Intelligent Data Analysis 11 (2007) 663–689
IOS Press

Anomaly detection in data represented as graphs

William Eberle[a,*] and Lawrence Holder[b]
[a]*Department of Computer Science, Tennessee Technological University, P.O. Box 5101, Cookeville, TN 38505, USA*
E-mail: weberle@tntech.edu
[b]*School of Electrical Engineering & Computer Science, Washington State University, Box 642752, Pullman, WA 99164-2752, USA*
E-mail: holder@wsu.edu

Received 20 November 2006
Revised 15 February 2007
Accepted 7 April 2007

Abstract. An important area of data mining is anomaly detection, particularly for fraud. However, little work has been done in terms of detecting anomalies in data that is represented as a graph. In this paper 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. In this paper, we validate all three approaches using synthetic data, verifying that each of the algorithms on graphs and anomalies of varying sizes, are able to detect the anomalies with very high detection rates and minimal false positives. We then further validate the algorithms using real-world cargo data and actual fraud scenarios injected into the data set with 100% accuracy and no false positives. Each of these algorithms demonstrates the usefulness of examining a graph-based representation of data for the purposes of detecting fraud.

Keywords: Graph-based anomaly detection, minimum description length principle, information theoretic compression

1. Introduction

Detecting anomalies in various data sets is an important endeavor in data mining. Using statistical approaches has led to various successes in environments such as intrusion detection. Recent research in graph-based anomaly detection has paved the way for new approaches that not only compliment the non-graph-based methods, but also provide mechanisms for handling data that cannot be easily analyzed with traditional data mining approaches. Using information theoretic, probabilistic and maximum partial substructure approaches, we have developed three novel algorithms for analyzing graph substructures

*Corresponding author.

for the purpose of uncovering all three types of graph-based anomalies: modifications, insertions and deletions.

The key to the algorithms presented in this paper lies in our definition of an anomaly. Basing our definition on the assumption that an anomaly is not random, for instance in the case of committing fraud, we believe that this type of anomaly should only be a minor deviation from the normal pattern. Because anyone who is attempting to commit fraud or hide devious activities would not want to be caught, it only makes sense that they would want their activities to look as real as possible. For example, 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]. Thus, if some set of data is represented as a graph, any nefarious activities should be identifiable by small modifications, insertions or deletions to the normative patterns within the graph.

Our first algorithm uses the *minimum description length* principle [10] to determine the normative pattern, and from that pattern, find patterns that while structurally similar, have some relational deviation that is within an acceptable level of change. By determining what substructure minimizes the description length of the graph, we are able to calculate the cost of transformation for instances within the graph that do not exactly match the discovered normative pattern, and as such, are indicative of an unexpected change.

Our second algorithm again determines the normative pattern as the one that minimizes the description length of a graph, but instead of looking at changes to this pattern we examine the *probability* of extensions to the pattern. If the normative pattern does not completely compress the graph, meaning there are other vertices and edges connected to the normative pattern, we examine each of these extensions in terms of the probability of their existence. If the probability of existence is low enough, we mark the instance as anomalous. We can then compress the graph by this anomalous instance, and repeat the process until there are no more extensions to the anomalous substructure.

Our third algorithm uses a trail of pattern expansion to discover the instances that are structurally deficient from the normative pattern. When we attempt to discover the pattern that minimizes the description length of the graph, we maintain a parental relationship between the structures. Once we have discovered the normative pattern, we traverse these relationships to find the instance that is the *maximum partial substructure*. In this case, we are looking for patterns that are unable to extend to the normative pattern, and are a maximal representation of that normative pattern. In other words, the maximum partial substructure is found in the instance that requires the fewest additions (IF they would have existed) for transforming the instance into an instance consisting of the normative structure.

Up until now, graph-based approaches to fraud detection have been limited to specific anomaly types and certain domains. Taking into account the "mind of the fraudster", we have developed algorithms that can discover any of three types of anomalous changes where the illegitimate actions consist of minor changes to a normal set of activities.

2. Previous work

Recently there has been an impetus towards analyzing data using graph theoretical 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 [12], 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 sub-graph perspective [9]. 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 sub-graph* detection which dealt with how anomalous a sub-graph (i.e., a substructure that is part of a larger graph) was to other sub-graphs. The idea was that sub-graphs that contained many common substructures were generally less anomalous than sub-graphs 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 took a different approach and applied what they called rarity measurements to the discovery of unusual links within a graph [8]. 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. 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 [1]. 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 in their analysis of a real-world data set: the famous Enron scandal [12]. 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 [9], 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 2005 SIGKDD Explorations, a couple of different approaches to graph-based anomaly detection were presented. Using just bipartite graphs, Sun et al. presented a model for scoring the normality of nodes as they relate to the other nodes [14]. 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. Rattigan and Jensen also went after anomalous links, this time via a statistical approach [10]. 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.

3. Graph-based anomalies

The idea behind the approach presented in this work is to find anomalies in graph-based data where the anomalous substructure (at least one edge or vertex) in a graph is part of (or attached to or missing from) a non-anomalous substructure, 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 approaches may aide in this respect, there does not appear to be any existing approaches that directly deal with this scenario.

Definition 1. A graph substructure S' is anomalous if it is not isomorphic to the graph's normative substructure S, but is isomorphic to S within X%.

X signifies the percentage of vertices and edges that would need to be changed in order for S' to be isomorphic to S. 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 as close to legitimate actions as possible. That makes this definition of an anomaly extremely relevant.

3.1. Types

For a graph-based anomaly, there are several possible changes in a graph that might occur:

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

It is also evident that these same situations can be applied to a substructure (i.e., multiple vertices and edges), and will be addressed as such. 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.

3.2. Assumptions

Many of the graph-based anomaly detection (or intrusion detection) approaches up to now have assumed that the data exhibits a power-law distribution. For example, much of the data that has been used in previous analysis has used items like the world-wide web, social networks, or other sources that convey a power-law behavior [5]. The advantage of the approaches presented in this work 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 is typical of most production data, such as telecommunications call traffic, financial transactions, and shipping manifests.

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 can 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 in Section 3.1, 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.

Assumption 5. Expected deviations in the graph are well represented so as to be distinguishable from anomalous deviations.

We base our algorithms on the presence of a normative pattern, and we are looking for unexpected deviations to that normative pattern. Thus, if we do not have enough examples that clearly distinguish a normative pattern, and the unexpected deviations from the expected deviations, our algorithms will be unable to discover the anomaly.

4. 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 we have an idea what is normal behavior, deviations from that behavior could constitute an anomaly. However, the issue with these 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]. Using a breadth-first search and Minimum Description Length (MDL) heuristic, each of the three anomaly detection algorithms uses GBAD to provide the 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:

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

where G is the entire graph, S is the substructure, $DL(G|)$ 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.

4.1. Information theoretic algorithm (*GBAD-MDL*)

The GBAD-MDL algorithm uses the Minimum Description Length (MDL) approach 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 high-level approach for the GBAD-MDL algorithm is, for a graph G:

- Find the best substructure S that minimizes the description length of G.
- Find all instances I_n, whose cost of transformation is less than a specified threshold, where the threshold is a user-defined parameter.
- Output all I_n whose (cost * frequency) is minimum.

"Cost of transformation" is the cost of transforming graph A into an isomorphism of graph B. We calculate this 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 substructure (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 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.

4.1.1. Algorithm

The detailed GBAD-MDL algorithm is as follows. For a graph G :

1. Find the top-k substructures S_i, where

   $$M(S_i, G) = DL(G|S_i) + DL(S_i)$$

   is the MDL value of S_i, and $M(S_i, G) \leqslant M(S_j, G)$ for all j, where $S_i, S_j \subseteq G$.
2. Find all instances of S_i such that $C(I_j, S_i) > 0$, where $C(I_j, S_i)$ is the cost of transforming the graph structure of I_j to match the graph structure of S_i.
3. For each I_j
   a. Determine the substructure definition S_j.
   b. Find all matching instances of $S_j$such that

      $$F(I_j) = I(S_j)$$

      where $I(S_j)$ is the number of *exact matching* instances of $S_j$in the list of instances for S_i. Another way to state this, is the value $F(I_j) = I(S_j)$is the number, or *frequency*, of instances that match I_j.
   c. Determine the *anomalousness* value such that

      $$A(I_j) = F(I_j) * C(I_j, S_i)$$

      where the lower the value, the more anomalous the instance.
4. Output all I_i where $A(I_i) \leqslant A(I_j)$ for all j, where $I_i, I_j \subseteq S_i$.

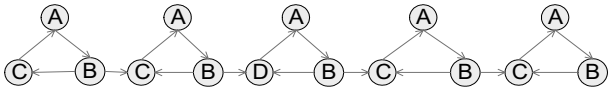

Fig. 1. Simple graph for GBAD-MDL example.

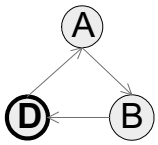

Fig. 2. Anomalous substructure from simple graph using GBAD-MDL.

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 is not found in the top normative pattern.

Throughout this work, whenever we indicate a relationship between substructures as $x \subseteq y$, we are referring to the fact that x is a *sub-graph* of y, rather than x is a subset of y.

4.1.2. Example

The following is a simple example of results obtained using our implementation of the GBAD-MDL algorithm described above. Take the fairly regular example shown in Fig. 1. Running the GBAD-MDL algorithm, the anomalous substructure, as shown in Fig. 2, is exactly the desired result. (The individual anomaly is in **bold**.) It should also be noted that no other substructures were reported as anomalous.

4.2. Probabilistic algorithm (*GBAD-P*)

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 than other extended instances. The high-level approach for the GBAD-P algorithm is:

- For a graph G, find the best substructure S that minimizes the description length of G.
- Compress G using S.
- For the newly compressed graph G
  - Find the single edge and vertex extension E that has the lowest probability P of existence from instances I_n of S.
  - Output instance I_n and E whose P is minimum.
  - Set S' to instance I_n s substructure.
- Compress G using S', and repeat the above steps if there are still substructures to consider.

At each iteration, the result will be the instance that consists of the best substructure pattern and an extension with the lowest probability of existence. The value associated with this instance represents the lowest regularity, where the lower the value, the more anomalous the structure.

4.2.1. Algorithm

The detailed GBAD-P algorithm is as follows:

1. Find the substructure S_i, where
   $$M(S_i, G) = DL(G|S_i) + DL(S_i)$$
   is the MDL value of S_i, and $M(S_i, G) \epsilon M(S_j, G)$ for all j, where $S_i, S_j \subseteq G$.
2. Compress G by S_i, and repeat Step 1.
3. Find all instances I_j that match the graph structure S_i.
4. For each instance I_j, create an extended instance I'_j that consists of the 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. For each I'_j,
   a. Find all matching instances of I'_j in the set I'.
   b. Determine the *anomalousness* value such that
      $$A(I'_j) = \left|I'_j\right| / \left|I'\right|$$
      where $\left|I'_j\right|$ is the cardinality of the set of instances that match I'_j, and $|I'|$ is the cardinality of the set of extended instances of S_i. $A(I'_j)$ 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, $\left|I'_j\right|$ can never be greater, and thus the value of $A(I'_j)$ will never be greater than 1.0.
6. Output I'_j where $A(I'_j) \leqslant A(I'_k)$ for all k, where $I'_j, I'_k \subseteq I'$, and its probability is less than a user specified threshold. This threshold will allow us some flexibility to indicate that we do not want to see instances that are not at least below some level of probability.
7. Compress $G$by the graph structure of I'_j.
8. Repeat Step 1.
9. Start again at Step 3.

4.2.2. Example

The following is a simple example of results obtained using our implementation of the GBAD-P algorithm described above. Take the example of a network-looking structure, as shown 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). 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

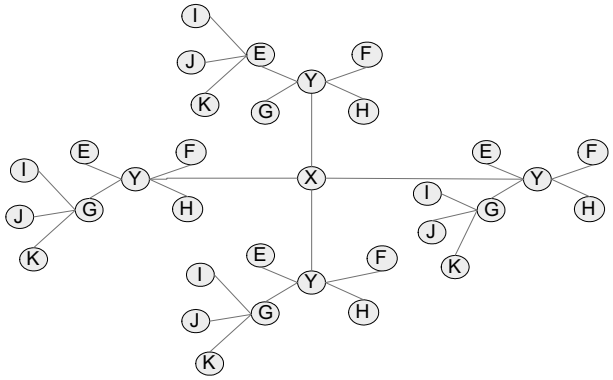

Fig. 3. Simple graph for GBAD-P example.

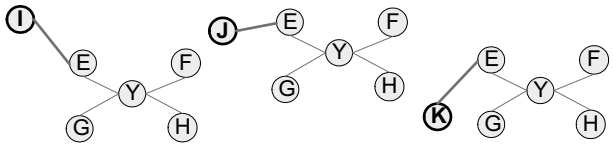

Fig. 4. Anomalous structures from GBAD-P example.

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.

4.3. Maximum partial substructure algorithm (*GBAD-MPS*)

The GBAD-MPS algorithm uses the MDL approach to discover the best substructure in a graph, then it examines all of the instances of similar substructures that are missing various edges and vertices. The high-level approach for the GBAD-MPS algorithm is:

- For a graph G, find the best substructure S that minimizes the description length of G.
- Find the instances of ancestor substructures of S (we will call the ancestor substructure S').
- Output all instances I_n of S' that are not part of any instances of S, but are closest in transformation cost (and lowest in frequency if cost of transformation is identical).

The result will be those instances that are the maximum possible partial substructures to the normative (or best) substructure. The value associated with the 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 to the best substructure without being included on the best substructure's instance list.

4.3.1. Algorithm

The detailed GBAD-MPS algorithm is as follows:

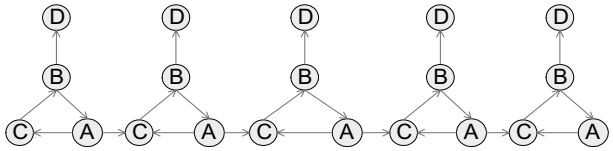

Fig. 5. Simple graph for GBAD-MPS example.

1. Find the substructure S, where

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

   is the MDL value of S, where $S \subseteq G$ and $M(S,G)$ is minimal. I is the list of instances that match S.
2. Find all ancestor substructures S'such that $S' \subseteq S$.
3. Find all instances $I'$of S'.
4. For each instance of I', if I'_n is not a subset of any instance of I, and its anomalous value (*cost of transformation * frequency*) is less than a user specified threshold, output I'_n as an anomalous instance.

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 be isomorphic to the best substructure).

4.3.2. Example

The following is a simple example of results obtained using our implementation of the GBAD-MDL algorithm described above. Take the example shown in Fig. 5. The normative pattern (best substructure) from this graph is shown in Fig. 6. 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. 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. However, this pattern is common to all of the normative instances. So, for usefulness, our implementation also reports the actual anomalous graph instance as specified in the input graph file.

5. Empirical evaluations on synthetic data

5.1. Synthetic Data

Synthetic graphs are created using a tool called *subgen* that randomly generates graphs based upon various parameters, and then modified, where:

– AV is the number of vertices in an anomalous substructure
– AE is the number of 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 consists of AV number of vertices and AE number of edges altered.

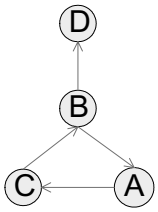

Fig. 6. Normative pattern from simple graph for GBAD-MPS example.

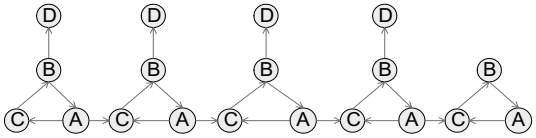

Fig. 7. Simple graph for GBAD-MPS example with deleted vertex and edge.

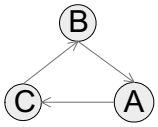

Fig. 8. Anomalous instance from deletion example when using GBAD-MPS.

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).

In addition, in order to better exemplify our definition of an anomaly, the distribution of randomly generated connections versus the anomalies is not the same. In other words, if randomly generated connections have the same frequency as the anomalies, there would be no way to distinguish between noise and anomalous behavior. Since our definition explicitly states that an anomaly is a slight deviation from the normal (to better hide the true nature of the action), its existence can not be as common as normal behaviors. Now, that does not mean that anomalies are completely different from noise, just that their frequency is not as prevalent. So, in order to achieve our goal, we will define in the *subgen* tool a distribution of 10-to-1 (i.e., random normal connections are ten times more likely than an anomaly).

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.

5.2. Metrics

Each test consists of a single graph from which 30 randomly altered graphs are generated. The output results consist of running the algorithms against those 30 graphs for the specified settings. The primary

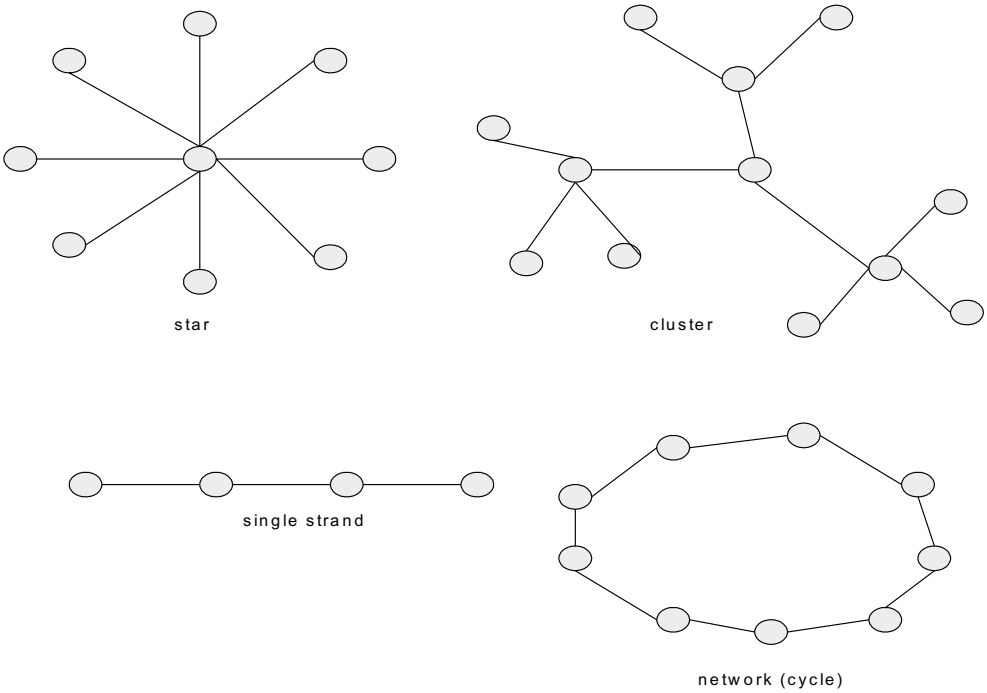

Fig. 9. Example normative pattern shapes.

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 running, 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 only one of the anomalous vertices, then that run would be considered a success for this measurement. 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.

5.3. Shapes

In addition to the types of anomalous changes, the "shape" of the normative pattern may affect the algorithms' abilities to discover the anomalies. Figure 9 shows some examples of possible patterns. In the following tests, we chose to use the *cluster* pattern as the baseline for the tests. While we could have chosen any of these patterns, we felt that this pattern was the most representative of the types of

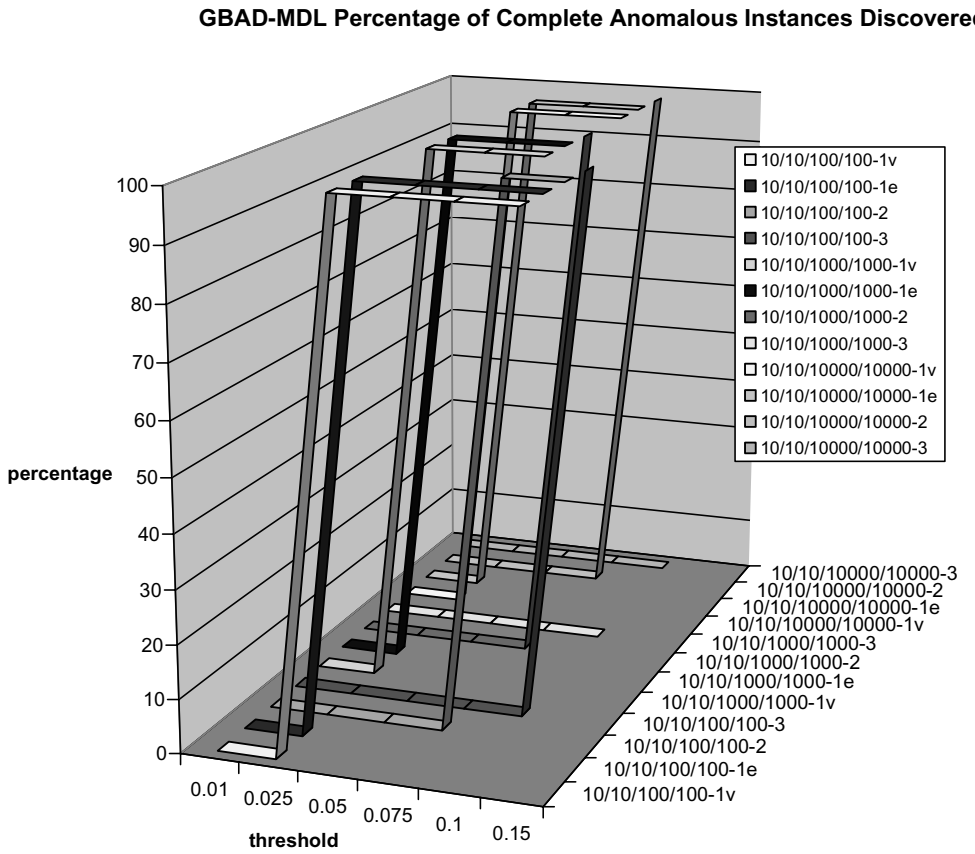

Fig. 10. Percentage of GBAD-MDL runs where all anomalies discovered.

real-world data that these approaches would be used for discovering anomalies. For example (which will be shown later), cargo shipments consist of ships, manifests, ports, etc. A nice graph representation of this type of data would be where certain entities are hubs for common information (e.g., shipper, port, etc.), such that multiple manifests could share identical information. There are many other types of pertinent information, such as telecommunication call records, terrorist networks, financial transactions, etc. which share this same type of ontology.

5.4. Information theoretic results

In this section, and the subsequent sections, for each approach, the synthetic experiments consist of *insertions* (i.e., additional vertices and edges connected to the normative pattern), *modifications* (i.e., vertices and edges modified within an instance of the normative pattern), and *deletions* (i.e., vertices and edges missing from what could have been an instance of the normative pattern). While each of the algorithms was designed to handle different types of anomalies, each approach will be evaluated as to their effectiveness across all types.

5.4.1. Modifications

Figure 10 shows the effectiveness of the GBAD-MDL approach on graphs of varying sizes with random anomalous modifications. (The results are the same for the experiments where at least part of the

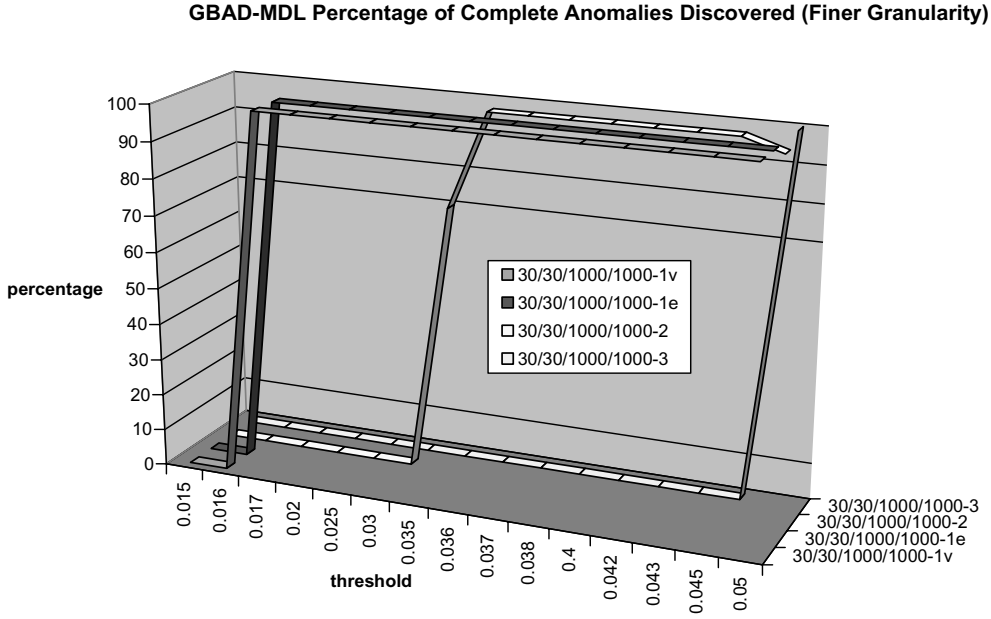

Fig. 11. Percentage of GBAD-MDL runs where all anomalies are discovered (finer granularity).

anomaly is discovered, and there are no false positives.) In Fig. 10 (and subsequent 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 very small synthetic tests (not shown for space reasons), when the threshold is high enough, (i.e., the threshold is equal to or higher than the percentage of change), this approach is able to find all of the anomalies. For example, when the normative pattern is of size 6 (3 vertices and 3 edges), we have to set the threshold higher than 0.1. This is done because with a normative pattern of size 6, even just a change of a single vertex and a single edge would require a threshold of at least 0.33 in order to discover such an anomaly. Thus, when we set the threshold to 0.035, the algorithm is able to find 100% of the anomalies for a change of size 2. Similarly, when we set the threshold to 0.2, GBAD-MDL is able to find all of the single anomalous modifications (as 0.167% of the normative pattern is changed).

For all tests where the threshold was 0.1 or less, no false positives are reported. However, when we increase the threshold to 0.2, a few false positives are reported. 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.

When the size of the normative pattern is larger, smaller thresholds can be used in order to uncover small changes to the structure. Figures 11, 12 and 13 represent the use of finer thresholds for the discovery of very small anomalies (1 to 3 changes) to a normative pattern of 30 vertices and 30 edges.

5.4.2. Insertions

Figures 14, 15 and 16 show the effectiveness of the GBAD-MDL approach on graphs of varying sizes with random anomalous insertions. On the smaller graphs, no matter how large the anomaly or the size of the threshold, none of the anomalous insertions are discovered. Even when unrealistically large thresholds (i.e., above 0.1) are used, we are unable to find any of the insertion anomalies.

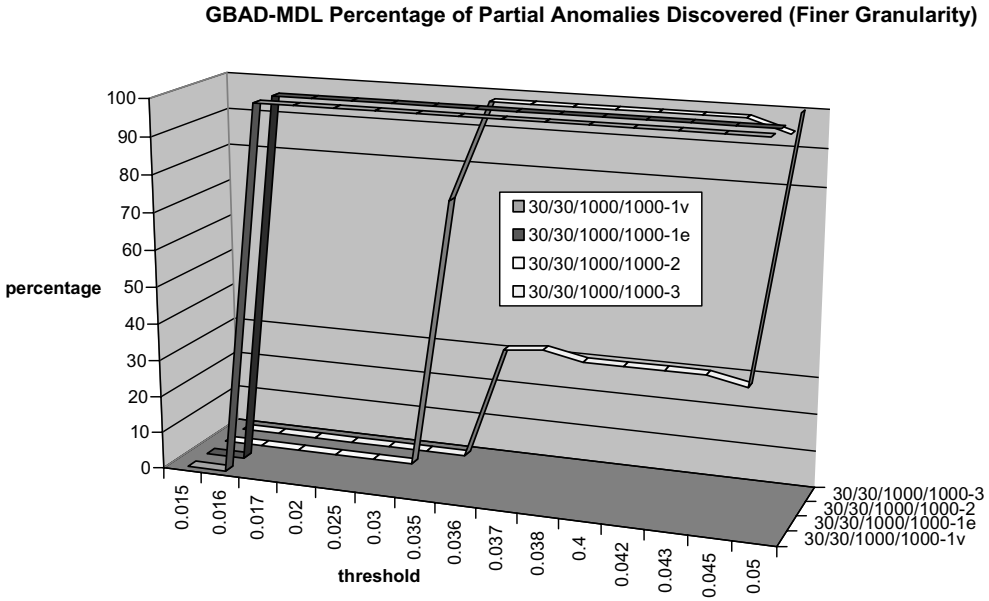

Fig. 12. Percentage of GBAD-MDL runs where at least one anomaly is discovered (finer granularity).

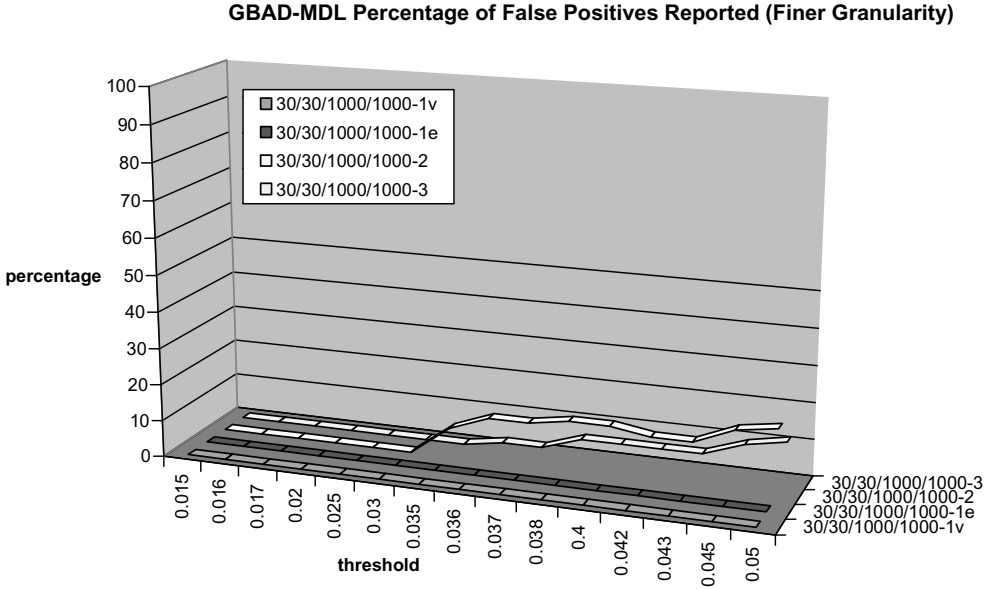

Fig. 13. Percentage of GBAD-MDL runs with false positives (finer granularity).

Two observations about the effectiveness of this algorithm on anomalous insertions are evident in these results. First, in the situation where the anomaly is a single edge and vertex, there is some success because the insertion is “close” to the normative pattern (a direct connection). Second, the effectiveness clearly drops off as the insertions get further away from the normative pattern, and the few successes can be attributed to the random insertions being close to the normative pattern (e.g., an anomaly of size 4 that

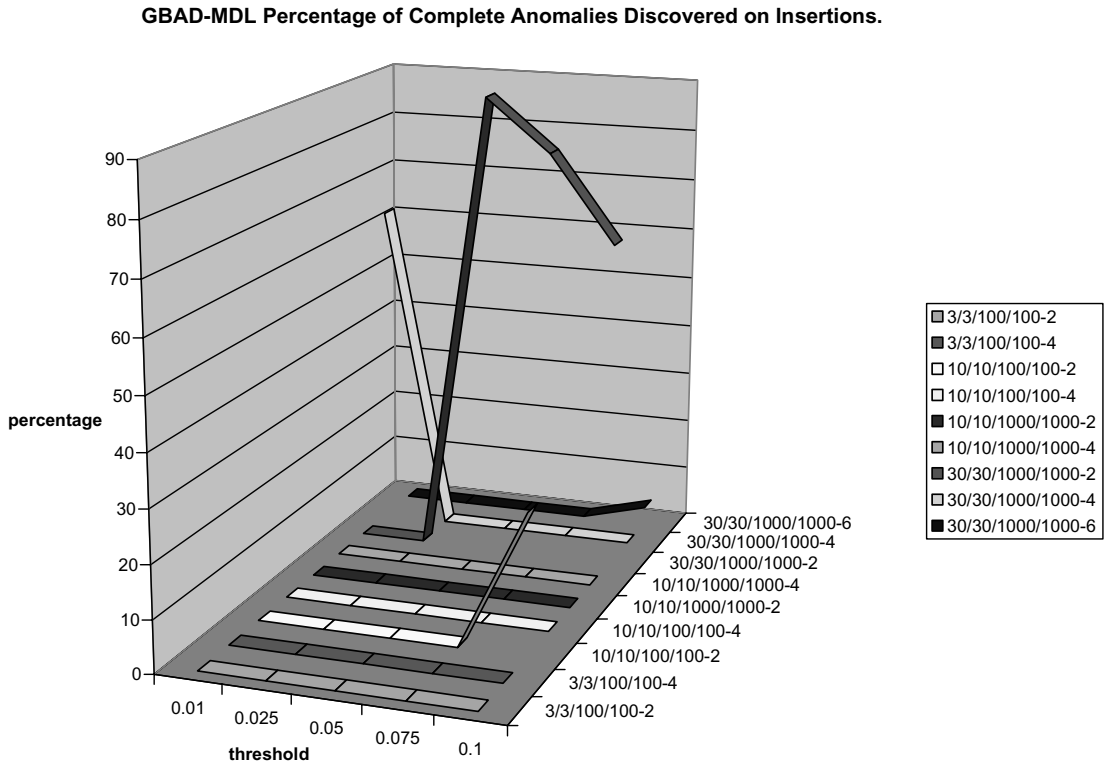

Fig. 14. Percentage of GBAD-MDL runs where all anomalous insertions are discovered.

consists of two edges and two vertices that are directly connected to the normative pattern via different vertices).

This is clearly not an effective solution for anomalous insertions.

5.4.3. Deletions

As expected, the results of running the GBAD-MDL algorithm on the graphs generated with random deletions are also not effective. In order to discover any of the anomalies, the threshold has to be set high (above 20%), however, that also results in very high false positives. The best result on a graph of 1000 vertices and 1000 edges is a 3.33% discovery rate, and 100% of the runs report false positives.

Again, this is not a surprise, as the GBAD-MDL algorithm was designed for discovering anomalous modifications and not deletions.

5.5. Probabilistic results

For each of the following tests, we will implement the following GBAD settings:

- prune substructures whose pattern matching value is less than their parent's (for performance)
- only analyze extensions found in the top 10 best substructures (for performance)
- allow the program to iterate (i.e., compress and run again) until there is nothing left to compress

It should be noted that for these experiments we will not perform any single vertex tests because one can not insert a new vertex without also inserting a new edge.

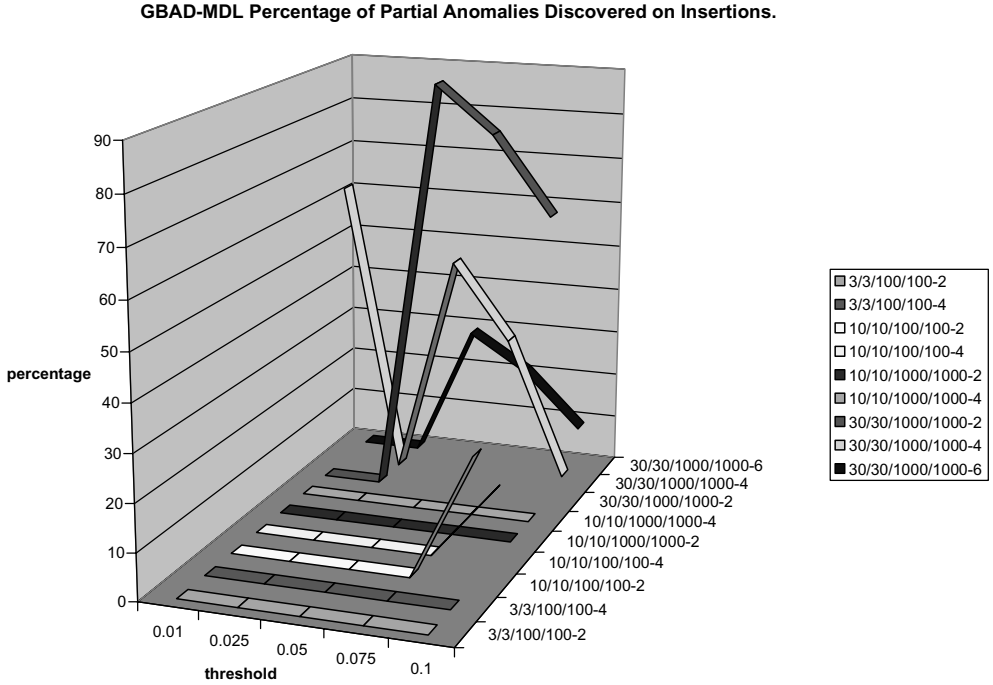

Fig. 15. Percentage of GBAD-MDL runs where at least one anomalous insertion is discovered.

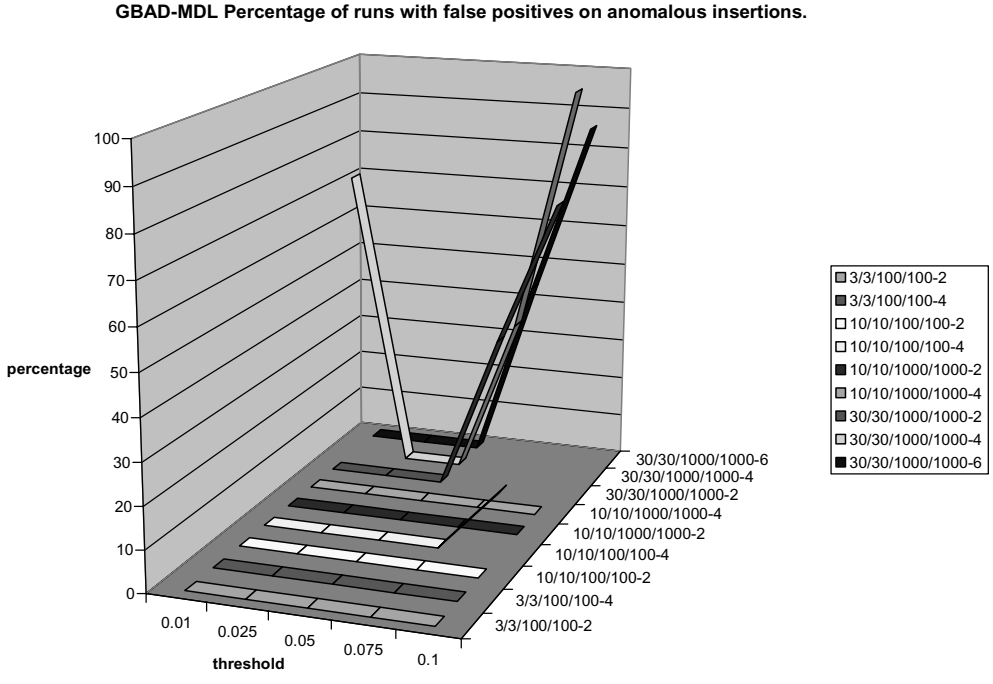

Fig. 16. Percentage of GBAD-MDL runs on anomalous insertions containing false positives.

Table 1
Percentage of discovery for GBAD-P runs on anomalous insertions

Graph size (norm pattern) <anomaly size>	All anomalies	Partial anomalies	False positives
100 vertices/100 edges (3 vertices/3 edges)			
< 1 vertex/ 1edge>	100	100	0
< 2 vertices/2 edges>	100	100	0
100 vertices/100 edges (10 vertices/10 edges)			
< 1 vertex/ 1edge>	100	100	0
< 2 vertices/ 2 edges>	100	100	0
< 3 vertices/3 edges>	100	100	0
< 4 vertices/ 4 edges>	100	100	0
< 5 vertices/5 edges>	93.33	93.33	0
1000 vertices/1000 edges (10 vertices/10 edges)			
< 1 vertex/ 1edge>	93.33	93.33	0
< 2 vertices/ 2 edges>	86.67	91.67	0
< 3 vertices/3 edges>	93.33	98.89	0
1000 vertices/1000 edges (30 vertices/30 edges)			
< 1 vertex/ 1edge>	96.67	96.67	0
< 2 vertices/ 2 edges>	82.76	84.48	0
< 3 vertices/3 edges>	60.0	70.0	0
10000 vertices/10000 edges (10 vertices/10 edges)			
< 1 vertex/ 1edge>	96.67	96.67	0
< 2 vertices/ 2 edges>	100	100	0
< 3 vertices/3 edges>	93.33	98.33	0
10000 vertices/10000 edges (30 vertices/30 edges)			
< 1 vertex/ 1edge>	100	100	0
< 2 vertices/ 2 edges>	93.10	95.69	0
< 3 vertices/3 edges>	90.0	95.56	0
< 4 vertices/ 4 edges>	96.0	96.0	0
< 5 vertices/5 edges>	96.43	99.64	0

5.5.1. Insertions

Table 1 shows the effectiveness of the GBAD-P approach on graphs of varying sizes with random anomalous insertions.

The positive results from these runs are the relatively high discovery rate (never lower than 82% except for one scenario) and the lack of false positives. The exception to the positives is the one scenario where only 60% of the complete anomalies (70% of at least some of the anomalies) are discovered. However, when the same size normative pattern and anomaly is used in a larger graph, the discovery rate is 90% (and 95.56% for some of the anomalies). We have hypothesized that it has something to do with the regularity of the data, where the more data one has the more regular the data becomes. Further investigation of this oddity will need to be performed.

It should be noted in the example with 1000 vertices/1000 edges and a normative pattern of 10 vertices/10 edges, that even though some 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.

5.5.2. Modifications

Running modification tests on graphs of 100 vertices/100 edges and 1000 vertices/1000 edges results in no anomalies, partial or complete, being discovered, and no false positives reported. The issue with

trying to find modifications using the Probabilistic approach lies in the way the algorithm examines extensions. Modifications are changes to the normative pattern, while the Probabilistic approach is examining edges and vertices that are connected to a normative pattern. Since modified anomalies are connected to substructures that are not the best substructure (or normative pattern), examination of extensions would not consider these anomalies. While we could look for smaller patterns (i.e., a substructure of the true normative pattern that does not include the modified vertices and edges), that would go against our principle of what is the normative (and best) pattern.

5.5.3. Deletions

Running deletion tests on graphs of 100 vertices/100 edges and 1000 vertices/1000 edges results in no anomalies, partial or complete, being discovered, and no false positives reported. Again, the Probabilistic approach is examining extensions from the normative pattern. Intuitively, it makes sense that anomalous deletions would be difficult to discover with this approach, as they could almost be considered the opposite of what we are hoping to uncover.

5.6. Maximum partial substructure results

For each of the following tests, we will implement the following GBAD settings:

- prune substructures whose pattern matching value is less than their parent's (for performance)
- only analyze the top 25 ancestral substructures (for performance)
- a cost of transformation (or anomalous) threshold of 8.0, so as to ignore substructures that would have too many changes to be considered an anomaly (based upon our definition of an anomaly)

It should also be noted that "partial-anomalies" do not apply for this approach. Either an instance is anomalous because it is missing some edges and vertices that exist in the normative pattern, or it is not considered anomalous.

5.6.1. Deletions

Table 2 shows the effectiveness of the GBAD-MPS approach on graphs of varying sizes with random anomalous deletions.

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 to what is shown. The reason that the number of best substructures and the threshold has 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. 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 rerun the algorithm with the new threshold.

One will notice that there is one strange discovery in that no anomalies were discovered in the large graphs where the anomaly consisted of a deleted edge. This may have something to do with the way we go about discovering the normative pattern by a single extension at a time. We need to further investigate the possibility that the complete normative pattern is not being discovered, leading to the inability of the GBAD-MPS algorithm to find the anomalous deletion.

Table 2
Percentage of discovery for GBAD-MPS runs on anomalous deletions

Graph size (norm pattern) <anomaly size>	All anomalies	False positives
100 vertices/100 edges (3 vertices/3 edges)		
< 1 vertex and associated edges>	100	0
< 1 edge>	100	0
< 1 vertex/1 edge>	100	0
100 vertices/100 edges (10 vertices/10 edges)		
< 1 vertex and associated edges>	100	0
< 1 edge>	100	0
< 1 vertex/1 edge>	100	0
< 2 vertices/1 edge>	100	0
1000 vertices/1000 edges (10 vertices/10 edges)		
< 1 vertex>	100	0
< 1 edge>	100	0
< 1 vertex/1 edge>	100	0
< 2 vertices/1 edge>	100	0
1000 vertices/1000 edges (30 vertices/30 edges)		
< 1 vertex>	100	0
< 1 edge>	100	0
< 1 vertex/1 edge>	100	0
< 2 vertices/1 edge>	100	0
< 2 vertices/2 edges>	100	0
< 2 vertices/3 edges>	100	0
< 3 vertices/3 edges>	100	0
10000 vertices/10000 edges (10 vertices/10 edges)		
< 1 vertex>	100	0
< 1 edge>	100	0
< 1 vertex/1 edge>	100	0
< 2 vertices/1 edge>	100	0
10000 vertices/10000 edges (30 vertices/30 edges)		
< 1 vertex>	100	0
< 1 edge>	0	0
< 1 vertex/1 edge>	100	0
< 2 vertices/1 edge>	100	0
< 2 vertices/2 edges>	100	0
< 2 vertices/3 edges>	100	0
< 3 vertices/3 edges>	100	0

5.6.2. Modifications

Running tests on graphs of 100 vertices/100 edges and 1000 vertices/1000 edges results in no anomalies, partial or complete, being discovered, and a 100% false positives rate.

5.6.3. Insertions

When running tests on 100 vertices/100 edges and 1000 vertices/1000 edges the GBAD-MPS algorithm does not discover any anomalies. As with the previous tests, since no anomalies are discovered in any of the tests up to this point (also there are no false positives), there is no reason to continue with these experiments.

5.7. Other types of normative patterns

Each of the above tests are run using the same type of normative pattern: a star cluster. However, it is theoretically possible that the shape of the normative pattern could have an effect on the effectiveness

Table 3
Percentage of complete anomalous instances found on runs with different patterns

Algorithm (anomaly type)	Star	Strand	Cycle
GBAD-MDL (Modification)	100	100	100
GBAD-P (Insertion)	40.0	93.33	93.33
GBAD-MPS (Deletion)	100	100	100

Table 4
Percentage of runs with anomalous parts found on runs with different patterns

Algorithm (anomaly type)	Star	Strand	Cycle
GBAD-MDL (Modification)	100	100	100
GBAD-P (Insertion)	40.0	93.33	93.33
GBAD-MPS (Deletion)	n/a	n/a	n/a

Table 5
Percentage of runs containing false positives on runs with different patterns

Algorithm (anomaly type)	Star	Strand	Cycle
GBAD-MDL (Modification)	0	0	0
GBAD-P (Insertion)	44.19	0	0
GBAD-MPS (Deletion)	0	0	0

of the algorithms. So, the following tests are devised in order to test some of the other patterns that are common, particularly as they might be used for real-world data. (Refer to section 5.3 for examples of each of these patterns.)

Each of the tests in Tables 3, 4 and 5 are executed with a graph size of approximately 500 vertices and 500 edges, a normative pattern of 11 vertices and 10 edges, and an anomalous change of 9.5% . All of the same GBAD parameter settings that were used in the previous tests are implemented here, and a threshold of 0.1 is used for the GBAD-MDL algorithm.

From these results we notice that except for the star normative pattern when running the GBAD-P algorithm, the results are the same across each algorithm for each of the different shapes. The case of the low discovery rate for a star-shaped normative pattern may again be traceable to an inconsistency in determining the normative pattern, and will need to be investigated further.

5.8. Performance

One of the factors to consider in evaluating these algorithms is their respective performances. Table 6 represents the average running times (in seconds) for each of the algorithms against each of the graph sizes for the anomaly types that were the most effectively discovered. For all of the Information Theoretic runs, the times are represented as a range, because the performance of this algorithm is dependent upon the threshold chosen. The higher the threshold, the longer the algorithm takes to execute, so there is a definite trade-off associated with the threshold choice.

6. Experiments on 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 US arrive via ships at ports of entry along the coast-lines.

Table 6
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

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 that will maximize effectiveness and minimize false searches.

The Customs and Border Protection (CBP) agency maintains shipping manifests in a system called PIERS (Port Import Export Reporting Service). This database is used for tasks such as reporting and data mining. Each entry (or row) in the PIERS tables consists of various information from a shipping manifest.

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. Figure 17 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 simulations of publicized incidents.

6.1. Random changes

Similar to what we did for the synthetic tests, we randomly modified, inserted and deleted small portions of the graph for randomly selected shipping entries. However, even though this data is not as regular as the synthetic data generated for the earlier tests, all three algorithms were able to successfully find all of their intended target anomalies with no false positives reported. This helps us validate further the usefulness of this approach when the anomaly consists of small modifications to the normative pattern.

6.2. Real-world scenarios

In Eberle and Holder's work [4], 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 US 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 US 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

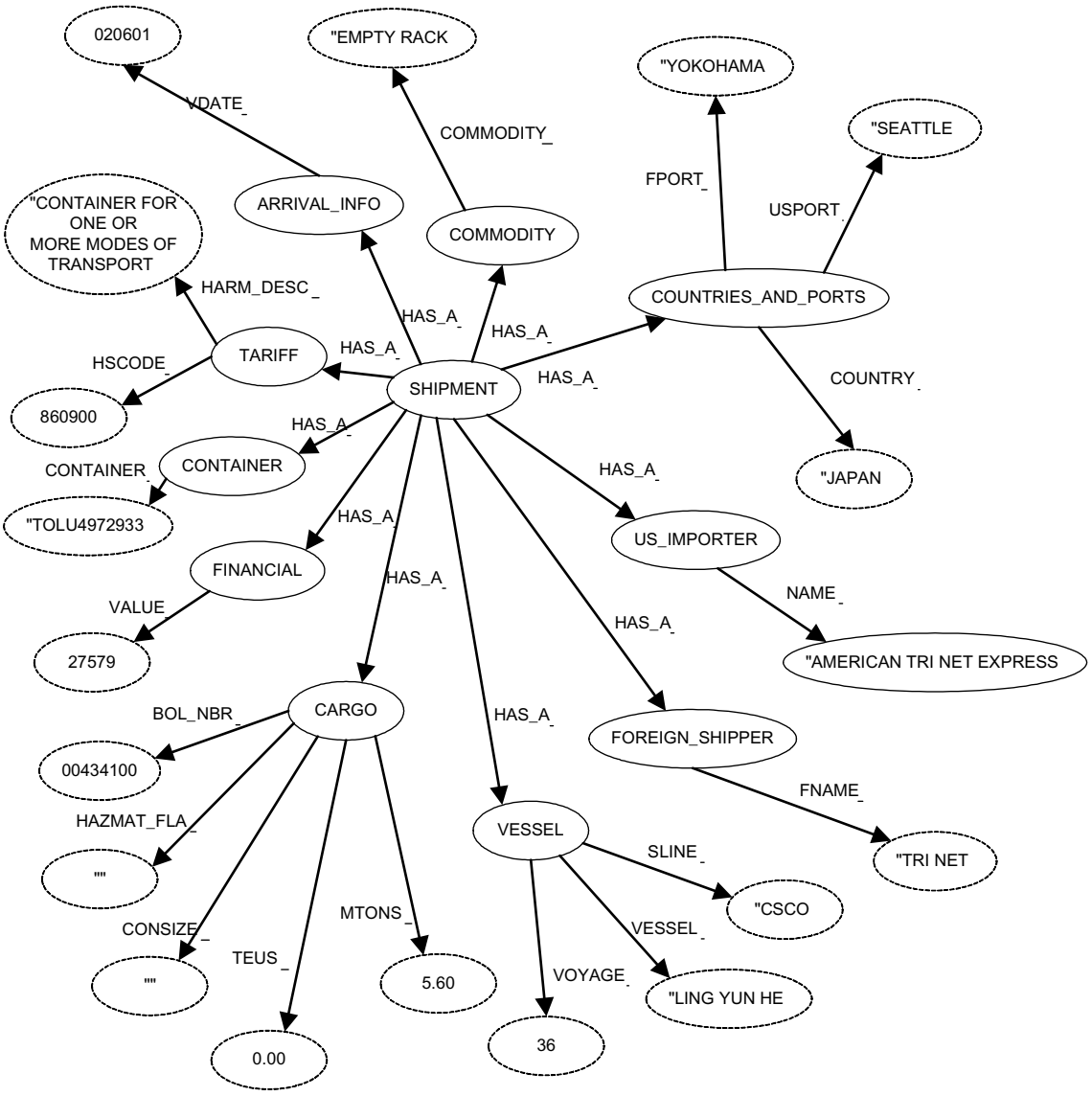

Fig. 17. Example of cargo information represented as a graph.

cargo looks like a normal shipment from Jamaica. Fig. 18 shows a graphical representation of a portion of the graph (for space reasons) containing the anomaly.

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 (shown as the bold edge and darkened vertex in Fig. 18), the GBAD-P algorithm is able to successfully discover the anomaly. Similarly, when the shipment instance in the graph is missing some financial information (the dotted and dashed edges and vertices in Fig. 18), GBAD-MPS reports the instance as anomalous.

According to the CBP, an estimated $2 billion in illegal textiles enter the US every year [3]. One of the more common methods of alluding authorities is accomplished using what is called transshipment. The

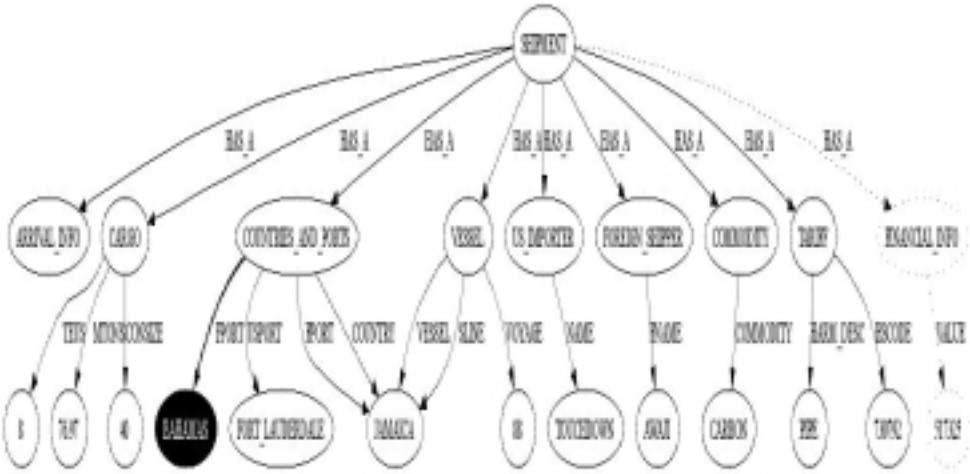

Fig. 18. Graph representation of cargo shipment containing the anomaly, with an insertion in bold and removals represented as dotted lines.

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 US 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.

6.3. Comparison to non-graph-based approach

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 (www.RuleQuest.com). The objective of the Gritbot tool is to look for anomalous values that would compromise the integrity of data to be analyzed by other data modeling tools such as Cubist, See5, etc.

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 value, the more possible anomalies are found.

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 $\sim$ 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 tools 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 this case, $\sim$250 out of the $\sim$1,000 records. When we arbitrarily modify 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 does 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 anomaloues (however, not the most anomalous). Why? It again gets to what Gritbot can determine to be statistically significant. Of all of the $\sim$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 $\sim$250 records is anomalous because it had enough other records containing the value "CSCO" to determine that its existence in these other records was signfiicant. In short, it really gets to 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 may not find the anomaly.

7. Experiments on 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].

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 US 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. 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 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.

8. Conclusions and future work

In this paper we presented our definition of a graph-based anomaly and how that is manifested in data that is represented as a graph. The purpose of this work 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.

First we described three novel algorithms, each with the goal of uncovering one of the specified anomalous types. With the aide of several simple examples, we were able to describe the approaches and the simplicity of their implementation. Then we 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. Finally, 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 presented results on data consisting of network intrusions where one of our algorithms (GBAD-MDL) was very successful in discovering the anomalies.

Currently, there is no connection between compressed substructures. In other words, once instances of a particular substructure have been compressed to a single vertex (i.e., when running multiple iterations of the GBAD-P algorithm), and a link (and vertex) extends from that compressed substructure, there is no information telling us what actual vertex is connected to that extension. If we can save that information for future iterations, that could prove to be useful for two reasons. One, we could possibly generate a

better probabilistic model for determining which extensions are actually more anomalous. Second, it would allow us to create a better picture of the anomaly, as an analyst would be able to view the entire chain of connections associated with the anomaly.

The minimum description length principle was used as the metric for determining the normative pattern, as well as by the GBAD-MDL algorithm for discovering substructures with minor modifications. This metric is a key component of the algorithms presented in this work. However, other graph-based approaches have used various other metrics for determining the normative pattern in a graph. Future work should include an analysis of these other metrics in lieu of the MDL approach.

Acknowledgements

This effort was partially sponsored by the Air Force Research Laboratory under agreement F30602-01-2-0570. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the US Government.

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, 2004, 112–124.
[2] D. Cook and L. Holder, Graph-based data mining, *IEEE Intelligent Systems* **15**(2) (2000), 32–41.
[3] 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).
[4] W. Eberle and L. Holder, *Detecting Anomalies in Cargo Shipments Using Graph Properties*. Proceedings of the IEEE Intelligence and Security Informatics Conference, 2006.
[5] M. Faloutsos, P. Faloutsos and C. Faloutsos, *On Power-law Relationships of the Internet Topology*. Proceedings of the conference on applications, technologies, architectures, and protocols for computer communications, SIGCOMM, 1999, 251–262.
[6] M. Hampton and M. Levi, Fast spinning into oblivion? Recent developments in money-laundering policies and offshore finance centres, *Third World Quartely* **20**(3) (June 1999), 645–656.
[7] KDD Cup 1999. Knowledge Discovery and Data Mining Tools Competition. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.
[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, 2003, 171–178.
[9] C. Noble and D. Cook, *Graph-Based Anomaly Detection*. Proceedings of the 9^{th} ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, 631–636.
[10] M. Rattigan and D. Jensen, The case for anomalous link discovery, *ACM SIGKDD Explor Newsl* **7**(2) (2005), 41–47.
[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, 2005, 74–81.
[13] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip and D. Zerkle, *GrIDS – A Graph Based Intrusion Detection System for Large Networks*. Proceedings of the 19^{th} 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) (2005), 48–55.
[15] US 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).