Metric Forensics: A Multi-Level Approach for Mining Volatile Graphs

Keith Henderson
Lawrence Livermore Lab
keith@llnl.gov

Tina Eliassi-Rad
Lawrence Livermore Lab
eliassi@llnl.gov

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

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

Lei Li
Carnegie Mellon University
leili@cs.cmu.edu

Koji Maruhashi
Fujitsu Laboratories Ltd.
maruhashi.koji@jp.fujitsu.com

B. Aditya Prakash
Carnegie Mellon University
badityap@cs.cmu.edu

Hanghang Tong
Carnegie Mellon University
htong@cs.cmu.edu

ABSTRACT

Advances in data collection and storage capacity have made it increasingly possible to collect highly volatile graph data for analysis. Existing graph analysis techniques are not appropriate for such data, especially in cases where streaming or near-real-time results are required. An example that has drawn significant research interest is the cyber-security domain, where internet communication traces are collected and real-time discovery of events, behaviors, patterns, and anomalies is desired. We propose METRICFORENSICS, a scalable framework for analysis of volatile graphs. METRICFORENSICS combines a multi-level "drill down" approach, a collection of user-selected graph metrics, and a collection of analysis techniques. At each successive level, more sophisticated metrics are computed and the graph is viewed at finer temporal resolutions. In this way, METRICFORENSICS scales to highly volatile graphs by only allocating resources for computationally expensive analysis when an interesting event is discovered at a coarser resolution first. We test METRICFORENSICS on three real-world graphs: an enterprise IP trace, a trace of legitimate and malicious network traffic from a research institution, and the MIT Reality Mining proximity sensor data. Our largest graph has ∼3M vertices and ∼32M edges, spanning 4.5 days. The results demonstrate the scalability and capability of METRICFORENSICS in analyzing volatile graphs; and highlight four novel phenomena in such graphs: elbows, broken correlations, prolonged spikes, and lightweight stars.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining; E.1 [**Data Structures**]: Graphs and networks

KDD'10, July 25–28, 2010, Washington, DC, USA.

General Terms

Algorithms, Design, Performance, Experimentation.

Keywords

Graph mining, temporal analysis, volatile graphs

1. INTRODUCTION

Given a stream of duration-stamped communication- or contact-events, how can we find suspicious behaviors, patterns, and anomalies in real-time or near real-time? How can we do attribution? For example, in a computer communication network, we would like to detect the interval that we are under attack, as well as the offending IP address (or addresses).

We define a "volatile graph" to be a stream of duration-stamped edges (in its simplest form: $\langle v_{src}, v_{dst}, start\_time, duration \rangle$), where we assume that there are potentially infinite number of nodes, and that edges may appear and disappear. Examples of volatile graphs include IP-to-IP communication graphs (either at the backbone or at the access-link) as well as physical proximity graphs (e.g., measured by bluetooth connections).

This paper introduces METRICFORENSICS which given a volatile graph is able to characterize it and detect interesting events at multiple levels (both temporally and topologically). At the global level, METRICFORENSICS computes and monitors a suite of graph metrics (e.g., the number of active nodes and links, the first few eigenvalues, their wavelet transforms, etc) at regular intervals. Only when a deviation from usual behavior is flagged, METRICFORENSICS follows through with a "drill down" approach, where the offending graph is studied at finer temporal and topological resolutions. In particular, flagged sub-graphs (community-level) and even individual nodes (local-level) are examined using more sophisticated and time-consuming metrics and analysis techniques (such as ego-net analysis). Thus, METRICFORENSICS is able to do *attribution* of the rare event, while maintaining high processing speed.

The contributions of METRICFORENSICS are as follows:

- *Effectiveness*: METRICFORENSICS spots strange activities, like "*elbows*" (where the observed behavior changes

while another phenomenon remains stable; see Section 4.1.1), broken correlations (where previously strong correlations disappear; see Section 4.1.2), prolonged spikes (where there is low volume but prolonged activity-level; see Section 4.1.3), and "*lightweight*" stars (i.e., vertices that form very big star-like structures but have lower than expected total incident edge-weights; see Section 4.3).

- *Scalability:* All the components of METRICFORENSICS are carefully chosen to be not only informative, but also fast to compute (linear on the measures of interest).
- *Flexibility* and *generality*: The METRICFORENSICS framework can easily include other modules in addition to the ones described, like spectral analysis, PageRank, etc. Moreover, the method can be applied to any type of volatile graphs (e.g., email/SMS communications).

METRICFORENSICS satisfies the six requirements of an emerging application. They are:

1. Requirements for the application: METRICFORENSICS' application is mining of highly volatile graphs represented as streams of duration-stamped edges. This application is useful in analysis of various communication data (e.g., IP traffic, phone-calls, blue-tooth connections, twitter feeds, etc). Its requirements are scalability and ability to run in a streaming or (near) real-time environment.
2. Approach: METRICFORENSICS is a multi-level graph-mining framework, which takes as input a stream of volatile graphs (represented by duration-stamped edges), constructs summary graphs, and conducts multi-level scalable graph analysis (such as eigen-value, fractal dimension, and correlation analyses). METRICFORENSICS is easily extensible to include analysis modules besides the ones presented here. See Figure 1.
3. Deployment: METRICFORENSICS has been released to various projects within Lawrence Livermore National Laboratory. We plan to deploy the system to our government sponsors and release the code as open source.
4. Evaluation: METRICFORENSICS was evaluated on three real-world volatile graphs from various domains. Our largest graph has $\sim$3M vertices and $\sim$32M edges, spanning 4.5 days. The results were verified by inspection of background information (such as deep packet-capture analysis) and discussions with analysts.
5. Pragmatic issues: METRICFORENSICS' framework is designed to be scalable (i.e., operate on graphs with millions of nodes and edges) and run in streaming or (near) real-time environments. See Tables 1 and 2.
6. Comparative evaluation: Existing approaches either fail in key requirements and functionalities (such as scalability, multi-level graph analysis, and attribution) or can be easily added as a module into METRICFORENSICS' extendable framework (see Section 2). For example, one of the closest systems is Redback [32], which takes a series of graphs and conducts single-level graph analysis. METRICFORENSICS and Redback are not directly comparable since Redback (a) puts the burden of discretizing the stream of edges on the user, (b) it does not conduct multi-level analysis, (c) it is not scalable (the largest graphs reportedly tested on Redback are hundreds of nodes), and (d) Redback is not open-source. METRICFORENSICS can be used to compare different analysis algorithms such as BGP-Lens [30] and fractal dimension analysis (see Section 4.1.3).

The outline of the paper is as follows: Section 1 provides an introduction to this work. Section 2 presents an overview of the related work. Section 3 describes our proposed framework, METRICFORENSICS. Section 4 presents experimental results on three real-world volatile graphs. Lastly, Section 5 provides some concluding remarks.

2. BACKGROUND AND RELATED WORK

We divide related work into four parts: (1) mining static graphs, (2) mining time-evolving graphs, (3) anomaly detection on graphs and finally (4) mining time series.

Mining Static Graphs. This work can be grouped into three levels. First, on the graph-level, there are discoveries on the statistical properties of some global metrics (e.g., degree distribution, diameter, first eigenvalue, etc) of the whole graph [3, 15, 8, 28]. Next, at the subgraph level, research has focused on frequent substructure discovery [35], graph partitioning, and community detection (eg [16]. Finally, at the individual node/edge level, past work includes link prediction [26], ranking [17], and proximity [33]. Note that almost all of the previous work deals with *only one* of the three levels, while METRICFORENSICS works on all the levels in a "drill down" way.

Mining Dynamic Graphs. Most work here has focused on community evolution [4, 13] and dynamic tensor analysis [31]. Again, most of the above analyze *only one* of the three levels (graph level, subgraph level, or node/edge level), and, typically, on *a single, fixed* time-granularity, in contrast to METRICFORENSICS.

Anomaly Detection on Graphs. Gibbons and Matias [18] proposed a fast method to compute "heavy-hitters" (that is, frequently-occurring items, like source-IP addresses). Lakhina et al. [24] suggested using entropy to characterize a connectivity matrix. Bunke et al. [9] use similar measures to spot differences between connectivity matrices and to report anomalies when these differences are too high. Additional work includes methods based on the minimum description length (MDL) principle [29, 10], classification-based methods [27], probabilistic measures [14], and spectral methods [21, 20]. In OddBall [2], they explicitly focus on the "individual" node-level by examining the 1-step-away ego-networks. For a comprehensive list, see a recent survey [12]. All such methods can be folded within METRICFORENSICS framework at the appropriate topological and temporal levels.

Mining Time Series. Related work here includes click-through rate estimation [1] and outlier detection [25]. Again, METRICFORENSICS can naturally incorporate these methods into its framework, such as BGP-Lens [30] for finding patterns and anomalies in Internet routing updates.

3. METRICFORENSICS

The flowchart for METRICFORENSICS is depicted in Figure 1. METRICFORENSICS is comprised of three distinct components: (1) a suite of graph metrics, (2) a collection of analysis techniques, and (3) a multi-level approach. We will de-

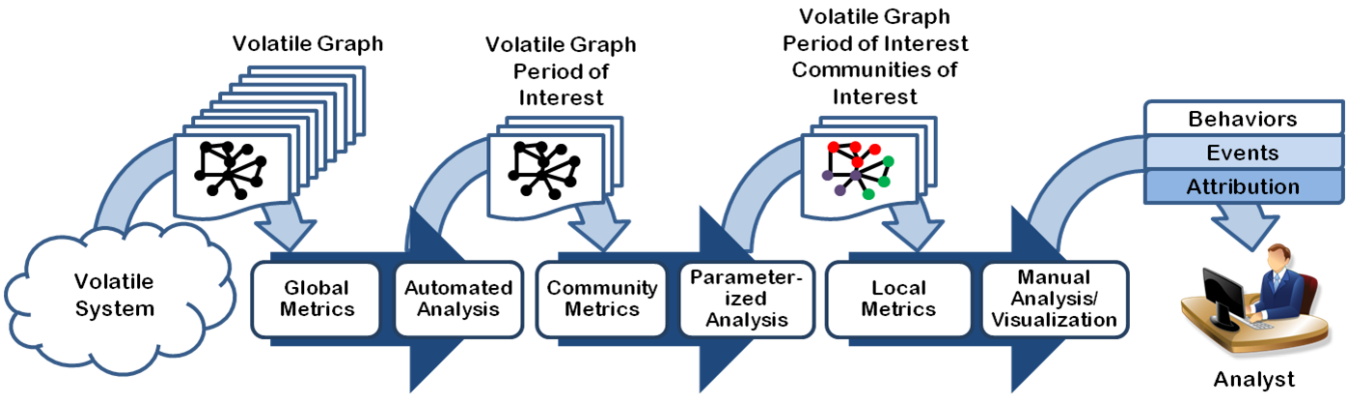


Figure 1: METRICFORENSICS' Flowchart

scribe each of these below. But first, we will briefly discuss METRICFORENSICS' data model for representing volatile graphs.

3.1 Data Model for Volatile Graphs

Highly volatile graphs, by definition, accumulate massive numbers of vertices and edges over time. However, during a given window of time, only a fraction of these vertices and edges are active. The METRICFORENSICS data model takes advantage of this behavior.

3.1.1 *Vertices and Edges*

Entities are represented as vertices in an ever-changing graph; they can appear or disappear at any time. If a vertex does not have any relationships in the current graph, it is effectively discarded until it forms new relationships. We do not assume knowledge of the full set of vertices, or even the total number N of vertices at any time during analysis. Likewise, relationships are represented as edges in the graph. Edges can be weighted or unweighted. They can have duration or be instantaneous. Multiple edges can exist between a pair of vertices at a given time (i.e., we are dealing with multi-graphs). Vertices and edges can have attribute data associated with them. It is assumed that in some cases these attributes require significant resources to access or analyze.

3.1.2 *Snapshot Graphs*

A *snapshot graph* is defined by its vertices V_t and edges E_t, which are active at time t. A snapshot graph can be viewed as an $N \times N$ adjacency matrix representing the graph at time t. The dynamic system is then comprised of many such matrices in sequence. Each time a vertex is added or deleted, or an edge appears or disappears, or an edge-weight is changed, a new snapshot graph is generated.[1]

3.1.3 *Summary Graphs*

Due to the high volatility of the data, it is neither computationally feasible nor analytically worthwhile to consider snapshot graphs in isolation. A *summary graph* summarizes all snapshot graphs a during time period T. It is represented by its vertices V_T and edges E_T. Many strategies are available for combining snapshot graphs, including:

- *Binary*: An unweighted edge (i, j) exists in the summary graph G_T if (i, j) exists in at least one snapshot graph during T.
- *Sum*: A weighted edge $w(i, j)$ exists in the summary graph G_T if (i, j) exists in any snapshot graph during T. Then, $w(i, j)$ is the sum of the weights of edges active at the beginning and during the interval T.
- *Max*: Similar to *Sum* except that $w(i, j)$ is the maximum value of element a_{ij} in the adjacency matrices of snapshot graphs for time interval T.

The frequency with which summary graphs are generated and analyzed is a parameter in METRICFORENSICS, and plays an important role in the multi-level component of the framework (see Section 3.4). Summary graphs can be generated after a fixed number of distinct snapshot graphs or after a fixed period of time. Our experiments demonstrate that the framework works across a reasonably large set of summary graph frequencies, and as a heuristic we tend to choose the frequency so that each summary graph represents no more than 100,000 unique snapshot graphs.

3.2 Suite of Graph Metrics

At the heart of METRICFORENSICS is a suite of graph metrics. These metrics are of varying levels of complexity and computational intensity. They are broadly classified into three groups based on their topological granularity: (1) *global*, (2) *community*, and (3) *local*. The framework is readily extendable to include any graph metrics. Moreover, it is not necessary to run all the metrics at all times.

3.2.1 *Global Metrics*

At the coarsest topological level, global metrics generally measure high-level properties of the graph and are largely agnostic to properties at the individual vertices. Table 1 lists a subset of METRICFORENSICS' global metrics. Several of the metrics have both unweighted and weighted versions; only the unweighted versions are listed here. Most are very fast to calculate, scaling linearly with the number of active vertices ($N_T = |V_T|$) or edges ($M_T = |E_T|$) in the time interval T. Currently, all of our implemented global metrics have complexity at most $O(N_T \log N_T + M_T)$.

[1]Data containing instantaneous edges (i.e., with duration of zero) will generate trivial snapshot graphs consisting of one edge at a time.

Basic Metrics	Time Complexity
Number of active vertices	$O(1)$
Number of active edges	$O(1)$
Average vertex degree	$O(1)$
Average edge weight	$O(1)$
Maximum vertex degree	$O(N_T)$
Connectivity Metrics	**Time Complexity**
Number of connected components	$O(M_T)$
Fraction of vertices in the largest component	$O(M_T)$
Number of articulation points	$O(M_T)$
Minimum spanning tree weight	$O(M_T)$
Spectral Metrics	**Time Complexity**
Top-k eigenvalues of the adjacency matrix	$O(N_T k^2 + M_T k)$
Stability Metrics	**Time Complexity**
Jaccard(V_T, V_{T-1})	$O(N_T)$
Jaccard(E_T, E_{T-1})	$O(M_T)$

Table 1: A subset of METRICFORENSICS' suite of global graph metrics

3.2.2 Community Metrics

A second collection of metrics examines the graph at its community-structure level. These algorithms are typically more computationally expensive than those for the global metrics. Many approaches to community discovery in graphs exist [19]. The results presented in Section 4 are based on *Cross-Associations* (XA) [11]. Regardless of the chosen community discovery algorithm, the metrics are similar.

Some community metrics are static, such as the fraction of vertices in the largest community or the number of communities. Others track changes in community structure, such as the variation of information [22] between successive assignments. If a particular vertex is of interest, then changes in its community can be easily tracked between successive summary graphs.

3.2.3 Local Metrics

The final group of metrics focuses on individual vertices. Local metrics often run too slowly to be applied to every vertex in each summary graph. Examples of local metrics include centrality metrics, OddBall [2], and impact metrics (e.g., leaving a single vertex out of the graph and recalculating other metrics to determine the impact of the vertex).

3.3 Collection of Analysis Techniques

The second component of METRICFORENSICS is a collection of analysis techniques. Broadly speaking, they fall into three categories: (1) *single metric analysis*, (2) *coupled metric analysis*, and (3) *non-metric analysis*. This component is similar to the suite of metrics in that it can easily accommodate the addition of other techniques.

3.3.1 Single Metric Analysis

Values for an individual metric across multiple summary graphs can be viewed as a time series. METRICFORENSICS leverages the multitude of time series analysis techniques to identify behaviors, events, and anomalies. For example, an Autoregressive Moving Average (ARMA) Model can be used to identify metric values that are abnormally large or small given recent values. Fourier analysis can identify periodic behavior, such as daily trends in graph properties. Wavelet analysis tools such as BGP-lens [30] identify patterns and anomalies in metric values. Other single-metric tools include lag plots, outlier detection techniques such as Local Outlier Factor [7] and fractal dimension analysis [6].

3.3.2 Coupled Metric Analysis

Techniques in this category consider two or more metrics in unison. The simplest such technique is correlation analysis. If K metrics are computed for a series of summary graphs, a $K \times K$ matrix, C, can be computed where C_{ij} is say the Pearson correlation between metrics i and j. Large values of $|C_{ij}|$ can identify redundant metrics. If such metrics vary widely in runtime complexity, then the slower ones can be omitted from future calculations. However, it is often useful to retain both metrics; if the computed values of two metrics typically demonstrate high correlation, a sequence of summary graphs that shows lower correlation is identified as an interesting event.

A useful example of coupled metric analysis involves various summary-graph edge-weighing strategies (Section 3.1.3). In particular, if metrics are computed simultaneously on summary graphs constructed using different strategies, such as *Sum* and *Max*, the resulting time series data are often highly correlated. In this case, a summary graph for which the metric-values do not demonstrate their typical relationship can be identified as an interesting event.

Other techniques can be applied to coupled metric data, such as outlier detection or clustering. For example, a clustering algorithm like k-means can be applied to two time series. Small clusters are labeled as interesting events or behaviors (see Section 4 for details).

3.3.3 Non-Metric Analysis

Techniques in this category do not involve the computed metrics (as described in Section 3.2). These techniques are *not* applied until an interesting event is discovered using the above techniques, but they are often useful for understanding the events. The primary existing techniques in this category are visualization tools and attribute data inspection.

METRICFORENSICS currently includes a novel 3D visualization tool that can display summary graphs rapidly and in an informative layout. It highlights vertices with high connectivity, and is used to quickly characterize a sequence of summary graphs that have been identified as interesting. The tool uses position (source vs. target vertex), size, and color to differentiate between vertices according to a user-specified collection of attributes. For example, the size of a vertex can show its degree, while the color can depict the vertex' betweenness centrality. See Figure 3a for a 2D snapshot of a summary graph by our visualization tool.

The second non-metric analysis technique involves inspection and processing of available attribute data. Vertices and edges in volatile graphs can have attributes. In some cases, more detailed attributes may be available at an increased cost of access. These should be retrieved only when necessary. For example, IP communication traces often have at least partial packet contents, but these are usually not available for fast inspection. While it is not feasible to consider every packet in detail, METRICFORENSICS can identify periods of time and sets of edges that may be of interest based on graph metrics or community structure. A user can then apply a full

packet capture (a.k.a. *pcap*) analysis tool to the identified regions.[2]

3.4 A Multi-Level Approach

METRICFORENSICS' multi-level approach allows for efficient use of computational resources. Due to the volatile nature of our data (e.g., IP network traces) and the varying complexity of metrics and analysis techniques, it is necessary to rely on fast techniques at coarse granularities (both temporally and topologically) to identify regions of interest, and then apply complex algorithms and tools only to interesting regions. METRICFORENSICS uses multiple levels in three distinct dimensions: (1) *time*, (2) *topology*, and (3) *analysis automation*.

The general approach involves performing METRICFORENSICS' metrics and analysis multiple times at different levels, starting with the coarsest and becoming finer at each iteration. Only those time periods identified as interesting at a coarse level are passed down to be analyzed at the next finer level. We generally identify three levels, based on the topological granularity levels (namely, global, community, and local). However, METRICFORENSICS supports any number of levels based on time granularity.

3.4.1 Time Granularity

The temporal scale of METRICFORENSICS can be controlled in two ways. First, the period of time in which summary graphs are analyzed can be adjusted. At the coarsest level, METRICFORENSICS operates on all available data, which in many cases can include streaming data. When an event is detected, only the relevant portion of the data is examined at finer levels.[3] Second, temporal granularity is adjusted by modifying the interval between summary graphs. At the coarsest level, summary graphs are generated less often than in finer levels. This "drill-down" approach is used to pinpoint changes in behavior of specific vertices (i.e., do attribution).

3.4.2 Topological Granularity

The axis of refinement here involves which set of graph metrics are applied. At the coarsest level, only the global topology of the graph is considered. Communities and individual nodes are not generally considered, with the exception of a small number of global statistics that track the identities of high-degree vertices. The global metrics are scalable and can be computed efficiently on each summary graph. When an event is discovered at this level, the period of interest is passed to the next (finer) level.

At the finer (regional) level, community-level metrics are calculated. By identifying communities that exhibit change, METRICFORENSICS can discard many vertices that have not changed their behavior. This information is subsequently used at the finest level of refinement, where local metrics are computed on vertices in the identified communities.

3.4.3 Analysis Automation Levels

The final difference between levels in METRICFORENSICS is the selection of analysis techniques. Some techniques, such as ARMA, are fully automated. These can be applied at any refinement level. Other tools and techniques like visualization and attribute analysis require user interaction and should only be applied to small sets of summary graphs.

[2]An example of a full pcap analysis tool is *Wireshark*: http://www.wireshark.org/.

[3]In a streaming setting, this is accomplished by maintaining a circular buffer that stores a fixed number of recent snapshot graphs.

4. EXPERIMENTS

We implemented METRICFORENSICS in Java (with some MATLAB modules) and ran experiments on a commodity machine Intel Core 2 Duo @2.93GHz with 4Gb of memory. Our experiments answer the following questions: (1) *Can* METRICFORENSICS *detect interesting events including anomalies?* (2) *Do the discovered interesting events tell us something new about the nature of volatile graphs?* (3) *Is* METRICFORENSICS *scalable and amenable to real-time (or near real-time) execution?*

Table 2 lists the graphs used in our experiments. ENTP is IP traffic collected at the perimeter of an enterprise network over 4.5 days in 2007. RMBT is the MIT Reality Mining's blue-tooth connections collected over 12 months.[4] LBNL is IP traffic collected on an internal enterprise network during a busy hour on 2004.12.15 on port #3 (TCP and UDP compression processes).[5] LBNL data includes scanning activities.

4.1 Experiments at the Global Level

We discuss some experiments at the global-level of our volatile graphs here. For brevity, we have removed many of results (such as our Fourier and wavelet analyses).

4.1.1 Eigen Analysis

Figure 2 depicts the two largest eigenvalues in the ENTP summary graphs. In particular, it shows the λ_1, λ_2 relationship under three different edge-weighing strategies. In the *maximum connections* strategy, the weight between vertices i and j is equal to the maximum number of simultaneously active connections between i and j during the summary graph's time interval T. Under the *number of connections* strategy, the weight between i and j is equal to the number of active connections between i and j when T started plus the number of connections between i and j during T. In the *sum of bytes* strategy the weight between i and j is the normalized sum of the flow-weights (i.e., number of bytes sent and received) when T started and the weights of flows that occurred between i and j during T. Regardless of the summary graphs' edge-weighing strategy, there are special regions where λ_1 is stable and λ_2 is changing, or vice versa. We also observe these special regions in the LBNL trace (see Figure 3), where they are elbow-shaped.

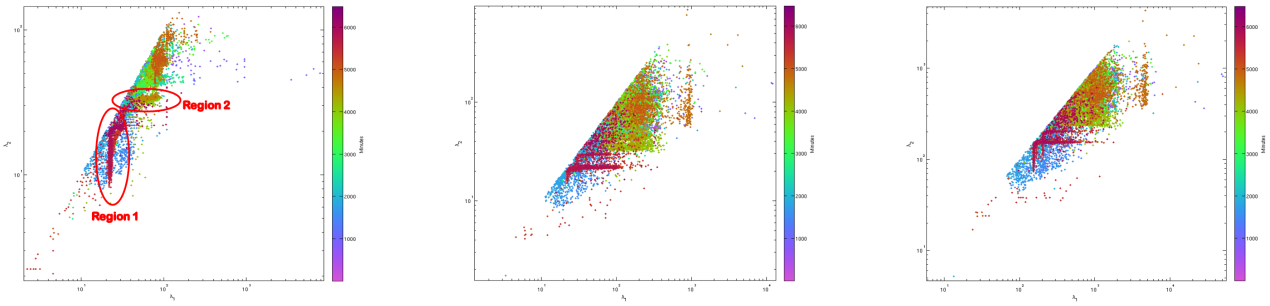
The large eigenvalues of a weighted graph typically correspond to either a single heavy edge, a vertex with high weighted degree, or a component with a large total weight. Thus, when we see a period of time when λ_1 is changing but λ_2 is steady, it is a result of the currently dominant phenomenon changing while the secondary phenomenon is stable (e.g., a single heavy edge changing weight while the structure of the giant component is steady). We refer to this as the "elbow" pattern because it appears as elbow-like structures (Figure 3). A trivial example here is a pair of heavy edges, (a, b) and (c, d), with $w(c, d) > w(a, b)$ initially. If $w(a, b)$ remains constant and $w(c, d)$ decreases such that eventually $w(c, d) < w(a, b)$, the corresponding eigenvalues will switch

[4]http://reality.media.mit.edu/

[5]http://www.icir.org/enterprise-tracing/download.html

Data Graph	# of Source Vertices	# of Target Vertices	# of Total Vertices	# Unique Edges	# Total Edges	Observation Time	Window Size	Runtime (wall clock)
ENTP	1,748,750	1,733,521	2,928,116	6,597,251	31,855,024	6480 min	0.5 min	107.75 min
RMBT	94	25,490	25,491	55,898	1,982,576	525.95K min	30 min	5.47 min
LBNL	3,268	2,837	3,317	15,577	9,258,309	60 min	0.0083 min	6.85 min

Table 2: Real-world networks used in experiments. Observation time is the span of time for which we have data. Window size is determined based on activity rate (e.g., IP traffic is faster than blue-tooth connections) and expected reaction time to events. Runtime is METRICFORENSICS' wall-clock time.



(a) Maximum Connections Strategy (b) Number of Connections Strategy (c) Sum of Bytes Strategy

Figure 2: λ_2 versus λ_1 under various edge-weighing strategies in the ENTP summary graphs (generated every 30 seconds). x-axis is λ_1 in log-scale; y-axis is λ_2 in log-scale. The color of a dot is the time that it was observed (in minutes): blue is earlier, red is later. Regardless of the summary graphs' edge-weighing strategy, there are interesting regions with elbow patterns where λ_1 is stable and λ_2 is changing, or vice versa.

so that λ_1 is always correlated with the larger-weight edge. Thus, during the initial period λ_1 tracks the changing $w(c,d)$; but once $w(c,d) < w(a,b)$, λ_1 becomes stable and λ_2 tracks the (c,d) edge.

Depending on the edge-weighting strategy employed, these periods may appear simply as horizontal or vertical sections (Figure 2) or they may appear as elbows (Figure 3). Regardless, the observed behavior is one phenomenon (heavy edge, heavy vertex, or heavy component) that is changing while another phenomenon remains stable.

4.1.2 *Correlation Analysis*

We computed the pairwise Pearson correlation coefficients between values of global metrics. For example, given summary graphs $G_{T_0} \cdots G_{T_t}$ we computed

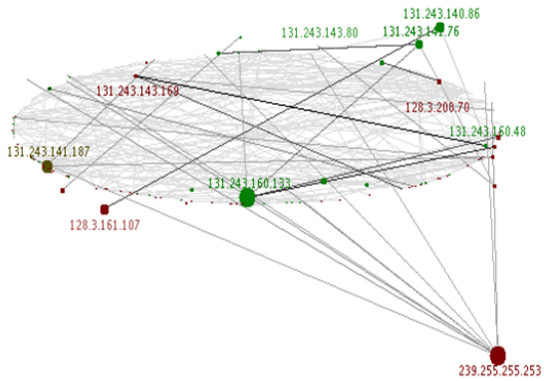
$$r([\lambda_1^{G_{T_0}} \cdots \lambda_1^{G_{T_t}}], [max\_wgt^{G_{T_0}} \cdots max\_wgt^{G_{T_t}}])$$

where max_wgt is the maximum edge weight. Figure 4 depicts the top-14 most correlated global metrics with λ_1 for the ENTP data. It shows that normally λ_1 is highly correlated with maximum edge weight; however in Region 1 of Figure 2a (where λ_1 is stable but λ_2 is changing), this correlation disappears. Indeed λ_1 is not correlated with any graph metric in this region. We observed this behavior on other data sets and other eigenvalues. For instance, λ_2 is highly correlated with the fraction of vertices in the largest component, except in regions like Region 2 of Figure 2. In these special regions (where λ_1 is changing but λ_2 is stable), λ_2 is highly correlated with the number of updates (e.g., addition or deletion of vertices and edges). We refer to this phenomena as "broken correlations" and observe that there are meta-level correlations between broken correlations and elbow patterns described above.

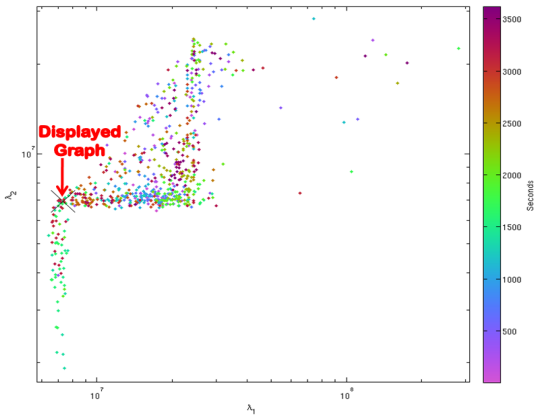
4.1.3 *Fractal Dimension Analysis*

Intuitively, fractal dimension [6] measures the burstiness of a collection of points. Human behavior is typically bursty [5] (e.g., disk accesses [34], email responses [23]). In our case, the points are in 1-dimensional space and correspond to communications at different times. For points that are uniformly distributed in time, the fractal dimension is near the dimensionality of a line (i.e., $D \approx 1$). For points that are all on the same time-tick (creating a single burst), the fractal dimension is the dimensionality of a point, $D = 0$. The Cantor set (constructed by recursively deleting the middle third of a line segment) has fractal dimension $D = \frac{\log(2)}{\log(3)} \approx 0.63$. Packets due to human behavior typically have fractal dimension somewhere in [0.7, 0.9] (with self-similar bursts at different time scales).

We computed the fractal dimensions of several graph metrics on our summary graphs (which can be regarded as a cloud of points on the time axis). Fractal dimensions were calculated for (disjoint) windows of width w = 3 hours on the ENTP data, w= 5 minutes on the LBNL data, and w =10 days on the RMBT data. These parameters were selected so that each window contains ~500 events (which is roughly the minimum size that fractal dimension analysis makes sense). The fractal dimension was stable for most time periods, around 0.9 on ENTP and LBNL and around 0.8 on RMBT data. This result suggests that RMBT data is more bursty than others. Interestingly, the fractal dimensions for some metrics (such as number of additions, number of deletions, number of up-



(a) Graph at 2004.12.15 20:06:51.348



(b) λ_2 versus λ_1

Figure 3: (a) The LBNL graph at 2004.12.15 20:06:51.348. The vertex colors indicate the recent position of the vertex: source (green) vs. destination (red). The elevation represents the same role but considers the entire history of the vertex. If the vertex is quiet, it slowly moves from green or red back to black; but will not change elevation. The IPs with names either have high weight at 2004.12.15 20:06:51.348 or have had high weight within the last 50 seconds. High weight is defined as 50% of the current maximum weight in the graph. (b) λ_2 versus λ_1 in the LBNL summary graphs (generated every 5 seconds with sum-of-flows strategy). x-axis is λ_1 in log-scale; y-axis is λ_2 in log-scale. The elbow patterns occur when the dominant phenomenon and the secondary phenomenon swap roles.

dates, and number of edges connecting to an IP outside the enterprise) suddenly drop to 0.6-0.8 in some periods on the ENTP data (Figure 5a). Specifically, the fractal dimension of the number of additions suddenly drops down in the early morning on 2007.11.15 (between 6 AM to 9 AM). Figure 5b shows the magnification of that interval, illustrating that the drop is due to a "prolonged spike" – i.e., an activity that has low-volume, but persists for a long time. We also observed this phenomena with wavelet analysis and the BGP-lens package [30]. For brevity, we omit the details of our wavelet analysis here.

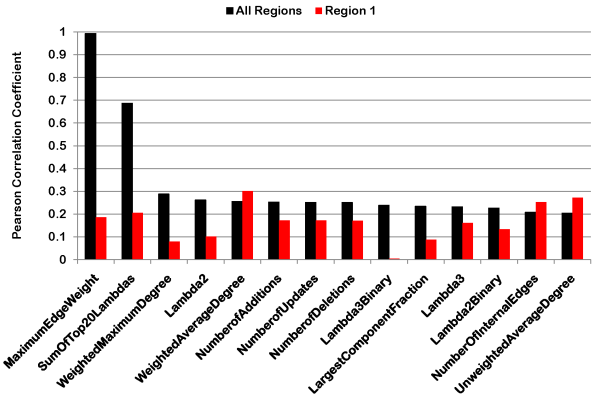
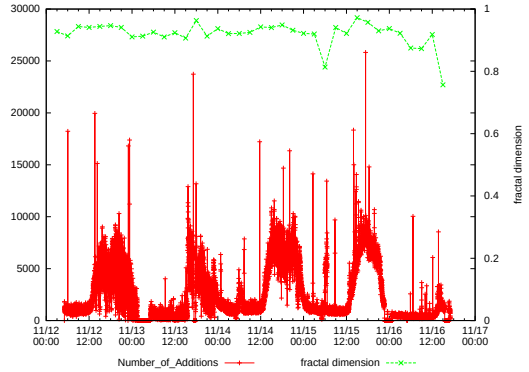
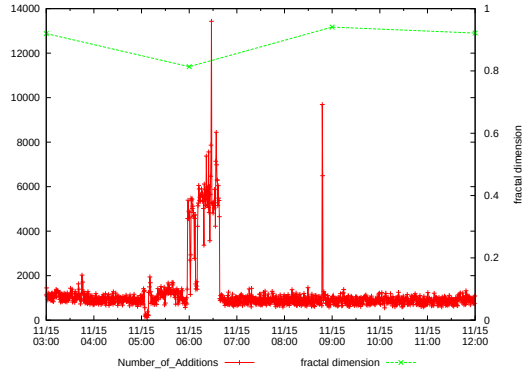


Figure 4: The top-14 graph metrics correlated with λ_1 in the ENTP data. The sharp drop in correlation for Region 1 of Figure 2 is very interesting and depicts a broken correlation.



(a) # Additions



(b) # Additions on the Magnified X-axis Scale

Figure 5: Fractal Dimension Analysis on ENTP data. Biplots: Number of updates (in red) and fractal dimension D (in green) versus time-stamp. (a) The full interval of analysis - note the drop of fractal dimension around 6 AM on 2007.11.15. (b) Magnification of the suspicious region, which has a "prolonged spike" (low volume, but prolonged activity-level). Notice the prolonged spikes in the number of edge-additions to the summary graphs.

4.2 Experiments at the Community Level

When a summary graph is flagged as interesting at the global-level, the next step is to analyze the flagged summary graph at its community-level. Figure 6a depicts the pairwise plot of λ_1 and the fraction of source nodes in the largest XA row-group (a.k.a. row-group fraction) during Region 1 from Figure 2. The points are clustered by k-means with $k = 8$, which produces five singleton clusters, one cluster of size 6, one cluster of size 37, and another one of size 352. While the singleton clusters and the small cluster of 6 are detectable from λ_1, the larger clusters have nearly identical centroids in λ_1 but are separable by their XA row-group assignments. For those summary graphs in the cluster of size 37 (dark red cluster in Figure 6a), vertices in the largest row-group are marked as suspicious and passed on for further local-level analysis. Figure 6b shows an exponential moving average of the XA row-group's variation-of-information (where lower values indicate more stable community structures) in LBNL. As pointed out by the red arrow, there is a noticeable dip between 750 and 1100 seconds from the start of the hour when communities are very stable. Figure 6c illustrates the XA row-group's variation of information against the same measure on column groups. We observe that they are correlated, but there are also instances for which one is abnormally high or low given the other (i.e., points in the dashed red circles). These indicate that there vertices whose row-groups are changing but not their column-groups (i.e., they are changing their behavior as source vertices but not as target vertices), and vice versa.

4.3 Experiments at the Local Level

When METRICFORENSICS detects interesting events in the given stream of volatile (or summary) graphs, it can zoom into those interesting graphs and perform more rigorous analysis. In such cases, the main goal is to find interesting (extreme, outlying, suspicious) *vertices* in a graph. To do so, features from the neighborhood of vertices are extracted. In particular, given a vertex, its neighbors, and the connections between them (a.k.a. the induced 1-step subgraph of the vertex or the egonet), METRICFORENSICS can employ a local-level analysis tool like OddBall [2]. OddBall computes the number of edges, the total weight of edges, etc and defines the vertices as points in a multi-dimensional feature space, in which it looks for anomalies.

Figure 7a shows the number of edges versus the number of vertices in the egonets of RMBT. Each point in the scatter plot corresponds to a particular vertex. Here, the dashed blue line with slope 2 corresponds to cliques and similarly the dashed black line corresponds to stars. We observe that most of the points lie on the blue line which indicates that a vast majority of vertices have neighborhoods that look like a clique. For RMBT, this is intuitive; all blue-tooth devices in a specific region will "see" each other, and hence form cliques. On the other hand, we also observe a second cluster of nodes that are neither cliques nor stars. The outlier points here are the black and the blue triangles, which indicate two big "lightweight" stars. Figure 7b shows the total edge-weight versus degree in RMBT. Here, the weights denote the number of times two devices where close enough to connect to each other. We again observe that vertices form two clusters. The two triangles shown in the circle are the same points as the ones discussed earlier in Figure 7a. These vertices not only form very big star-like structures, but also their total edge-weights are lower than expected. Hence, we refer to these as "lightweight" stars.

We performed similar analysis on the LBNL graph, where the vast majority of the vertices form star-like structures. This is intuitive since the LBNL data is a sample of the network traffic over a limited amount of time (≈1 hour), so we have partial information about the interactions between all vertices. Figure 7c and 7d show the total weight versus the number of edges in the egonets of nodes in the LBNL graph without and with scanning activities, respectively. Here the weights denote the total number of packets sent between pairs of machines. On each plot, we show the top 100 anomalies we detected using a simple metric of the distance from the fitting line. Note that we were able to detect non-scanner vertices that sent much fewer packets than expected compared to the number of machines they connected with (point shown in square on the figures) as well as detect scanners with a similar behavior: fewer packets than the norm over links (points shown in circle on the same figures).

For the ENTP data, we observed a massive increase (of 10x) in communications around 9 AM on 2007.11.12. Looking at the flow data, we observed a pair of machines that opened over 10K connections in about a minute on a BitTorrent related port. Moreover in early morning hours of 2007.11.13, we observed an order of magnitude increase in λ_1 of the weighted summary graph but did not see a corresponding jump in λ_1 computed on the unweighted summary graph. This was a case where looking at traffic volume alone could not detect the single heavy edge that caused λ_1 to spike for several minutes. For brevity, we omit the plots.

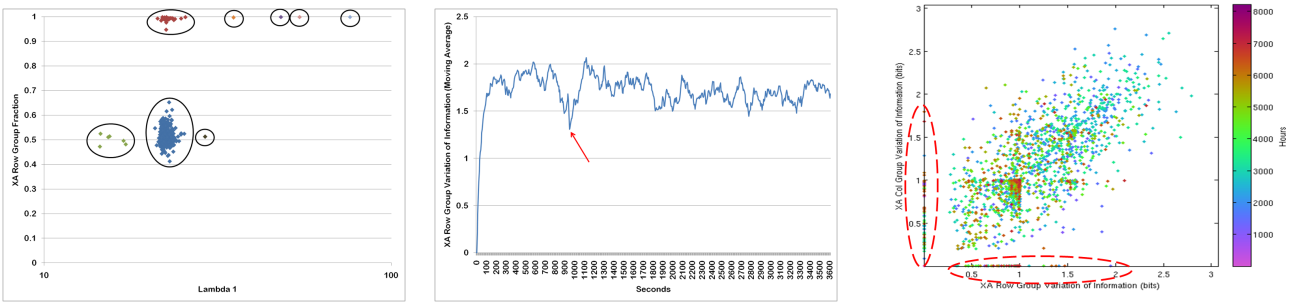
5. CONCLUSIONS

Volatile graphs (such as IP-to-IP communication graphs) are becoming more ubiquitous in network science applications. Challenges associated with mining of such graphs include dealing with an ever-changing graph, analysis in streaming or real-time fashion, and analysis at multiple temporal and topological granularities. In this paper, we presented METRICFORENSICS: a multi-level framework for mining volatile graphs that addresses the aforementioned challenges; and illustrated the generality and applicability of METRICFORENSICS on several large real-world volatile graphs (with up to ∼32M edges). Its strong points are the following:

- METRICFORENSICS is *effective*, capable of spotting suspicious patterns like the "elbow" pattern, prolonged spikes, broken correlations, and more.
- It is *scalable*, with carefully chosen operations, fast-to-compute components (eigenvalues, wavelets, etc), and global-to-local architecture, for efficient runtimes.
- It is *flexible*, general and extensible, with room for many more components, in addition to ones used (fractal analysis, OddBall, wavelets, etc.)

6. ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. DE-AC52-07NA27344. LLNL-CONF-432792.

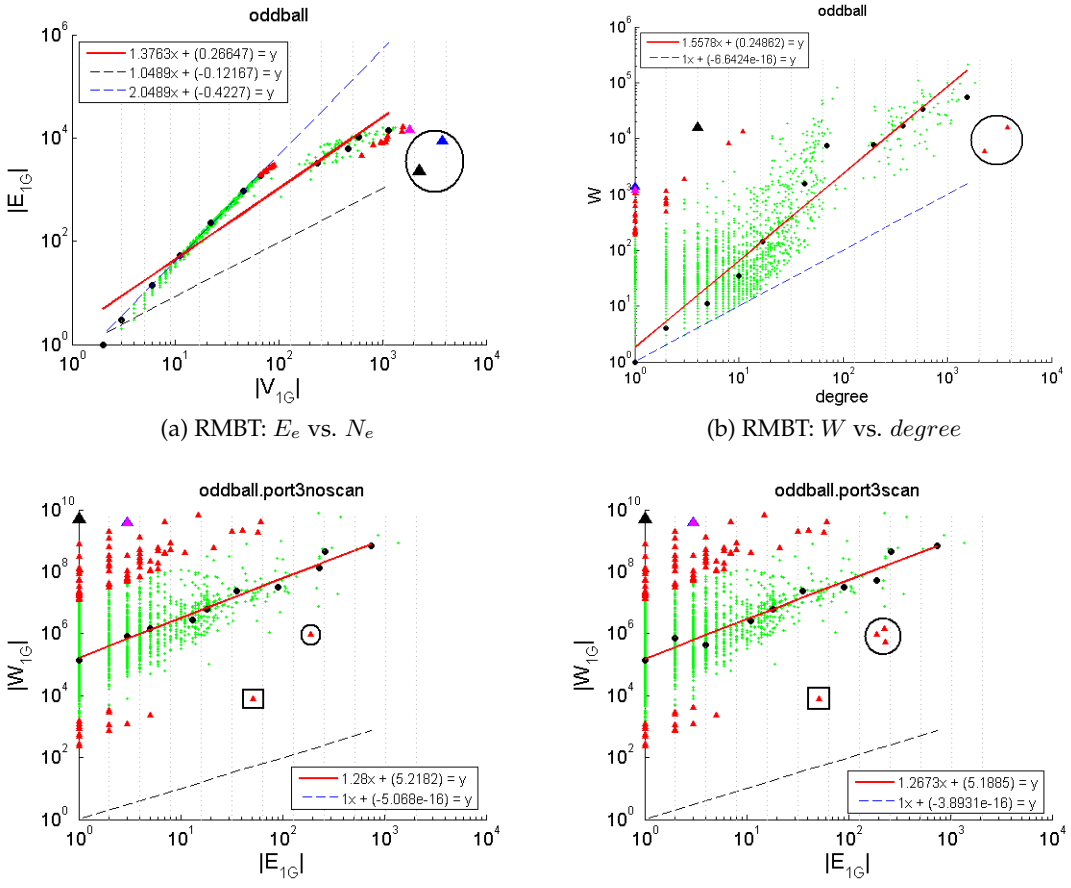


(a) ENTP: k-Means Clusters for Row-Group Fraction vs. λ_1

(b) LBNL: Row-Group's Variation-of-Information

(c) RMBT: Variation-of-Information on Column- vs. Row-Group

Figure 6: Community-Level Experiments with Cross-Associations (XA). (a) During Region 1 (shown in Figure 2a), ENTP has behaviors that can be detected using XA (dark red cluster) but not using other metrics (λ_1 shown). (b) For the LBNL data, there is a pronounced increase in community stability for about 5 minutes. (c) RMBT includes times where source vertices form stable communities but targets do not, and vice versa as encircled by the dashed red ovals.



(a) RMBT: E_e vs. N_e

(b) RMBT: W vs. $degree$

(c) LBNL': W_e vs. E_e

(d) LBNL: W_e vs. E_e

Figure 7: Local-Level Experiments with OddBall. LBNL' is the LBNL data without scanning activity. For a given vertex, W and $Degree$ are its sum of edge-weights and its number of neighbors, respectively. N_e, E_e, and W_e are the number of vertices, edges, and the total weight of all edges in a vertex' egonet, respectively. The vertices circled have "lightweight" star-like neighborhoods.

7. REFERENCES

[1] D. Agarwal, A. Z. Broder, D. Chakrabarti, D. Diklic, V. Josifovski, and M. Sayyadian. Estimating rates of rare events at multiple resolutions. In *KDD*, pages 16–25, 2007.

[2] L. Akoglu, M. McGlohon, and C. Faloutsos. OddBall:

Spotting anomalies in weighted graphs. In *PAKDD*, 2010.

[3] R. Albert, H. Jeong, and A.-L. Barabasi. Diameter of the world wide web. *Nature*, (401):130–131, 1999.

[4] L. Backstrom, D. P. Huttenlocher, J. M. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD*, pages 44–54, 2006.

[5] A.-L. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435:207–211, 2005.

[6] A. Belussi and C. Faloutsos. Estimating the selectivity of spatial queries using the 'correlation' fractal dimension. In *VLDB*, pages 299–310, 1995.

[7] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *SIGMOD*, pages 93–104, 2000.

[8] A. Z. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. L. Wiener. Graph structure in the web. *Computer Networks*, 33(1-6):309–320, 2000.

[9] H. Bunke, P. J. Dickinson, A. Humm, C. Irniger, and M. Kraetzl. Graph sequence visualisation and its application to computer network monitoring and abnormal event detection. In A. Kandel, H. Bunke, and M. Last, editors, *Applied Graph Theory in Computer Vision and Pattern Recognition*, volume 52 of *Studies in Computational Intelligence*, pages 227–245. Springer, 2007.

[10] D. Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *PKDD*, pages 112–124, 2004.

[11] D. Chakrabarti, S. Papadimitriou, D. Modha, and C. Faloutsos. Fully automatic cross-associations. In *KDD*, pages 79–88, 2004.

[12] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), 2009.

[13] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *KDD*, pages 153–162, 2007.

[14] W. Eberle and L. B. Holder. Mining for structural anomalies in graph-based data. In *DMIN*, 2007.

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

[16] G. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 35(3):66–71, 2002.

[17] F. Geerts, H. Mannila, and E. Terzi. Relational link-based ranking. In *VLDB*, pages 552–563, 2004.

[18] P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *SIGMOD*, pages 331–342, 1998.

[19] K. Henderson, T. Eliassi-Rad, S. Papdimitriou, and C. Faloutsos. HCDF: A hybrid community discovery framework. In *SDM*, 2010.

[20] S. Hirose, K. Yamanishi, T. Nakata, and R. Fujimaki. Network anomaly detection based on eigen equation compression. In *KDD*, pages 1185–1194, 2009.

[21] T. Idé and H. Kashima. Eigenspace-based anomaly detection in computer systems. In *KDD*, pages 440–449, 2004.

[22] B. Karrer, E. Levina, and M. E. J. Newman. Robustness of community structure in networks. *Phys. Rev. E*, 77(046119), 2008.

[23] J. M. Kleinberg. Bursty and hierarchical structure in streams. In *KDD*, pages 91–101, 2002.

[24] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *SIGCOMM*, pages 217–228, 2005.

[25] J.-G. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. In *ICDE*, pages 140–149, 2008.

[26] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM*, pages 556–559, 2003.

[27] C. Liu, X. Yan, H. Yu, J. Han, and P. S. Yu. Mining behavior graphs for "backtrace" of noncrashing bugs. In *SDM*, 2005.

[28] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[29] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *KDD*, pages 631–636, 2003.

[30] B. A. Prakash, N. Valler, D. Andersen, M. Faloutsos, and C. Faloutsos. BGP-lens: Patterns and anomalies in internet routing updates. In *KDD*, pages 1315–1324, 2009.

[31] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *KDD*, pages 374–383, 2006.

[32] V. Syrotiuk, K. Shaukat, Y. Kwon, M. Kraetzl, and J. Arnold. Application of a network dynamics analysis tool to mobile ad hoc networks. In *MSWiM*, pages 36–43, 2006.

[33] H. Tong, C. Faloutsos, and J.-Y. Pan. Random walk with restart: Fast solutions and applications. *Knowledge and Information Systems: An International Journal (KAIS)*, 14(3):327–346, 2008.

[34] M. Wang, N. H. Chan, S. Papadimitriou, C. Faloutsos, and T. Madhyastha. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. In *ICDE*, pages 507–516, 2002.

[35] D. Xin, J. Han, X. Yan, and H. Cheng. Mining compressed frequent-pattern sets. In *VLDB*, pages 709–720, 2005.