# Discovering Structural Anomalies in Graph-Based Data

**Conference Paper** · November 2007

DOI: 10.1109/ICDMW.2007.91 · Source: IEEE Xplore

CITATIONS
139

READS
668

**2 authors:**

William Eberle
Tennessee Technological University
**73** PUBLICATIONS **1,357** CITATIONS
SEE PROFILE

Lawrence Holder
Washington State University
**247** PUBLICATIONS **6,261** CITATIONS
SEE PROFILE

# Discovering Structural Anomalies in Graph-based Data

**William Eberle**
Department of Computer Science and Engineering
University of Texas at Arlington
Box 19015, Arlington, TX 76019-0015
eberle@cse.uta.edu

**Lawrence Holder**
School of Electrical Engineering & Computer Science
Washington State University
Box 642752, Pullman, WA 99164-2752
holder@wsu.edu

Keywords: anomaly detection, graph, minimum description length, information theoretic, fraud

## 1. Abstract

The ability to mine data represented as a graph has become important in several domains for detecting various structural patterns. One important area of data mining is anomaly detection, particularly for fraud, but 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 paper we present graph-based approaches to uncovering anomalies in domains where the anomalies consist of entity/relationship alterations that closely resemble non-anomalous behavior. We introduce three novel 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 and uses the minimum description length principle to discover those substructure instances that contain anomalous entities and relationships.

Using synthetic and real-world data, we evaluate the effectiveness of each of these algorithms in terms of each of the types of anomalies. Each of these algorithms demonstrates the usefulness of examining a graph-based representation of data for the purposes of detecting fraud.

## 2. Introduction

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

[Noble and Cook 2003] used the SUBDUE application to look at the problem of anomaly detection from both the anomalous substructure and anomalous sub-graph perspective. 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. 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.

[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 were interesting or not, without having any preconceived notions of meaningful patterns. One of the disadvantages of this approach was that while it was domain independent, it assumed that the user was querying the system to find interesting relationships regarding certain nodes.

The AutoPart system presented a non-parametric approach to finding outliers in graph-based data [Chakrabarti 2004]. Part of this approach was to look for outliers by analyzing how edges that were removed from the overall structure affected the minimum descriptive length (MDL) of the graph [Rissanen 1989]. 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.

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 2003] had proposed, they hypothesized that the important nodes (or people) were the ones who had the greatest effect on the entropy of the graph when they were removed. However, in this approach, the idea of important nodes did not necessarily mean that they were anomalous.

In the 2005 SIGKDD Explorations, a couple of different approaches to graph-based anomaly detection were presented. Using just bipartite graphs, [Sun et al. 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. 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.

## 3. Graph-based Anomalies

Setting up fraudulent web-sites, "phishing" for credit cards, stealing calling cards, and creating bogus bank accounts are just some of the countless examples of scams that have succumb everyone from the individual investor to large corporations. In every case, the fraudster has attempted to swindle their victim and hide their dealings within a morass of data that has become proverbially known as the "needle in the haystack". Yet, even when the data is not relatively large in size, the ability to discover the nefarious actions is still ultimately difficult due to the *mimicry* of the perpetrator.

The idea behind the approach presented in this paper is to find anomalies in graph-based data where the anomalous substructure in a graph is part of (or attached to or missing from) a 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**: *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. 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." [Hampton and Levi 1999]. That makes this definition of an anomaly extremely relevant.

### Anomaly Types

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

1. *A vertex exists that is unexpected.*
2. *An edge exists that is unexpected.*
3. *The label on 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 (or a self-edge to a vertex) is absent.*

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

### Assumptions

Many of the graph-based anomaly detection approaches up to now have assumed that the data exhibits a power-law distribution. The advantage of the approaches presented in this paper is that it does not assume the data consists of a power-law behavior. In fact, no standard distribution model is assumed to exist. All that is required is that the data is *regular*, which is typical of most production data, such as telecommunications call traffic, financial transactions, and cargo shipments. In other words, most real-world data consisting of user transactions exhibits regular patterns of behavior.

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

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

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

**Assumption 2**: *The graph is regular.*

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

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

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

**Assumption 4**: *The normative pattern is connected.*

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

## 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 one has an idea what is normal behavior, deviations from that behavior could constitute an anomaly. However, the issue with those approaches is that one has to have the data in advance in order to train the system, and the data has to already be labeled (i.e., fraudulent versus legitimate).

Our work has resulted in the development of three algorithms, which we have implemented using a tool called GBAD (Graph-based Anomaly Detection). GBAD is an *unsupervised* approach, based upon the SUBDUE graph-based knowledge discovery system [Cook and Holder 1998]. 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:

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

where $G$ is the entire graph, $S$ is the substructure, $DL(G|S)$ is the description length of $G$ after compressing it using $S$, and $DL(S)$ is the description length of the substructure.

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

### *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.

***Algorithm***

The detailed GBAD-MDL algorithm is as follows:

For a graph *G:*

```
Find the top-k substructures S_i, that minimizes $M(S_i,G) \le M(S_j,G)$ for all j,
    where $S_i, S_j \subseteq G$.
Find all matching instances of $S_i$ such that the cost of transformation, $C(I_j,S_i)$, for the graph
    structure of $I_j$ to match the graph structure of $S_i$, is greater than 0.
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.
Output all $I_i$ where $A(I_i) \le A(I_j)$ for all j,
    where $I_i, I_j \subseteq S_i$.
```

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 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 also 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 is not found in the top normative pattern.

Through out this paper, 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*.

***Example***

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

In [Noble and Cook 2003], they presented the example shown in Figure 4.1.

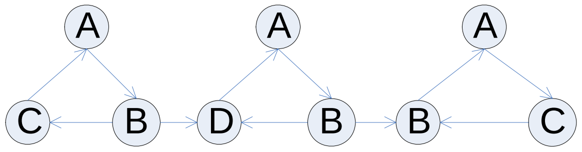

**Figure 4.1. Noble and Cook example.**

Running the GBAD-MDL algorithm on this example results in the anomalous substructure as shown in Figure 4.2.

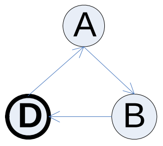

**Figure 4.2. Anomalous substructure from Noble and Cook example using GBAD-MDL.**

With Noble and Cook's approach, the **D** vertex is shown to be the anomaly. While correct, the importance of this new approach is that a larger picture is provided regarding its associated substructure. In other words, not only are we providing the anomaly, but we are also presenting the context of that anomaly within the graph (the individual anomaly is in **bold**.) It should also be noted that no other substructures were reported as anomalous (i.e., no false positives).

## *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 possible extensions.

***Algorithm***

The detailed GBAD-P algorithm is as follows:

**Find the top-*k* substructures** $S_i$, that minimizes $M(S_i, G) \leq M(S_j, G)$ for all *j*, where $S_i, S_j \subseteq G$.

**Compress** $G$ by $S_i$, and **Find the top-k substructures** again.

**Find all instances** $I_j$ that match the substructure $S_i$.

**Create extended instances** $I'_j$ that consist 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 $S_i$.

**Determine anomalous value** for each $I'_j$ by finding matching instances of $I'_j$ in set $I'$, and calculating the value of $A(I'_j) = |I'_j| / |I'|$ where $|I'_j|$ is the cardinality of the set of instances that match $I'_j$, and $|I'|$ is the cardinality of the set of extended instances of $S_i$.

**Output** $I'_j$ where $A(I'_j) \leq A(I'_k)$ for all *k*, where $I'_j, I'_k \subseteq I'$, and its probability is less than a user acceptable threshold.

**Compress** $G$ by the graph structure of $I'_j$.

**Repeat** the **Find the top-k substructures** step and start again at the **Find all instances** step.

$A(I'_j)$ 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.

***Example***

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

Take the fairly regular example shown in Figure 4.3.

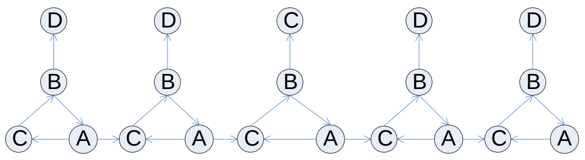

**Figure 4.3. Simple graph for GBAD-P example.**

After one iteration, the best substructure is shown in Figure 4.4.

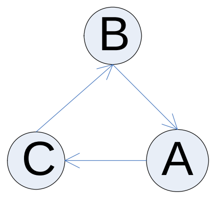

**Figure 4.4. Best substructure from simple graph for GBAD-P example.**

Then, on the second iteration, this substructure is compressed to a single vertex, extensions are evaluated, and the resulting anomalous substructure is shown in Figure 4.5 (with the compressed substructure expanded just for this visualization):

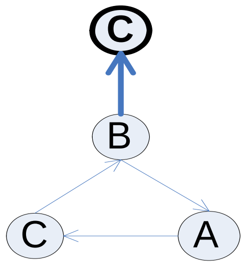

**Figure 4.5. Anomalous substructure from simple graph using GBAD-P.**

Again, the edge and vertex (shown in **bold** in Figure 4.5) is labeled as the actual anomaly, but the entire anomalous substructure is output for possible analysis.

## *Maximum Partial Substructure (GBAD-MPS)*

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.

***Algorithm***

The detailed GBAD-MPS algorithm is as follows:

**Find the top-*k* substructures** $S_i$, that minimizes $M(S_i, G) \leq M(S_j, G)$ for all *j*, where $S_i, S_j \subseteq G$.

**Find all ancestor** substructures $S'$ such that $S' \subseteq S$.

**Find all instances** $I'$ of $S'$.

**Determine the anomalous value** for each instance of $I'$, if $I'_n$ is not a subset of any instance of $I$, as the *cost of transformation * frequency.*

**Output** $I'_n$ 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 best substructure).

***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 Figure 4.6.

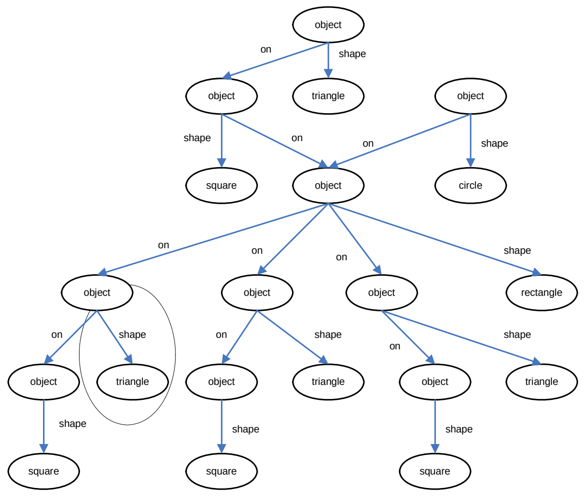

**Figure 4.6. Simple graph for GBAD-MPS example (with edge and vertex that will be removed).**

The normative pattern from this graph is shown in Figure 4.7.

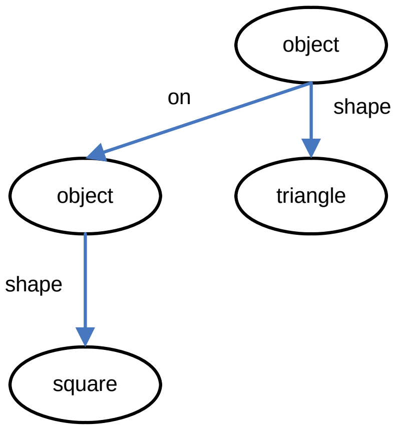

**Figure 4.7. Normative pattern from simple graph used for GBAD-MPS example.**

Suppose we remove the edge and its associated vertex shown circled in Figure 4.6. In this case, the parts shown in Figure 4.8 are removed.

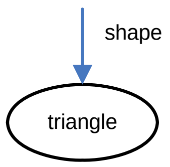

**Figure 4.8. Parts removed from simple graph in GBAD-MPS example.**

Running the GBAD-MPS algorithm on this modified graph results in the anomalous instance shown in Figure 4.9 being discovered.

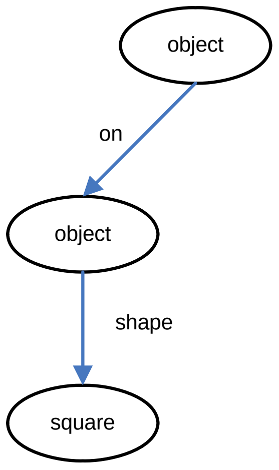

**Figure 4.9. Anomalous instance from simple graph when using GBAD-MPS.**

So, in this case, the instance was anomalous because it did not contain the link to a triangle.

## 5. Synthetic Experiments

The following sections contain the results when applying the algorithms to randomly-generated synthetic data.

### *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.

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 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.

## Metrics

Each test consists of a single graph from which 30 randomly altered graphs are generated. The output shown consists 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 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 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.

## Information Theoretic Results

For each of the following GBAD-MDL tests, to improve performance, we will prune substructures whose pattern matching values are less than their parent's. While this could result in a loss of accuracy, previous testing has shown this to not be the case in most situations.

### *Modification Tests*

Figure 5.1 and Figure 5.2 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. (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. In fact, in the following section, some lower more granular thresholds are used to show a clearer picture of the trends.

When we use graphs with larger normative patterns, smaller thresholds need to be used in order to uncover small changes to the structure. For example, the results shown in Figure 5.3, Figure 5.4 and Figure 5.5 (for graphs of 1000 vertices/1000 edges, with a normative pattern of 30 vertices/30 edges) represent the use of finer thresholds for the discovery of very small anomalies (percentage wise).

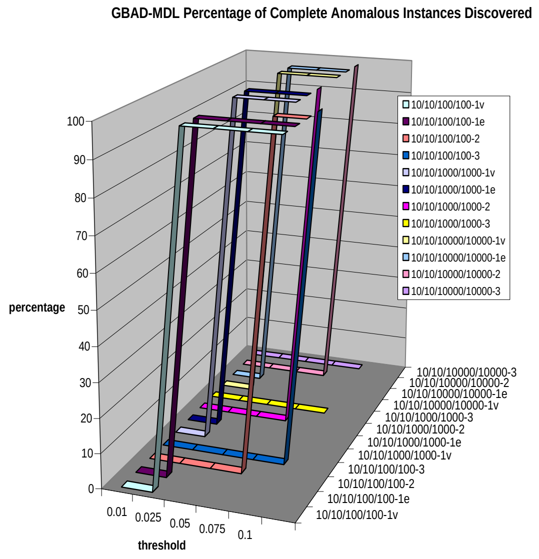

**Figure 5.1. Percentage of GBAD-MDL runs where all anomalies discovered.**

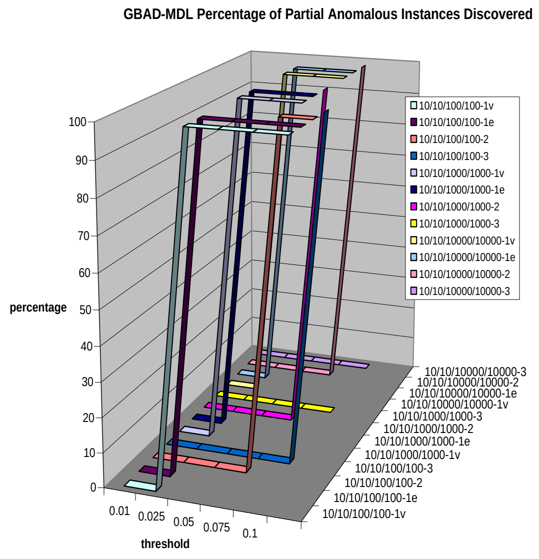

**Figure 5.2. Percentage of GBAD-MDL runs where at least one anomaly is discovered.**

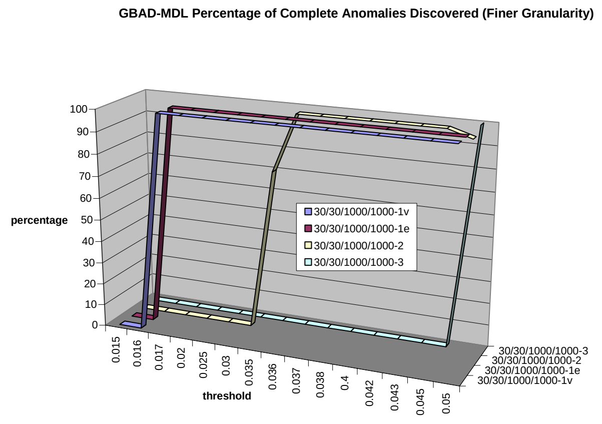

**Figure 5.3. Percentage of runs where all anomalies discovered (finer granularity).**

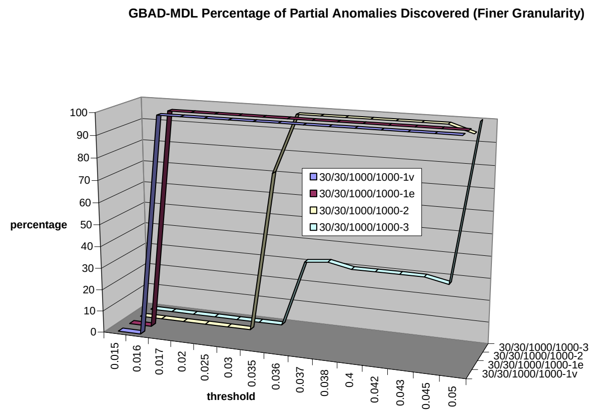

**Figure 5.4. Percentage of runs where at least one anomaly discovered (finer granularity).**

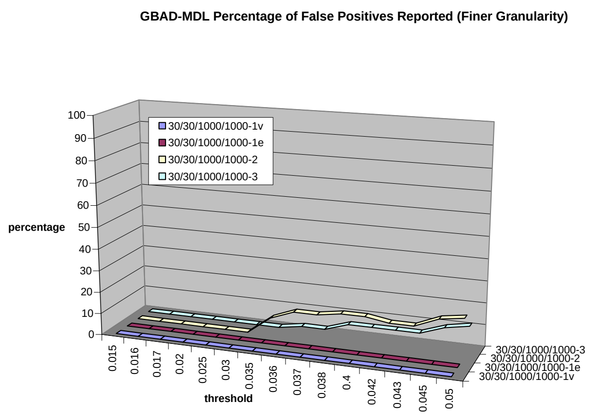

**Figure 5.5. Percentage of runs containing false positives (finer granularity).**

## Probabilistic Results

For each of the following tests, we implemented the following GBAD settings:

- pruned substructures whose pattern matching value is less than their parent's (for performance)
- only analyzed extensions found in the top 10 best substructure's (for performance)
- allowed the program to iterate (i.e., compress and run again) until there was 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. (Again, for space reasons, only a portion of the results are shown.)

### *Insertion Tests*

Figure 5.6 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.

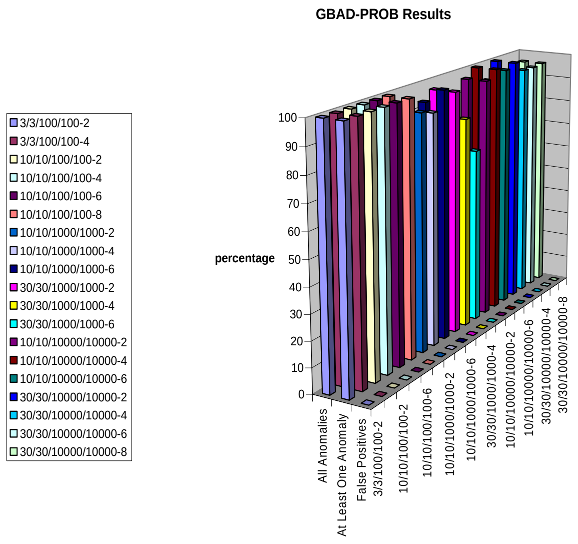

**Figure 5.6. GBAD-P results on anomalous insertions.**

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).

As was mentioned earlier, this approach is incrementally discovering single extensions from compressed substructures. Since the random generation of the anomalies could introduce multiple edges between the same two vertices, compressing the normative pattern at an iteration could result in a second duplicate edge being missed or ignored, as it would just appear to be a self-edge to a compressed substructure.

## Maximum Partial Substructure Results

For each of the following tests, we implemented the following settings:

- pruned substructures whose pattern matching value is less than their parent's (for performance)
- only analyzed 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.

***Deletion Tests***

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 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.

## Performance

One of the factors to consider in evaluating these algorithms is their respective performances. For graphs of 100 vertices and 100 edges, the average running times for all three algorithms were one second or less. However, while the GBAD-P and GBAD-MPS algorithms ran for up to 2200 seconds when presented with the graphs of 10,000 vertices and 10,000 edges, the running times for the GBAD-MDL algorithm were significantly more. 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.

## Summary

With an almost 100% discovery rate for each algorithm on all of the graphs, using varying sizes of normative patterns and anomalies, each approach has clearly shown to be useful at discovering a specific type of anomaly. While the algorithms do not appear to be useful outside of their intended targets, no graphs of any size or any anomaly went undetected by all three approaches.

One of the advantages of these algorithms lies in the fact that they are not implementation dependant. While we are using GBAD as a tool to run these algorithms, they can be implemented with any graph-based pattern discovery tool, as long as that tool maintains a list of substructures and instances that are being evaluated. Another advantage is that these algorithms do not just return the pattern of the anomaly – they also return the actual anomalous instances within the data. In a real-world scenario, that can be invaluable to an analyst who may need to act upon a fraud situation before the losses are too great. The disadvantage of these algorithms is that they are focused on specific anomalies: modifications, insertions or deletions. Thus, in a real-world setup, it would require that all three algorithms be used in conjunction, as the type of anomaly would most likely be unknown.

## 6. Real-World Experiments

One area that has garnered much attention recently is the analysis and search of imports into the United States. The largest number of imports into the U.S. arrive via ships at ports of entry along the 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.

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 6.1 shows a portion of the actual graph that we will use in our anomalous detection experiments.

While we were not given any labeled data from the CBP (i.e., which shipments were illegal, or anomalous, and which ones were not), we can draw some results from random changes and from simulations of publicized incidents.

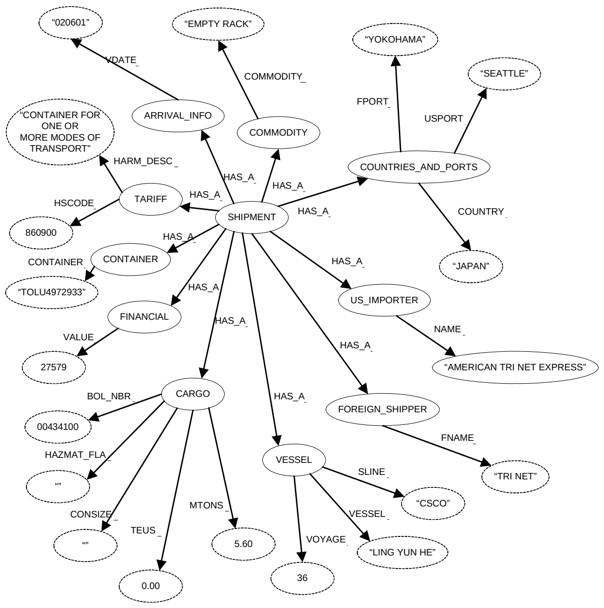

**Figure 6.1. Example of cargo information represented as a graph.**

## 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.

## Real-world Scenarios

In [Eberle and Holder 2006], real-world cargo shipment occurrences were generated so as to show how graph properties can be used to determine structural anomalies in graphs. While that approach was successful in discovering graphs that contained anomalies, the exact anomalies were not part of the output. Using the GBAD algorithms on these same data sets, we can display the actual anomalies.

One example is from a press release issued by the U.S. Customs Service. The situation is that almost a ton of marijuana is seized at a port in Florida [U.S. Customers 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, food and bicycles from Jamaica. Figure 6.2 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 Figure 6.2), 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 Figure 6.2), GBAD-MPS reports the instance as anomalous.

According to the CBP, an estimated $2 billion in illegal textiles enter the U.S. every year [Customs and Border Protection 2003]. One of the more common methods of alluding authorities is accomplished using what is called transshipment. The CBP defines transshipment as “A false declaration or information given in order to circumvent existing trade

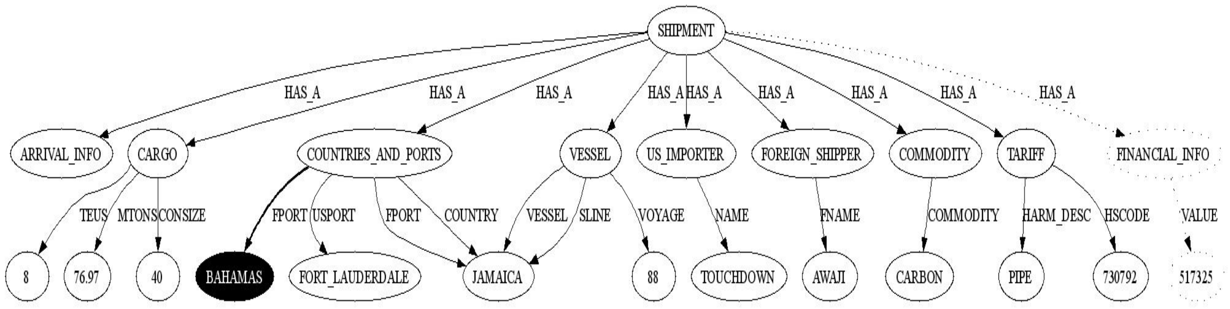

**Figure 6.2. Graph representation of cargo shipment containing the anomaly, with an insertion in bold and removals represented as dotted lines.**

laws for the purpose of avoiding quotas, embargoes or prohibitions, or to obtain preferential duty treatment." In order to circumvent quotas, the fraudster will change the country of origin of their goods. For example, they may ship the goods into Canada or Mexico, change the country-of-origin, and ship into the U.S. free from tariffs under the North American Free Trade Agreement (NAFTA).

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

## 7. Conclusions and Future Work

The three algorithms presented in this paper are able to discover an anomaly when it consists of a small change to the normative pattern. Using the minimum description length principle and probabilistic approaches, we have been able to successfully discover anomalies in graphs and normative patterns of varying sizes with minimal to no false positives. Results from both synthetic and real-world data demonstrate the effectiveness of the approaches.

Our next step in this effort is to modify the algorithms to handle data that contains a mixture of anomalous types. If a particular data set contains more than one type of anomaly (e.g., modification and deletion), these algorithms are unable to consistently discover the anomalies. The issue lies in the fact that a change can alter the normative pattern such that the basis for a particular algorithm is no longer valid. For instance, if the anomaly in a graph consists of a deletion and a modification, the deletion results in an instance that is smaller than the normative pattern, and as such, the GBAD-MDL algorithm will not analyze the structure, thus missing the anomalous modification. We are pursuing a couple of different alternatives to solve this problem, including the carrying along of all instances regardless of their size. This would allow GBAD-MDL and GBAD-P to perform comparisons and extensions (respectively) to instances that would not have been considered in the current implementation.

## 8. References

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.

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

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

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

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.

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.

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

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

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.

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)