OPAvion: Mining and Visualization in Large Graphs

Leman Akoglu*, Duen Horng Chau*, U Kang*, Danai Koutra*, Christos Faloutsos
School of Computer Science
Carnegie Mellon University
{lakoglu, dchau, ukang, danai, christos}@cs.cmu.edu

ABSTRACT

Given a large graph with millions or billions of nodes and edges, like a who-follows-whom Twitter graph, how do we scalably compute its statistics, summarize its patterns, spot anomalies, visualize and make sense of it? We present OPAvion, a graph mining system that provides a scalable, interactive workflow to accomplish these analysis tasks. OPAvion consists of three modules: (1) The *Summarization* module (Pegasus) operates off-line on massive, disk-resident graphs and computes graph statistics, like PageRank scores, connected components, degree distribution, triangles, etc.; (2) The *Anomaly Detection* module (OddBall) uses graph statistics to mine patterns and spot anomalies, such as nodes with many contacts but few interactions with them (possibly telemarketers); (3) The *Interactive Visualization* module (Apolo) lets users incrementally explore the graph, starting with their chosen nodes or the flagged anomalous nodes; then users can expand to the nodes' vicinities, label them into categories, and thus interactively navigate the interesting parts of the graph.

In our demonstration, we invite our audience to interact with OPAvion and try out its core capabilities on the Stack Overflow Q&A graph that describes over 6 million questions and answers among 650K users.

Categories and Subject Descriptors

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

General Terms

Design, Experimentation, Algorithms

Keywords

cloud computing, anomaly detection, visualization

*Authors in alphabetical order

SIGMOD 12, May 20–24, 2012, Scottsdale, Arizona, USA.

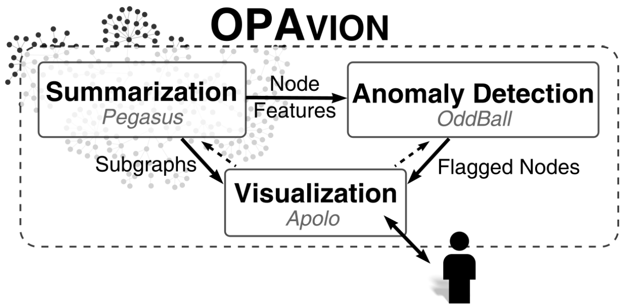


Figure 1: System overview. OPAvion consists of three modules: the *Summarization* module (Pegasus) provides scalable storage and algorithms to compute graph statistics; the *Anomaly Detection* module (OddBall) flags anomalous nodes whose egonet features deviate from expected distributions; the *Visualization* module (Apolo) allows the user to visualize connections among anomalies, expand to their vicinities, label them into categories, and interactively explore the graph.

1. INTRODUCTION

We have entered the era of big data. Massive graphs measured in terabytes or even petabytes, having billions of nodes and edges, are now common in numerous domains, such as the link graph of the Web, the friendship graph of Facebook, the customer-product graph of Netflix, eBay, etc. How to gain insights into these data is the fundamental challenge. How do we find patterns and anomalies in graphs at such massive scale, and how do we visualize them?

We present OPAvion, a system that provides a scalable, interactive workflow to help people accomplish these analysis tasks (see Figure 1). Its core capabilities and their corresponding modules are:

- The **Summarization** module (Pegasus) provides massively scalable graph mining algorithms to compute essential statistics for the whole graph, such as PageRank, connected components, degree distribution, etc. It also generates plots that summarize these statistics, to reveal apparent deviations from the expected graph's properties (see Figure 2). Pegasus has been tested to work efficiently on a huge graph with 1 billion nodes and 6 billion edges [4].

- The **Anomaly Detection** module (OddBall) automatically detects anomalous nodes in the graph—for each node, ODDBALL extracts features from its egonet (induced subgraph formed by the node and its neighbors) and flags nodes whose feature distributions deviate from those of other nodes'. Example features include: number of nodes, total edge weight, principal eigenvalue, etc. These flagged nodes are great points for analysis, as they are potentially new and significant information (see Figure 3a for example anomalies).

- The **Visualization** module (APOLO) provides an interactive visualization canvas for analysts to further their investigation. For example, flagged anomalous nodes can be transferred to the visualization to show their connections (see Figure 3b). Different from most graph visualization packages that visualize the full graph, APOLO uses a built-in machine learning method called BELIEF PROPAGATION to guide the user to expand to other relevant areas of the graph; the user specifies nodes of interest as *exemplars* and APOLO suggests nodes that are in their close proximity, and their induced subgraphs, for further inspection.

We will demonstrate how OPAVION tightly integrates the above modules and combines their strengths to support the analysis workflow: (1) Run in offline mode, the *Summarization* module (PEGASUS) stores the full graph, computes graph statistics and each node's egonet features; (2) The *Anomaly Detection* module (ODDBALL) retrieves egonet features from PEGASUS to perform real-time detection of anomalous nodes, and generates interactive plots that rank and highlight them; (3) The user can choose among those flagged nodes and instruct the *Visualization* module (APOLO) to visualize them, to reveal non-trivial connections among them (see Figure 3 for an example); the user can also spatially arrange nodes, label them into categories, and expand to the nodes' neighborhoods to incrementally explore interesting parts of the graph.

Our audience will be invited to try out the core capabilities of OPAVION on the STACK OVERFLOW Q&A graph (`http://stackoverflow.com`), which describes over 6 million questions and answers among 650K users. In the graph, nodes are STACK OVERFLOW users, and a directed edge points from the user who asks a question, to the user who answers it.

2. SYSTEM OVERVIEW

OPAVION consists of three modules: *Summarization*, *Anomaly Detection*, and *Visualization*. The block-diagram in Figure 1 shows how they work together in OPAVION. The following subsections briefly describe how each module works.

2.1 Summarization

How do we handle graphs with billions of nodes and edges, which do not fit in memory? How to use parallelism for such Tera- or Peta-scale graphs? PEGASUS provides massively scalable graph mining algorithms to compute a carefully selected set of statistics for the whole graph, such as diameter, PageRank, connected components, degree distribution, triangles, etc. PEGASUS is based on the observation that many graph mining operations are essentially repeated matrix-vector multiplications. Based on the observation, PEGASUS implements a very important primitive called GIM-V (Generalized Iterated Matrix-Vector multiplication) which is a generalization of the plain matrix-vector multiplication. Moreover, PEGASUS provides fast algorithms for GIM-V in MAPREDUCE, a distributed computing platform for large data, achieving (a) good scale-up on the number of available machines and edges, and (b) more than 9 times faster performance over the non-optimized GIM-V.

Here is the list of the algorithms supported by PEGASUS.

Structure Analysis. PEGASUS extracts various features in graphs, including degree, PageRank scores, personalized PageRank scores, radius [5], diameter, and connected components [4]. The extracted features can be analyzed for finding patterns and anomalies. For example, degree or connected component distributions of real world graphs often follow a power law as shown in Figure 2, and the deviations from the power law line indicate an anomaly (e.g. spammers in social networks, and a set of web pages replicated from a template).

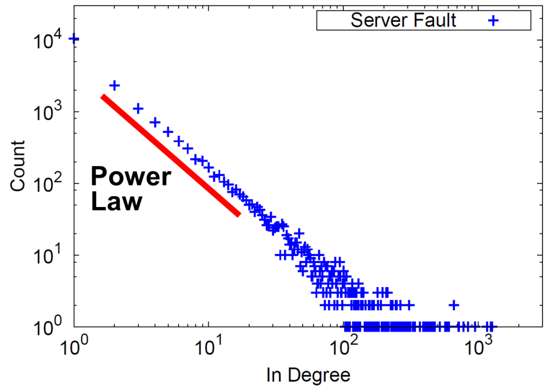


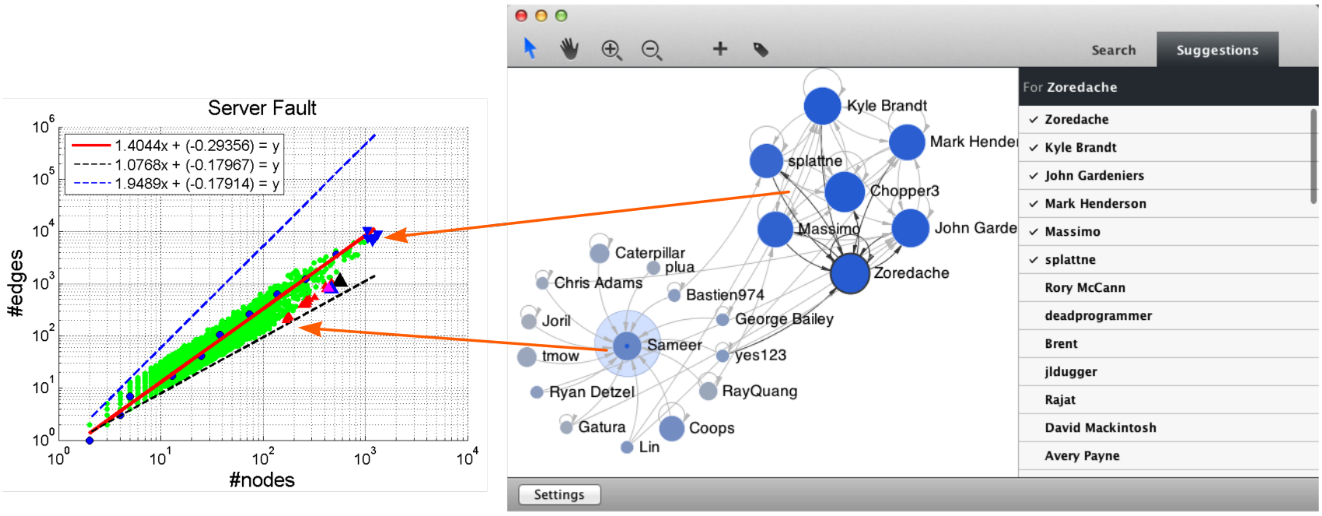
Figure 2: Degree distribution in 'Stack Overflow' Q&A graph. Real world graphs often have power-law degree distribution, as marked with the red line, and the deviations from the power law line indicate anomalies (e.g. replicated 'robots' in social networks).

Eigensolver. Given a graph, how can we compute near-cliques, the count of triangles, and related graph properties? All of them can be found quickly if we have the first few eigenvalues and eigenvectors of the adjacency matrix of the graph [6, 3]. Despite their importance, existing eigensolvers do not scale well. PEGASUS provides a scalable eigensolver [3] which can handle billion-scale, sparse matrices. An application of the eigensolver is the triangle counting which can be used to find interesting patterns. For example, the analysis of the number of participating triangles vs. the degree ratio in real world social networks reveals a surprising pattern: few nodes have the extremely high ratio, indicating spamming or tightly connected suspicious accounts.

2.2 Anomaly Detection

The anomaly detection module ODDBALL [1] consists of three main components: (1) feature extraction, (2) pattern mining, and (3) anomaly detection. In the following we explain each component in detail.

I. Feature extraction. The first step is to extract features from a given graph that would characterize the nodes. We choose to study the local neighborhood, that is the 'egonet', features of each node. More formally, an egonet is



(a) (b)

Figure 3: (a) Illustration of Egonet Density Power Law on the 'Stack Overflow' Q&A graph. Edge count E_i versus node count N_i (log-log scales); red line is the least squares fit on the median values (dark blue circles) of each bin; dashed black and blue lines have slopes 1 and 2 respectively, corresponding to stars and cliques. The top anomalies deviating from the fit are marked with triangles. (b): Screenshot of the visualization module working with the anomaly detection module, showing a "star" (Sameer, at its center, is a red triangle flagged on the left plot), and a "near-clique" (blue triangles). Nodes are Stack Overflow users and a directed edge points from a user who asks a question, to another user who answers it. Node size is proportional to node's in-degree. Here, user Sameer (the star's center) is a maven who has answered a lot of questions (high in-degree) from other users, but he has never interacted with the other mavens in the near-clique who have both asked and answered numerous questions.

defined as the induced subgraph that contains the node itself (ego), its neighbors, as well as all the connections between them. Next, we extract several features from the egonets, for example, number of nodes, number of triangles, total weight, eigenvalue, etc. As we extract a dozen of features from all the egonets in the graph, feature extraction becomes computationally the most expensive step, especially for peta-scale graphs. Thanks to the PEGASUS module introduced in §2.1, the heavy-lifting of this component is efficiently handled through HADOOP.

II. Pattern mining. In order to understand how the majority of the 'normal' neighborhoods look like (and spot the major deviations, if any), we search for patterns and laws that capture normal behavior. Several of the features we extract from the egonets are inherently correlated. One example is the number of nodes N_i and edges E_i in egonet i: E_i is equal to the number of neighbors ($=N_i - 1$) for a perfect star-neighborhood, and is about N_i^2 for a clique-like neighborhood, and thus capture the density of the egonets.

We find that for real graphs the following Egonet Density Power Law holds: $E_i \propto N_i^\alpha, \quad 1 \leq \alpha \leq 2$. In other words, in log-log scales E_i and N_i follow a linear correlation with slope α. Fig. 3 illustrates this observation, for the example dataset 'Stack Overflow' Q&A graph, in which nodes represent the users and edges to their question answering interactions. Plots show E_i versus N_i for every egonet (green points); the larger dark blue circles are the median values for each bucket of points after applying logarithmic binning on the x-axis [1]; the red line is the least squares(LS) fit on the median points (regression on median points together with high influence point elimination ensures a more robust LS fit to our data). The plots also show a dashed blue line of slope 2, that corresponds to cliques, and a black dashed line of slope 1, that corresponds to stars. We notice that the majority of egonets look like neither cliques nor stars, but somewhere inbetween (e.g. exponent α=1.4 for the example graph in Figure 3). The axes are in log-log scales.

ODDBALL looks for patterns across many feature pairs and their distributions and yields a 'collection' of patterns. Therefore, ODDBALL generates a different ranking of the nodes by anomalousness score for each of these patterns. As a result, it can help anomaly *characterization*; the particular set of patterns a node violates explains what 'type' of anomaly that node belongs to.

III. Anomaly detection. Finally, we exploit the observed patterns in anomaly detection since anomalous nodes would behave away from the normal pattern. Let us define the y-value of a node i as y_i and similarly, let x_i denote the x-value of node i for a particular feature pair $f(x, y)$. Given the power law equation $y = Cx^\alpha$ for $f(x, y)$, we define the anomaly score of node i to be $score(i) = \frac{max(y_i, Cx_i^\alpha)}{min(y_i, Cx_i^\alpha)} * log(|y_i - Cx_i^\alpha| + 1)$, which intuitively captures the "distance to fitting line". The score for a point whose y_i is equal to its fit Cx_i^α is 0 and increases for points with larger deviance.

2.3 Interactive Visualization

The APOLO [2] module (see Figure 3b) provides an interactive environment to visualize anomalous nodes flagged by ODDBALL, and those nodes' neighborhoods, revealing connections that help the user understand why the flagged nodes are indeed anomalous. Should the user want to see more than the flagged nodes' direct neighbors, he can instruct APOLO to incrementally expand to a larger neighborhood. Typically, as for many other visualization software, such expansions could pose a huge problem because thousands of new nodes and edges could be brought up, clouding the screen and overwhelming the user. Instead, APOLO uses its built-in machine learning algorithm called BELIEF PROPAGATION (BP) to help the user find the most relevant areas to visualize. The user specifies nodes of interest, such as several flagged nodes as *exemplars*, and BP automatically infers which other nodes may also be of interest to the user. This way, all other nodes can be ranked by their relevance relative to the exemplar nodes, allowing the user to add only a few of the top-ranking nodes into the visualization.

BP is a message passing algorithm over link structure similar to spreading activation, but is uniquely suitable for our visualization purpose, because it *simultaneously* supports: (1) multiple user-specified exemplars; (2) dividing exemplars into any number of groups, which means each node has a relevance score for each group; and (3) linear scalability with the number of edges, allowing APOLO to generate real-time suggestions.

3. DEMONSTRATION PLAN

In our demonstration, we will invite our audience to interact with OPAVION and try out its core capabilities on a large graph: the STACK OVERFLOW Q&A graph, which describes over 6 million questions and answers among 650K users. In the graph, nodes are STACK OVERFLOW users, and a directed edge points from the user who asks a question, to the user who answers it.

In preparation for the demo, the *Summarization* module (PEGASUS) pre-computes the statistics of the graph (e.g., degree distribution, PageRank scores, radius, connected components) and creates plots that show their distributions. During the demo, the *Anomaly Detection* module (ODDBALL), using the pre-computed graph statistics, detects anomalous nodes in real time, and shows them in interactive plots (e.g., Figure 3a). The user can mouse-over the flagged nodes, and instruct OPAVION to show them in the *Visualization* module (APOLO).

In the visualization, users can interact with and navigate the graph, either from a node they like, or from the flagged anomalies. The user can spatially arrange nodes, and expand their vicinities to reveal surprising connections among the flagged anomalous nodes. For example, Figure 3b shows two subgraphs that include nodes flagged in Figure 3a (as blue and red triangles): a "star" subgraph with the user *Sameer* at its center, and a "near-clique" subgraph (users include Massimo, Chopper3, etc.). Node size is proportional to the node's in-degree. The visualization reveals that Sameer is a maven who has answered a lot of questions, having a high in-degree, but never asked any questions; on the other hand, the other mavens in the near-clique have a lot of discussion among themselves and never involve Sameer. It is an great example that shows that two vastly different anomalous subgraphs—star and near-clique—can actually be very close in the full graph (in this case, Sameer is only two hops away from the near-clique). The visualization helps with this type of discovery.

We will invite our audience to try out OPAVION, turn themselves into analysts, and make discoveries.

Acknowledgments

Research was sponsored by the Defense Threat Reduction Agency under contract No. HDTRA1-10-1-0120, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. This work is also partially supported by an IBM Faculty Award, a Google Focused Research Award, and a Yahoo Research Alliance Gift. 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, the U.S. Government, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

4. REFERENCES

[1] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *PAKDD*, 2010.

[2] D. H. 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 *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 167–176, New York, NY, USA, 2011. ACM.

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

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

[5] 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.

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