2013 IEEE 13th International Conference on Data Mining Workshops

Incremental Anomaly Detection in Graphs

William Eberle
Department of Computer Science
Tennessee Technological University
Cookeville, TN USA
weberle@tntech.edu

Lawrence Holder
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA USA
holder@wsu.edu

***Abstract*— The advantage of graph-based anomaly detection is that the relationships between elements can be analyzed for structural oddities that could represent activities such as fraud, network intrusions, or suspicious associations in a social network. However, current approaches to detecting anomalies in graphs are computationally expensive and do not scale to large graphs. 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 this work, we describe methods for graph-based anomaly detection via graph partitioning and windowing, and demonstrate their ability to efficiently detect anomalies in data represented as a graph.**

Keywords- Anomaly detection, graph mining, dynamic graphs.

I. INTRODUCTION

Recent research efforts have involved the representation 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. More importantly, 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 [1]. 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.

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. In a previous approach to graph-based anomaly detection, called GBAD [2], we 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 [3], 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. We have developed such a method for pattern learning and anomaly detection in streams (PLADS) depicted in Figure 1. In this paper we describe the PLADS approach and demonstrate its effectiveness and scalability for large datasets.

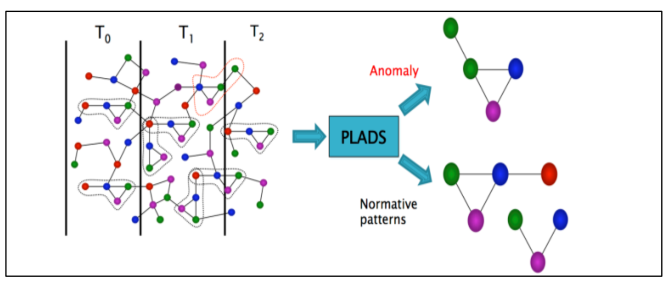

Figure 1. Network monitoring scenario for PLADS. Information about entities and relationships streams in over time, and PLADS maintains a current set of normative patterns and anomalies.

II. GRAPH-BASED ANOMALY DETECTION

A graph is a set of nodes and a set of edges, where each edge 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 the set V of vertices (or nodes), the set E of edges (or links) between the vertices, and the set L 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.

The idea behind the GBAD approach to graph-based anomaly detection is to find anomalies in graph-based data where the anomalous substructure (or subgraph) in a graph is part of (or attached to or missing from) a *normative pattern*. We assume a measure $d(G_1,G_2)$ of the distance between two graphs G_1 and G_2. Several such measures have been developed, but we focus on a graph edit distance measure that computes the cost (number of additions, deletions and changes of a node, edge or label) of transforming G_1 into a graph that is isomorphic to G_2.

Definition: *A substructure S_A is anomalous in graph G if $(0 < d(S_A,S) < T_D)$ and $(P(S_A|S,G) < T_P)$, where S is a normative pattern in G, T_D bounds the maximum distance 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 is $d(S_A,S) * P(S_A|S,G)$, 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 and G 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 obtain or hide information. The United Nations Office on Drugs and Crime states the first fundamental law of money laundering as "The more successful a money-laundering apparatus is in imitating the patterns and behavior of legitimate transactions, the less the likelihood of it being exposed" [4].

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, 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.

III. GBAD

The PLADS approach is based on previous work on static graph-based anomaly detection (GBAD) [2]. Here we briefly review the GBAD approach. There are three general *categories of 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 minimum description length (MDL) approach is used to determine the *best substructure*(s) as the one that minimizes the following:

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

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. Previous work found that a structural representation of such data can improve one's ability to detect anomalies in the behaviors of entities being tracked [6]. 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 newly-labeled node. 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 [2].

IV. Initial Observations

Take the example of a cyber-security threat where there is the leaking of information by employees with access to confidential and sensitive information. One of the Visual Analytics Science and Technology (VAST) 2009 mini-challenges involved various aspects of a fictional insider threat scenario where someone is leaking information [7]. The goal of these challenges is to allow contestants to apply various visual analysis techniques to discover the spy and their associated actions. The VAST data set consists of the activities (card swipes and network traffic) of 60 employees at an embassy over the month of January in 2008.

Starting with a graph of embassy employee activity data on January 29, 2008, consisting of 180 transactions (5,058 vertices and 4,878 edges), we *randomly inserted an extra edge and vertex* into the graph, representing a *potentially* anomalous *insertion*. We then ran GBAD on the entire graph, targeting anomalous insertions. This results in the targeted anomaly being discovered in 2,364 seconds with no false positives. The normative pattern (shown in Figure 2) consists of 8 vertices and 7 edges. We then divided the graph into 10 graph partitions, where each partition consists of 18 transactions, and ran GBAD on each *partition* individually. This results in a much shorter running time, with the longest running partition taking only 215 seconds. However, while the targeted anomaly is discovered, it also results in 189 false positives being reported.

In these experiments, the normative pattern is the same across all partitions. This leads us to make three observations. First, if we know the total number of instances of the best substructure, the targeted anomalous substructure would have a similar anomalous score as the reported most anomalous substructure. Second, and even more important, if we know the numbers of instances of each anomalous substructure across all of the partitions, our targeted anomalous substructure would come out on top by itself. And third, if we keep track of the best/most anomalous score across the partitions, we would be able to remove some false positives.

In short, if we can effectively detect anomalies across multiple graphs, we can more efficiently handle not only very large graphs that are static (by partitioning them into multiple smaller graphs), but also graphs that represent a continuous *stream* of information.

V. Related Work

Early work by Cook and Noble [8] 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. [9] also addressed anomaly detection in one large graph, but their target was to identify only anomalous nodes. Both of the above approaches assumed 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 [10]. 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 [11][12][13]. Basically, these approaches propose a tradeoff between the available internal memory and the number of passes it requires.

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 [14]. 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 [15]. 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 [16]. 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*. 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 [17] based on the GBAD approach [2], that type of approach does not address the issue of scalability.

VI. An Approach To Graph-Based Anomaly Detection on Partitions

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* 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 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.

A. Experiments Using Cyber-Security Example

First, we will show the PLADS algorithm applied to a subset of the cyber-security insider threat data presented earlier. In this experiment, we analyze just the movements of the employees throughout the embassy over the specified month of January in 2008. This set consists of card swipes as employees enter various rooms in the embassy.

1) GBAD

As input to GBAD, the data is represented as a graph, composed of 39,331 vertices and 38,052 edges, where movement, building, and type of room are depicted as vertices and edges indicating direction and movement between rooms. The normative pattern for this graph is depicted in Figure 2. After running GBAD on the entire graph, two anomalous substructures are discovered (one of the substructures is shown in Figure 2). However, it took 14,347 seconds to discover the anomalous substructure when analyzing the entire graph.

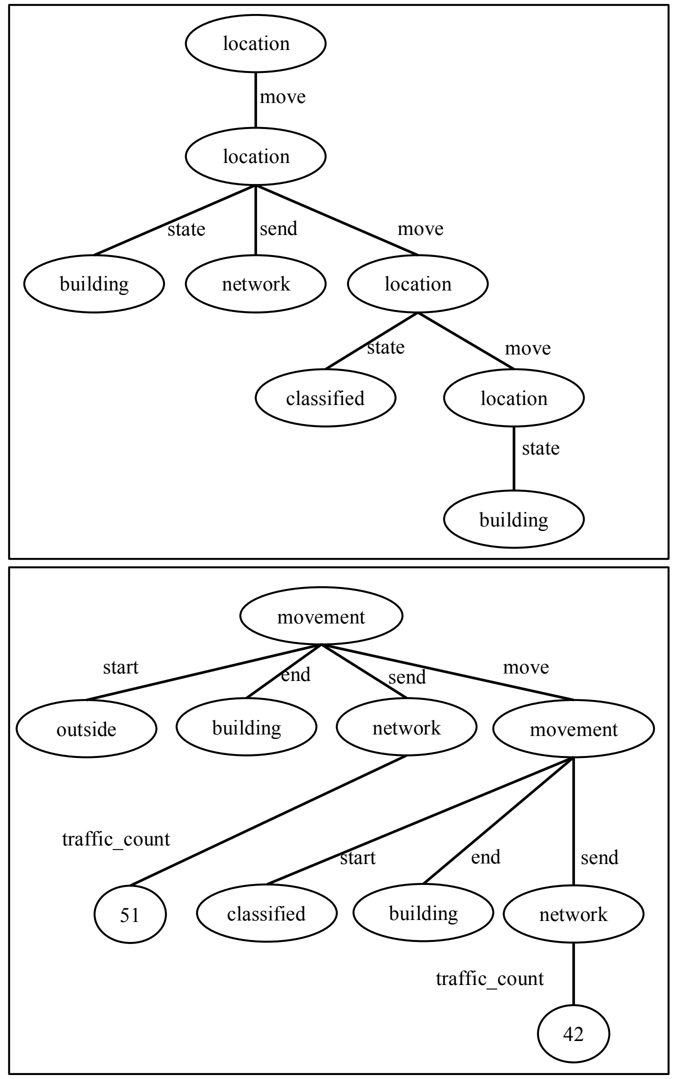

Figure 2. Normative pattern (top) and anomalous substructure (bottom) discovered in the VAST embassy insider challenge dataset.

2) PLADS

We applied the PLADS approach to the same dataset, divided into ten equal-sized partitions. We arbitrarily chose to initially process the first 5 (N) partitions of the graph. Running them in parallel, all of the partitions finish processing in 293 seconds, each producing 3 (M) normative patterns. We then examine all of the partitions' normative patterns, searching for the best normative substructure among them (i.e., the substructure that maximizes the value of size * frequency). The result is a normative pattern P identical to the normative pattern shown in Figure 2. Next, each partition discovers anomalous substructures based upon P. Only 1 substructure is reported as anomalous across all of the partitions, with the longest running partition taking 328 seconds. Since there is only one anomalous substructure reported, evaluation is trivial. (It should also be noted that this is one of the targeted anomalous substructures discovered when the graph was processed in its entirety.)

Processing data as streams can be handled in two ways. Either we can always remove the oldest partition, or we can remove any partitions that are older than some time threshold T (i.e., a sliding window). For this example, we will do the former, removing the oldest partition and processing a new partition (e.g., removing partition 1 and processing partition 6). We then discover the best substructure in the new partition, so that we can determine the best normative pattern among all of the remaining partitions. However, while the reported normative pattern in partition 6 is different, it is not better than the best substructure reported by the other five partitions. So, we use the same best substructure on partition 6, and no anomalous substructures are discovered (in 106 seconds). Also, since we are using the best substructure from a previous iteration, we do not have to re-discover any anomalous substructures in the older partitions.

At the next iteration (partition 2 is removed and partition 7 is added), we discover that the normative pattern has not changed (i.e., it is still the best substructure across all of the active partitions). Again, 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 257 seconds) no anomalous substructures. This same behavior continues over partitions 8 and 9, using 207 and 301 seconds respectively. However, on partition 10, the same best substructure is reported, but a new anomalous substructure is reported of equal "anomalousness" (in 501 seconds) to the substructure discovered in partition 3. This happens to be the second anomalous substructure discovered when the entire, non-partitioned graph was processed.

So, we are able to implement a graph-based anomaly detection approach on data that represents movements of people, and successfully discover the same two anomalous substructures (with no false positives) within a streaming approach in a fraction of the time (1,993 seconds) it took to process the entire graph (14,347 seconds). However, this graph is rather sparse (i.e., few edges compared to the number of vertices). Next we will examine results on a denser graph that also represents data that can be streamed.

B. *Experiments Using Network Traffic Data*

The Cooperative Association for 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/). Using GBAD, we analyzed the CAIDA AS (Autonomous Systems) data set for normative patterns and possible anomalies [1]. The AS data set represents the topology of the internet as the composition of various Autonomous Systems. Each of the AS units represents routing points through the internet.

1) *GBAD*

We 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 3 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 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 took 59,743 seconds to discover the anomaly.

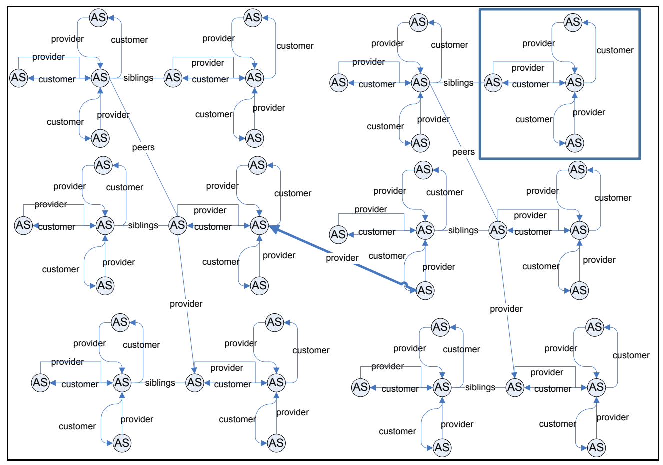

Figure 3. Normative pattern (square) and anomaly (bold) discovered in the CAIDA dataset.

2) *PLADS*

In order to demonstrate the potential effectiveness of an *incremental* approach to graph-based anomaly detection, we apply the PLADS algorithm to this same CAIDA data set divided into ten equal-sized partitions. We again arbitrarily chose to initially process the first 5 (*N*) partitions of the graph. Running them in parallel, all of the partitions finish processing in 210 seconds, each with 3 (*M*) normative patterns. We then examine all of the partitions' normative patterns, searching for the best normative substructure among them. The result is the normative pattern shown in Figure 4, which is smaller than the normative pattern found when running on the entire graph (see Figure 3). Based upon the best substructure from among all of the partitions, we then search for all anomalous substructures related to that normative pattern. The result is that 166 substructures are reported as anomalous across all of the partitions, with the longest running partition taking 112 seconds. We then examine all of the reported anomalous substructures across the partitions, and the result is that 2 substructures are reported as equally anomalous. However, at this point, neither of the substructures are the targeted anomalous substructures.

Similar to the previous example, we handle the data incrementally by removing the oldest partition and processing a new partition. We then discover the best substructure on the new partition, so that we can determine the best normative pattern among all of the remaining partitions. The result is the discovery of the normative pattern shown in Figure 3 (i.e., the same normative pattern from processing the entire graph) in 92 seconds. Since the normative pattern has changed since the last iteration, we have each partition *re-discover* any anomalous substructures *based upon* the new normative pattern. Examining all of the reported anomalous substructures across the partitions, we discover that the most anomalous substructure (found in partition 5) is the one that was identified when we ran GBAD on the entire graph.

At the next iteration (partition 2 is removed and partition 7 is added), we discover in 45 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 58 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), we discover that the best normative pattern across all of the partitions is different from the previous iteration (see Figure 4). 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, if you compare their "anomalousness" to the one reported earlier (shown in Figure 3), one can see that there are more instances of this newly reported anomalous substructure, so the anomalous substructure discovered earlier would still be 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 new normative pattern stays the same, and the anomalousness of reported substructures lessens (i.e., becomes more common), still leaving us with the targeted anomalous instance.

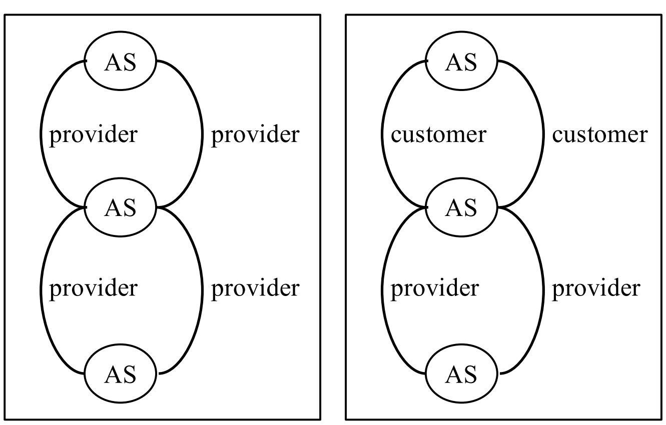

Figure 4. Normative pattern early (left) and later (right) in the "stream".

So, we are able 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 (642 seconds) it took to process the entire graph (59,743 seconds). Even the overhead associated with comparing normative patterns and anomalous substructures across partitions is negligible, as the number of substructures to evaluate from each partition is minimal.

C. *Experiments Using Larger Synthetic Graphs*

While the previous real-world data set experiments analyze some interesting scenarios, the data sets are relatively small and the ability to control the anomalies is limited. So, in order to validate our approach on larger graphs and vary the substructures, we used a synthetic graph generator to generate a sparse graph of ~2M vertices and edges. While not on the order of what most would define as "big data" [18], where graphs consist of billions of nodes, processing partitions of this size using the PLADS approach could quickly add up to this scale.

1) GBAD

The graph consists of a specified normative subgraph of 10 vertices and 9 edges, with random substructures of varying levels of anomalousness (i.e., frequency of their existence) injected into the graph. After running GBAD on the complete graph, the anomalous substructure, consisting of an unexpected edge and vertex, is discovered as shown in Figure 5 (attached to the normative pattern). However, the normative pattern and anomalous substructure are discovered in 276,873 seconds (i.e., over 3 days) - hardly useful in a real-world environment.

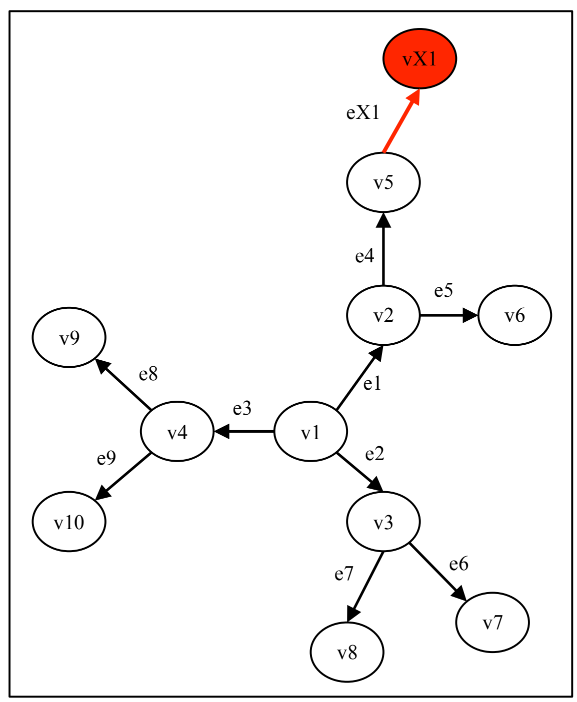

Figure 5. Normative pattern and anomalous substructure in synthetic graph.

2) Streaming GBAD

Again, to demonstrate the potential effectiveness of a partition-based incremental *streaming* approach to graph-based anomaly detection, we apply the PLADS algorithm to this same synthetic graph. For this example, we have divided the original graph into 100 partitions, where each partition consists of approximately 19,000 vertices and edges. We initially process the first 20 (*N*) partitions of the graph. The choice of an initial 20 partitions is somewhat arbitrary, as our goal is to just get a representative sample with which to start analyzing. Running them in parallel, all of the partitions finish processing in 9 seconds, each with 3 (*M*) normative patterns. It should be noted that even if the partitions are not processed in parallel, it only takes 136 seconds to process the 20 partitions serially. However, we will also use *T*=20 as the size of our processing window (i.e., partitions retained in memory).

We then examine all of the partitions' normative patterns, searching for the best normative substructure among them. The result is the normative pattern shown in Figure 5, which is the same (targeted) normative pattern found when running on the entire graph. Based upon the best substructure from among all of the partitions (previous step), we then search for all anomalous substructures related to that normative pattern. The result is that only 2 anomalous instances (from partition 17) are reported as anomalous across all of the partitions. At this point, the anomalous substructure is not the targeted anomalous substructure.

Processing partitions 21-33 results in no new normative patterns or anomalies, at a total processing time of 92 seconds. Processing partition 34 does not report a new normative pattern, but 2 instances of an anomalous substructure are discovered. The new anomalous substructure is evaluated against the current best anomalous substructure, and it is discovered to be the same anomalous substructure (albeit, still not the targeted anomaly at this point). Thus, we now have 4 instances of the current best anomalous substructure. This step takes 8 seconds.

Processing partitions 35-57 results in no new normative patterns or anomalies, at a total processing time of 165 seconds. Processing partition 58 does not report a new normative pattern, but 3 instances of a new anomalous substructure are discovered. The anomalous substructure is evaluated against the current best anomalous substructure, and is found to be different and more anomalous. Thus, this new substructure becomes the current best anomalous substructure. This step takes 7 seconds, and is still not the targeted anomaly.

Processing partitions 59-78 results in no new normative patterns or anomalies, at a total processing time of 141 seconds. Processing partition 79 does not report a new normative pattern, but 3 instances of an anomalous substructure are discovered. The anomalous substructure is evaluated against the current best anomalous substructure, and it is discovered to be the same anomalous substructure. Thus, we now have 6 instances of the current best anomalous substructure. This step takes 7 seconds. It is also interesting to note that in terms of "global anomalousness", our visibility is limited to the partitions that are retained in memory. If we could compare the current best anomalous substructure to the one discovered in partition 17 (as well as partition 34), the older substructures would be more anomalous.

Processing partitions 80-84 results in no new normative patterns or anomalies, at a total processing time of 35 seconds. Processing partition 85 does not report a new normative pattern, but 1 instance of an anomalous substructure is discovered. The anomalous substructure is evaluated against the current best anomalous substructure, and is found to be different and more anomalous. Thus, this new substructure, *which is the targeted anomaly discovered when analyzing the entire graph* (shown in Figure 5), becomes the current best anomalous substructure. This step takes 7 seconds. Processing partitions 86-100 results in no new normative patterns or anomalies, at a total processing time of 103 seconds.

Again, we are able to implement a graph-based anomaly detection approach on a larger graph that is able to successfully discover the same targeted anomalous substructure within a streaming approach in a fraction of the time (574 seconds) it took to process the entire graph (276,873 seconds). Timings for each partition are shown in Figure 6, with partitions containing anomalous substructures shown in red at the top. As noted earlier, the size of the window (i.e., the number of partitions retained in memory) does affect what anomalous substructures are discovered. At the end of processing all 100 partitions, the targeted anomalous substructure is reported as the best. However, if the targeted anomalous substructure was in an early partition, a later partition may report a less anomalous substructure (as opposed to when we process the non-partitioned graph), because the targeted anomalous substructure will have fallen out of the window when *T*=20.This fits into the idea of *concept drift* when handling data as a stream, whereby subgraphs that are reported as anomalous may lessen in their "anomalousness" as new information is received.

Running this same experiment using different values for *T* (the initial number of partitions, *N*, is set to the same value), we discover that:

- *T*=1: the newest partition always has the current best anomalous substructure (if any).
- *T*=5: due to the scarcity of injected anomalous substructures, only one partition in the window contains an anomalous substructure, leaving the newest partition to always have the current best anomalous substructure.
- *T*=10: when the targeted anomalous substructure is discovered at partition 85, within the window is the previous best anomalous substructure, which would be replaced by the targeted anomalous substructure because it is more anomalous (score wise). (Same results for *T*=15.)

In addition to accuracy, the running times are also slightly affected by the change in *T*, as shown in Figure 7. We see that total running time decreases as the window size *T* increases due to the increased ability to exploit parallelism across

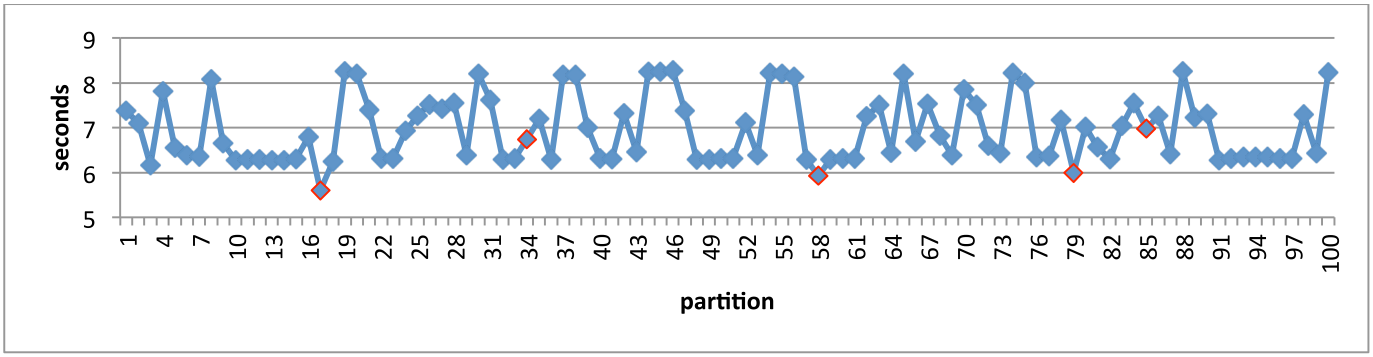

Figure 6. Partition timings, where highlighted partitions indicate newly-discovered anomalies.

partitions in the window. Thus, *T* (and *N*) is bound by the number of processors available. While there is some overhead associated with having to compare substructures across partitions, it is minimal from a run-time perspective as well as system resources. Also, the value chosen for *M* (number of normative patterns) can be increased with minimal impact to performance. From a user perspective, one must determine how truly "normative" is a pattern that is not in the top *M*.

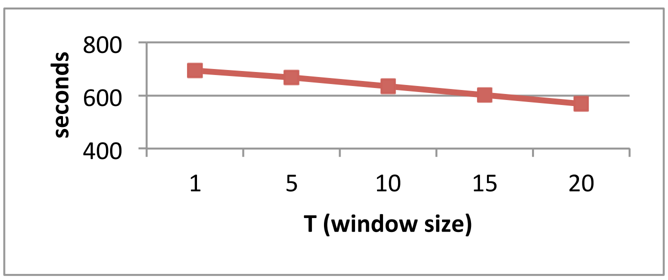

Figure 7. Running times on synthetic graph based upon window size (T).

VII. 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 parallel partitioning approach that can discover anomalous substructures. 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 real-world data from a cyber-threat scenario, and actual network traffic between autonomous systems, we are able to discover the anomalies with minimal false-positives using a parallel partitioning approach to processing segments of the entire graph. While there are several different approaches to graph-based knowledge discovery and anomaly detection, the algorithm presented is not dependent on a single approach. In future work, we will examine the scalability of such an approach to "big data" sizes as well as high-speed streams. In addition, we would like to further examine this approach in a variety of other real-world domains, such as social networks.

Acknowledgment

Support for CAIDA's Internet Traces is provided by the National Science Foundation, the US Department of Homeland Security, and CAIDA Members.

References

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

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

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

[4] M. Hampton and M. Levi, "Fast spinning into oblivion? Recent developments in money-laundering policies and offshore finance centres," Third World Quarterly, 1999, Vol. 20, Num. 3, p. 645-656.

[5] B. McLean and P. Elkind, "The Smartest Guys in the Room: The Amazing Rise and Scandalous Fall of Enron," Portfolio Trade Publishing, 2004.

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

[7] W. Eberle, L. Holder and J. Graves, "Detecting Employee Leaks Using Badge and Network IP Traffic," IEEE Symposium on Visual Analytics Science and Technology, October 2009.

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

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

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

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

[12] G. Aggarwal, M. Datar, S. Rajagopalan, and Ruhl, M., "On the streaming model augmented with a sorting primitive," Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2004.

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

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

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

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

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

[18] U. Kang and C. Faloutsos (2012). Big Graph Mining: Algorithms and Discoveries. ACM SIGKDD Explorations, 14(2):29-36.