Mining for Structural Anomalies in Graph-based Data

William Eberle and Lawrence Holder, *Members, IEEE*

Abstract—**In this paper we present graph-based approaches to mining for anomalies in domains where the anomalies consist of unexpected entity/relationship alterations that closely resemble non-anomalous behavior. We introduce three novel algorithms for the purpose of detecting anomalies in all possible types of graph changes. Each of our algorithms focuses on a specific graph change and uses the minimum description length principle to discover those substructure instances that contain anomalous entities and relationships. Using synthetic and real-world data, we evaluate the effectiveness of each of these algorithms in terms of each of the types of anomalies. Each of these algorithms demonstrates the usefulness of examining a graph-based representation of data for the purposes of detecting fraud.**

I. Introduction

Recently there has been an impetus towards analyzing multi-relational data using graph theoretic methods. Not to be confused with the mechanisms for analyzing "spatial" data, graph-based data mining approaches are an attempt at analyzing data that can be represented as a graph (i.e., vertices and edges). Yet, while there has been much written as it pertains to graph-based data mining for intrusion detection [13], very little research has been accomplished in the area of graph-based anomaly detection.

Using information theoretic, probabilistic and maximum partial substructure approaches, we have developed three novel algorithms for analyzing graph substructures for the purpose of uncovering all three types of graph-based anomalies: modifications, insertions and deletions. In this paper, we define what we consider to be an anomaly as it relates to graphs. Then, we present the algorithms along with some examples, followed by our results using randomly-generated synthetic graphs two real-world data sets. Finally, we conclude with some related work, conclusions and future work.

II. Related Work

Lin and Chalupsky [8] applied what they called *rarity* measurements to the discovery of unusual *links* within a graph. Using various metrics to define the commonality of paths between nodes, the user was able to determine whether a path between two nodes were interesting or not, without having any preconceived notions of meaningful patterns. One of the disadvantages of this approach was that while it was domain independent, it assumed that the user was querying the system to find interesting relationships regarding certain nodes.

The AutoPart system presented a non-parametric approach to finding outliers in graph-based data [1]. Part of this approach was to look for outliers by analyzing how edges that were removed from the overall structure affected the minimum descriptive length (MDL) of the graph [11]. Representing the graph as an adjacency matrix, and using a compression technique to encode node groupings of the graph, he looked for the groups that reduced the compression cost as much as possible. However, with this approach, only the effect of edge removals on a graph's structure are examined.

In 2005, the idea of entropy was also used by Shetty and Adibi [12] in their analysis of a real-world data set: the famous Enron scandal. They used what they called "event based graph entropy" to find the most interesting people in an Enron e-mail data set. Using a measure similar to what [9] had proposed, they hypothesized that the important nodes (or people) were the ones who had the greatest effect on the entropy of the graph when they were removed. However, in this approach, the idea of important nodes did not necessarily mean that they were anomalous.

In the 2005 SIGKDD Explorations, a couple of different approaches to graph-based anomaly detection were presented. Using just bipartite graphs, Sun et al. [14] presented a model for scoring the normality of nodes as they relate to the other nodes. Again, using an adjacency matrix, they assigned what they called a "relevance score" such that every node *x* had a relevance score to every node *y*, whereby the higher the score the more related the two nodes. Rattigan and Jensen [10] also went after anomalous links, this time via a statistical approach. Using a Katz measurement, they used the link structure to statistically predict the likelihood of a link. While it worked on a small dataset of author-paper pairs, their single measurement just analyzed the links in a graph.

III. Graph-based Anomalies

Setting up fraudulent web-sites, "phishing" for credit cards, stealing calling cards, and creating bogus bank accounts are just some of the countless examples of scams that have succumb everyone from the individual investor to large corporations. In every case, the fraudster has attempted to swindle their victim and hide their dealings within a morass of data that has become proverbially known

William Eberle is with the Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019 USA. (e-mail: eberle@cse.uta.edu).

Lawrence Holder is with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164 USA. (e-mail: holder@wsu.edu).

as the “needle in the haystack”. Yet, even when the data is not relatively large in size, the ability to discover the nefarious actions is still ultimately difficult due to the *mimicry* of the perpetrator.

The idea behind the approach presented in this paper 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 substructure*. This definition of an anomaly is unique in the arena of graph-based anomaly detection, as well as non-graph-based anomaly detection. The concept of finding a pattern that is "similar" to frequent, or good, patterns, is different from most approaches that are looking for unusual or “bad” patterns. While other non-graph-based data mining approaches may aide in this respect, there does not appear to be any existing approaches that deal with this scenario.

Definition: Given a graph G with a normative substructure S, a substructure S' is considered anomalous if the difference d between S and S' satisfies $0 < d <= X$, where X is a user-defined threshold and d is a measure of the unexpected structural difference between two sub-graphs of a graph.

The importance of this definition lies in its relationship to fraud detection (i.e., deceptive practices that are intended to illegally obtain or hide information). If a person or entity is attempting to commit fraud, they will do all they can to hide their illicit behavior. To that end, their approach would be to convey their actions as close to legitimate actions as possible. 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.” [6].

A. Anomaly Types

For a graph-based anomaly, there are several situations that might occur:

1. *A vertex exists that is unexpected.*
2. *An edge exists that is unexpected.*
3. *The label on vertex is different than was expected.*
4. *The label on edge is different than was expected.*
5. *An expected vertex is absent.*
6. *An expected edge between two vertices is absent.*

In essence, there are three general *categories of anomalies*: insertions, modifications and deletions. Insertions would constitute the first two situations; modifications would consist of the third and fourth situation; and deletions would categorize the last two situations.

B. Assumptions

Many of the graph-based anomaly detection approaches up to now have assumed that the data exhibits a power-law distribution. The advantage of the approaches presented in this paper is that it does not assume the data consists of a power-law behavior. In fact, no standard distribution model is assumed to exist. All that is required is that the data is *regular*, which in general means that the data is “predictable”. While there are many data sets that are not regular in nature, many of the real-world data sets that are examined for fraudulent activity consist of user transactions that exhibit regular patterns of behavior.

In order to address our definition of an anomaly, we make the following assumptions about the data.

Assumption 1: The majority of a graph consists of a normative pattern, and no more than X% of the normative pattern is altered in the case of an anomaly.

Since our definition implies that an anomaly constitutes a minor change to the prevalent substructure, we chose a small percentage (e.g., 10%) to represent the most a substructure would be changed in a fraudulent action.

Assumption 2: The graph is regular.

If a graph were irregular, the ability to distinguish between anomalies and noise would be prohibitive.

Assumption 3: Anomalies consist of one or more modifications, insertions or deletions.

As was described earlier, there are only three types of changes that can be made to a graph. Therefore, anomalies that consist of structural changes to a graph must consist of one of these types.

Assumption 4: The normative pattern is connected.

In a real-world scenario, we would apply this approach to data such as cargo shipments, telecommunication traffic, financial transactions or terrorist networks. In all cases, the data consists of a series of nodes and links that share common nodes and links. Certainly, graphs could contain potential anomalies across disconnected substructures, but at this point, we are constraining our research to only connected anomalies.

IV. GRAPH-BASED ANOMALY DETECTION ALGORITHMS

Most anomaly detection methods use a supervised approach, which requires some sort of baseline of information from which comparisons or training can be performed. In general, if one has an idea what is normal behavior, deviations from that behavior could constitute an anomaly. However, the issue with those approaches is that one has to have the data in advance in order to train the system, and the data has to already be labeled (i.e., fraudulent versus legitimate).

Our work has resulted in the development of three algorithms, which we have implemented using a tool called GBAD (Graph-based Anomaly Detection). GBAD is an *unsupervised* approach, based upon the SUBDUE graph-based knowledge discovery system [2]. SUBDUE is a graph-based knowledge discovery system that finds structural, relational patterns in data representing entities and relationships. Using a greedy beam search and Minimum Description Length (MDL) heuristic, each of the three anomaly detection algorithms uses SUBDUE to provide the top substructure, or normative pattern, in an input graph. In our implementation, the 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.

Using GBAD as the tool for our implementation, we have developed three separate algorithms: GBAD-MDL, GBAD-P and GBAD-MPS. Each of these approaches is intended to discover all of the possible graph-based anomaly types as set forth earlier.

A. Information Theoretic Algorithm (GBAD-MDL)

In order to implement the GBAD-MDL algorithm, we first use SUBDUE to discover the best substructure. In addition to providing the normative pattern using an MDL evaluation, SUBDUE also provides two other features: the ability to specify inexact matching as a percentage of the normative substructure, and a list of all instances that match the best substructure. SUBDUE terminates processing when there are no more extensions to candidate substructures, whereas the GBAD-MDL algorithm continues processing the best substructure, analyzing its instances for the one that is closest in transformation cost to the normative pattern.

First, the algorithm modifies the best substructure list by determining which substructure is actually the true normative pattern. Since an inexact matching was used, it is possible that the top substructure specified in SUBDUE (i.e., the best substructure), may not be the true normative pattern. So, a search is performed on the list of instances, finding the pattern that is the most frequent, and replacing the previously specified best substructure with its structure.

Second, the new list of instances is compared to the new best substructure, and each instance is given an anomalous score equal to its cost of transformation (for transforming the instance into the normative pattern). Then, for each instance in the list that matches this instance (i.e., isomorphic), the anomalous score is increased by the value of the cost of transformation, where the score is equal to the cost of transformation times frequency.

For the last step, our GBAD-MDL implementation finds the anomalous instance (or instances, if their anomalous scores match), and flags the individual vertices and edges that are anomalous. This is accomplished by comparing the structure of the anomalous instance with the normative substructure, and for each vertex and edge in the anomalous instance that does not have a match in the normative pattern, a flag is set. So, in the end, when the anomalous instance is output by this implementation, there is an indicator next to each individual anomaly.

The following is a simple example of results obtained using our implementation of the GBAD-MDL algorithm described above.

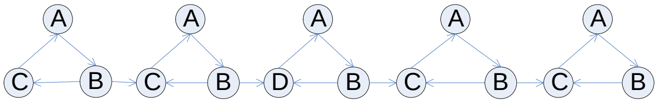

Fig. 1. Simple graph for GBAD-MDL example.

Running the GBAD-MDL algorithm, the anomalous substructure, as shown in Fig. 2, is:

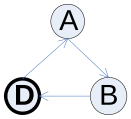

Fig. 2. Anomalous substructure from simple graph using GBAD-MDL.

which is exactly the desired result. (The individual anomaly is in **bold**.) It should also be noted that no other substructures are reported as anomalous. The above is similar to the example that was presented in the paper by Noble and Cook [9]. In their work they used the SUBDUE application to look at the problem of anomaly detection from the anomalous substructure and sub-graph perspective.

B. Probabilistic Algorithm (GBAD-P)

In order to implement the GBAD-P algorithm, we again used SUBDUE to discover the best substructure. In addition, we also used two other features provided by SUBDUE: maintaining a list of all instances that match the best substructure; iterating multiple times, compressing the graph by the best substructure at each iteration. When enough iterations are specified, SUBDUE terminates processing when any more attempts at compressing the graph would not result in a further reduction in its MDL. After the first iteration, where the graph is compressed by the normative pattern, the GBAD-P algorithm analyzes extensions from each instance of the best substructure at each iteration, looking for the ones with the lowest probability of occurring.

First, SUBDUE’s logic for extensions is modified to only extend one edge at each iteration. While the first iteration works as-is in terms of performing extensions in order to find the best substructure, subsequent iterations only process single edge extensions from the newly compressed substructure. This allows the GBAD-P algorithm to eventually evaluate the probability of individual extensions.

Second, the algorithm modifies the list of best substructure list by finding the best substructure that contains the compressed normative pattern from the first iteration. This is done to ensure that at each iteration we are still working from the normative pattern. The first substructure in the list that contains the compressed normative pattern is moved to the top of the list as the best substructure (since the list is already in order by value).

Third, for the newly defined best substructure, all of its instances are evaluated in terms of their *probability* amongst themselves. For each instance, a simple evaluation is calculated where the probability of the instance is the number of matching instances divided by the total number of instances, all within the list of instances for the best substructure. This value is then set as the anomalous score for the corresponding instance.

After each iteration, our GBAD-P implementation prints the anomalous instance (or instances, if their anomalous scores match). The output is similar to what is produced by the GBAD-MDL algorithm, except that the score is a value from 0.0-1.0, and it is done after each iteration (except for the first). By doing this over each iteration, it allows one to view the growth of the anomaly, one edge at a time.

The following is a simple example of results obtained using our implementation of the GBAD-P algorithm on a network-looking structure, as shown in Fig. 3.

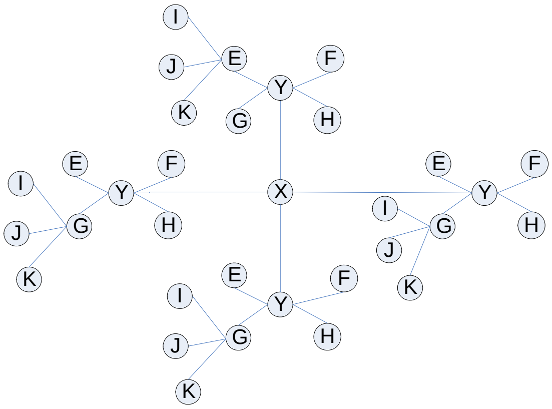

Fig. 3. Network-type graph.

In Fig. 3**,** there is a central node (labeled **X**) with four connected identical star structures (each with a center node labeled **Y**). Each of these star structures has an identical smaller substructure (made up of vertices labeled **I**, **J** and **K**) connected to it. However, one of the star structures has the **IJK** substructure connected to its vertex labeled **E**, while the others have it connected to their vertex labeled **G**.

Running the GBAD-P algorithm on this graph results in the following three structures labeled as anomalous, as shown in Fig. 4 (after the second iteration).

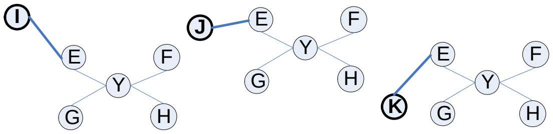

Fig. 4. Anomalous structures when limit increased on GBAD-P example.

So, in essence, while it did report the anomaly as three different substructures (all equal in probability), the complete anomaly is discovered. It should also be noted that on subsequent iterations, no more anomalous substructures are found. (All of the subsequent candidates have a probability of 100%.) This is because on the following iteration, the instances of the best substructure are compressed to a single vertex, and the other vertices (**I**, **J** and **K**), are linked to that single vertex, with no former knowledge of where they linked (i.e., whether they linked to **E** or **G**). Possible future work could include a modification to this approach to keep track of the original connections for further evaluation.

C. Max Partial Substructure Algorithm (GBAD-MPS)

In order to implement the GBAD-MPS algorithm, again we used SUBDUE to discover the best substructure. In addition, we also used another feature provided by SUBDUE: specifying the beam width of the search. By default, SUBDUE uses a beam width of 4 which signifies that it will only keep the top 4 substructures after evaluating each extension. While this heuristic has proven to be successful in SUBDUE's ability to discover the normative pattern, in order to be able to analyze substructures that never extended to the normative pattern, which is necessary for this algorithm, we need to extend the beam width so that other substructures can be evaluated for anomalies. This allows for us to keep track of those instances that are not direct ancestors of the normative pattern. In the end, SUBDUE terminates processing when there are no more extensions to candidate substructures, while the GBAD-MPS algorithm continues processing all of the ancestral substructures, looking for the one that is closest in transformation cost to the normative pattern

First, a list of substructures is maintained that consists of substructures (and their instances) that at some point during SUBDUE processing were used in evaluating their potential for being the normative pattern. Even if a substructure fails to make the "best" list at some point, it is still maintained on this list as a possible anomalous substructure. While this list can be rather large (and deserves some future memory-saving analysis), since the normative pattern is not known at this point, it has to be maintained until the final evaluation.

Second, the algorithm takes this list of substructures and compares each substructure to the normative pattern. If a substructure matches within the user specified anomalous threshold (cost of transformation), each of its instances is compared to the instances of the normative pattern. If an instance overlaps one of the normative pattern's instances (i.e., all of its edges and vertices are found in one of the normative instances), the instance is thrown out because it could eventually extend to the normative pattern.

Third, each instance in the candidate list of instances is given an anomalous score equal to its cost of transformation (for transforming the instance into the normative pattern). Then, for each instance in the list that is isomorphic to another instance in the list, its anomalous score is increased by the value of its cost of transformation (i.e., cost of transformation * frequency).

In the end, our GBAD-MPS implementation prints the anomalous instance (or instances, if their anomalous scores match). This output is a little different from the other two algorithms in that no anomalous vertices and edges are indicated, just the entire anomalous instance. Since what is anomalous is the lack of structure, a comparison of the normative pattern to the anomalous substructure yields the anomalous differences.

The following is a simple example of results obtained using our implementation of the GBAD-MPS algorithm described above.

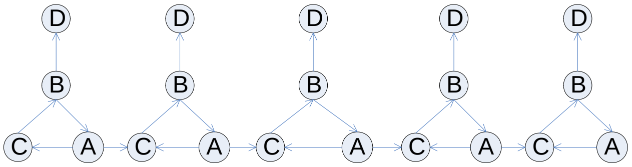

Fig.5. Simple graph for GBAD-MPS example.

The normative pattern from this graph is shown in Fig. 6.

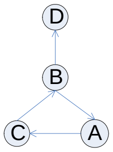

Fig. 6. Normative pattern from simple graph for GBAD-MPS example.

Now, suppose we remove one of the edges and its associated vertex, from one of the instances of this normative pattern, creating the graph shown in Fig. 7.

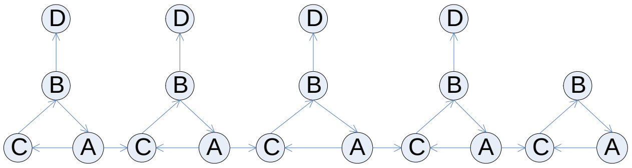

Fig. 7. Simple graph for GBAD-MPS with deleted vertex and edge.

In other words, we removed one of the **D** vertices and its associated edge. Running the maximum partial substructure approach on this modified graph, results in the anomalous instance shown in Fig. 8.

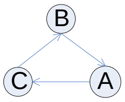

Fig. 8. Anomalous instance from deletion example using GBAD-MPS.

However, this pattern is common to all of the normative instances. So, for usefulness, we report the actual anomalous graph instance specified in the input graph file.

V. SYNTHETIC EXPERIMENTS

For our synthetic experiments, we created graphs using a tool called *subgen* [3] that generates graphs based upon user-specified parameters, including:

- total number of vertices and edges
- list of possible vertex and edge labels and their probabilities
- substructure pattern
- amount of connectivity

Using these parameters, *subgen* computes the number of instances that need to be generated by calculating the size of the graph and dividing by the size of the substructure pattern (i.e., what we want to be the normative pattern). After the graph is built from these instances, randomly-labeled vertices (based upon their probabilities) are added in order to achieve the desired graph size. Then randomly-labeled edges (again based upon their probabilities) are added in order to achieve the specified connectivity level. Finally, any additional edges are added in order to achieve the desired graph size.

In order to be consistent across all experiments, we chose a star-cluster pattern as our normative pattern (i.e., a node with connections to several other nodes, and each of those nodes with several connections to other nodes). The choice of this pattern was somewhat arbitrary, but it also resembles many types of real-world data, such as networks, calling trees, and financial transactions. Each synthetic graph consisted of substructures containing a normative pattern (V number of vertices and E number of edges), connected to each other by one or more random connections, and each test consisted of AV number of anomalous vertices and AE number of anomalous edges.

Fig. 9 shows the effectiveness of the GBAD-MDL approach. For graphs of varying sizes, from 100 vertices/edges to 10,000 vertices/edges, with a normative pattern consisting of 10 vertices/9 edges, the results were identical across the spectrum. In this figure, the X axis represents the thresholds, the Y axis is the percentage of anomalies discovered, and the Z axis indicates the sizes of the anomalies.

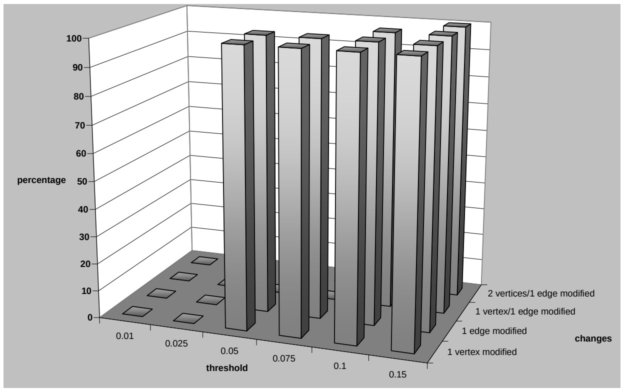

Fig. 9. GBAD-MDL runs where all anomalies discovered

As expected, when the threshold is increased to accommodate the size of the anomaly with respect to the normative pattern, the anomalies are discovered 100% of the time. The drawback is that as the threshold is increased, so is the running time of the algorithm, and false positives, like noise, will increase (i.e., the size of the reported anomaly is equal to or smaller than that of the true anomaly).

Without changing any parameters, experiments using GBAD-P and GBAD-MPS resulted in less than a 100% discovery rate across all tests. However, when we increased SUBDUE's beam width parameter so that GBAD could be provided a larger set of substructure instances to evaluate, the result was a 100% discovery rate. The reason that the number of substructures to evaluate has to be increased is that as the size of the anomaly grows (i.e., the number of vertices and edges inserted or deleted increases), the further away the cost of transformation for the anomalous instance is from the normative pattern. In addition, unlike with the GBAD-MDL tests, there were no false positives reported from any of the GBAD-P or GBAD-MPS synthetic tests. Using varying sizes of normative patterns and anomalies, each approach has shown to be useful at discovering a specific type of anomaly. While the algorithms do not appear to be useful outside of their intended targets, no graphs of any size or any anomaly went undetected by all three approaches.

One of the advantages of these algorithms is that they do not just return the pattern of the anomaly – they also return the actual anomalous instances within the data. In a real-world scenario, that can be invaluable to an analyst who may need to act upon a fraud situation before the losses are too great. The disadvantage of these algorithms is that they are focused on specific anomalies: modifications, insertions or deletions. Thus, in a real-world scenario, it would require that all three algorithms be used in conjunction, as the type of anomaly would most likely be unknown.

VI. REAL-WORLD EXPERIMENTS

A. Cargo Shipments

One area that has garnered much attention recently is the analysis and search of imports into the United States. The largest number of imports into the U.S. arrive via ships at ports of entry along the coasts. Thousands of suspicious cargo, whether illegal or dangerous, are examined by port authorities every day. Due to the volume, strategic decisions must be made as to which cargo should be inspected, and which cargo will pass customs without incident. A daunting task that requires advanced analytical capabilities to maximize effectiveness and minimize false searches.

Using shipping data obtained from the CBP (http://www.cbp.gov/), we are able to create a graph-based representation of the cargo information where row/column entries are represented as vertices, and labels convey their relationships as edges. Fig. 10 shows a portion of the actual graph that we will use in our anomalous detection experiments. While we were not given any labeled data from the CBP (i.e., which shipments were illegal, or anomalous, and which ones were not), we can draw some results from random changes and from simulations of publicized incidents.

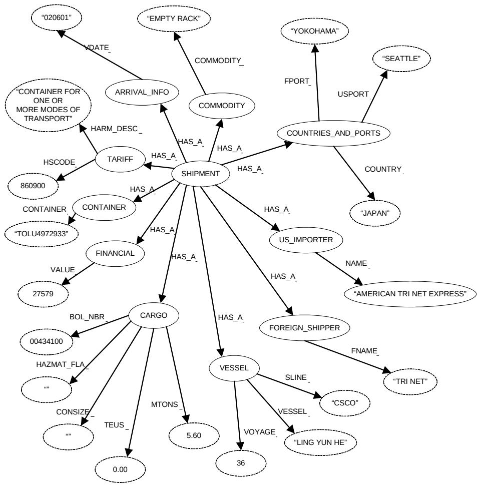

Fig. 10 Example of cargo data represented as a graph.

In [5], real-world cargo shipment occurrences were generated so as to show how graph properties can be used to determine structural anomalies in graphs. While that approach was successful in discovering graphs that contained anomalies, the exact anomalies were not part of the output. Using the GBAD algorithms on these same data sets, we can display the actual anomalies.

One example is from a press release issued by the U.S. Customs Service. The situation is that almost a ton of marijuana is seized at a port in Florida [15]. In this drug smuggling scenario, the perpetrators attempt to smuggle contraband into the U.S. without disclosing some financial information about the shipment. In addition, an extra port is traversed by the vessel during the voyage. For the most part, the shipment looks like it contains a cargo of toys, food and bicycles from Jamaica. When we run all three algorithms on this graph, GBAD-MDL is unable to find any anomalies, which makes sense considering none of the anomalies are modifications. When the graph contains the anomalous insertion of the extra traversed port, the GBAD-P algorithm is able to successfully discover the anomaly. Similarly, when the shipment instance in the graph is missing some financial information, GBAD-MPS reports the instance as anomalous.

According to CBP, an estimated $2 billion in illegal textiles enter the U.S. every year [4]. One of the more common methods of eluding authorities is accomplished using what is called transshipment. The CBP defines transshipment as "A false declaration or information given in order to circumvent existing trade laws for the purpose of avoiding quotas, embargoes or prohibitions, or to obtain preferential duty treatment." In order to circumvent quotas, the fraudster will change the country of origin of their goods. For example, they may ship the goods into Canada or Mexico, change the country-of-origin, and ship into the U.S. free from tariffs under the North American Free Trade Agreement (NAFTA).

In order to simulate this real-world example, we randomly changed the country of origin on one of the shipments to "CANADA". While the GBAD-P and GBAD-MPS algorithms were unsuccessful in discovering this anomaly (as was expected), the GBAD-MDL algorithm was able to clearly mark the instance that contained the anomaly. At first it was surprising that just a change in the country of origin would have that effect, and given perhaps a different set of data, this would not have been as effective. But, in this case, all of the shipments had a normative pattern that included Asian ports of origin. So, by altering the originating country to Canada, the GBAD-MDL was able to clearly notice the structural oddity.

B. Network Intrusions

One of the more applied areas of research when it comes to anomaly detection can be found in the multiple approaches to intrusion detection. The reasons for this are its relevance to the real world problem of networks and systems being attacked, and the ability of researchers to gather actual data for testing their models. Perhaps the most used data set for this area of research and experimentation is the 1999 KDD Cup network intrusion dataset [7].

The KDD Cup data consists of connection records, where a connection is a sequence of TCP packets. Each connection record is labeled as either "normal", or one of 37 different attack types. Each record consists of 31 different features (or fields), with features being either continuous (real values) or discrete. In the 1999 competition, the data was split into two parts: one for training and the other for testing. Groups were then allowed to train their solutions using the training data, and were then judged based upon their performance on the test data.

Since the GBAD approach uses unsupervised learning, we will run the algorithms on the test data so that we can judge our performance versus other approaches. Also, because we do not know the possible structural graph changes associated with network intrusions, we will have to run all three algorithms to determine which algorithms are most effective for this type of data. Each test contains 50 essentially random records, where 49 are normal records and 1 is an attack record, where the only controlled aspect of the test is that there is only one attack record per data set. This is done because the test data is comprised of mostly attack records, which does not fit our definition of an anomaly, where we are assuming that anomalous substructures are rare. Fortunately, this again is a reasonable assumption, as attacks would be uncommon in most networks.

Not surprisingly, each of the algorithms has a different level of effectiveness when it comes to discovering anomalies in intrusion data. Using GBAD-MDL, our ability to discover attacks is relatively successful. Across all data sets, 100% of the attacks are discovered. However, all but the *apache2* and *worm* attacks produce some false positives. 42.2% of the test runs do not produce any false positives, while runs containing *snmpgetattack*, *snmpguess*, *teardrop* and *udpstorm* attacks contribute the most false positives. False positives are even higher for the GBAD-P algorithm, and the discovery rate of actual attacks decreases to 55.8%. GBAD-MPS shows a similarly bad false positive rate at 67.2%, and a lower discovery rate at 47.8%.

It is not surprising that GBAD-MDL is the most effective of the algorithms, as the data consists of TCP packets that are structurally similar in size across all records. Thus, the inclusion of additional structure, or the removal of structure, is not as relevant for this type of data, and any structural changes, if they exist, would consist of value modifications.

VII. Conclusion

The three algorithms presented in this paper are able to discover an anomaly when it consists of a small change to the normative pattern. Using the minimum description length principle and probabilistic approaches, we have been able to successfully discover anomalies in graphs and normative patterns of varying sizes with minimal to no false positives. Results from both synthetic and real-world data demonstrate the effectiveness of the approaches. We are pursuing experiments on other domains that can be represented as graphs, including citation and social networks. While our results are effective in detecting anomalies in a security area such as cargo shipments, other possible applications of these approaches include post-9/11 terrorist networks and the Enron e-mail datasets (e.g., detecting anomalies in e-mail patterns among executives).

References

[1] D. Chakrabarti. *AutoPart: Parameter-Free Graph Partitioning and Outlier Detection*. Knowledge Discovery in Databases: PKDD 2004, 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, 112-124, 2004.

[2] D. Cook and L. Holder. *Graph-based data mining*. IEEE Intelligent Systems 15(2), 32-41, 1998.

[3] D. Cook and L. Holder. *Substructure Discovery Using Minimum Description Length and Background Knowledge*. Artificial Intelligence Research, 1:231-255, 1994.

[4] Customs and Border Protection Today, "Illegal textile entries: a way to save a few bucks?", March 2003. (http://www.cbp.gov/xp/CustomsToday/2003/March/illegal.xml)

[5] W. Eberle and L. Holder. *Detecting Anomalies in Cargo Shipments Using Graph Properties*. Proceedings of the IEEE Intelligence and Security Informatics Conference, 2006.

[6] M. Hampton and M. Levi. Fast spinning into oblivion? Recent developments in money-laundering policies and offshore finance centres. Third World Quartely, Volume 20, Number 3, June 1999, pp. 645-656, 1999.

[7] KDD Cup 1999. Knowledge Discovery and Data Mining Tools Competition. 1999. *(http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html)*

[8] S. Lin and H. Chalupsky. *Unsupervised Link Discovery in Multi-relational Data via Rarity Analysis*. Proceedings of the Third IEEE ICDM International Conference on Data Mining, 171-178, 2003.

[9] C. Noble and D. Cook. *Graph-Based Anomaly Detection*. Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 631-636, 2003.

[10] M. Rattigan and D. Jensen. *The case for anomalous link discovery*. ACM SIGKDD Explor. Newsl., 7(2):41-47, 2005.

[11] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Company, 1989.

[12] J. Shetty and J. Adibi. *Discovering Important Nodes through Graph Entropy: The Case of Enron Email Database*. KDD, Proceedings of the 3rd international workshop on Link discovery, 74-81, 2005.

[13] S. Staniford-Chen et al.. *GrIDS – A Graph Based Intrusion Detection System for Large Networks*. Proceedings of the 19th National Information Systems Security Conference, 1996.

[14] J. Sun, H. Qu, D. Chakrabarti and C. Faloutsos. *Relevance search and anomaly detection in bipartite graphs*. SIGKDD Explorations 7(2), 48-55, 2005.

[15] U.S. Customs Service: *1,754 Pounds of Marijuana Seized in Cargo Container at Port Everglades*. November 6, 2000. (http://www.cbp.gov/hot-new/pressrel/2000/1106-01.htm)