A Partitioning Approach to Scaling Anomaly Detection in Graph Streams

William Eberle
Department of Computer Science
Tennessee Technological University
Box 5101, Cookeville, TN, 38505
931-372-3278(office), 931-372-3686(fax)
weberle@tntech.edu

Lawrence Holder
School of Electrical Engineering and Computer Science
Washington State University
Box 642752, Pullman, WA 99164
509-335-6138(office), 509-335-3818(fax)
holder@wsu.edu

***Abstract*—Due to potentially complex relationships among heterogeneous data sets, recent research efforts have involved the representation of this type of complex data as a graph. For instance, in the case of computer network traffic, a graph representation of the traffic might consist of nodes representing computers and edges representing communications between the corresponding computers. However, computer network traffic is typically voluminous, or acquired in real-time as a stream of information. In previous work on static graphs, we have used a compression-based measure to find normative patterns, and then analyzed the close matches to the normative patterns to indicate potential anomalies. However, while our approach has demonstrated its effectiveness in a variety of domains, the issue of scalability has limited this approach when dealing with domains containing millions of nodes and edges. To address this issue, we propose a novel approach called *Pattern Learning and Anomaly Detection on Streams,* or PLADS, that is not only scalable to real-world data that is streaming, but also maintains reasonable levels of effectiveness in *detecting anomalies*. In this paper we present a partitioning and windowing approach that partitions the graph as it streams in over time and maintains a set of normative patterns and anomalies. We then empirically evaluate our approach using publicly available network data as well as a dataset that represents e-commerce traffic.**

Keywords- Graph-based, knowledge discovery, anomaly detection, streaming data.

I. INTRODUCTION

Institutions build data warehouses that store terabytes of information ranging from simple personal information to complex attributes that represent relationships between entities. Hellerstein calls it "the revolution of data" – so much data that even the term "big data" is being used to describe the phenomena [24]. Users then try to "mine" these databases for patterns to help them understand not only what the data means, but also discover those patterns that are unexpected or anomalous. Traditionally, methods for discovering patterns consist of "machine learning" approaches that use techniques such as classification, clustering, nearest neighbors, and statistics [19]. These methods have also focused mostly on the attributes of entities and ignored the *relationships* between entities.

Recent research efforts have involved the representation of this type of complex data as a *graph*, in order to analyze the relational *structure* in the data. This research has touched on a wide range of graph-theoretic approaches that have been applied to a wide variety of domains. While some successes have been demonstrated, they have either been specific to a particular data set, a particular type of graph, or a particular graph algorithm. In addition, they have not dealt with the scalability issues associated with "big data" when attempting to learn patterns and anomalies in data represented as a graph. For instance, in the case of computer network traffic, a graph representation of the traffic might consist of nodes representing computers and edges representing communications between the corresponding computers. In addition, other potential data sources for aiding in the analysis of the network traffic could include details about the individual users, location of the computer nodes, or even switch information. Adding these heterogeneous data sets to the network traffic, represented as a graph, could provide the basis for discovering interesting structural patterns and anomalies, which may alert a security analyst to the potential threat in the form of a network intrusion attempt, denial-of-service attack, or worms. However, computer network traffic is typically voluminous, or acquired in real-time as a *stream* of information. For example, CAIDA (www.caida.org) provides a data repository to the research community for the analysis of internet traffic [7]. In one example of network traffic collected by CAIDA, representing a dynamic denial-of-service (DDOS) attack at a single location, every second produced an average of 3,992 transactions, for a total of 2,395,234 transactions over a 10 minute span.

In order to lay the foundation for this effort, we hypothesize that a real-world, meaningful definition of a *graph-based anomaly* is an *unexpected deviation to a normative pattern*. Such anomalies are associated with illicit activity that tries to mimic normal behavior. Our initial approach to graph-based anomaly detection, called GBAD [13], used a compression-based measure to find normative patterns, and then analyzed the close matches to the normative patterns to determine if they meet the above definition of an anomaly. However, while this approach has demonstrated its effectiveness in a variety of domains [11], the issue of *scalability* has limited this approach when dealing with domains containing millions of nodes and edges. Furthermore, many graphs of interest are dynamic, i.e., changes to the graph are streaming in over time. This further complicates the analysis, because we cannot just analyze a static graph, but would need to analyze snapshots of the graph over time. However, this streaming graph scenario also offers an opportunity for methods that can update the current set of patterns and anomalies based on only the changes to the graph, rather than repeated analyses on the large graph snapshots.

To address this issue, we propose a novel approach called *Pattern Learning and Anomaly Detection on Streams,* or PLADS, that is not only scalable to real-world data that is streaming, but also maintains reasonable levels of effectiveness in *detecting anomalies*. We have developed a partitioning approach that partitions the graph as it streams in over time and maintains a set of normative patterns and anomalies biased toward the set of normative patterns found in the current "time window". In the next section, we discuss related work on graph-based anomaly detection and graph streams. We then present our proposed PLADS algorithm, followed by some empirical results on the CAIDA network data and synthetic e-commerce data.

II. Related Work

Early work by Cook and Noble [20] on anomaly detection in one large graph defined anomalies as structural outliers, i.e., after compressing the graph based on normative patterns, the remaining structure was considered anomalous. More recent work by Akoglu et al. [4] also addressed anomaly detection in one large graph, but their target was to identify only anomalous nodes. In addition, there has been work in the area of what is called "big graph mining", using approaches built on top of the MapReduce technology. One such approach called Pegasus, addresses the problem of graph mining using a distributed approach, but not in the context of anomaly detection [16]. Also, all of the above approaches assume a static graph.

One potential solution to handling very large graphs is to view the graph as a "stream" and processing the graph one, or a few edges, at a time. Previous work in this area has provided a few different approaches to handle *graph streams*. One approach is to use what is called a *semi-streaming model* as a way of studying massive graphs whose edge sets cannot be stored in memory. For example, Feigenbaum et al.'s work presents semi-streaming constant approximation algorithms for un-weighted as well as weighted matching problems, as well as an improvement for handling bipartite graphs [14]. By considering a set of classical graph problems in their semi-streaming model, they were able to demonstrate that certain approximations to the problems can be achieved. Other work has generalized this approach to different graph problems, such as the shortest paths in directed graphs, and used intermediate temporary streams as a means of resolving the space issues [9][3][22], or dealt primarily with bipartite graphs [23]. Basically, these approaches propose a tradeoff between the available internal memory and the number of passes it requires.

Early work also included the statistical analysis of data flows for the detection of anomalies. Duffield et al. limited the data flow, in order to deal with the issue of scalability, through sampling, which has the potential for missing anomalies [10]. Cormode et al. focused their work on the goal of heavy-change detection (i.e., drastic flow changes in a network), which misses the small changes/deviations characteristic when one is attempting to go unnoticed in their illicit behavior [8]. In addition, neither of these previous approaches deal with data represented as a graph.

Another approach is to examine the problem of *clustering* massive graph streams and use a technique for creating *hash-compressed micro-clusters* from graph streams [2]. Addressing the issues with large disk-resident graphs, the compressed micro-clusters are designed using a hash-based compression of the edges onto a smaller domain space.

Recently, others have attempted to mine frequent closed subgraphs in non-stationary data streams. One such approach called AdaGraphMiner, maintains only the current frequent closed graphs, utilizing estimation techniques with theoretical guarantees [6]. Empirical experiments have demonstrated the effectiveness of this approach on graph streams representing chemical molecules and structural representations of cancer data. In addition, there have been recent attempts to discover outliers in massive network streams. Using what is called a structural connectivity model, some researchers have attempted to handle the issue of sparseness in massive networks by dynamically partitioning the network [1], while others have exploited the graph structure of a network using various partitioning approaches [18]. Using techniques such as reservoir sampling methods that compress a graph stream, one can search for structural summaries of the underlying network. The goal of this type of outlier detection is to identify graph objects which contain unusual bridging edges, or edges between regions of a graph that rarely occur together.

However, all of the approaches so far have not addressed the issue of *scalability* associated with performing *graph-based anomaly detection* on "big data" graphs represented as streams. While some approaches have detected outliers in graph streams, their objective is to identify unusual clusters of subgraphs in the graph by analyzing the *statistical* nature of the existence of edges, as opposed to discovering anomalies in the *structure* of a graph, or graph stream. In addition, while some work has attempted to discover anomalous subgraphs using an ensemble-based approach [21] based on the GBAD approach [13], that type of approach does not address the issue of scalability.

III. Graphs

A graph is a set of nodes and a set of links, where each link connects either two nodes or a node to itself. More formally, we use the following definition.

***Definition**: A labeled graph G = (V,E,L) consists of three sets V, E and L. V is the set of vertices (or nodes), E is the set of edges (or links) between the vertices, and L is the set of string labels assigned to each of the elements of V and E.*

Much work has been done using *graph*-based representations of data. Using *vertices* to represent entities such as people, places and things, and *edges* to represent the relationships between the entities, such as friend, lives-in and owns, allows for a much richer expression of data than is present in the standard textual or tabular representation of information. Representing various data sets like telecommunications call records, financial information and social networks in a graph form allow us to discover *structural* properties in data that are not evident using traditional data mining methods.

A. Graph-Based Anomaly Detection

The idea behind our approach to graph-based anomaly detection is to find anomalies in graph-based data where the anomalous substructure in a graph is part of (or attached to or missing from) a *normative pattern*.

Definition: A substructure S_A is anomalous in graph G if $(0 < d(S_A,S) < T_D)$ and $(P(S_A|S) < T_P)$, where S is a normative pattern in G, T_D bounds the maximum distance (d) an anomaly S_A can be from the normative pattern S, and T_P bounds the maximum probability of S_A.

Definition: The anomalous score of an anomalous substructure S_A based on the normative substructure S in graph G as $d(S_A,S) * P(S_A|S)$, where the smaller the score, the more anomalous the substructure.

The distance between two graphs can be due to the addition, removal or modification of structure from one graph to the other. The probability of S_A given S is based on the frequency of S_A among all graphs within distance T_D of S. Therefore, the more anomalous substructure is that which is closer to the normative pattern and appears with lower probability. The importance of this definition lies in its relationship to any deceptive practices that are intended to illegally obtain or hide information. The United Nations Office on Drugs and Crime states the first fundamental law of money laundering as "The more successful money-laundering apparatus is in imitating the patterns and behavior of legitimate transactions, the less the likelihood of it being exposed" [15].

The advantage of *graph-based anomaly detection* is that the *relationships* between entities can be analyzed for structural oddities in what could be a rich set of information, as opposed to just the entities' attributes. However, graph-based approaches have been prohibitive due to computational constraints. Because graph-based approaches typically perform subgraph isomorphisms, a known NP-complete problem, in order to address this issue, most approaches use some type of heuristic to arrive at an approximate solution. However, this is still problematic, and in order to use graph-based anomaly detection techniques in a real-world environment, we need to take advantage of the structural/relational aspects found in dynamic, streaming data sets.

B. GBAD

The PLADS approach we are proposing is based on our previous work on static graph-based anomaly detection (GBAD) [13]. Here we briefly review the GBAD approach.

There are three general categories of structural anomalies in a graph: insertions, modifications and deletions. Insertions would constitute the presence of an unexpected vertex or edge. Modifications would consist of an unexpected label on a vertex or edge. Deletions would constitute the unexpected absence of a vertex or edge. GBAD discovers each of these types of anomalies. Using a greedy beam search and a minimum description length (MDL) heuristic, GBAD first discovers the best substructure, or normative pattern, in an input graph. The MDL approach is used to determine the *best substructure*(s) as the one that minimizes the following.

$$M(S,G) = DL(G\,|\,S) + DL(S)$$

where G is the entire graph, S is the substructure, $DL(G|S)$ is the description length of G after compressing it using S, and $DL(S)$ is the description length of the substructure.

The GBAD approach is based on the exploitation of *structure* in data represented as a graph. We have found that a structural representation of such data can improve our ability to detect anomalies in the behaviors of entities being tracked [12]. GBAD discovers anomalous instances of structural patterns in data that represent entities, relationships and actions. GBAD uncovers the relational nature of the problem, rather than solely the traditional statistical deviation of individual data attributes. Attribute deviations are evaluated in the context of the relationships between structurally similar entities. In addition, most anomaly detection methods use a supervised approach, requiring labeled data in advance (e.g., illicit versus legitimate) in order to train their system. GBAD is an *unsupervised* approach, which does not require any baseline information about relevant or known anomalies. In summary, GBAD looks for those activities that appear to match normal/legitimate/expected transactions, but in fact are *structurally* different.

Once GBAD finds a normative pattern and anomalies in a graph, it can then iterate to find additional anomalies. First, GBAD compresses the graph using the normative pattern, i.e., replacing each instance of the normative pattern with a single node with a new label. Then, GBAD is executed on this compressed graph to again find normative patterns and anomalies. This process can continue for multiple iterations to find more and more normative patterns, and anomalies to them, throughout the graph, and at different levels of abstraction as the graph is further compressed.

For more details regarding the GBAD algorithms, the reader can refer to [13].

C. Data

We evaluate GBAD and PLADS on two different datasets. The first, called the Berlin SPARQL Benchmark, is a synthetic generator of e-commerce data. The second dataset contains real network traffic and was obtained from the Center for Applied Internet Data Analysis (CAIDA).

The Berlin SPARQL Benchmark [5] is a suite of benchmarks for comparing the performance of various system architectures. The benchmark is built around an e-commerce use case in which a set of products is offered by different vendors and consumers have posted reviews about products. This synthetic e-commerce data can be scaled up to represent larger and denser volumes of traffic.

The Berlin data model contains the following classes: *Product, ProductType, ProductFeature, Producer, Vendor, Offer, Review,* and *Person*. (We refer the reader to the Berlin website for more details regarding the model.) We chose to represent this data as a graph where classes are nodes, associations are edges, and attributes are linked to their corresponding class (node). Figure 1 shows our graph representation, including, as shown in the lower right, that for each class (i.e., Product, ProductType, etc.) there are also edges and vertices representing a corresponding name, month, day, and year.

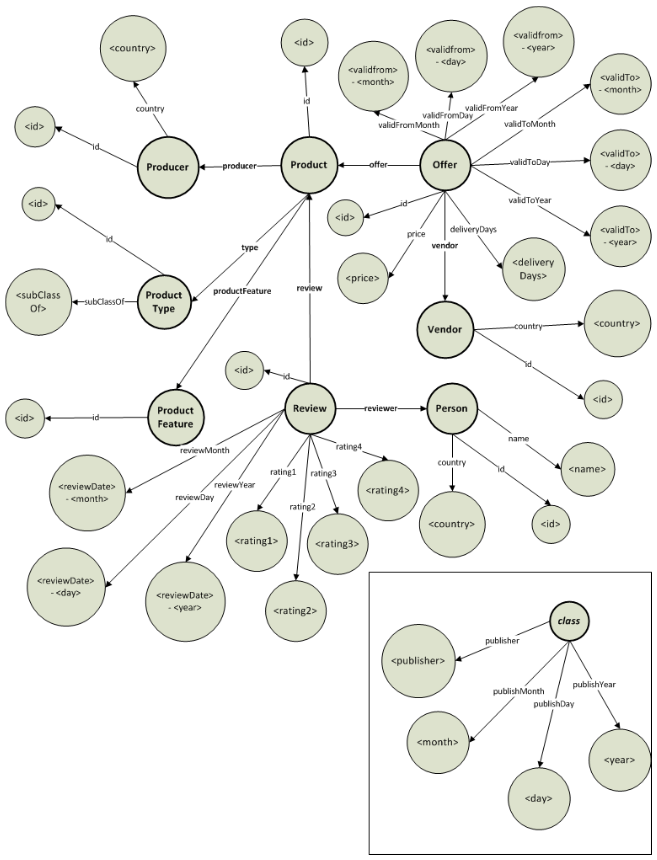

Figure 1. Graph representation of Berlin data.

Given that the "anomalies" inherent in the Berlin data are uninteresting, and do not represent the types of anomalies that one might find in real data of this type (i.e., products being sold and associated reviews), we "seed"ed the data with the following "real-world" anomalies: an invalid publisher scenario, and an invalid publication year scenario.

The Center for Applied Internet Data Analysis (CAIDA) is a publicly available resource for the analysis of IP traffic. Through a variety of workshops, publications, tools, and projects, CAIDA provides a forum for the dissemination of information regarding the interconnections on the internet. One of the core missions of CAIDA is to provide a data repository to the research community that will allow for the analysis of internet traffic and its performance (www.caida.org/data/). We will specifically use the CAIDA AS data set that represents the topology of the internet as the composition of various Autonomous Systems. Each of the AS units represents routing points through the internet (see example in Figure 2).

D. *GBAD Results*

First, using the CAIDA data set, we are able to represent the data as a graph composed of 24,013 vertices and 98,664 edges, with each AS depicted as a vertex and an edge indicating a peering relationship between the AS nodes. Figure 2 shows a portion of the AS graph, where the rectangle indicates the normative pattern and the emboldened edge indicates the anomalous structure found by GBAD.

This example also shows the advantage of using a graph-based approach on a complex structure. While the data indicates many provider/customer relationships, of which the norm is a particular AS being the provider to three different customers, the anomaly indicates an unusual connection between two ASes. Such an inconspicuous structure would probably be missed by a human analyst, and shows the potential of an approach like GBAD to find these anomalies in network traffic data. However, GBAD takes 14,259 seconds to discover the anomaly.

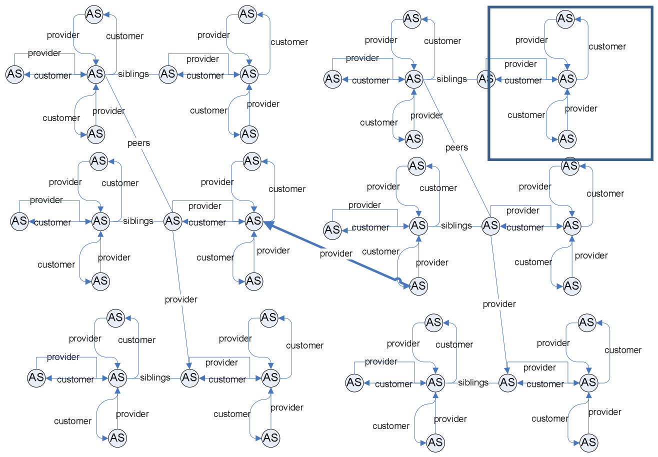

Figure 2. Normative pattern (square) and anomaly (bold) discovered in the CAIDA dataset.

We also ran GBAD on a single graph of the Berlin data that represents 100 products (35,994 vertices and 38,395 edges). GBAD reports the normative pattern shown in Figure 3 (left), and is able to successfully discover the anomalous substructure, as shown in Figure 3 (right). However, the running time is less than desirable, as it takes 5,630 seconds to complete.

IV. PATTERN LEARNING AND ANOMALY DETECTION ON STREAMS (PLADS)

The advantages associated with graph-based anomaly detection are well-documented, providing a myriad of approaches for discovering structural and relational anomalies. However, they have been limited to static domains, or data sets that are relatively small in size – certainly nothing on the order of what we would call "big data". Our preliminary experiments have shown that we *can*

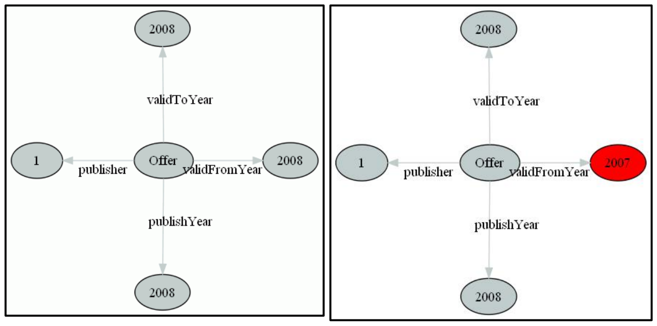

Figure 3. Normative pattern (left) and anomaly (right) in CAIDA data.

devise an approach whereby if we take into account smaller, individual partitions (i.e., a segment of the data that is processed individually in parallel with other partitions) *in terms of what we know* about other partitions, we can not only provide similar accuracy (i.e., precision, or relevance of reported anomalous substructures, and recall, or fraction of relevant substructures reported) but do it in a fraction of the time. In order to formalize our approach, we propose the PLADS algorithm, which accepts as input a set of *N* graph partitions either by partitioning a static graph, or fed in over time.

PLADS (input graph partitions)

```
1. Process $N$ partitions in parallel
   a. Each partition discovers top $M$ normative patterns.
   b. Each partition waits for all partitions to discover
      their normative patterns.
2. Determine best normative pattern $P$ among $NM$
   possibilities.
3. Each partition discovers anomalous substructures based
   upon $P$.
4. Evaluate anomalous substructures across partitions and
   report most anomalous substructure(s).
5. Process new partition
   a. If oldest partition(s) has exceeded a threshold $T$
      (based upon criteria such as the number of available
      partitions or the time-stamped-age of the partition),
      remove partition(s) from further processing.
   b. Determine top $M$ normative patterns from new
      partition.
   c. Determine best normative pattern $P'$ among all
      active partitions.
   d. If ($P' \neq P$), each partition discovers new anomalous
      substructures based upon $P'$.
   e. Else, only new partition discovers anomalous
      substructure(s).
   f. Evaluate anomalous substructures across partitions
      and report most anomalous substructure(s).
   g. Repeat.
```

This is a generic algorithm for applying graph-based anomaly detection methods to streaming data. The user can apply any normative pattern discovery techniques and any graph-based anomaly detection algorithms with this approach. For the purpose of demonstrating the PLADS approach, we use GBAD (defined earlier) for determining what are the *normative patterns* and what are the *anomalous substructures*.

The parameters to the PLADS algorithm are defined as follows:

N – number of partitions in the sliding window. This will be the initial number of graph partitions processed in parallel, and the number of partitions considered for determining the normative pattern and the substructures that are anomalous as each subsequent partition is processed.

M – number of normative patterns to retain. This will be the number of normative patterns saved from each graph partition to compare against other graph partitions.

It should be mentioned that the size of each partition (i.e., number of vertices and edges) is not necessarily the same. For instance, as we will show, we used the well-known graph-partitioning Metis tool [18] to create our graph partitions. While the Metis tool will create partitions of roughly equal-size, they are not usually exactly the same. Therefore, in the empirical results shown in the next section, we will experiment with (1) varying the number of partitions being processed – the more partitions that are created, the smaller their size, and (2) different values for the *N* and *M*. We can then analyze the effect that this has on both scalability as well as accuracy when it comes to detecting the anomalous substructures and reducing the false positives. Eventually, in the next phase of our streaming approach to graph-based anomaly detection, we will address the *automation* of partition size selection, whereby the system can determine when a partition should be processed based upon the speed at which data is streaming.

V. Empirical Results Using PLADS

In order to evaluate an *incremental* approach to graph-based anomaly detection, we apply the PLADS algorithm initially to the Berlin data set, and then to the CAIDA data set, eventually extending the number of edges in the real-world CAIDA data so as to demonstrate the scalability of our approach. All of our experiments are performed using a Debian 6.0, 64-bit server with 24GB of RAM and 8 CPUs.

A. Berlin Data

The Berlin data generator creates an XML file of products along with associated information, offers, and reviews. When you generate the XML file, you specify how many products you want to be generated, and all product, offers, and reviews are generated between the dates of March 15, 2008 and June 19, 2008. From this, we decide to create partitions based upon offer and review times (i.e., if a product has an offer or review), *where each partition* consists of all offers and reviews for products in a specified "time window" (e.g., partition 1 contains all product offers and reviews made on March 15th, partition 2 contains all product offers and reviews made on March 16th, etc.). We use this concept with the objective of not only verifying the feasibility of the PLADS approach, but also to apply the concept of graphs that are "streaming".

So, we test our PLADS partitioned approach on streaming graphs that represent 100 products and all of their associated information over the entire available time frame (i.e., March 15, 2008 to June 19, 2008). We create 100 partitions, where each partition represents a single day.

1. *Process N partitions in parallel.* We arbitrarily choose to process the first 5 (*N*) partitions of the graph. Running them in parallel, PLADS is able to process all of the partitions in under 2 seconds, each with 20 (*M*) normative patterns. This value of *N* is also used as the size of our processing window (i.e., how many partitions will be retained in memory). It should be noted that we will experiment with different values for *N* later, allowing us to analyze the effects of this choice.

2. *Determine best normative pattern P among NM possibilities.* The best substructure is different for each partition. Since each partition represents offers and reviews for a particular day, each original normative pattern differs by the *publishDay* value – i.e., on March 16th, the best substructure includes a *publishDay* value of "16"; on March 17th, the best substructure includes a *publishDay* value of "17", etc. However, across the *N* partitions, when PLADS analyzes the top *M* substructures in each partition for the best normative substructure among them, it discovers that the normative pattern *P* in Figure 4 (i.e., based upon month, which in this case is March) is the best.

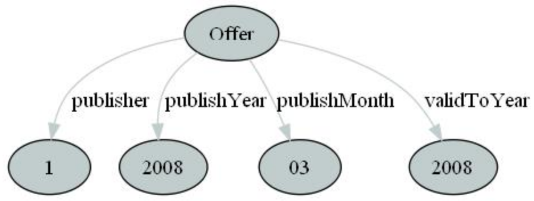

Figure 4. Normative pattern from first graph partition.

3. *Each partition discovers anomalous substructures based upon P.* Based upon the best substructure *P* from among all of the partitions, PLADS then searches for all anomalous substructures related to that normative pattern. The result is that no substructures are reported as anomalous across all of the partitions. This step takes less than 2 seconds when processing all partitions in parallel.

4. *Evaluate anomalous substructures across partitions and report most anomalous substructure.* Since no anomalous substructures are discovered in the previous step, there is no need to examine anomalous substructures across the partitions.

5. *Process new partition.* PLADS then processes one partition at a time, handling the data as a "sliding time window", removing the oldest partition and processing a new partition. We observe two behaviors when it comes to processing *N* partitions. First, if all partitions are from the same month (March, April, May or June), then the normative pattern (when evaluated across all of the *N* partitions) is similar to one shown in Figure 4, except with different values for the *publishMonth*. Second, when the partitions span two different months, while individual partitions report normative patterns dealing with the corresponding month, when evaluated across all of the current partitions, the normative pattern becomes the one shown in Figure 3. In other words, there are not enough instances from a particular month to be dominant, so the normative pattern associated with the entire set becomes the best *P*. Also, in terms of accuracy, only the targeted anomalous substructure is discovered – no false positives in this partition.

In addition to achieving the same accuracy as when GBAD was run on the entire graph, PLADS is able to process the partitioned graph in 486 seconds – an order of magnitude faster than the 5,630 seconds to process the entire graph.

B. CAIDA Data

We apply the PLADS approach to the same CAIDA data set used earlier when running GBAD on the entire graph. However, this time, we use the METIS tool [17] to partition the graph into 10 (arbitrarily) smaller, roughly equal-sized subgraphs. We also arbitrarily chose to initially process the first 5 (*N*) partitions of the graph. Running them in parallel, all of the partitions finish processing in 256 seconds, each with 5 (*M*) normative patterns (*Step 1*). PLADS then examines all of the partitions' normative patterns, searching for the best normative substructure among them (*Step 2*). The result is the normative pattern shown in Figure 5 (left), which is smaller than the normative pattern found when running on the entire graph (see Figure 2). Based upon the best substructure from among all of the partitions, we then search for all anomalous substructures related to that normative pattern (*Step 3*). The result is that 42 substructures are reported as anomalous across all of the partitions, with the longest running partition taking 252 seconds. PLADS then examines all of the reported anomalous substructures across the partitions, and the result is that 2 substructures are reported as equally anomalous (*Step 4*). However, at this point, in this experiment, neither of the substructures are the targeted anomalous substructures.

For the remaining graph partitions, PLADS iteratively analyzes the data by removing the oldest partition and processing a new partition – a sort of "sliding window" (*Step 5*). First, partition 1 is removed from consideration, and partition 6 is added to the sliding window. PLADS discovers the best substructure on the new partition, so that it can determine the best normative pattern among all of the remaining partitions. The result is the discovery of the normative pattern shown in Figure 2 (i.e., the same normative pattern from processing the entire graph) in 94 seconds. Since the normative pattern has changed since the last iteration (left-hand-side of Figure 5), PLADS processes each current partition in order to *re-discover* any anomalous substructures *based upon* the new normative pattern.

Examining all of the reported anomalous substructures across the partitions, PLADS discovers that the most anomalous substructure (found in partition 5) is the one that was identified by GBAD on the entire graph.

At the next iteration (partition 2 is removed and partition 7 is added), PLADS discovers in 48 seconds that the normative pattern has not changed (i.e., it is still the best substructure across all of the active partitions). In this case, only the new partition needs to be analyzed for any anomalous substructures, as the anomalies would not change for the already processed partitions. Analysis of the results from the new partition (partition 7) yields (in 109 seconds) no substructures more anomalous than what were already discovered.

Taking this scenario one more iteration (partition 3 is removed and partition 8 is added), PLADS discover that the best normative pattern across all of the partitions is different from the previous iteration (see right-hand-side of Figure 5). So, similar to two iterations back, all of the active partitions need to be re-evaluated based upon this new best substructure. The result is two new anomalous substructures. However, when PLADS compares their "anomalousness" to the one reported earlier (shown in Figure 2), it is discovered that there are more instances of this newly reported anomalous substructure, so the anomalous substructure discovered earlier is still the most anomalous.

After two more iterations of adding and removing partitions (i.e., processing all of the partitions that represented the single graph), the normative pattern stays the same, but partition 5 moves out of the sliding window and a new anomalous substructure is reported.

So, PLADS allows us to implement a graph-based anomaly detection approach on network data that is able to successfully discover the same anomalous substructure within an incremental approach in a fraction of the time (1358 seconds) it took to process the entire graph (14,259 seconds). Even the overhead associated with just managing the normative patterns and anomalous substructures, including the transferring of information between processing partitions, is negligible.

C. Scalability

While our results above demonstrate the ability of PLADS to maintain detection accuracy and reduce discovery times by an order of magnitude, in order to scale up to graph streams that may produce millions, if not billions, of edges, we decided to scale up our real-world data set by replicating the data that was provided by CAIDA. Table 1 shows results of experiments with scaled-up AS traffic.

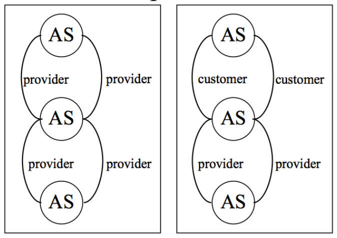

Figure 5. Normative pattern early (left) and later (right) in "stream".

Table 1. Results from runs using scaled-up CAIDA AS traffic

Edges	**GBAD time (secs)**	**Total parti-tions**	**N**	**PLADS time (secs)**	**PLADS rate (edges/sec)**
125,601	14,259	10	5	1,147	109.5
			6	1,030	121.9
			7	990	126.9
134,429		20	5	1,212	110.9
			6	1,075	125.1
			7	1,031	130.4
139,321		30	5	1,173	118.8
			6	1,021	136.4
			7	1,041	133.8
251,201	60,800	20	5	2,491	100.8
			6	2,444	102.8
			7	2,350	106.9
261,710		40	5	2,528	103.5
			6	2,556	102.4
			7	2,494	104.9
278,641		60	5	2,251	123.8
			6	2,291	121.6
			7	2,335	119.3
628,001	436,570	50	5	7,647	82.1
680,605		100	5	7,124	95.5
701,529		150	5	6,153	114.1
			6	5,561	126.2
			7	5,868	119.6
12,444,753	> 7 days	1,000	5	197,669	62.9
13,316,837		2,000	5	187,268	71.1
13,936,929		3,000	5	180,304	77.3
15,252,956		4,000	5	103,800	146.9

The results shown in Table 1 compare GBAD and PLADS. The table shows varying graph sizes, varying numbers of partitions (which varies the size of the partitions – i.e., the more partitions, the smaller the size of the partitions), and varying numbers of partitions retained in the sliding window (*N*). The "PLADS Rate" is calculated as the number of edges divided by the total running time. We can use that rate as an idea of how fast we can process data that is streaming, and not just the processing of individual graph partitions. From these results one can observe that the rate at which edges are processed grows linearly to the number of partitions and to the value of *N*. It should also be noted that we observed that varying the value for *M* does not have a noticeable effect on the PLADS performance. Figure 6 illustrates the scalability improvement of PLADS over GBAD for N=5 and the largest number of partitions.

While this particular work is attempting to address the *scalability* issues associated with graph-based anomaly detection, the anomaly detection accuracy of PLADS is the same as GBAD when applied to a single graph of all of the data. In all experiments, the targeted anomalous substructures are discovered, and are reported as being the *most anomalous* during their existence in the sliding window. If we handle the data as a stream, whereby a partition only has knowledge about its current partition as well as previous partitions, we achieve a false positive rate of 14%. However, if we evaluate ALL the partitions individually, and compare their discoveries once all the partitions have been processed, the result is a false positive rate of 6%.

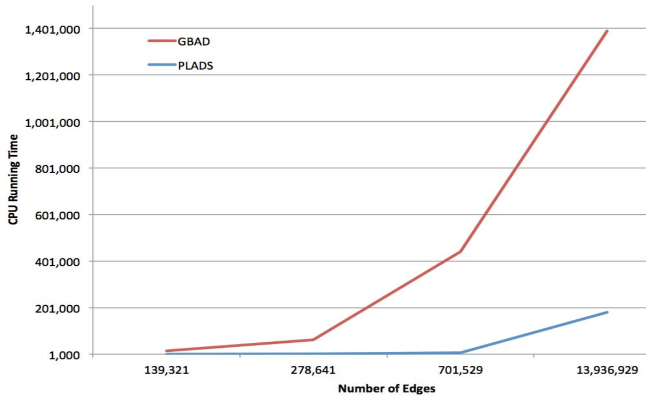

Figure 6: Scalability of GBAD vs. PLADS.

VI. Conclusions and Future Work

Handling large or streaming graphs provides the opportunity to handle complex data sets that are well-suited for graph-based approaches. We have proposed a method for analyzing graphs using a partitioned approach that can discover anomalous substructures – particularly ones that indicate potential criminal activity. We have also demonstrated the scalability of our approach with an order-of-magnitude improvement in the running-times of a graph-based anomaly detection approach. Using data from both cyber-threat and e-commerce scenarios, we have been able to discover anomalies with minimal false-positives using a partitioning approach to processing segments of the entire graph. In addition, we have demonstrated the ability to discover targeted anomalous substructures in graphs that have been scaled-up to represent potentially larger domains. In our next step, we will develop an incremental approach that processes only the stream of *graph changes* over time, where normative patterns and anomalies are updated only as necessary based on the impact of the changes. Going to a purely streaming approach we allow us to remove the "boundary issues" associated with anomalous substructures that could span graph partitions. We then hope to develop parallel implementations of these approaches to take advantage of high-performance computing platforms and further improve the scalability of the PLADS framework. In addition, while there are several different approaches to graph-based knowledge discovery and anomaly detection, the algorithm presented is not dependent on the GBAD approach. In future work, we hope to further examine this approach in a variety of other real-world, "big data" domains. In particular, we are going to evaluate our approaches on a Nokia mobile data set, as well as actual traffic flows collected from our institution-level network. Using these data sets will allow us to target an approach that can eventually "keep up" with real-world network traffic.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant Nos. 1318913 and 1318957. Support for CAIDA's Internet Traces is provided by the National Science Foundation, the US Department of Homeland Security, and CAIDA Members.

References

[1] Aggarwal, C., Zhao, Y. and Yu, P. 2011. Outlier Detection in Graph Streams. *ICDE*.

[2] Aggarwal, C., Zhao, Y. and Yu, P. 2010. On Clustering Graph Streams. *SIAM*.

[3] Aggarwal, G., Datar, M., Rajagopalan, S. and Ruhl, M. 2004. On the streaming model augmented with a sorting primitive. *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.

[4] Akoglu, L, McGlhon, M., and Faloustsos, C. 2010. OddBall: Spotting Anomalies in Weighted Graphs. *PAKDD*.

[5] Berlin SPARQL Benchmark http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/index.html.

[6] Bifet, A., Holmes, G., Pfahringer, B. and Gavaldà, R. 2011. Mining frequent closed graphs on evolving data streams. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (KDD '11).

[7] The CAIDA AS Relationships Dataset, http://www.caida.org/data/active/as-relationships.

[8] Cormode, G., & Muthukrishnan, S. 2004. What's New: Finding Significant Differences in Network Data Streams. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies* , pp. 1534–1545.

[9] Demetrescu, C., Finocchi, I., and Ribichini, A. 2006. Trading Off Space for Passes in Graph Streaming Problems. *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA*).

[10] Duffield, N., Lund, C., & Thorup, M. 2002. Properties and prediction of flow statistics from sampled packet streams. In *Proceedings of the 2nd ACM SIGCOMM* , pp. 159–171.

[11] Eberle, W.. Holder, L., and Graves, J. 2011. Insider Threat Detection Using a Graph-based Approach. *Journal of Applied Security Research*, Volume 6, Issue 1, pp. 32-81.

[12] Eberle, W., Holder, L. and Cook, D. 2009. Identifying Threats Using Graph-Based Anomaly Detection, in: *Machine Learning in Cyber-Trust*, J. Tsai and P. Yu (Editors), Springer.

[13] Eberle, W. and Holder, L. 2007. Anomaly Detection in Data Represented as Graphs. *Intelligent Data Analysis.* Vol. 11, No. 6.

[14] Feigenbaum, J., Kannan, S., McGregor, A., Suri, S. and Zhang, J. 2005. On Graph Problems in a Semi-Streaming Model. *Theoretical Computer Science*.

[15] Hampton, M. and Levi, M. 1999. Fast spinning into oblivion? Recent developments in money-laundering policies and offshore finance centres. *Third World Quarterly*, 20(3): 645-656.

[16] Kang, U. and Faloutsos, C. 2012. Big Graph Mning: Algorithms and Discoveries. *SIGKDD Explorations*, Volume 14, Issue 2.

[17] Karypis, G. 2011. METIS: A Software Package for Partitioning Unstructured Graphs, Version 5.0, August 4, 2011.

[18] Leskovec, J., Lang, K., Dagupta, A., and Mahoney, M. 2008. Statistical properties of community structure in large social and information networks. *WWW*, pages 695-704.

[19] Mitchell, T. 1997. Machine Learning. *McGraw Hill.*

[20] Noble, C. and Cook, D. 2003. Graph-Based Anomaly Detection. *SIGKDD.*

[21] Parveen, P., Evans, J., Thuraisingham, B., Hamlen, K., Khan, L. 2011. Insider Threat Detection Using Stream Mining and Graph Mining. *Privacy, Security, Risk and Trust (PASSAT), IEEE International Conf. on Social Computing (SocialCom)*, pp.1102-1110.

[22] Sarma, A., Gollapudi, S. and Panigrahy, R. 2008. Estimating PageRank on Graph Streams. *AMC PODS*.

[23] Jimeng, S., Huiming, Q., Chakrabarti, D., & Faloutsos, C. 2005. Neighborhood formation and anomaly detection in bipartite graphs. *IEEE International Conference on Data Mining, ICDM,* pp. 418–425.

[24] Weinberger, D. 2010. Data, Data Everywhere. *The Economist*, February 27, 2010.