Streaming Data Analytics for Anomalies in Graphs

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

Lawrence Holder
Washington State University
Box 642752, Pullman, WA 99164-2752
509-335-6138 (phone)
509-335-3818 (fax)
holder@wsu.edu

Abstract— **Protecting our nation's infrastructure and securing sensitive information are critical challenges for both industry and government. Due to the *complex* and *diverse* nature of the environments which can expose attacks or terrorism activity, one must not only be able to deal with attacks that are *dynamic*, or constantly changing, but also take into account the *structural* aspects of the networks and the relationships among communication events. However, analyzing a massive, ever-growing graph will quickly overwhelm currently-available computing resources. One potential solution to the issue of handling very large graphs is to handle data as a "stream". In this work, we present an approach to processing a stream of changes to the graph in order to efficiently identify any changes in the normative patterns and any changes in the anomalies to these normative patterns without processing all previous data. The overall framework of our approach is called PLADS for Pattern Learning and Anomaly Detection in Streams. We evaluate our approach on a dataset that represents people movements and actions, as well as a scalable, streaming data generator that represents social network behaviors, in order to assess the ability to efficiently detect known anomalies.**

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

I. INTRODUCTION

Detecting potential attacks to our nation's infrastructure, whether it is via insider threats or social threats to the populace are critical challenges for industry as well as our government. With the plethora of information that is transmitted in all aspects of today's culture, physical as well as environmental attacks pose serious consequences to individuals, corporations, governments, and society as a whole. A good example is the spread of the Ebola virus, where the potential spread due to contact is becoming exponentially difficult to follow, and could potentially become difficult to manage both organizationally as well as financially [1].

In order to address the issue of analyzing complex networks for patterns and anomalies, one must provide methods of monitoring and rapidly detecting pattern changes and any associated anomalies. However, due to the *complex* and *diverse* nature of these networked environments, one must not only be able to deal with threats that are *dynamic*, but also take into account the *structural* aspects of the networks and the relationships among communication events.

In previous work, we represented networks and various security related information using a graph and developed the graph-based anomaly detection (GBAD) approach that was able to detect anomalies with high accuracy and low false positive rates. However, the GBAD approach required the analysis of a graph containing all information up to a current point in time. Analyzing a massive, ever-growing graph will quickly overwhelm currently-available computing resources. One potential solution to the issue of handling very large graphs, and one we explore in this work, is to handle data as a "stream". Instead of processing an entire graph, which represents a complete set of data (or at least a very large portion of the data), one can process the graph a few edges at a time. Recent work in this area has focused on finding patterns in streams of small, independent graph transactions or on outliers in global graph properties. In this work, we present an approach to processing a stream of changes to the graph in order to efficiently identify any changes in the normative patterns and any changes in the anomalies to these normative patterns without processing all previous data.

The overall framework of our approach is called PLADS for Pattern Learning and Anomaly Detection in Streams. We evaluate our approach on two different synthetic data sets. The first represents the movements and actions of employees at an embassy where an insider threat activity is occurring. We will use this dataset to present our approach on heterogeneous data where there is anomaly ground-truth, and partition the data so as to simulate a streaming approach. The second data source is a streaming social network generator. The generator will allow us to scale the amount of posts, likes, etc. for users that are static (i.e., do not change), and social network activity that is dynamic (i.e., streaming). These data sources will not only allow us to evaluate the accuracy of detecting anomalies, but also, because of the data volume, the *scalability* of our methods. In addition, using these diverse data sets will allow us to demonstrate a general approach to structural anomaly detection that could be applied to a wide array of relevant security threat scenarios.

II. GRAPH-BASED ANOMALY DETECTION

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 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.*

978-1-4799-1737-2/15/$31.00 ©2015 IEEE

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, GPS movements 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 our 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*.

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 as $d(S_A,S) * P(S_A|S,G)$, where the smaller the score, the more anomalous the substructure.*

The distance between two graphs can be due to a difference in 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 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. For instance, take the example situation that occurred at the Enron Corporation [2]. Using anomaly detection on graphs that represent e-mail correspondences (Figure 1), such as those between executives at Enron, might help in the prevention of lost revenues, pensions, and jobs. However, graph-based approaches have been prohibitive due to computational constraints. Because graph-based approaches typically perform subgraph isomorphisms, 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*.

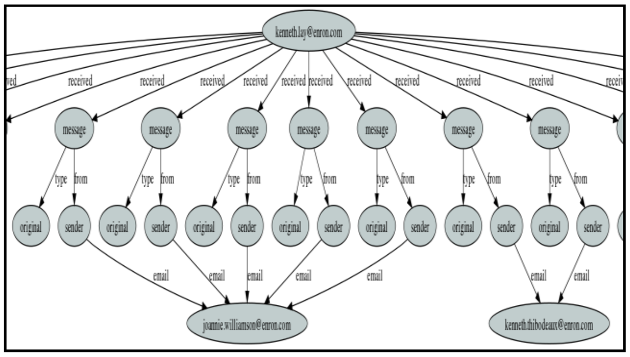


Figure 1. Example partial graph of Enron e-mail correspondences.

III. GBAD

The PLADS approach is based on our previous work on static graph-based anomaly detection (GBAD) [3]. 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) (i.e., normative pattern) 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. Using a beam search (a limited length queue of the best few patterns that have been found so far), the algorithm grows patterns one edge at a time, continually discovering what substructures best compress the description length of the input graph. The strategy implemented is that after extending each substructure by one edge, it evaluates each extended substructure based upon its compression value (the higher the better). A list is maintained of the best substructures, and this process is continually repeated until either there are no more substructures to consider or a user-specified limit is reached.

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 [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.

For more information regarding the GBAD algorithms, the readers should refer to [3].

IV. Related Work

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 *streaming graphs*. 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 [4]. Other work has evaluated approaches to different graph properties, such as shortest paths in directed graphs [7], counting triangles [5][6], or finding the maximum clique [16]. In the work by Jha et al., they propose a single-pass streaming algorithm that maintains a real-time estimate of the number of triangles of a graph, by storing only a fraction of edges [5]. Pavan et al. present a novel space-efficient algorithm for counting and sampling triangles in a massive graph whose edges arrive as a stream [6]. 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 [8].

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 graph, utilizing estimation techniques with theoretical guarantees [9]. 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 by dynamically partitioning the network [10]. 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. Other non-outlier detection approaches have involved sampling schemes to sample from the stream of graph edges. In the work by Ahmed et al., they use sampling techniques to primarily deal with estimating certain graph properties, like triangle counts [14].

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 [11] based on the GBAD approach [3], that type of approach does not address the issue of scalability.

V. A Streaming Partitioned Approach To Graph-Based Anomaly Detection

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". What our experiments have shown us is 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 following algorithm. PLADS 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.

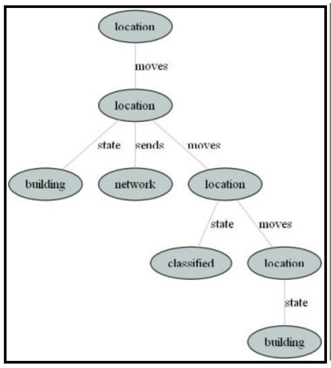
It should be mentioned that the size of each partition (i.e., number of vertices and edges) is not necessarily the same. For instance, with the first data set representing employee movements and actions at an embassy, we will partition the graph down to the day level (~2000 edges per partition), where each partition presents a day at the embassy. Then, for the second data set, we will experiment with partitions that represent not only a day, but also partitions down to the hour and minutes level. In previous work, we analyzed the effect that varying the values of *M* and *N* has on the running-times as well as accuracy [12].

A. *Experiments Using Insider Threat Example*

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 [13]. 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.

1) *GBAD*

As input to GBAD, the entire data set 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 3, where it shows that an employee somehow got into the classified area without ever badging in). However, it took 14,347 seconds to discover the anomalous substructure when analyzing the entire graph.



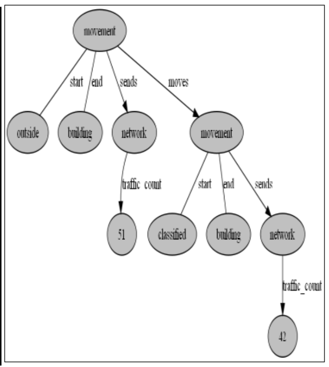


Figure 2. Normative pattern of movements and transactions (left).
Figure 3. Anomalous substructure of movement/transactions (right).

2) *PLADS*

In order to demonstrate the potential effectiveness of the PLADS approach, we applied the algorithm to the same dataset that represents cyber-threat activity, following the steps outlined in the algorithm:

1. *Process N partitions in parallel.*

   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.

2. *Determine best normative pattern, P, among NM possibilities.*

   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 that is identical to the normative pattern shown in Figure 2.

3. *Each partition discovers anomalous substructures based upon P.*

   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 1 substructure is reported as anomalous across all of the partitions, with the longest running partition taking 328 seconds.

4. *Evaluate anomalous substructures across partitions and report most anomalous substructure.*

   For this example, since there is only one anomalous substructure reported, evaluation is trivial. (It should be noted that this is one of the anomalous substructures discovered when the graph was processed in its entirety.)

5. *Process new partition.*

   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 on 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 four partitions. So, we use the 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 (e.g., 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 a total of 257 seconds) no anomalous substructures.

   This same behavior continues over partitions 8 and 9, in 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-like 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), and not very large. So, now we will examine results on denser graphs that actually represent streaming data.

B. *Experiments Using Social Network Genrator*

The Linked Stream Benchmark (LSBench) data generator allows one to generate data that represents users as the static data, and their actions as a data stream, including gps locations, posts, and photo albums, as well as "like"s and "know"s (https://code.google.com/p/lsbench/). Using the provided *sibgenerator* tool, we can generate RDF triplets of varying sizes and periods of time, that contain user information, and their associated locations, devices used, postings, photos, likes, and whom they know. Figure 4 shows the schema of the data as shown on the LSBench web-site.

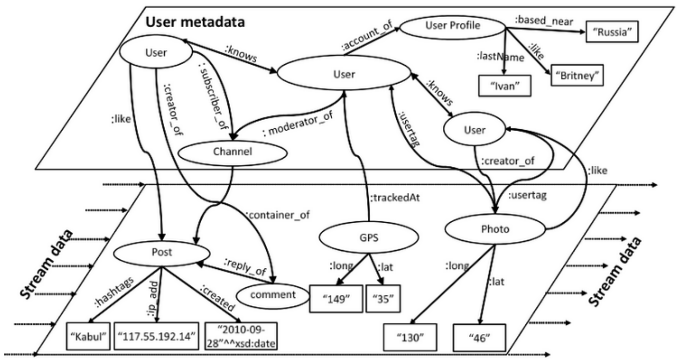


Figure 4. Logical schema of the stream data.

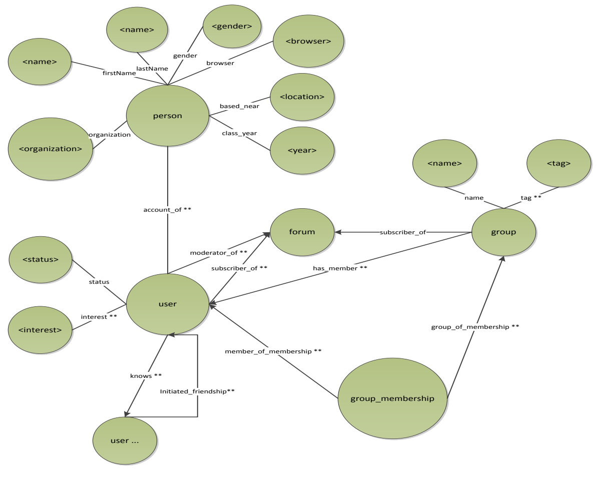


Figure 5. Graph topology of user information.

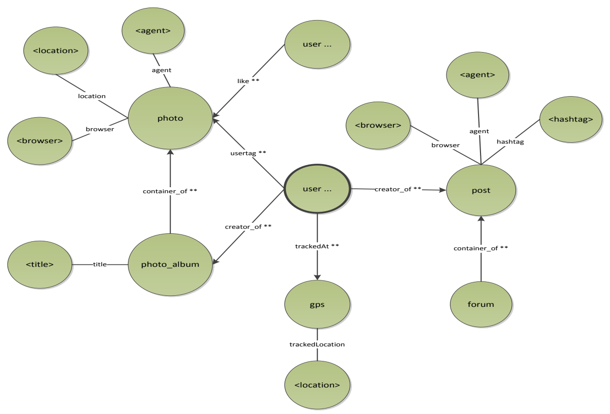


Figure 6. Graph topology of streaming data.

We then converted the *sibgenerator* output into a graph format, representing the LSBench objects as vertices, and actions as edges. Figure 5 presents the graph topology of the static user information - information that is repeated in a partition where an action takes place. Figure 6 presents the graph topology of the streaming data. It should also be noted that a quick analysis of data from the generator shows that values created for the *firstName*, *lastName*, *organization*, and *based_near* fields (associated with a user) are highly unique. Table 1 shows an example from a generation of 500 users.

Table 1. Number of unique values.

Field name	Entity	Num unique values/total values
firstName	user	431/500
lastName	user	413/500
organization	user	416/500
based_near	user	236/500

Because there is such a high diversity amongst these values with a uniform probability, and from an anomaly detection perspective we are not interested in discovering the differences between these values, we will remove these particular fields from the final generated input graphs.

Varying the number of users from anywhere between 500 and 10,000 users results in between 5,000 and up to 150,000 edges per graph input file (i.e., partition), depending upon the time period of the captured data, which is anywhere from every 15 minutes to every day. We quickly discover that the generator is consistent in its generation of patterns, and despite us also varying the number of minutes, hours and days, as well as the probabilities of occurrence of values (as much as the generator will allow us), the normative pattern is fairly consistent.

To demonstrate the potential of this approach to discovering anomalies in a social network, we will take the example of a network generated for 500 users using the default *sibgenerator* settings. We will generate graph partitions that are composed of all activity broken down to the hour (i.e., midnight-00:59AM, 1:00AM-1:59AM, etc.) on February 28th. (This is just one arbitrary example to show the potential of this approach, but we observe similar results throughout many different configurations.) Using PLADS, we discover the normative pattern shown in Figure 7.

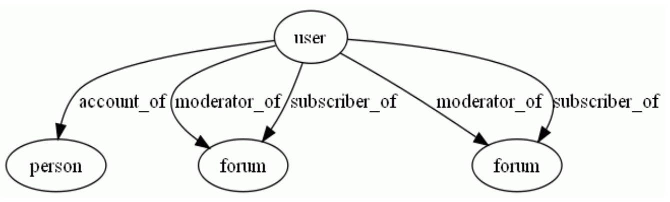


Figure 7. Normative pattern.

The substructure shown in Figure 7 indicates that the normative pattern is a user that moderates/subscribes to at least two forums. Starting at 6:00AM, PLADS reports two users showing an interest in (i.e., are a fan of) "Cyndi Lauper", despite the fact that there are 34 other users that are "Lauper" fans throughout the entire day. What makes this interesting is that starting at 7:00AM, PLADS then reports a single instance of a user – one of the "Lauper" users reported previously - that belongs to a Japanese group called "Andrea Bocelli". This is interesting because this is the only user (out of 10) that does not fit the pattern for users in this group, as all of the other members of the group do not moderate or subscribe to any forums. In short, PLADS discovers a user that fits the normative pattern, but is interested in "Cyndi Lauper" while also belonging to a group devoted to "Andrea Bocelli" – a group that seems solely devoted to the Italian tenor, as none of its others members moderate or subscribe to any forums – unlike this lone "Lauper" fan. While somewhat inconspicuous, this scenario can be mapped to more serious situations in which a clandestine group attempts to remain inconspicuous by acting like other groups.

In these experiments, PLADS discovers all of the targeted anomalies. As reported in our previous work [12], when 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%. With these experiments using the social network generator data, we observe a similar false positive rate, and we achieve a slightly better processing average of 222 edges/second. The improved edge processing rate can probably be attributed to the diversity of values produced by the *lsbench* data generator, which reduces the possible candidates during beam generation. In regards to the high FP rate, if we were to increase the size of the processing window, we would be able to reduce the number of false positives by observing more repetitions of the incorrectly-identified anomalous substructures (i.e., noise), and thus excluding them from being considered as anomalies of interest, albeit at the sacrifice of speed.

VI. Conclusions and Future Work

Detecting potential attacks, whether it is via insider threats or social threats to the populace are critical challenges for industry as well as our government. In order to address the issue of analyzing complex networks for patterns and anomalies, one must provide methods of monitoring and rapidly detecting anomalies. In previous work, we represented networks and various security related information using a graph and developed the graph-based anomaly detection (GBAD) approach that was able to detect anomalies with high accuracy and low false positive rates. In this work we present our streaming approach called PLADS for Pattern Learning and Anomaly Detection in Streams. We then evaluated the relevance of our approach on two different synthetic data sets: one representing an insider threat scenario, and the other demonstrating a social network scenario. PLADS allows us to process information that is represented in data streams, discovering patterns and anomalies with minimal false-positives, with an order-of-magnitude speed-up over the traditional GBAD approach. Next, 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 will allow us to remove the "boundary issues" associated with anomalous substructures that could span graph partitions. We will also develop parallel implementations of these approaches to take advantage of high-performance computing platforms and further improve the scalability of the PLADS framework.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant Nos. 1318913 and 1318957.

References

[1] A. Sorkin, "Calculating the Grim Economic Costs of Ebola Outbreak," *N.Y. Times*, October 13, 2014.

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

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

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

[5] M. Jha, C. Seshadhri, and A. Pinar, "A space efficient streaming algorithm for triangle counting using the birthday paradox," *KDD*, 2013, pp. 589-597.

[6] A. Pavan, K. Tangwongsan, S. Tirthapura, and K. L. Wu, "Counting and sampling triangles from a graph stream," *VLDB* 6, 14, 2013, pp. 1870-1881.

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

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

[9] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavaldà, "Mining frequent closed graphs on evolving data streams," *KDD*, 2011.

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

[11] 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), IEEE International Conf. on Social Computing (SocialCom)*, pp.1102-1110, 2011

[12] W. Eberle and L. Holder, "A Partitioning Approach to Scaling Anomaly Detection in Graph Streams," *First International Workshop on High Performance Big Graph Data Management, Analysis, and Mining (BigGraphs), IEEE BigData Conference*, October 2014.

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

[14] N. Ahmed, J. Neville, and R. Kompella, "Network sampling: from static to streaming graphs", *TKDD*, 2013.