Identifying Threats Using Graph-based Anomaly Detection

William Eberle
Department of Computer Science
Tennessee Technological University
Box 5101
Cookeville, TN 38505
weberle@tntech.edu

Lawrence Holder
School of Electrical Engineering and Computer Science
Washington State University
Box 642752
Pullman, WA 99164
holder@wsu.edu

Diane Cook
School of Electrical Engineering and Computer Science
Washington State University
Box 642752
Pullman, WA 99164
cook@eecs.wsu.edu

Abstract

Much of the data collected during the monitoring of cyber and other infrastructures is structural in nature, consisting of various types of entities and relationships between them. The detection of threatening anomalies in such data is crucial to protecting these infrastructures. We present an approach to detecting anomalies in a graph-based representation of such data that explicitly represents these entities and relationships. The approach consists of first finding normative patterns in the data using graph-based data mining and then searching for small, unexpected deviations to these normative patterns, assuming illicit behavior tries to mimic legitimate, normative behavior. The approach is evaluated using several synthetic and real-world datasets. Results show that the approach has high true-positive rates, low false-positive rates, and is capable of detecting complex structural anomalies in real-world domains including email communications, cell-phone calls and network traffic.

1. Introduction

Maintaining the security of our infrastructure, whether physical or cyber, requires the ability to detect threats to the infrastructure. Modern threats are sophisticated, multifaceted, coordinated and attempt to mimic normal activity. Detecting such threats requires approaches that consider many different types of activities and the relationships between them. We describe an approach that represents the activities and relationships as a graph, mines the graph for normative patterns, and then searches for unexpected deviations to the normative patterns. These unexpected deviations may indicate the presence of a threat to the infrastructure being monitored.

The ability to mine data represented as a graph has become important in several domains for detecting various structural patterns (Cook and Holder 2006). One important area of data mining is anomaly detection, particularly for fraud. However, less work has been done in terms of detecting anomalies in graph-based data. While there has been some previous work that has used

statistical metrics and conditional entropy measurements, the results have been limited to certain types of anomalies and specific domains.

In this chapter we present graph-based approaches to uncovering anomalies in domains where the anomalies consist of unexpected entity/relationship alterations that closely resemble non-anomalous behavior. We have developed three algorithms for the purpose of detecting anomalies in all three types of possible graph changes: label modifications, vertex/edge insertions and vertex/edge deletions. Each of our algorithms focuses on one of these anomalous types, using the minimum description length principle to first discover the normative pattern. Once the common pattern is known, each algorithm then uses a different approach to discover particular anomalous types. The first algorithm uses the minimum description length to find anomalous patterns that closely compress the graph. The second algorithm uses a probabilistic approach to examine pattern extensions and their likelihood of existence. The third algorithm analyzes patterns that come close to matching the normative pattern, but are unable to make some of the final extensions leading to the construction of the normative pattern.

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 threats, where some individual or entity is cloaking their illicit activities through an attempt at closely resembling normal behavior.

The next section describes the general area of graph-based learning and the algorithm underlying our ability to find normative patterns in graphs. Section 3 defines the problem of graph-based anomaly detection, and Section 4 presents the GBAD system, which consists of three variants for detecting different types of graph-based anomalies. Section 5 presents experimental results evaluating GBAD on several synthetic and real-world datasets related to threat detection. We discuss related work in section 6, and conclude in section 7.

2. Graph-based Learning

While not specific to anomaly detection, there are several approaches to handling the first stage of detecting anomalies, which is the discovery of the normative pattern in data represented as a graph. One approach called gSpan returns all *frequent* subgraphs in a database that is represented as a graph (Yan and Han 2002). Using a depth-first search (DFS) on the input graphs, the algorithm constructs a hierarchical search tree based upon the DFS code assigned to each graph. Then, from its canonical tree structure, the algorithm performs a pre-order traversal of the tree in order to discover the frequent subgraphs.

Another approach is found in FSG, which is similar to gSpan in that it returns all of the frequent subgraphs in a database of transactions that have been represented as a graph (Kuramochi and Karypis 2004). However, unlike gSpan, FSG uses an Apriori-style breadth-first search. The algorithm takes the input graphs and performs a level-by-level search, growing patterns one edge at a time. The core of the FSG algorithm lies in its candidate generation and counting that are used to determine the frequent subgraphs.

In order to mine large graphs for frequent subgraphs, Huan et al. proposed a maximal frequent subgraphs approach called SPIN as an improvement to gSpan (Huan et al. 2004). By mining only subgraphs that are not part of any other frequent subgraphs, they are able to reduce the number of mined patterns by orders of magnitude. This is accomplished by first mining all frequent trees from a graph, and then reconstructing all maximal subgraphs from the mined trees. Zeng et al. looked at the problem of dense graphs by mining the properties of quasi-cliques (Zeng et al. 2006). Using a system called Cocain, they propose several optimization techniques for pruning the unpromising and redundant search spaces. To help combat the computational complexity of subgraph isomorphism, Gudes et al. proposed a new Apriori-based algorithm using disjoint paths (Gudes et al. 2006). Following a breadth-first enumeration and what they called an

"admissible support measure", they are able to prune candidate patterns without checking their support, significantly reducing the search space. MARGIN is another maximal subgraph mining algorithm that focuses on the more promising nodes in a graph (Thomas et al. 2006). This is accomplished by searching for promising nodes in the search space along the "border" of frequent and infrequent subgraphs, thus reducing the number of candidate patterns.

The goal of SUBDUE is to return the substructures that best compress the graph (Holder et al. 1994). 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 which subgraphs best compress the description length of the input graph. The core of the SUBDUE algorithm is in its compression strategy. After extending each subgraph pattern by one edge, it evaluates each extended subgraph 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 subgraphs that can compress or a user-specified limit is reached.

While each of these approaches is successful at pattern discovery, we will use the SUBDUE compression evaluation technique as the basis for our underlying discovery of the normative patterns. While the gSpan application is not publicly available, there are a few reasons why we found FSG to not be an ideal candidate for our implementation. One reason for our choice of pattern learner lies with the format expected by the FSG application. SUBDUE can effectively discover normative patterns whether it is given all transactions or data as one entire graph, or if each transaction is defined as individual subgraphs. As a graph data miner, FSG shows the frequency of a pattern based upon the number of transactions defined in the graph input file. So, if a graph is not delineated by individual transactions, the frequency of every pattern is 1, and thus very difficult to determine which pattern is the most frequent. However, in some later work by Kuromachi and Karypis, they improve upon this with an approach called Grew that is able to better handle large graphs that consist of connected subgraphs (Kuromachi and Karypis 2004). Another reason we prefer SUBDUE lies in the FSG approach to determining the normative pattern based upon frequency. While tests on various graphs showed SUBDUE and FSG returned the same normative pattern, when the tests involved a graph where the normative pattern is not found across all transactions (e.g., noise), the frequent pattern is not found unless the FSG support percentile is reduced. The issue then is knowing what support percentile should be used for a specific run. Specifying 100% support will result in the normative pattern being lost if the pattern is not found in every transaction, while using a lower percentile may result in other (smaller) normative patterns being found. In short, SUBDUE allows us to find the normative pattern in data that may be less regular or contain some noise. As will be shown in the following section, this is critical to the success of discovering anomalies.

3. Graph-based Anomaly Detection

Before we lay the groundwork for our definition of a graph-based anomaly, we need to put forth a framework for the definition of a graph. In general, 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 definitions (Gross and Yellen 1999) (West 2001):

Definition: A graph $G = (V, E, L)$ is a mathematical structure consisting of three sets V, E and L. The elements of V are called *vertices* (or nodes), the elements of E are the *edges* (or links) between the vertices, and the elements of L are the string *labels* assigned to each of the elements of V and E.

Definition: A vertex (or node) is an entity (or item) in a graph. For each vertex there is a labeled vertex pair (v, l) where v is a vertex in the set V of vertices and l is a string label in the set L of labels.

Definition: An edge (or link) is a labeled relation between two vertices called its endpoints. For each edge there is a labeled edge pair (e, l) where e is an edge in the set E of edges and l is a string label in the set L of labels.

Definition: An edge can be directed or undirected. A *directed* edge is an edge, one of whose endpoints is designated as the *tail*, and whose other endpoint is designated as the *head*. An *undirected* edge is an edge with two unordered endpoints. A multi-edge is a collection of two or more edges having identical endpoints.

Much research has been done recently 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 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 allows us to discover *structural* properties in data that are not evident using traditional data mining methods.

The idea behind the approach presented in this work is to find anomalies in graph-based data where the anomalous subgraph (at least one edge or vertex) in a graph is part of (or attached to or missing from) a non-anomalous subgraph, or the *normative pattern*. 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 directly deal with this scenario.

Definition: Given a graph G with a normative substructure S, a subgraph S', and a difference d between S and S', let $C(d)$ be the cost of the difference and $P(d)$ be the probability of the difference. Then the subgraph S' is considered anomalous if $0 < A(S') \leq X$, where X is a user-defined threshold and $A(S') = C(d) * P(d)$ is the anomaly score.

The importance of this definition lies in its relationship to fraud detection (i.e., any sort of 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 to be as close to legitimate actions as possible. That makes this definition of an anomaly extremely relevant.

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

1. *The label on a vertex is different than was expected.*
2. *The label on an edge is different than was expected.*
3. *A vertex exists that is unexpected.*
4. *An edge exists that is unexpected.*
5. *An expected vertex is absent.*
6. *An expected edge between two vertices (or a self-edge to a vertex) is absent.*

These same situations can also be applied to a subgraph (i.e., multiple vertices and edges), and will be addressed as such. In essence, there are three general *categories of anomalies*: modifications, insertions and deletions. Modifications would constitute the first two situations;

insertions would consist of the third and fourth situations; and deletions would categorize the last two situations.

4. GBAD Approach

Most anomaly detection methods use a supervised approach, which requires a 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 in the GBAD (Graph-based Anomaly Detection) system. GBAD is an *unsupervised* approach, based upon the SUBDUE graph-based knowledge discovery system (Cook and Holder 2000). 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 (subgraph), 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.

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. The GBAD-MDL algorithm uses a Minimum Description Length (MDL) heuristic to discover the best substructure in a graph, and then subsequently examines all of the instances of that substructure that "look similar" to that pattern. The detailed GBAD-MDL algorithm is as follows:

1. **Given input graph** *G:*
2. **Find the top-*k* substructures** S_i, that minimize $M(S_i,G)$, where S_i is a subgraph of *G*.
3. **Find all close-matching instances** I_j of S_i such that the cost $C(I_j,S_i)$ of transforming I_j to match the graph structure of S_i is greater than 0.
4. **Determine anomalous value** for each I_j by building the substructure definition S_j, finding all *exact* matching instances of S_j such that $F(I_j)$ is the frequency of instances that match I_j, and calculating the value $A(I_j) = F(I_j) * C(I_j,S_i)$, where the lower the value, the more anomalous the instance.
5. **Output** all I_j minimizing $A(I_j)$.

The cost of transforming a graph A into an isomorphism of a graph B is calculated by adding 1.0 for every vertex, edge and label that would need to be changed in order to make A isomorphic

to B. The result will be those instances that are the "closest" (without matching exactly) in structure to the best structure (i.e., compresses the graph the most), where there is a tradeoff in the cost of transforming the instance to match the structure, as well as the frequency with which the instance occurs. Since the cost of transformation and frequency are independent variables, multiplying their values together results in a combinatory value: the lower the value, the more anomalous the structure.

It should be noted that the value of substructure *S* will include the instances that do not match exactly. It is these inexact matching instances that will be analyzed for anomalousness. It should also be noted that we are only interested in the top substructure (i.e., the one that minimizes the description length of the graph), so *k* will always be 1. However, for extensibility, the *k* can be adjusted if it is felt that anomalous behavior may be found in more than one normative pattern.

The GBAD-P algorithm also uses the MDL evaluation technique to discover the best substructure in a graph, but instead of examining all instances for similarity, this approach examines all *extensions* to the normative substructure (pattern), looking for extensions with the lowest probability. The subtle difference between the two algorithms is that GBAD-MDL is looking at instances of substructures with the same characteristics (i.e., size, degree, etc.), whereas GBAD-P is examining the probability of extensions to the normative pattern to determine if there is an instance that when extended beyond its normative structure is traversing edges and vertices that are probabilistically less likely than other possible extensions.

The detailed GBAD-P algorithm is as follows:

1. **Find the top-*k* substructures** S_i, that minimize $M(S_i,G)$, where S_i is a subgraph of G.
2. **Compress** G by S_i, and **Find the top-k substructures** again.
3. **Find all instances** I_j that match the substructure S_i.
4. **Create extended instances** $I_j^{'}$ that consist of an original instance with an additional extension of an edge and a vertex, such that $I_j \subseteq I_j^{'}$, and $I_j^{'} \subseteq I^{'}$, where $I^{'}$ is the set of all extended instances of S_i.
5. **Determine anomalous value** for each $I_j^{'}$ by finding matching instances of $I_j^{'}$ in set $I^{'}$, and calculating the value of $A(I_j^{'}) = \left|I_j^{'}\right| / \left|I^{'}\right|$ where $\left|I_j^{'}\right|$ is the cardinality of the set of instances that match $I_j^{'}$, and $\left|I^{'}\right|$ is the cardinality of the set of extended instances of S_i.
6. **Output** $I_j^{'}$ minimizing $A(I_j^{'})$ and where $A(I_j^{'})$ is less than a user acceptable threshold.
7. **Compress** G by the graph structure of $I_j^{'}$.
8. **Repeat** step 1 and then start again at step 3.

$A(I_j^{'})$ is the *probability* that a given instance should exist given the existence of all of the extended instances. Again, the lower the value, the more anomalous the instance. Given that

$|I^{'}|$ is the total number of possible extended instances, $|I_j^{'}|$ can never be greater, and thus the value of $A(I_j^{'})$ will never be greater than 1.0.

The GBAD-MPS algorithm again uses the MDL approach to discover the best substructure in a graph, then it examines all of the instances of *parent* (or ancestral) substructures that are missing various edges and vertices. The value associated with the parent instances represents the cost of transformation (i.e., how much change would have to take place for the instance to match the best substructure). Thus, the instance with the lowest cost transformation (if more than one instance have the same value, the frequency of the instance's structure will be used to break the tie if possible) is considered the anomaly, as it is closest (maximum) to the best substructure without being included on the best substructure's instance list.

The detailed GBAD-MPS algorithm is as follows:

1. **Find the top-*k* substructures** S_i, that minimize $M(S_i, G)$, where S_i is a subgraph of G.
2. **Find all ancestor** substructures S' such that $S' \subseteq S_i$.
3. **Find all instances** I' of S'.
4. **Determine the anomalous value** for each instance I' as $A(I') = F(I') * C(I', S_i)$.
5. **Output** I' as an anomalous instance if its anomalous value is less than a user specified threshold.

By allowing the user to specify a threshold, we can control the amount of "anomalousness" that we are willing to accept. By our definition of an anomaly, we are expecting low transformation costs (i.e., few changes for the anomalous instance to match the normative substructure).

5. Experimental Results

We have performed several experiments evaluating GBAD on both synthetic and real-world data sets. Results on synthetic data show that GBAD is able to accurately detect different types of target anomalies with low false-positive rates. Results on real-world data show the types of anomalies that can be detected in data describing network intrusions, email communications, cargo shipments, cell-phone communications, and network traffic.

5.1. Synthetic Data

We constructed numerous synthetic graphs that were randomly generated based on the following parameters.

- AV is the number of anomalous vertices in an anomalous substructure
- AE is the number of anomalous edges in an anomalous substructure
- V is the number of vertices in the normative pattern
- E is the number of edges in the normative pattern

Each synthetic graph consists of substructures containing the normative pattern (with V number of vertices and E number of edges), connected to each other by one or more random connections, and each test anomaly consists of AV number of vertices and AE number of edges altered.

For *modification* anomalies: an AV number of vertices and AE number of edges, from the same randomly chosen normative instance, have their labels modified to randomly-chosen, non-duplicating labels (e.g., we do not replace a vertex labeled "X" with another vertex labeled "X").

For *insertion* anomalies: randomly inserted AV vertices and AE edges, where the initial connection of one of the AE edges is connected to either an existing vertex in a randomly chosen normative instance, or to one of the already inserted AV vertices.

For *deletion* anomalies: randomly chosen AV vertices and AV edges, from a randomly chosen normative instance, are deleted along with any possible "dangling" edges (i.e., if a vertex is deleted, all adjacent edges are also deleted).

Due to our definition of an anomaly, all tests will be limited to changes that constitute less than 10% of the normative pattern. Again, since anomalies are supposed to represent slight deviations in normal patterns, an excessive change to a pattern is irrelevant. However, in order to analyze the effectiveness of these approaches beyond the upper bounds, we will also perform some tests at deviations above 10%.

Each of the above is repeated for each algorithm, varying sizes of graphs, normative patterns, thresholds, iterations and sizes of anomalies (where the size of the anomaly is $|AV| + |AE|$). Also, due to the random nature in which structures are modified, each test will be repeated multiple times to verify its consistency.

5.1.1. Metrics

Each test consists of a single graph from which 30 randomly-altered graphs are generated. The output shown consists of the average of the results of running the algorithms against those 30 graphs for the specified settings. The primary three metrics calculated are:

1. Percentage of runs where the *complete* anomalous substructure was discovered.
2. Percentage of runs where at least *some* of the anomalous substructure was discovered.
3. Percentage of runs containing *false positives*.

After the algorithm has completed, the first metric represents the percentage of success when comparing the results to the known anomalies that were injected into the data. If all of the anomalies are discovered for a particular run, that is counted as a success for that run. For example, if 27 out of the 30 runs found all of their anomalies, the value for this metric would be 90.0.

The second metric represents the percentage of runs where at least one of the injected anomalies was discovered. For example, if the anomaly consisted of 3 vertices and 2 edges that had their labels changed, and the run reported one of the anomalous vertices, then that run would be considered a success. Obviously, this metric will always be at least as high as the first metric.

The last metric represents the percentage of runs that reported at least one anomaly that was not one of the injected anomalies. Since it is possible that multiple reported anomalous instances could have the same anomalous value, some runs may contain both correct anomalies and false ones. Further tuning of these algorithms may enable us to discover other measurements by which we could "break the tie" when it comes to calculating an anomalous score.

5.1.2. Information Theoretic Results: GBAD-MDL

Figures 1 and 2 show the effectiveness of the GBAD-MDL approach on graphs of varying sizes with random anomalous modifications. In these figures, the X axis represents the thresholds, the Y axis is the percentage of anomalies discovered, and the Z axis indicates the sizes of the normative patterns, graphs and anomalies. For example, "10/10/100/100-1v" means a normative pattern with 10 vertices and 10 edges in a graph with 100 vertices and 100 edges and an anomaly consisting of a modification to 1 vertex. (Only a portion of the results are shown for space reasons, and other tests showed similar results.)

In the small synthetic test, when the threshold is high enough, (i.e., the threshold is equal to or higher than the percentage of change), it is clear that this approach was able to find all of the anomalies. The only time false positives are reported is when the threshold is 0.2. For a threshold of 0.2, we are basically saying that we want to analyze patterns that are up to 20% different. Such a huge window results in some noise being considered (along with the actual anomalies, as all of the anomalous instances are discovered). Fortunately, our definition of what is truly an anomaly would steer us towards observing runs with lower thresholds.

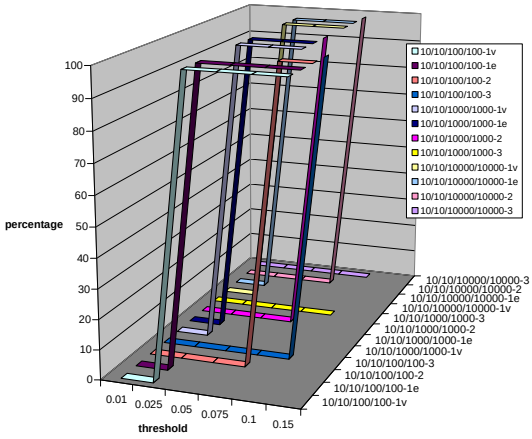

Figure 1. Percentage of GBAD-MDL runs where all anomalies discovered.

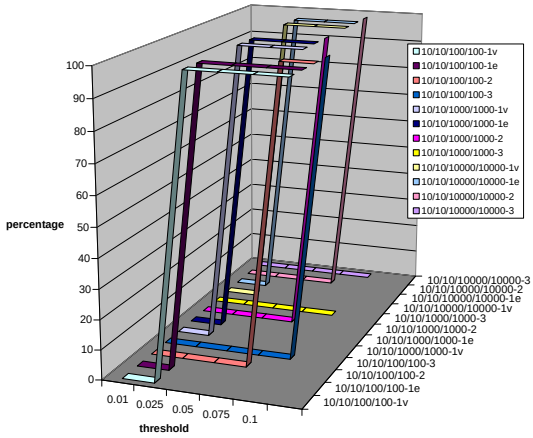

Figure 2. Percentage of GBAD-MDL runs where at least one anomaly is discovered.

5.1.3. Probabilistic Results: GBAD-P

Figure 3 shows the effectiveness of the GBAD-P approach on graphs of varying sizes with random anomalous insertions. It should be noted in this example that even though unrealistic anomaly sizes were used (representing 20-30% of the normative pattern), this approach is still effective. This same behavior can be observed in larger graphs as well.

As a further experiment, we also tried this approach on different distributions, varying the number of vertices versus the number of edges (e.g., adding more edges than vertices by creating more edges between existing vertices), and also lessening the distribution difference between

noise and anomalies. In all cases, the results were relatively the same, with never less than 96.67% of the anomalous instances being found for anomalies of size 8 (40% of the normative pattern) or less, with the lowest discovery rate being 90% for an anomaly of size 10 (50% of the normative pattern).

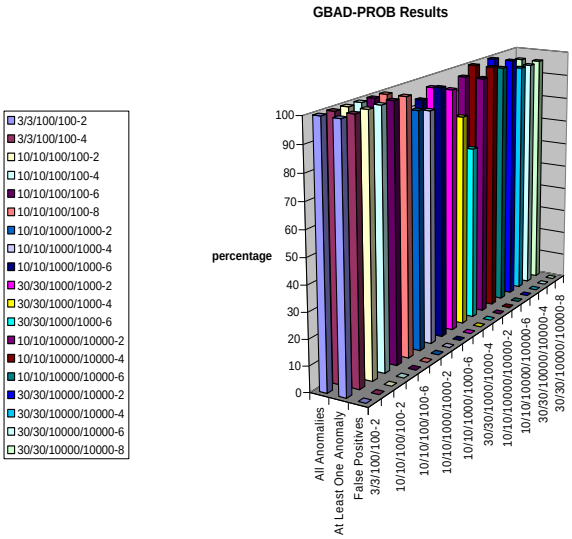

Figure 3. GBAD-P results on anomalous insertions.

5.1.4. Maximum Partial Substructure Results: GBAD-MPS

For all tests, across all sizes of graphs and anomalies, the GBAD-MPS algorithm is able to discover all of the anomalous deletions while at the same time reporting no false positives. Initially, the results from the runs on the graph with 1000 vertices and 1000 edges, where the normative pattern consists of 30 vertices and 30 edges, were not good. However, when we increase the number of substructures (to analyze) to 100, and increase the anomalous threshold (i.e., cost of transformation * frequency) to 50.0, the results improve. So, a good rule of thumb is to choose an anomalous threshold based upon the size of the normative pattern. For instance, GBAD could be run first to determine the normative pattern, then based upon the size of the normative pattern, we can determine the maximum size of an anomaly (e.g., around 10%), choose a cost of transformation that would allow for the discovery of an anomaly that size, and then when we rerun the algorithm with the new threshold, the result is complete discovery.

It should be noted that the reason the number of best substructures and the threshold had to be increased is that as the size of the anomaly grows (i.e., the number of vertices and edges deleted increases), the further away the cost of transformation for the anomalous instance is from the normative pattern.

5.1.5. Performance

One of the factors to consider in evaluating these algorithms is their respective performances. Table 1 represents the average running times (in seconds) for each of the algorithms against varying graph sizes for the anomaly types that were the most effectively discovered (i.e., the types of anomalies that each algorithm was targeted to discover). Overall, the running times were slightly shorter for the non-targeted anomalies, as the algorithms for the most part did not have any anomalies to process.

Table 1 Running-times of algorithms (in seconds).

Graph Size (Normative Pattern Size) / Algorithm (Anomaly Type)	100v 100e (6)	100v 100e (20)	1,000v 1,000e (20)	1,000v 1,000e (60)	10,000v 10,000e (20)	10,000v 10,000e (60)
GBAD-MDL (Modification)	0.05-0.08	0.26-15.80	20.25-55.20	31.02-5770.58	1342.58-15109.58	1647.89-45727.09
GBAD-P (Insertion)	1.33	0.95	30.61	18.52	745.45	2118.99
GBAD-MPS (Deletion)	0.14	0.07	4.97	75.59	242.65	813.46

Because the GBAD-MDL algorithm uses a matching threshold, the performance of the algorithm is dependent upon the threshold chosen. The higher the threshold, the longer the algorithm takes to execute, so there is a trade-off associated with the threshold choice. Even on graphs of 10,000 vertices and 10,000 edges, the running times varied anywhere from 1342 seconds to 45,727 seconds, depending upon the threshold chosen. The GBAD-MDL algorithm is tractable given the input parameters. Because of the graph matching that is performed, and the fact that the GBAD-MDL algorithm needs to examine more than just instances of the normative pattern, the user-defined threshold provides a means by which the threat detection analyst can decide how much of the entire graph they want to analyze. The larger the graph, as well as the number of non-matching subgraphs one wants to analyze for anomalous structure, the greater the runtime for this algorithm. Future work on this algorithm will involve a reduction in the number of graph matches by reducing the number of subgraphs that are relevant in terms of matching.

The ability to discover the anomalies is sometimes limited by the *resources* allocated to the algorithm. Given a graph where the anomalous substructure consists of the *minimal* deviation from the normative pattern, if a sufficient amount of processing time and memory is provided, all of these algorithms will discover the anomalous substructure with no false positives. However, the ability to discover anomalies (per our definition) is also hampered by the amount of *noise* present in the graph. The issue is that if noise is a smaller deviation from the normative pattern than the actual anomaly, it may score higher than the targeted anomaly (depending upon the frequency of the noise). Of course, one might say that noise is an anomaly in that it is not normal; however, it is probably not an insider threat, which is the goal of these approaches.

Now, the presence of noise does not eliminate the algorithms' abilities to discover the anomalous substructure. It only results in more false positives being detected if the anomalous score of the noisy structure is better than the desired anomalous substructure. That is where another trade-off is necessary that can be found in most threat detection systems: adjusting thresholds to find a balance of false-positives versus true anomalies. Future work in this area will include not only improved heuristics to reduce the number of graph matches that is performed, but also algorithmic changes to analyze the distribution of vertex and edge labels (especially numeric values) that will aid in the differentiation between seemingly similar labels, thereby reducing the effect of noise.

5.2. Real-world Datasets

5.2.1. Network Intrusion

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 (KDD Cup 1999).

In 1998, MIT Lincoln Labs managed the DARPA Intrusion Detection Evaluation Program. The objective was to survey and evaluate research in intrusion detection. The standard data set consisted of a wide variety of intrusions simulated in a military network environment. The 1999 KDD Cup intrusion detection dataset consists of a version of this data. For nine weeks, they simulated a typical U.S. Air Force local-area network, initiated multiple attacks, and dumped the raw TCP data for the competition.

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. The graph representation consisted of a central "record" vertex with 31 outgoing edges, labeled with the feature names, connecting to vertices labeled with the value for that feature. In the 1999 competition, the data was split into two parts: one for training and the other for testing. Groups were 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 ran 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 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, so 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 detection data. Using GBAD-MDL, our ability to discover the 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 an even worse 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.

In order to better understand the effectiveness of the GBAD algorithms on intrusion detection data, we will compare our results with the graph-based approaches of Noble and Cook (Noble and Cook 2003). They proposed two approaches to discovering anomalies in data represented as a graph. Their *anomalous substructure detection* method attempts to find unusual substructures within a graph by finding those substructures that compress the graph the least, compared to our GBAD-MDL approach which uses compression to determine which substructures are closest to the best substructure (i.e., the one that compresses the graph the most). In their results, they use the inverse of the ratio of true anomalies found over the total number of anomalies reported, where the lower the value (i.e., a greater percentage of the reported anomalies are the network attacks), the more effective the approach for discovering anomalies.

The other approach presented is what they call *anomalous subgraph detection*. The objective with this method is to compare separate structures (subgraphs) and determine how anomalous each subgraph is compared to the others. This is similar to our approaches in that the minimum description length is used as a measurement of a substructure's likelihood of existence within a graph. However, in order to implement this approach, the graph must be divided into clearly defined subgraphs so that a proper comparison can be performed. The basic idea is that every subgraph is assigned a value of 1, and the value decreases as portions of the subgraph are compressed away. In the end, the subgraphs are ranked highest to lowest, with the higher the value (i.e., closest to, or equal to, 1), the more anomalous the subgraph. While this works well on intrusion detection data, their approach is restricted to domains where a clear delineation (i.e., subgraphs) must be defined. In other words, the delineation occurs when a graph can be sub-divided into distinct subgraphs, with each subgraph representing a common entity. For example, in domains like terrorist or social networks, this type of delineation may be difficult and subjective.

Using the same set of KDD Cup intrusion detection data as set forth previously, we can compare GBAD-MDL (since it performed the best on this set of data) to both of these approaches, using the same anomalous attack ratios used by Noble and Cook. The ratio used in their work is an inverse fraction of correctly identifying the attacks among all of the attacks reported. For example, if 10 anomalies are reported, but only 1 of them is the actual attack, then the fraction is 1/10, and the inverse is the score of 10, where obviously the lower the score the better. Their anomalous substructure detection method achieves an average anomalous ratio of 8.64, excluding the *snmpgetattack* and *snmpguess* attacks, while using the same scoring ratio, GBAD-MDL generates an average of 7.22 *with snmpgetattack* and *snmpguess* included. In Noble and Cook's paper, both the *snmpgetattack* and *snmpguess* attacks were excluded from the anomalous substructure detection approach results because they had high average attack values of around 2211 and 126 respectively (i.e., too many false positives). However, GBAD-MDL is much more successful at discovering these two attack types, as their respective averages are 8.55 and 7.21. Then, for their anomalous subgraph detection approach, they get an average ranking of 4.75, whereas the GBAD-MDL algorithm is able to achieve a better average ranking of 3.02. Figure 4 shows the ranking results for each of the different attack types.

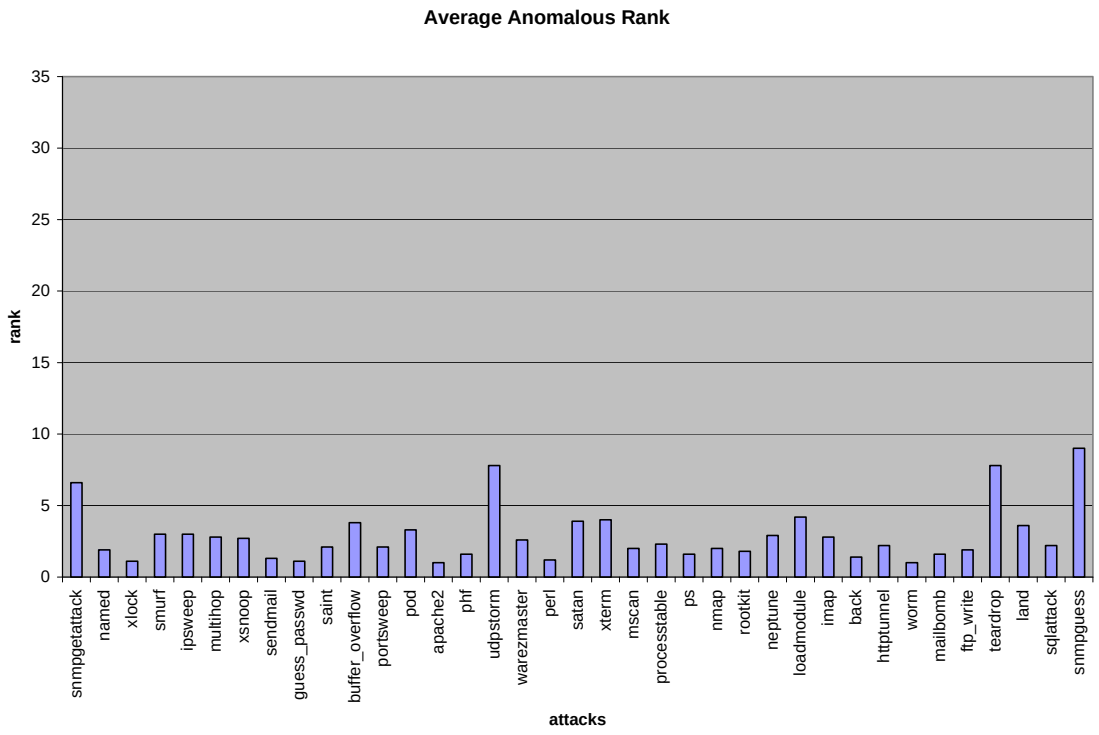

Figure 4. Average anomalous ranking using GBAD-MDL on KDD intrusion detection data.

These results, when compared to the ones presented in Noble and Cook's paper (Noble and Cook 2003), not only show an overall average improvement, but also again show a significant improvement when it comes to effectively discovering the *snmpgetattack* and *snmpguess* attacks, which both had values over 20 using the anomalous subgraph detection approach, whereas the GBAD-MDL algorithm was under 10 for both attack types. It should also be noted that the false positives are mostly due to the fact that we have to increase the anomalous threshold in order to detect some of the anomalous patterns. Unlike our assumption that anomalies are small deviations from the normative pattern, several of the attack records are actually large deviations from the norm.

5.2.2. Enron E-mail

One of the more recent domains that has become publicly available is the data set of e-mails between management from the Enron corporation. The Enron e-mail dataset was made public by the Federal Energy Regulatory Commission during its investigation. After subsequent data integrity resolutions, as well as the removal of some e-mails due to requests from affected employees, William Cohen at CMU put the dataset on the web for researchers. From that dataset, Shetty and Adibi further cleaned the dataset by removing duplicate e-mails, and putting the final set into a publicly available database (http://www.isi.edu/~adibi/Enron/Enron.htm). This dataset contains 252,759 messages from 151 employees distributed in approximately 3000 user-defined folders.

This Enron e-mail database consists of messages not only between employees but also from employees to external contacts. In addition to providing the e-mails, the database also consists of

employee information such as their name and e-mail address. However, since we do not have information about their external contacts, we decided to limit our graph to the Enron employees and just their correspondences, allowing us to create a more complete "social" structure. In addition, since the body of e-mails consists of many words (and typos), we limited the textual nature of the e-mails to just the subject headers. From these decisions, we created a graph consisting of the structure shown in Figure 5. The message vertex can have multiple edges to multiple subject words, and multiple recipient type edges (i.e., TO, CC and BCC) to multiple persons. Running the GBAD algorithms on this data set produced the small normative pattern shown in Figure 6.

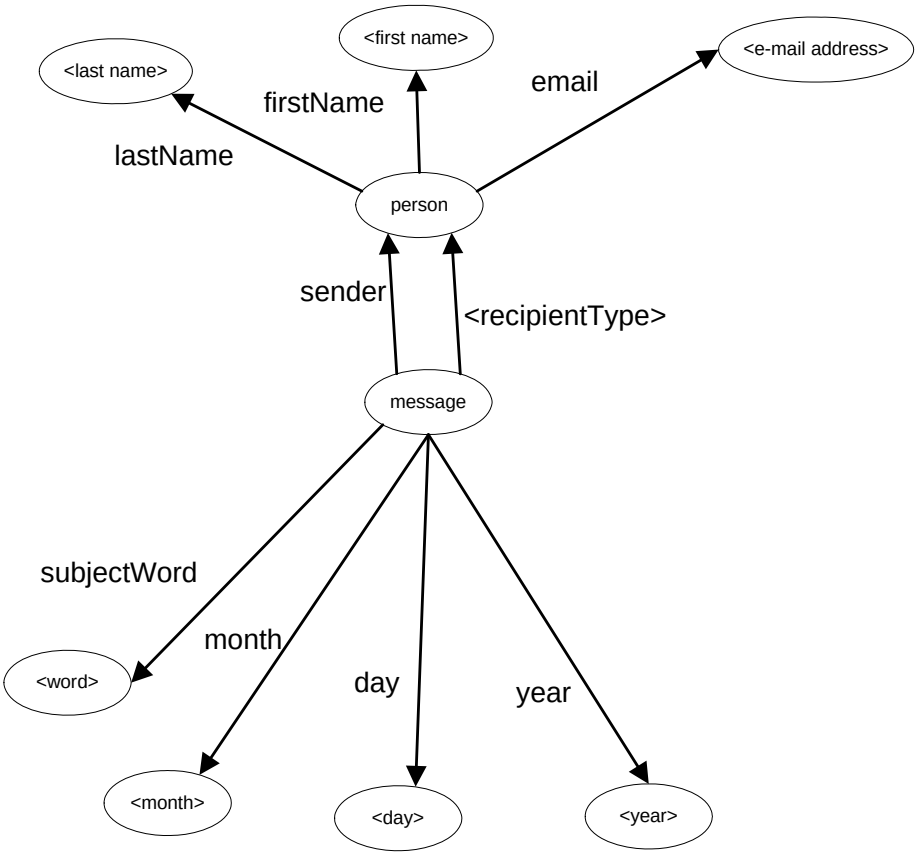

Figure 5. Graphical representation of Enron e-mail.

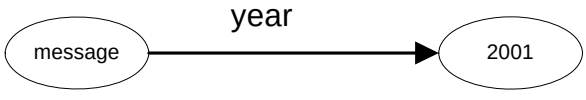

Figure 6. Normative pattern from Enron e-mail data set.

This was an expected normative pattern because most of the e-mail was from the year 2001, with little regularity beyond the fact that messages were sent. Considering the small size of the normative pattern, we did not run the GBAD-MDL and GBAD-MPS algorithms, as clearly

nothing of importance would be derived from a modification or deletion to this normative pattern. However, running the GBAD-P algorithm resulted in the substructure shown in Figure 7.

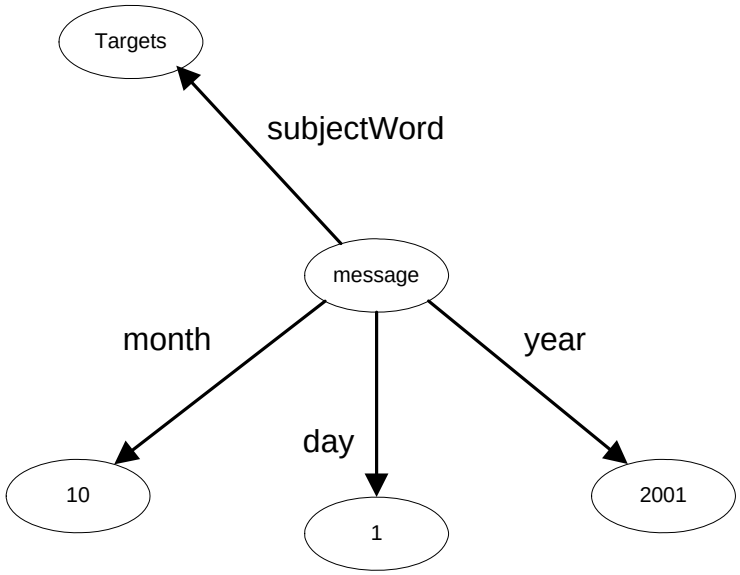

Figure 7. Results from running GBAD-P on Enron e-mail data set.

It is interesting to point out that 859 messages were sent on October 1, 2001, and of all of the messages in the data set, this was the only one with a subject of "Targets", and 1 of only 36 messages in the entire dataset that had the word "Targets" anywhere in its subject line, and no messages anywhere that were a response to this message.

Also, analyzing just the transfer of e-mails between the employees, GBAD has been able to discover various anomalous patterns as they relate to specific individuals. For instance, in the case of Enron employee Mark Taylor (one of the employees with a high volume of e-mail traffic), the *normative* pattern consists of sending e-mails to three other employees. While the employees that are involved in e-mails with Mr. Taylor varies, the GBAD-P algorithm identified one instance as anomalous because it included a link to another employee, Tana Jones, who was not found anywhere else to have been part of a correspondence that contained this normative pattern associated with Mr. Taylor. While Ms. Jones is involved in many e-mails in the Enron dataset, the *structural* anomaly associated with the normative pattern of communication for Mr. Taylor results in GBAD detecting this instance of an anomalous substructure.

5.2.3. Cargo Shipments

One area that has garnered much attention recently is the analysis and search of imports into the United States. A large number of imports into the U.S. arrive via ships at ports of entry along the coast-lines. Thousands of suspicious cargo, whether it be 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 Customs Border and Protection Agency (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. Figure 8 shows a portion of the actual graph that we used in our experiments.

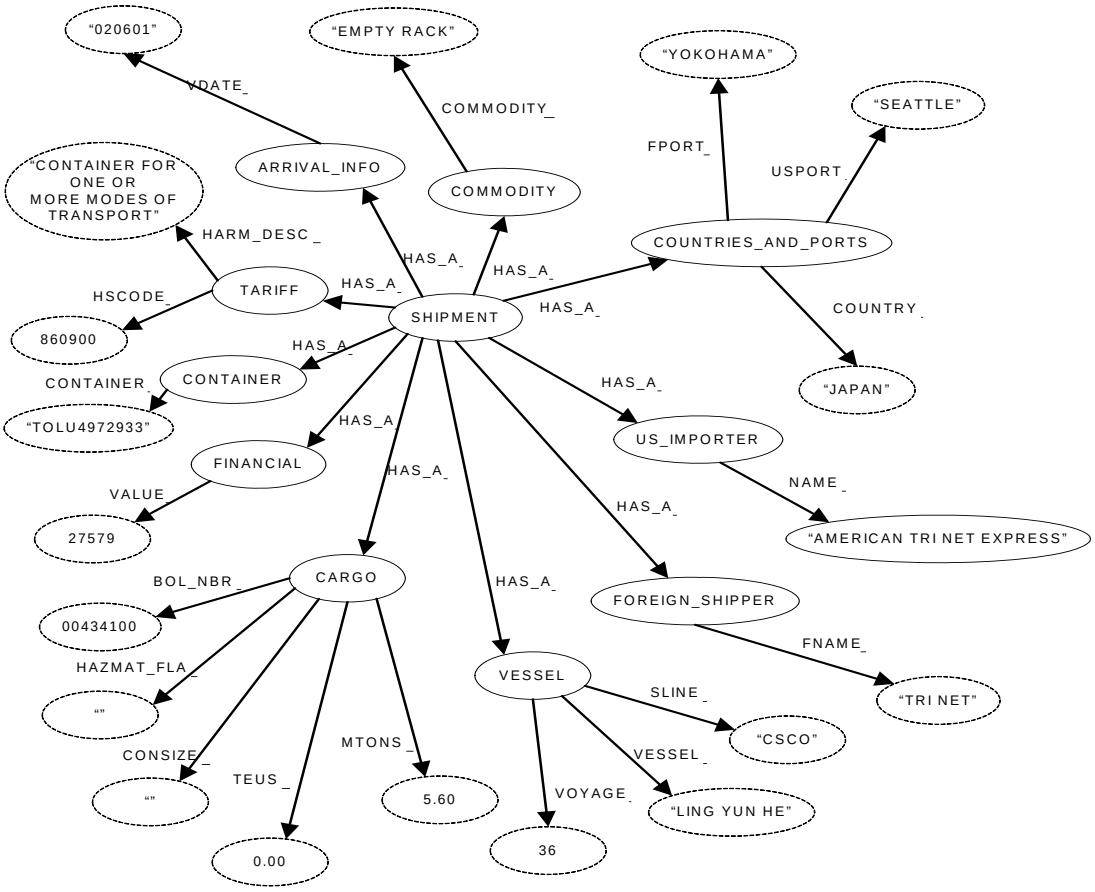

Figure 8. Example graph of cargo information.

While we were not given any labeled data from the CBP, we can draw some results from simulations of publicized incidents. Take for instance the example from a press release issued by the U.S. Customs Service. The situation was that almost a ton of marijuana was seized at a port in Florida (US Customs Service 2000). 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 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.

There are many different non-graph-based machine learning approaches to anomaly detection. In order to compare our results to a non-graph-based approach, we chose perhaps the most popular approach to anomaly detection - the class of approaches known as *clustering*. The idea behind clustering is the grouping of similar objects, or data. Clustering is an unsupervised approach whose goal is to find all objects that are similar where the class of the example is unknown (Frank and Witten 2005). From an anomaly detection perspective, those objects that fall outside a cluster (*outliers*), perhaps within a specified deviation, are candidate anomalies. Due to the popularity of this approach, and the fact that it is an unsupervised approach (like

GBAD), we evaluate the effectiveness of the simple k-Means clustering approach on the cargo shipment data using the WEKA tool (WEKA). For the simple k-Means approach (*SimpleKMeans*), given a set of n data points in d-dimensional space R^d, and an integer k, the problem is to determine a set of k points in R^d, called centers, so as to minimize the mean squared distance from each data point to its nearest center (Kanungo et al. 2000).

First, we randomly select 200 cargo records and generate a graph from the chosen records. Second, we run the graph through SUBDUE to determine the normative pattern in the graph. Then, we generate multiple anomalies of each of the anomaly types (modifications, insertions and deletions), where each of the induced anomalies is similar to the real-world examples mentioned earlier (e.g., change in the name of a port), and the anomaly is part of a normative pattern. Only choosing anomalies that are small deviations (relative to the size of the graph), all three of the GBAD algorithms are able to successfully find all of the anomalies, with only one GBAD-MDL test and one GBAD-P test reporting false positives.

Using these same 200 records, we then convert the data into the appropriate WEKA format for the k-Means algorithm. For each of the tests involving what were vertex modifications in the graph, and are now value or field modifications in the text file, the k-Means algorithm is able to successfully find all of the anomalies. Similar to how we have to adjust GBAD parameters, we have to increase the default WEKA settings for the number of clusters and the seed, as the defaults do not produce any anomalous clusters on the cargo test set which consists of 12 attributes for 200 records. By increasing the number of clusters to 8 and the seed to 31, we are able to discover the anomalous modifications, as shown in the example in Figure 9.

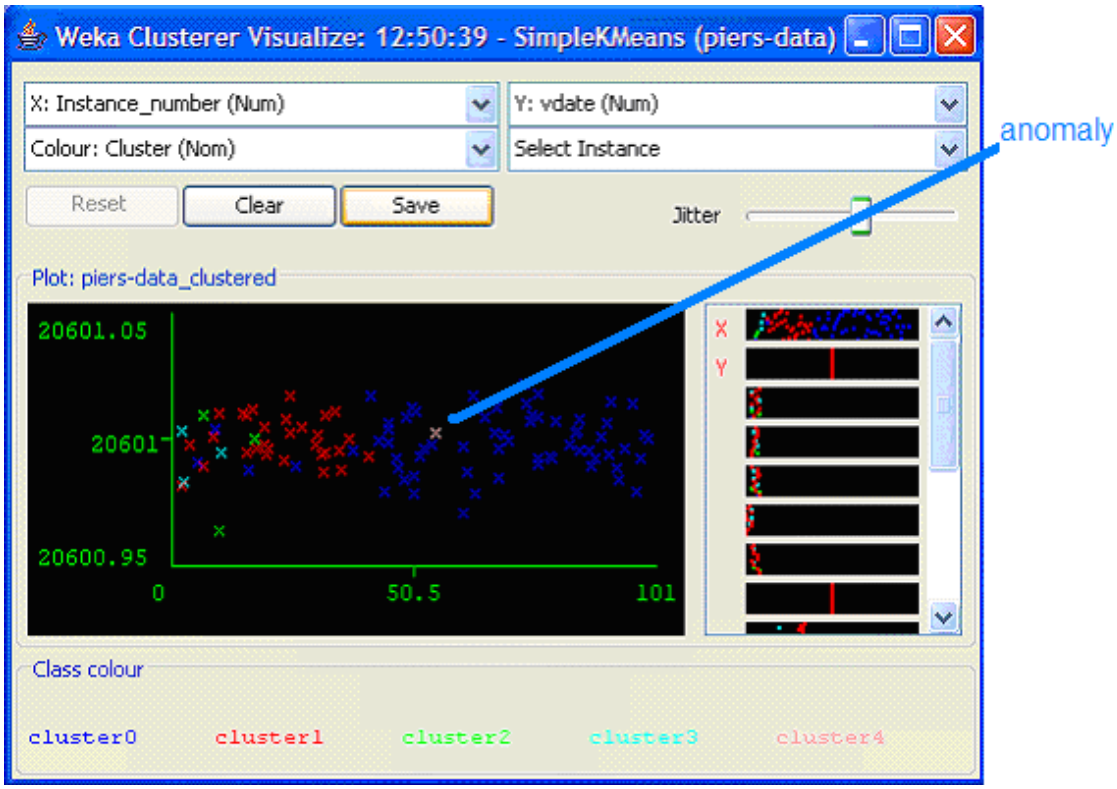

Figure 9. Results from k-Means clustering algorithm on cargo data with anomalous modification.

In these tests, a cluster is considered anomalous if it contains only a single instance.

However, for insertions and deletions, the k-Means approach is not effective, but in some ways, that is to be expected. The k-Means algorithm assumes that every specified record is of the same length. So, in order to simulate anomalous insertions, extra attributes must be added to represent the extra vertices, where the values for those attributes are NULL, unless an anomalous insertion is present. Yet, despite the additional non-NULL attribute when the anomaly is present, the k-Means algorithm never reports an anomalous cluster for any of the tests. When we increase the number of clusters and the seed, it only increases the number of false positives (i.e., clusters of single instances that are not the anomalous instances). This is surprising in that we would have assumed that the unique value for an individual attribute would have been discovered. However, again, we are attempting to simulate a structural change, which is something that the k-Means algorithm (or other traditional clustering algorithms) is not intended to discover.

Similarly, we can only simulate an anomalous deletion by replacing one of the record's attributed values with a NULL value. Again, at first, we would have considered this to be the same as a modification, and clearly identifiable by the k-Means algorithm. But, the algorithm was unable to find the anomalous deletion in any of the tests, which leads us to believe that the presence of a NULL value has an effect on the functionality of the k-Means algorithm. The importance of these tests is to show that for some anomalies (specifically modifications) traditional machine learning approaches like clustering are also effective, and at the same time, the inability to discover anomalous insertions and deletions further justifies the use of an approach like GBAD for structural anomalies. In addition, approaches like k-Means are only able to report the anomalous record – not the specific anomaly within the record.

The use of the k-Means clustering algorithm for anomaly detection and intrusion detection has been reported in other research efforts (Portnoy 1999)(Caruso and Malerba 2004). For more information on how the WEKA tools work, please refer to the WEKA website (WEKA).

We also compared our algorithms against a traditional non-graph-based anomaly detection approach found in the commercially available application called Gritbot, from a company called RuleQuest (http://www.RuleQuest.com/). The objective of the Gritbot tool is to look for anomalous values that would compromise the integrity of data that might be further analyzed by other data modeling tools.

There are two required input files for Gritbot: a .names file that specifies the attributes to be analyzed, and a .data file that supplies the corresponding comma-delimited data. There are several optional parameters for running Gritbot, of which the most important is the "filter level". By default, the filter level is set at 50%. The lower the filter level percentage, the less filtering that occurs, resulting in more possible anomalies being considered.

In order to compare Gritbot to our GBAD algorithms, we gave Gritbot the same cargo data files used in the previous experiments (formatted to the Gritbot specifications). Using the default parameters, no anomalies were reported. We then lowered the filter level to 0 (which specifies that all anomalies are requested). In every case, anomalies were reported, but none of the anomalies reported were the ones we had injected into the data set. So, we increased the number of samples from 200 shipments to ~1000 shipments, so that Gritbot could infer more of a statistical pattern, and then randomly injected a single modification to the country-of-origin attribute. In the cargo data files, all of the country-of-origins were "JAPAN", except for the randomly selected records where the value was changed to "CHINA". Again, Gritbot did not report this anomaly (i.e. 1020 cases of "JAPAN" and one case of "CHINA"), and instead reported a couple of other cases as anomalous.

While we consider the existence of a record with "CHINA" as anomalous, Gritbot does not view that as an anomaly. The issue is that Gritbot (and this is similar to other outlier-based approaches), does not treat discrete attributes the same as numeric attributes. This is because Gritbot views continuous distributions (such as "age") as a much easier attribute to analyze

because the distribution of values leads to certain expectations. While discrete distributions are more difficult because there is not a referential norm (statistically), it limits the tool's ability to provide its user with a comprehensive list of anomalies. That is not to say that Gritbot will not discover anomalous discrete values - it will if it can determine a statistical significance. For example, we found (when examining by hand) records that contained a significant number of identical attribute values (e.g., COUNTRY, FPORT, SLINE, VESSEL). In our data set, approximately 250 out of the approximately 1,000 records had identical SLINE values. When we arbitrarily modified the SLINE value of one of these cases from "KLIN" to "PONL" (i.e., another one of the possible SLINE values from this data set), Gritbot did not report the anomaly. When we changed it to "MLSL", Gritbot still did not report it. However, when we changed it to "CSCO", Gritbot reported that case as being anomalous (albeit, not the most anomalous). Why? This behavior is based on what Gritbot can determine to be statistically significant. Of all of the ~1,000 records, only 1 has an SLINE value of "MLSL", and only 3 have a value of "PONL". However, there are 123 records with an SLINE value of "CSCO". Thus, Gritbot was able to determine that a value of "CSCO" among those ~250 records is anomalous because it had enough other records containing the value "CSCO" to determine that its existence in these other records was significant. In short, the behavior depends upon the definition of what is an anomaly.

Gritbot's approach to anomaly detection is common among many other outlier-based data mining approaches. However, in terms of finding what we would consider to be anomalous (small deviations from the norm), Gritbot's approach typically does not find the anomaly.

5.2.4. Internet Movie Database (IMDb)

Another common source of data mining research is the Internet Movie Database (http://www.imdb.com/). This database consists of hundreds of thousands of movies and television shows, with all of the credited information such as directors, actors, writer, producers, etc. In their work on semantic graphs, Barthelemy et al. proposed a statistical measure for semantic graphs and illustrated these semantic measures on graphs constructed from terrorism data and data from the IMDb (Barthelemy et al, 2005). While they were not directly looking for anomalies, their research presented a way to measure useful relationships so that a better ontology could be created. Using bipartite graphs, Sun et al. presented a model for scoring the normality of nodes as they related to the other nodes (Sun et al, 2005). Using the IMDb database as one of their datasets, they analyzed the graph for just anomalous nodes.

In order to run our algorithms on the data, due to the voluminous amount of information, we chose to create a graph of the key information (title, director, producer, writer, actor, actress and genre) for the movies from 2006. Running the GBAD algorithms on this data set produced the normative pattern shown in Figure 10.

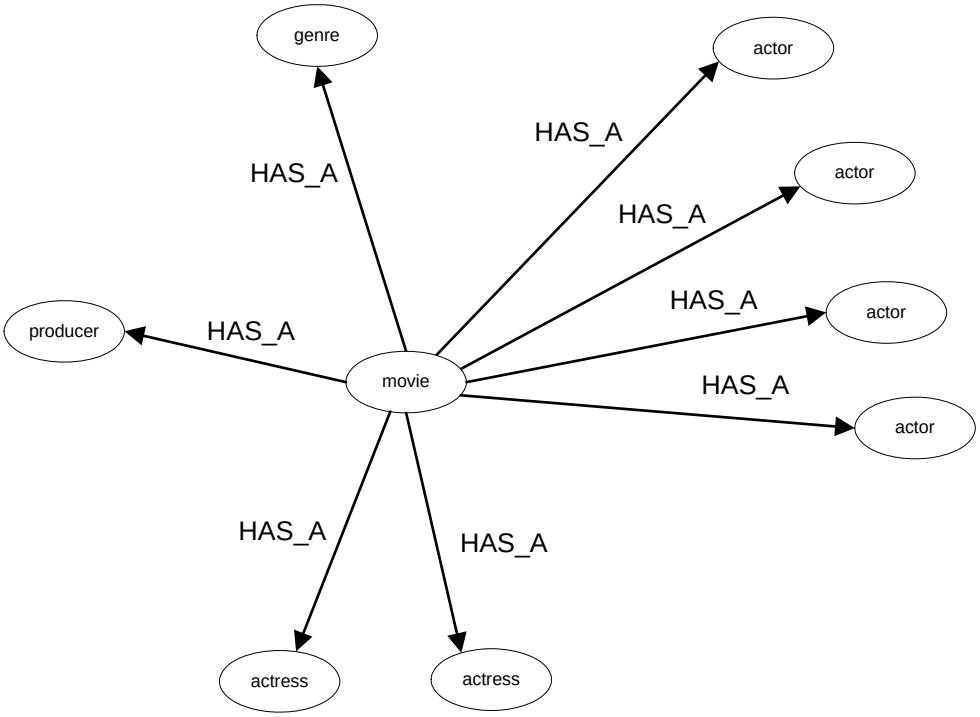

Figure 10. Normative pattern from graph representation of movie data.

This pattern is not surprising, as movies typically consist of multiple actors and actresses. However, because of the size of the database, we chose a beam width that would produce seemingly relevant substructures within a reasonable amount of time. For instance, all of our runs are made with a default beam width of 4, resulting in reasonable running-times of approximately 15 minutes each. In contrast, with a beam width of 7, the running-times are nearly tripled. However, this does result in some additional elements being discovered, like another producer and more actors, so there is always the trade-off of time versus knowledge discovery. One may also wonder why some of the other movie elements like director and writer were not discovered. (Even with the aforementioned beam width increase they are not discovered.) This is due to the fact that several genres are reality shows, which do not require directors and writers, and many documentaries do not have writers credited in the IMDb.

Running the GBAD-MDL and GBAD-MPS algorithms on the IMDb data produced a variety of anomalies that all scored equally. The GBAD-MDL algorithm reported a single anomalous instance where an actor label was replaced by an anomalous actress label. The GBAD-MPS algorithm reported multiple anomalous instances consisting of a writer replacing an actor, a writer replacing the producer, another genre replacing the producer, a director replacing the producer, another actor replacing the producer and another actress replacing the producer. For the GBAD-P algorithm, the anomalous extension consisted of the title of the movie. Considering every movie has a different title (in most cases), this was an expected anomalous extension.

5.2.5. VAST Cell-phone Data

As part of the IEEE Symposium on Visual Analytics Science and Technology (VAST), the VAST challenge each year presents a contest whereby the goal is to apply visual analytics to a provided set of derived benchmark data sets. For the 2008 challenge (http://www.cs.umd.edu/hcil/VASTchallenge08/), contestants are presented with four mini-

challenges, in addition to the grand challenge of addressing the complete data set. Yet, while the goal of the challenge is to target new visual analytics approaches, it is still possible to apply these graph-based anomaly detection algorithms to the same data sets. For instance, one of the data sets consists of cell-phone traffic between inhabitants of the fictitious island of Isla Del Sueño. The data consists of 9,834 cell-phone calls between 400 people over a 10-day period. The mini-challenge is to describe the social network of a religious group headed by Ferdinando Cattalano and how it changes over the 10-day period. The graph of the cell-phone traffic is represented as shown in Figure 11.

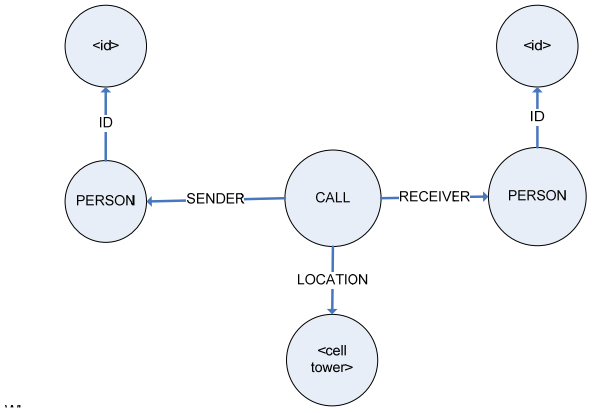

Figure 11. Graph representation of a cell-phone call from the VAST dataset.

Applying the GBAD algorithms to this information results in several structural anomalies within the data, especially when particular individuals are analyzed in terms of their calling patterns. For instance, when we look at the calling patterns of individuals who correspond with Ferdinando Cattalano, identified in the challenge as ID 200, one notices several anomalous substructures, including some who contact Ferdinando on days that are out of the ordinary, and even some individuals who call others outside of their normal chain of cell-phone calls. In addition, a graph of the *social network* of phone usage (i.e., a phone call between two individuals indicating a social interaction), was also applied to the GBAD approaches, yielding additional anomalous behavior between targeted persons. From these results we are able to determine the members of the normative social network surrounding Ferdinando and how the network begins to breakdown after about day 6.

5.2.6. CAIDA

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 (http://www.caida.org/data/). Using GBAD, we analyzed the CAIDA AS (Autonomous Systems) data set for normative patterns and possible anomalies. 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. For the purposes of analysis, the data is represented 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. The normative pattern for this graph is depicted in Figure 12. After running GBAD-P on the AS graph, the anomalous substructure discovered is shown in Figure 13.

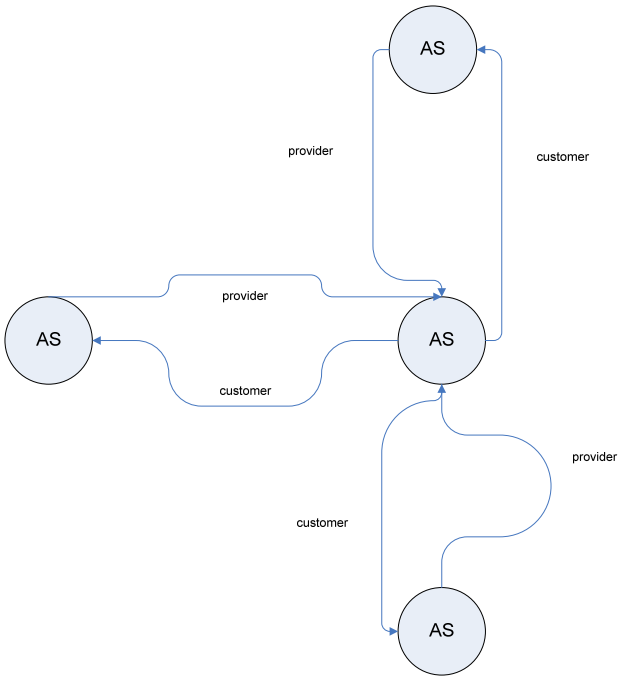

Figure 12. Normative pattern discovered in the CAIDA dataset.

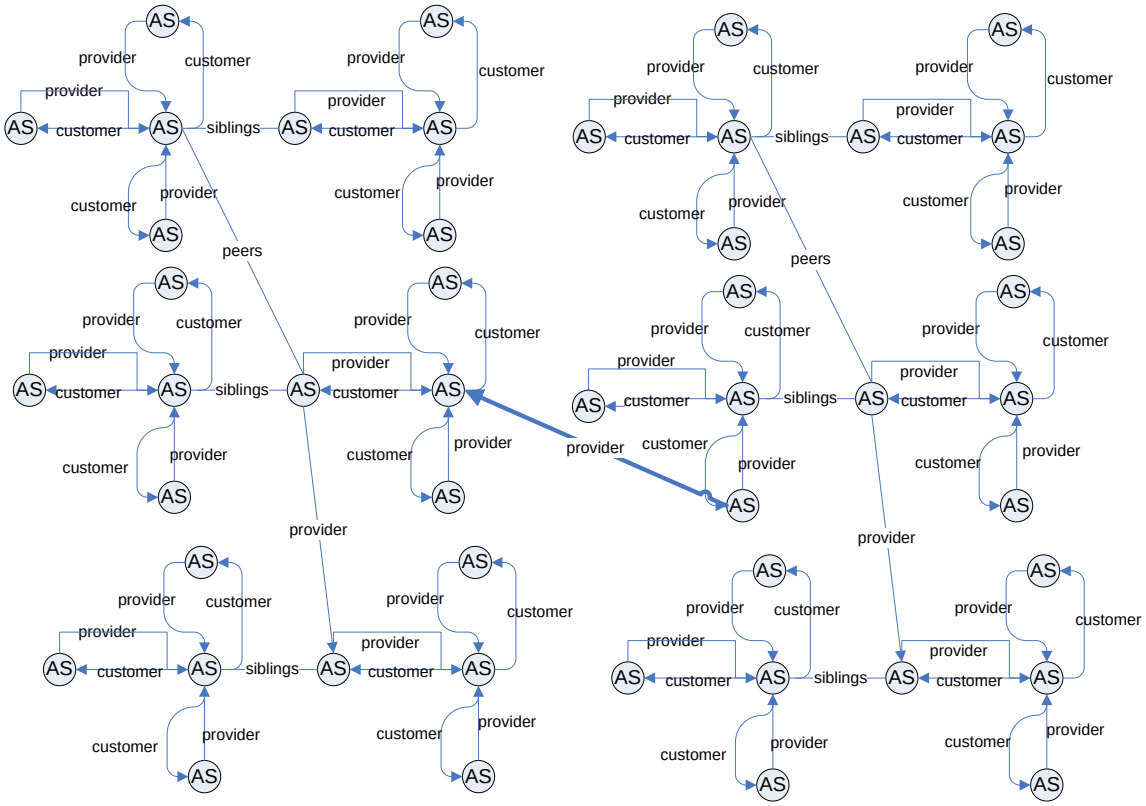

Figure 13. Anomalous pattern discovered in the CAIDA dataset, where the crucial anomalous component is the central "provider" edge (emphasized).

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, this single substructure indicates an unusual connection between two ASes. Such a complex structure would probably be missed by a human analyst, and shows the potential of an approach like GBAD to find these complex anomalies in network traffic data.

5.3. Other Domains

Since 9/11, one of the more common domains used in data mining consists of terrorist activity and relationships. Organizations such as the Department of Homeland Security use various techniques to discover the inherent patterns in the network representation of known terrorists and their relations (Kamarck 2002). Much research has been applied to not only understanding terrorist networks (Sageman 2004), but also discovering the patterns that discriminate the terrorists from the non-terrorists (Taipale 2003). Much of this area of research has also been applied to what is known as social network analysis, which is a more general term for the measuring and mapping of relationships between people, places and organizations (Mukherjee and Holder 2004).

Through the Evidence Assessment, Grouping, Linking, and Evaluation (EAGLE) program, under the auspices of the Air Force Research Lab (AFRL), we have been able to gather counter-terrorism data. While the data is simulated, it does represent scenarios based on input from various intelligence analysts (Holder et al. 2005). The data represents different terrorist organization activities as they relate to the exploitation of vulnerable targets. Our goal is to use this data as another example of real-world data to further validate the effectiveness of this approach. If a terrorist can be distinguishable from a non-terrorist by a small deviation in a normative pattern, we should be able to discover actual terrorist instances within a network of people and relationships.

Another domain worth investigating is data from the Financial Crimes Enforcement Network (FinCEN, http://www.fincen.gov). The purpose of FinCEN is to analyze financial transactions for possible financial crimes including terrorist financing and money laundering. Again, if illegal transactions consist of small deviations from normal transactions, we should be able to uncover genuine fraudulent activity within a network of people and their related monetary dealings.

Similar techniques could be applied to a myriad of domains, including telecommunications call records and credit card transactions. In short, any data source where transactions and relationships can be represented structurally as a graph, and possible anomalous behavior consists of minor deviations from normal patterns, these approaches to graph-based anomaly detection could prove to be a viable alternative to more traditional anomaly detection methods. In addition, by analyzing the effectiveness of our algorithms against real-world, labeled data sets, we can establish a baseline of comparison that can be used in subsequent anomaly detection endeavors.

6. Related Work

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 intrusion detection (Staniford-Chen et al. 1996), very little research has been accomplished in the area of graph-based anomaly detection.

In 2003, Noble and Cook used the SUBDUE application to look at the problem of anomaly detection from both the anomalous substructure and anomalous subgraph perspective (Noble and

Cook 2003). They were able to provide measurements of anomalous behavior as it applied to graphs from two different perspectives. *Anomalous substructure* detection dealt with the unusual substructures that were found in an entire graph. In order to distinguish an anomalous substructure from the other substructures, they created a simple measurement whereby the value associated with a substructure indicated a degree of anomaly. They also presented the idea of *anomalous subgraph* detection which dealt with how anomalous a subgraph (i.e., a substructure that is part of a larger graph) was to other subgraphs. The idea was that subgraphs that contained many common substructures were generally less anomalous than subgraphs that contained few common substructures. In addition, they also explored the idea of conditional entropy and data regularity using network intrusion data as well as some artificially created data.

(Lin and Chalupsky 2003) took a different approach and 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 was 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. In other words, the unusual patterns had to originate or terminate from a user-specified node.

The AutoPart system presented a non-parametric approach to finding outliers in graph-based data (Chakrabarti 2004). Part of Chakrabarti's 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. 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. Nodes were put into groups based upon their entropy.

In 2005, the idea of entropy was also used by (Shetty and Adibi 2005) 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 Noble and Cook 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. Thus, the most interesting node was the one that brought about the maximum change to the graph's entropy. However, in this approach, the idea of important nodes did not necessarily mean that they were anomalous.

In the December 2005 issue of SIGKDD Explorations, two different approaches to graph-based anomaly detection were presented. Using just *bipartite* graphs, (Sun et al. 2005) 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. The idea was that the nodes with the lower normality score to *x* were the more anomalous ones to that node. The two drawbacks with this approach were that it only dealt with bipartite graphs and it only found anomalous nodes, rather than what could be anomalous substructures. In (Rattigan and Jensen 2005), they 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.

In (Eberle and Holder 2006), we analyzed the use of *graph properties* as a method for uncovering anomalies in data represented as a graph. While our initial research examined many of the basic graph properties, only a few of them proved to be insightful as to the structure of a graph for anomaly detection purposes: average shortest path length, density and connectedness. For a measurement of *density*, we chose to use a definition that is commonly used when defining social networks (Scott 2000). For *connectedness*, we used a definition from (Broder et al. 2000).

Then, for some of the more complex graph properties, we investigated two measurements. First, there is the maximum *eigenvalue* of a graph (Chung et al. 2003). Another, which was used in identifying e-mail "spammers", is the *graph clustering coefficient* (Boykin and Roychowdhury 2005). We tested these approaches on the aforementioned cargo data when injecting one of two real-world anomalies related to drugs and arms smuggling. No significant deviations are seen using the average shortest path or eigenvalue metrics. However, there are visible differences for the density, connectedness and clustering coefficient measurements. One issue with this approach is that while graphs are indicated as anomalous, this does not identify the specific anomaly within what could be a very large graph. However, the algorithms presented in this work rectify that problem by not only indicating a graph contains an anomaly, but more importantly, they identify the specific anomaly and its pertinent structure within the graph.

7. Conclusions

The purpose of this chapter was to present an approach for discovering the three possible graph anomalies: modifications, insertions and deletions. Using a practical definition of fraud, we designed algorithms to specifically handle the scenario where the anomalies are small deviations to a normative pattern. We have described three novel algorithms, each with the goal of uncovering one of the specified anomalous types. We have validated all three approaches using synthetic data. The tests verified each of the algorithms on graphs and anomalies of varying sizes, with the results showing very high detection rates with minimal false positives. We further validated the algorithms using real-world cargo data and actual fraud scenarios injected into the data set. Despite a less regular set of data, normative patterns did exist, and changes to those prevalent substructures were detected with 100% accuracy and no false positives. We also compared our algorithms against a graph-based approach using intrusion detection data, again with better discovery rates and lower false positives. We also looked at other real-world datasets where we were able to show unusual patterns in diverse domains. In short, the GBAD algorithms presented here are able to consistently discover graph-based anomalies that are comprised of the smallest deviation of the normative pattern, with minimal false positives.

There have been many approaches to anomaly detection over the years, most of which have been based on statistical methods for determining outliers. As was shown here, recent research in graph-based approaches to data mining have resulted in new methods of anomaly detection. This work shows promising approaches to this problem, particularly as it is applied to threat detection. However, there are still many avenues to be explored. One avenue is to detect anomalies in weighted graphs, where the relational edges are associated with weights reflecting the extent of the relationship (e.g., trust). Another avenue is to detect anomalies in graphs changing over time both in terms of their attributes and structure. Techniques for detecting anomalies in real-time within such data are crucial for securing our cyber-infrastructure against modern, sophisticated threats.

References

Barthélemy, M., Chow, E. and Eliassi-Rad, T, *Knowledge Representation Issues in Semantic Graphs for Relationship Detection*. AI Technologies for Homeland Security: Papers from the 2005 AAAI Spring Symposium, AAAI Press, 2005, pp. 91-98.

Boykin, P. and Roychowdhury, V. *Leveraging Social Networks to Fight Spam*. IEEE Computer, April 2005, 38(4), 61-67, 2005.

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A. and Wiener, J. *Graph Structure in the Web*. Computer Networks, Vol. 33, 309-320, 2000.

Caruso, C. and Malerba, D. *Clustering as an add-on for firewalls*. Data Mining, WIT Press, 2004.

Chakrabarti, D. *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.

Chung, F., Lu, L., Vu, V. *Eigenvalues of Random Power Law Graphs*. Annals of Combinatorics, 7, 21-33, 2003.

Cook, D. and Holder, L. *Graph-based data mining*. IEEE Intelligent Systems 15(2), 32-41, 2000.

Cook, D. and Holder, L. *Mining Graph Data*. John Wiley and Sons, 2006.

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

Frank, E. and Witten, I. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufman, Second Edition, 2005.

Gross, J, and Yellen, J. *Graph Theory and Its Applications*. CRC Press. 1999.

Gudes, E. and Shimony, S. *Discovering Frequent Graph Patterns Using Disjoint Paths*. IEEE Transactions of Knowledge and Data Engineering, 18(11), November 2006.

Holder, L., Cook, D. and Djoko, S. *Substructure Discovery in the SUBDUE System*. Proceedings of the AAAI Workshop on Knowledge Discover in Databases, pp. 169-180, 1994.

Holder, L., Cook, D., Coble, J., and Mukherjee, M. *Graph-based Relational Learning with Application to Security*. Fundamenta Informaticae Special Issue on Mining Graphs, Trees and Sequences, 66(1-2):83-101, March 2005.

Huan, J., Wang, W. and Prins, J. *SPIN: Mining Maximal Frequent Subgraphs from Graph Databases*. Knowledge Discovery and Data Mining, KDD '04, 2004.

KDD Cup 1999. Knowledge Discovery and Data Mining Tools Competition. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. 1999.

Kamarck, E. *Applying 21st Century Government to the Challenge of Homeland Security.* Harvard University, PriceWaterhouseCoopers, 2002.

Kanungo, T, Mount, D., Netanyahu, N., Piatko, C., Silverman, R. and Wu, A. *The Analysis of a Simple k-Means Clustering Algorithm*. Proceedings on the 16th Annual Symposium on Computational Geometry, 100-109, 2000.

Kuramochi, M. and Karypis, G. *An Efficient Algorithm for Discovering Frequent Subgraphs*. IEEE Transactions on Knowledge and Data Engineering, pp. 1038-1051, 2004.

Kuramochi, M. and Karypis, G. *Grew – A Scalable Frequent Subgraph Discovery Algorithm.* IEEE International Conference on Data Mining (ICDM '04), 2004.

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

Mukherjee, M. and Holder, L. *Graph-based Data Mining on Social Networks*. Workshop on Link Analysis and Group Detection, KDD, 2004.

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

Portnoy, L., Eskin, E. and Stolfo, S. *Intrusion detection with unlabeled data using clustering*. Proceedings of ACM CSS Workshop on Data Mining Applied to Security, 2001.

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

Sageman, M. *Understanding Terror Networks*. University of Pennsylvania Press, 2004.

Scott, J. *Social Network Analysis: A Handbook*. SAGE Publications, Second Edition, 72-78, 2000.

Shetty, J. and Adibi, J. *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.

Staniford-Chen, S., Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J. Levitt, K., Wee, C., Yip, R. and Zerkle, D. *GrIDS – A Graph Based Intrusion Detection System for Large Networks*. Proceedings of the 19th National Information Systems Security Conference, 1996.

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

Taipale, K. *Data Mining and Domestic Security: Connecting the Dots to Make Sense of Data*. Columbia Science and Technology Law Review, 2003.

Thomas, L., Valluri, S. and Karlapalem, K. *MARGIN: Maximal Frequent Subgraph Mining*. Sixth International Conference on Data Mining (ICMD '06), 109-1101, 2006.

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)

WEKA, *http://www.cs.waikato.ac.nz/~ml/index.html*.

West, D. *Introduction to Graph Theory*. Prentice-Hall International. Second Edition. 2001.

Yan, X. and Han, J. *gSpan: Graph-Based Substructure Pattern Mining*. Proceedings of International Conference on Data Mining, ICDM, pp. 51-58, 2002.

Zeng, Z., Wang, J., Zhou, L. and Karypis, G. *Coherent closed quasi-clique discovery from large dense graph databases*. Conference on Knowledge Discovery in Data, SIGKDD, 797-802, 2006.