. Conf. on Data Mining*, Apr. 2004.

[4] D. H. Chau, S. Pandit, and C. Faloutsos. Detecting fraudulent personalities in networks of online auctioneers. *PKDD*, 2006.

[5] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150, 2004.

[6] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM*, pages 251–262, Aug-Sept. 1999.

[7] M. Kamel. Computing the singular value decomposition in image processing. In *Proceedings of Conference on Information Systems*, Princeton, 1984.

[8] U. Kang and C. Faloutsos. Beyond 'caveman communities': Hubs and spokes for graph compression and mining. In *ICDM*, 2011.

[9] U. Kang, B. Meeder, and C. Faloutsos. Spectral analysis for billion-scale graphs: Discoveries and implementation. In *PAKDD (2)*, pages 13–25, 2011.

[10] U. Kang, H. Tong, J. Sun, C.-Y. Lin, and C. Faloutsos. Gbase: a scalable and general graph management system. In *KDD*, pages 1091–1099, 2011.

[11] U. Kang, C. Tsourakakis, and C. Faloutsos. Pegasus: A peta-scale graph mining system - implementation and observations. *ICDM*, 2009.

[12] U. Kang, C. E. Tsourakakis, A. P. Appel, C. Faloutsos, and J. Leskovec. Radius plots for mining tera-byte scale graphs: Algorithms, patterns, and observations. In *SDM*, pages 548–558, 2010.

[13] U. Kang, C. E. Tsourakakis, A. P. Appel, C. Faloutsos, and J. Leskovec. Hadi: Mining radii of large graphs. *ACM Trans. Knowl. Discov. Data*, 5:8:1–8:24, February 2011.

[14] U. Kang, C. E. Tsourakakis, and C. Faloutsos. Pegasus: mining peta-scale graphs. *Knowl. Inf. Syst.*, 27(2):303–325, 2011.

[15] J. Leskovec, J. M. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*, pages 177–187, 2005.

[16] K. Maruhashi, F. Guo, and C. Faloutsos. Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In *ASONAM*, pages 203–210, 2011.

[17] M. Mcglohon, L. Akoglu, and C. Faloutsos. Weighted graphs and disconnected components: patterns and a generator. *KDD*, pages 524–532, 2008.

[18] B. A. Prakash, M. Seshadri, A. Sridharan, S. Machiraju, and C. Faloutsos. Eigenspokes: Surprising patterns and community structure in large graphs. *PAKDD*, 2010.

[19] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos. Power-laws and the AS-level internet topology. *IEEE/ACM Transactions on Networking*, 11(4):514–524, 2003.

[20] C. Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *ICDM*, 2008.