

Compression versus Frequency for Mining Patterns and Anomalies in Graphs

William Eberle
Tennessee Technological University
Department of Computer Science
Cookeville, TN USA
1-931-372-3278
weberle@tntech.edu

Lawrence Holder
Washington State University
School of Electrical Engineering & Computer Science
Pullman, WA. USA
1-509-335-6138
holder@wsu.edu

ABSTRACT

Discovering patterns in data represented as a graph has been an important focus of research in a variety of domains such as the web, biological data, and networks. In general, the two different approaches to discovering the normative pattern in a graph have focused on either *frequency* or *compression*. In addition, the ability to discover anomalies in data represented as a graph has demonstrated advantages over more traditional data mining approaches. However, one of the major issues with graph-based approaches is the run-time performance associated with detecting anomalies. In this paper, we analyze the differences between using frequency versus compression as it pertains to the discovery of both normative and anomalous patterns. Using synthetic and real-world graphs for our experiments, we explore the varying effects of subgraph and anomaly discovery based upon the size of the graph, the number of normative patterns within a subgraph, and the number of random edges (noise) that can affect subgraph discovery. In addition, we explore the scalability of such an approach, with the potential application to real-world problems.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications - *Data Mining*.

General Terms

Algorithms.

Keywords

Graph-based, knowledge discovery, anomaly detection

1. INTRODUCTION

Over the last decade, several methods have been developed for mining data represented as a graph. One important area of graph mining is the discovery of frequent subgraphs in a set of graphs or within one large graph. With frequent subgraph miners, the most interesting substructure is the largest one (or ones) that meet the minimum support – a parametric value that is chosen by the user. Whereas, compression-based graph miners discover those

subgraphs that maximize the amount of compression that a particular substructure provides a graph. The algorithms associated with these two approaches are not only different, but they also may result in dramatic performance differences, both in running-times as well as accuracy when it comes to pattern discovery.

In addition, the ability to discover anomalies is a vital task for a wide range of organizations, such as businesses or national defense, and involves diverse applications, such as fraud and intrusion detection. Traditionally, methods for discovering anomalies consist of supervised and unsupervised approaches using techniques such as classification, clustering, nearest neighbors, and statistics [2]. One of the primary issues with these approaches is their inability to handle complex, structural data. One approach to this issue involves the detection of anomalies in data that is represented as a graph. The advantage of *graph-based anomaly detection* is that the *relationships* between elements can be analyzed, as opposed to just the elements' attributes, for structural oddities in what could be a complex, rich set of information. However, attempts at applying graph-based approaches to anomaly detection have encountered a few challenges.

This paper compares these two approaches and provides some empirical results that highlight their differences. Experimenting with graphs of different size, structure and density, as long as the frequent subgraph mining approach has a sufficient minimum support threshold, both approaches discover the same normative pattern, albeit with significantly different running times. Similar running time differences are observed when anomaly detection is performed, and while the reported anomalies are identical between the two approaches, the compression approach appears more susceptible to noise that result in false positives.

2. FREQUENT SUBGRAPH MINING

There have been various implementations of *frequent subgraph* miners. Approaches like GASTON [16] and gSpan [17] return all *frequent* substructures in a database that is represented as a graph. Using a depth-first search on the input graphs, the algorithm constructs a hierarchical search tree based upon the DFS code assigned to each graph. Then, from its canonical tree structure, these algorithms perform a traversal of the tree in order to discover the frequent subgraphs. Other approaches, such as Grew and FSG also return all of the frequent subgraphs (substructures) in a database of transactions that have been represented as a graph [18]. However, unlike GASTON and gSpan, they use an Apriori-style breadth-first search. The algorithm takes the input graphs

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MLG '11 San Diego, CA, USA

Copyright 2011 ACM 978-1-4503-0834-2 ...\$10.00.

and performs a level-by-level search, growing patterns one edge at a time. The core of the algorithm lies in its candidate generation and counting that are used to determine the frequent subgraphs. In order to mine large graphs for frequent subgraphs, Huan et al. propose a maximal frequent subgraphs approach called SPIN as an improvement to gSpan [19]. By mining only subgraphs that are not part of any other frequent subgraphs, they are able to reduce the number of mined patterns by orders of magnitude. This is accomplished by first mining all frequent trees from a graph, and then reconstructing all maximal subgraphs from the mined trees. Zeng et al. looked at the problem of dense graphs by mining the properties of quasi-cliques [20]. Using a system called Cocain, they propose several optimization techniques for pruning the unpromising and redundant search spaces. To help combat the subgraph isomorphism issue, Gudes et al. propose an Apriori-based algorithm using disjoint paths [21]. Following a breadth-first enumeration and what they called an “admissible support measure”, they are able to prune candidate patterns without checking their support, significantly reducing the search space. MARGIN is another maximal subgraph mining algorithm that focuses on the more promising nodes in a graph [22]. This is accomplished by searching for promising nodes in the search space along the “border” of frequent and infrequent subgraphs, thus reducing the number of candidate patterns.

There have also been several attempts at creating graph miners that use maximal *compression*. Implementations such as GraphScope [23] and SUBDUE [24] return the substructures that compress the graph the best using the Minimum Description Length (MDL) principle. SUBDUE uses a beam search (a limited length queue of the best few patterns that have been found so far), growing patterns one edge at a time, continually discovering what substructures best compress the description length of the input graph. Graphscope also uses an MDL compression to discover graphs, albeit in the context of graphs changing over time. This approach examines graph “snapshots”, and when a new graph snapshot cannot fit well into an old segment (in terms of compression), a change-point is introduced and a new segment starts at that time-stamp. The purpose of this approach is to find communities of time-evolving graphs along with the change points, all represented in a matrix. The core of both of these approaches is in their compression strategy.

3. GRAPH ANOMALY DETECTION

Recently, there have been significant strides in the development of graph-based approaches to anomaly detection. In 2003, Noble and Cook used the SUBDUE application to look at the problem of anomaly detection from both the anomalous substructure and anomalous subgraph perspective [12]. 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 subgraph* detection which dealt with how anomalous a subgraph (i.e., a substructure that is part of a larger graph) was to other subgraphs.

Lin and Chalupsky [26] took a different approach and applied what they called rarity measurements to the discovery of unusual *links* within a graph. Using various metrics to define the commonality of paths between nodes, the user was able to determine whether a path between two nodes was interesting or not, without having any preconceived notions of meaningful patterns. The AutoPart system presented a non-parametric

approach to finding outliers in graph-based data [27]. Part of Chakrabarti’s approach was to look for outliers by analyzing how *edges* that were removed from the overall structure affected the minimum description length 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. Using just *bipartite* graphs, Sun et al. [14] presented a model for scoring the normality of nodes as they relate to the other nodes. Again, using an adjacency matrix, they assigned what they called a “relevance score” such that every node x had a relevance score to every node y , whereby the higher the score the more related the two nodes. The idea was that the nodes with the lower normality score to x were the more anomalous ones to that node. In [28], 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.

Other approaches include Wan et al. [15] who use a link-based event detection method that clusters similar vertices together and then considers deviations from each vertex’s individual profile. Whereas, Huang uses probabilities to generate models that predict the likelihood of links with the topology of a graph [10]. More recently, Akoglu et al. present an algorithm called *OddBall* that searches weighted graphs based upon a set of rules to determine whether or not an anomaly exists [1]. Eberle and Holder [5] introduced the GBAD (Graph-Based Anomaly Detection) approach which uses information theoretic, probabilistic and maximum partial substructure approaches to discover all three types of graph-based anomalies: modifications, insertions and deletions. In their approach, anomalies are viewed as unexpected deviations from normative patterns, rather than just unexpected structure [6][8] [7][8].

However, with all of these approaches, the primary issue continues to be scalability. While certain small-world domains (e.g., Enron e-mail [13][4]) as well as specific graph representations (e.g., bipartite [14] and time-evolving graphs [25]) have yielded improvements in the run-time performance, application in a large real-world setting has been elusive. We propose the use of a frequent subgraph mining approach to help address these performance issues.

4. TWO APPROACHES

We have chosen two systems, GASTON and SUBDUE, as representatives of frequency and compression-based approaches.

4.1 Methodologies

The goal of GASTON is to return all frequent substructures in a graph using a depth-first search on the input graphs. The core of the GASTON algorithm (and other approaches, like gSpan) is the construction of its hierarchical search tree based upon the DFS code assigned to each graph. Using its canonical tree structure, the algorithm performs a traversal of the tree in order to discover the frequent sub-graphs. This search through the DFS codes is repeated on each edge until its frequency drops below the user specified minimum support threshold.

The key to the GASTON algorithm lies in its canonical labels, search strategy, and effective pruning strategy. The canonical labeling system that is employed partitions the graph according to a DFS lexicographic order that is a minimum DFS encoding. Because there could be many possible DFS trees, it chooses the

one with the smallest lexicographic value. GASTON also uses a traversal of the DFS code tree, where it is able to prune false positives before they are compared (unlike what happens in approaches like FSG). In addition, GASTON's performance is improved by first searching for frequent paths, then searching for frequent trees, and then finally for frequent subgraphs.

The goal of SUBDUE is different from frequent subgraph miners in that it chooses to return the substructures that compress the graph the best. Using a beam search (a limited length queue of the best few patterns that have been found so far), the algorithm grows patterns one edge at a time, continually discovering what substructures best compress the description length of the input graph. After extending each substructure by one edge, it evaluates the substructure based upon its compression value (the higher the better). A list is maintained of the best substructures, and this process is continually repeated until either there are no more substructures to compress or a user-specified limit is reached.

There are two key components to the SUBDUE algorithm. First, there is the use of a compression methodology. The idea behind this approach is that there could be interesting patterns that are less frequent than other patterns. In short, using the Minimum Description Length principle, the graph is represented as the number of bits needed to encode an adjacency matrix representation of the graph. Thus, the substructure that reduces the number of bits the most (by replacing all instances of the substructure with a single vertex) is considered the better substructure. Second, SUBDUE collects the instances of a subgraph by finding those that match (graph isomorphism) from a list of candidates, rather than performing a subgraph isomorphism test each time. SUBDUE further constrains the graph isomorphism test to run in polynomial time. In addition, several optimization steps have been incorporated into SUBDUE to reduce the frequency with which it is performed. Some of these optimization techniques are controlled by an optional *-prune* parameter, while others are intrinsic. For instance, the *-prune* option tells SUBDUE to prune the search space by discarding substructures whose value is less than that of their parent's substructure. Since the evaluation heuristics are not monotonic, pruning may cause SUBDUE to miss some good substructures, however, it will improve the running time.

While both of these applications have some obvious algorithmic differences in their choices (e.g., canonical depth-first tree versus compression), there is one key difference in their results. GASTON returns the *frequent* sub-graphs from a set of input (transaction) graphs. SUBDUE returns the sub-graphs that are believed to represent the graph the best by resulting in the graph's maximum *compression*. Though the ultimate goal for both of these approaches is to find interesting patterns, the result could be a different set of patterns being discovered.

4.2 Algorithmic Analysis

Because of the algorithmic choices each methodology uses, there are differences in the amount of memory/space that is used for their internal structures. GASTON uses a sparse adjacency list representation of the graphs, and with its tree structure and pruning strategy (the depth-first search allows for removal of edges and whole graphs from the search space after all relevant patterns for that edge or graph has been processed), it uses a minimal amount of memory. Whereas, SUBDUE uses a

significant amount of memory due to the graph structures it builds and the maintenance of best substructures and graph instances.

Several optimization techniques have been employed by both of these systems, albeit different types of optimization. GASTON, as already mentioned, achieves its optimization through its pruning strategy, and of course, it does not do candidate generation. SUBDUE implements not only an optional aggressive form of pruning the list of best substructures, it also performs some heuristic measures to avoid some repetitive comparisons. Each of these optimizations not only reduces the amount of memory needed, they also increase their relative speed. However, SUBDUE does make more calls to the graph isomorphism test than GASTON, and the canonical tree ordering implemented in GASTON minimizes its search space.

While in general SUBDUE may be slower than GASTON, it does have several significant features. For instance, SUBDUE can handle both directed and undirected input graphs. (And other frequent subgraph miners have similar graph input restrictions, such as dealing with only bipartite graphs, or ignoring labels.) In addition, SUBDUE can perform inexact graph matching. By specifying a threshold, the user can indicate a level of acceptance for patterns that may not match completely. While it may be possible to implement some of these features into GASTON (or other approaches like FSG and gSpan), that could be problematic. For instance, allowing for directed graphs in GASTON could cause its tree structure to become a forest, and ultimately cause issues with both its traversal and pruning strategies.

It should also be mentioned that one of the advantages of the SUBDUE approach is its ability to handle more complex graphs in multiple domains. GASTON does not appear to be able to handle large, regular frequent sub-graphs (i.e., long patterns), and has only been documented on small graphs for specific domains. In addition, both GASTON and SUBDUE are not able to handle non-structural attribute values. For instance, the difference between 50.0001 and 50.0002 is handled the same as if the two values were 1 and 1000 (i.e., two uniquely different values).

There are two algorithmic disadvantages with SUBDUE. First, SUBDUE is a greedy algorithm. While this is useful from a performance perspective, there is the possibility that some frequent sub-graphs could be missed. Second, the compression mechanism used by SUBDUE is lossy. In other words, one cannot reverse the compression and restore the original graph.

One final observation regarding these algorithms should be made. While GASTON should be a significantly better performer in terms of memory and speed, it is also extremely complex. Considering some of the enhancements that would need to be made to make GASTON a more robust application, certain enhancements will increase the running time. Another possibility could be to incorporate some of GASTON's features, such as the canonical form for subgraph generation, into SUBDUE.

5. NORMATIVE PATTERN DISCOVERY

We now present the empirical results from experiments comparing the two approaches. The following algorithm represents the methodology used for constructing our synthetic input graphs:

Given:

$S = \text{subgraph}$

$N = \text{number of transactions}$

C = maximum copies of S per transaction
 E = maximum extra edges per transaction

Do:
 $T = \{ \}$
for $n = 1$ to N
 $t = c$ copies of S (where c is drawn from $Uniform(1, C)$)
Add extra random edges e to t (where e is drawn from $Uniform(1, E)$, e 's label randomly selected from unique labels, e 's vertices randomly selected from vertices in t)
Add t to T
return T (set of N transactions)

Using the above methodology, we kept the size of the subgraph S constant (10 vertices and 9 edges), and varied the other parameters as follows:

- N : 10, 100, 1000, 10000
- C : 1, 2, 3
- E : 5%, 10%, 15%, 20% (where the number of random edges is based upon the size of transaction)

We chose to use SUBDUE and GASTON in our experiments for one primary reason: very few graph mining tools are publicly available. (In addition, if needed, we can analyze/modify the code as they both are open-source.) For SUBDUE, we will use version 5.2 (www.subdue.org). All SUBDUE runs will use the default options (i.e., no parameters specified). The one exception to this is the use of the `-undirected` parameter. Since GASTON does not allow directed graphs, we can only compare undirected graphs. This parameter tells SUBDUE to only handle edges as undirected.

For GASTON, we will use version 1.1 (www.liacs.nl/~snijssen/gaston). Again, we will use the default options (i.e., no parameters), except for varying minimum support thresholds (*mst*). Here we make a slight modification to the baseline code. GASTON only outputs all substructure instances that meet the MST. So, in order to test our hypothesis, we will modify GASTON to output the single (best) substructure that maximizes (frequency * size). This will be in direct contrast to what SUBDUE discovers, where the best substructure is defined to be the subgraph that compresses the graph the best (MDL).

It should be noted that in the following sections, the x-axis, moving from left to right, represents increasing difficulty, based upon the number of copies (c) and edges (e).

5.1 Sparse Graphs

For these experiments, we analyzed graphs which are considered *sparse* (average degree < 2). Sparse graphs are very representative of real-world phenomenon such as network traffic, citation networks, and social networks. We chose a subgraph pattern S consisting of 10 vertices and 9 edges and embedded it into each of the N transactions. We vary the number of copies of S in each transaction according to the C parameter. We then add additional edges to each transaction according to the E parameter.

For the experiments shown in Figure 1, where $N=10$, when $C=3$, SUBDUE discovers the normative pattern in under 0.3 seconds, whereas for GASTON discovery of the normative pattern with an *mst* of 10% and number of extra random edges (E) at 10%, takes ~2 hours, and more than 16 hours when $E > 10\%$.

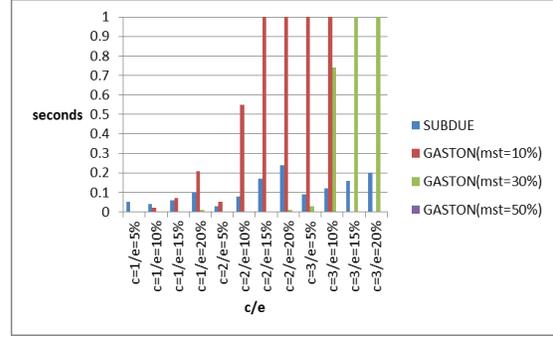


Figure 1. Runs for $N=10$ on sparse graphs.

In Figure 2, where $N=100$, when $C=3$, SUBDUE completes in under 5 seconds, whereas for GASTON discovery of the normative pattern with an *mst* of 10% and number of extra random edges (E) at 5%, takes ~7 minutes, and more than 3 hours when $E > 5\%$. Also, with an *mst* of 30%, runs with $E=15\%$ take ~19 minutes and runs with $E > 15\%$ do not complete in < 3 hours.

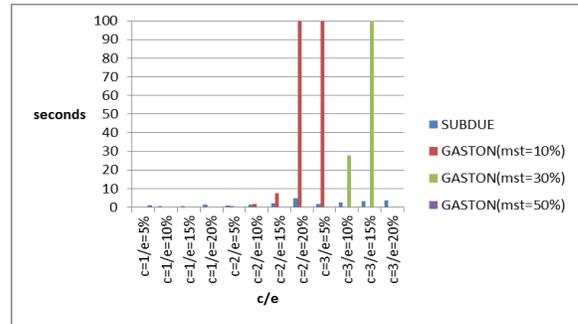


Figure 2. Runs for $N=100$ on sparse graphs.

For the experiments shown in Figure 3, where $N=1000$, when $C=3$, SUBDUE discovers the best substructure in under 4 minutes, whereas for GASTON discovery of the normative pattern with an *mst* of 10% and number of extra random edges (E) at 5%, takes ~13 minutes, and more than 3 hours when $E > 5\%$. Also, with an *mst* of 30%, runs with $E=15\%$ take over 2 hours and runs with $E > 15\%$ do not complete in less than 10 hours.

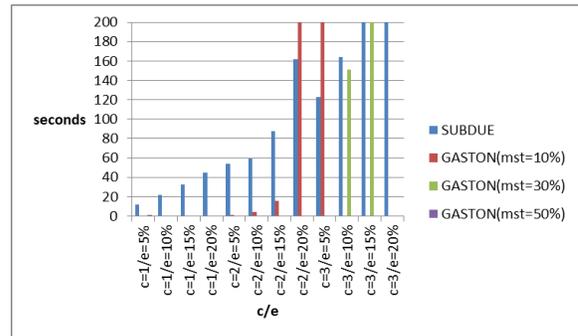


Figure 3. Runs for $N=1000$ on sparse graphs.

In Figure 4, where $N=10000$, SUBDUE runs range from a max of 16 hours to a minimum of 3 hours. GASTON performs well on all runs when the *mst* is at least 50% (less than one second in the worst case), but even at an *mst* of 30%, the runs grow significantly when more connectivity is introduced. Most runs with an *mst* of 10% had to be terminated after a few hours.

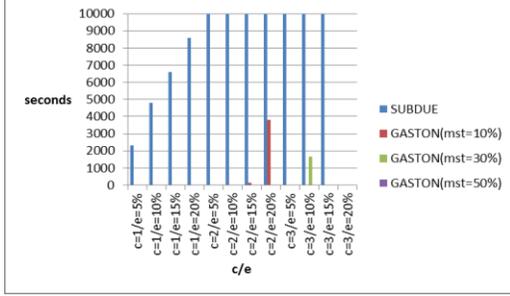


Figure 4. Runs for $N=10000$ on sparse graphs.

For all SUBDUE and GASTON runs that completed, the seeded normative pattern is reported as the best substructure, with no extra random edges. In addition, in all cases, both approaches report all instances of the best substructure. So, not only does the frequent subgraph mining approach produce identical results to the compression approach, but it is clearly faster when the number of transactions is large. Some of the performance degradation in GASTON, when the minimum support threshold is low, is due to the *connectedness* of the graph. The synthetic graphs that were generated consisted of a mixture of edges between vertices in the same subgraph and vertices in different subgraphs. The less connected the graph, the faster the running times. While it does not appear to affect GASTON when the threshold is at least 50%, runs under that minimum support threshold, in some cases, do not finish in a reasonable amount of time (albeit still better than the compression approach in some cases). Also, as is the case with all frequent subgraph mining approaches, the minimum support is parametric, which requires a user to make a decision about an appropriate threshold. With SUBDUE, there is no such threshold.

5.2 Dense Graphs

For these experiments, we analyzed graphs which are considered *dense* (average degree > 2). We chose a subgraph pattern S that is a complete graph consisting of 5 vertices and 10 edges (each vertex has degree 4) and embedded it into each of the N transactions. We vary the number of copies of S in each transaction according to the C parameter. We then add additional edges to each transaction according to the E parameter.

For the experiments shown in Figure 5, where $N=10$, the discovery of normative patterns by SUBDUE grows at a nonlinear rate, whereas the running times are negligible for GASTON when the mst is 30% or 50%. Only when the mst is 10% does it fluctuate based upon the number of extra random edges (E).

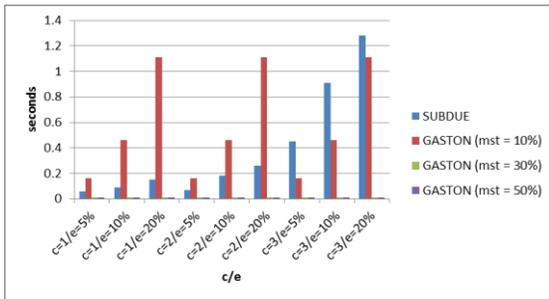


Figure 5. Runs for $N=10$ on dense graphs.

In Figure 6, where $N=100$, all SUBDUE runs complete in under 17 seconds. However, GASTON struggles to discover the normative pattern when there is more than one normative pattern

(C) per transaction. With an mst of 50%, the longest run is 1760 seconds. However, for mst of 10%, runs did not complete within 24 hours when there is at least 3 normative patterns per transaction, and an mst of 30% also struggles on graphs with more random edges (E), not finishing within 24 hours either. In short, except for when $E=20\%$ (i.e., the amount of random edges that add more connectivity to the graph), an mst of 50% is sufficient to discover the normative pattern and complete within a second for GASTON. Similar behavior is observed when $N=1000$.

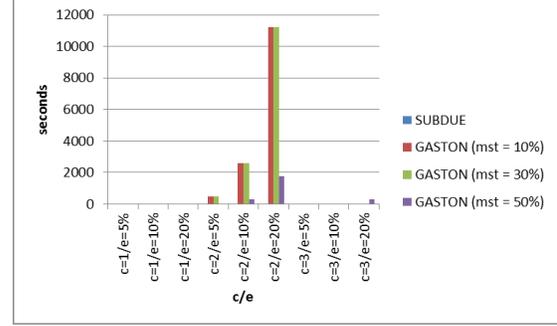


Figure 6. Runs for $N=100$ on dense graphs.

In Figure 7, where $N=10000$, SUBDUE runs range from a maximum of 31 hours to minimum running times of 1 hour. GASTON performs better on all runs when the mst is at least 50% (less than 1324 seconds in the worst case). However, runs with an mst of 10% and 30% do not complete in under 24 hours with $C=3$ (except for the case where the number of random edges is small and the mst is 30%), and trends upwards in running times similar to SUBDUE.

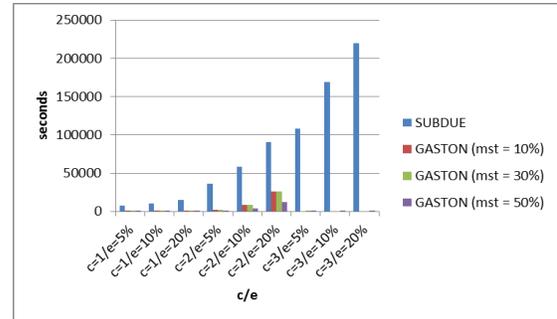


Figure 7. Runs for $N=10000$ on dense graphs.

In summary, across all of the experiments, when the graphs are small, SUBDUE discovers all of the normative patterns in a reasonable amount of time. However, a frequent subgraph mining approach like GASTON is consistent in its discovery and running times when the minimum support threshold is at 50%, plus executes faster than SUBDUE on larger graphs.

5.3 Real-World Example

In order to evaluate the two approaches on real-world data, we chose to use the publicly available data set of e-mails between employees from the Enron Corporation. The Enron e-mail dataset consists of not only messages, but also employee information such as their full name and work title. We created graphs based on the “social network” and company position of employees that start a “chain” of e-mails, where a chain consists of the originating e-mail and any subsequent replies or forwards to that e-mail. Each graph consists of substructures represented as shown in Figure 8.

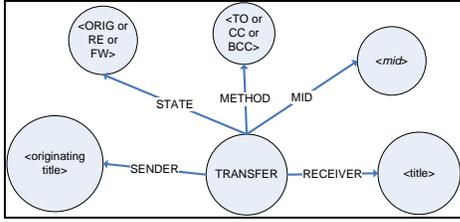


Figure 8. Substructure topology for Enron e-mail.

In this representation, a graph consists of individual, disconnected substructures that represent the “flow” of each e-mail that originates from someone with a specified title (e.g., Director). An e-mail can be sent by one or more TRANSFERS to one or more individuals with varying employment titles (represented by an arrow to show who sent the message to whom), and can either be sent back (as a reply or forward) to the <originating title>, or forwarded/replied on to other <title> entities.

There are many different employee titles within Enron, for example Presidents, each with expected different patterns of behavior. So, in order to determine the interesting patterns that each approach would discover, we created graphs that consist of all personnel grouped by their respective title. For instance, if a person is a VP, all of the e-mails that originated from them would be included in the graph of VPs. In addition, we will vary the *mst* on the frequent subgraph mining approach to see what effect that has on the discovered normative pattern. On the experiments of e-mails originating from Traders, GASTON, with an *mst* of 50%, reports the best substructure as the one shown in Figure 9.

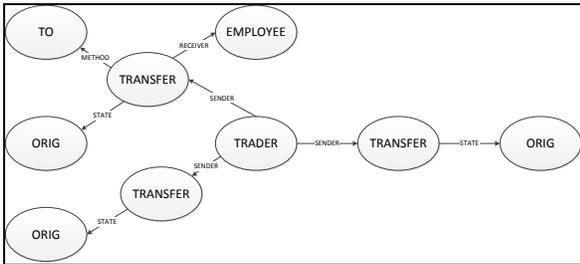


Figure 9. Best substructure for Traders with *mst*=50%.

However, if we lower the minimum support threshold below 50%, the normative pattern begins to shrink. Figure 10 shows the normative pattern when the *mst* value is below 50%.

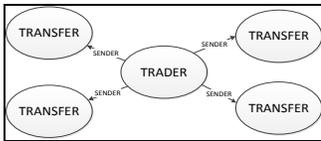


Figure 10. Best substructure for Traders with *mst* < 50%.

Clearly, a more interesting (structurally) pattern exists when the threshold represents a value whereby the normative pattern exists in at least half of the transactions. In the case where the minimum support is lower, we discover more instances of the pattern shown above in Figure 10, but even a slight increase from 45 to 50 allows one to see a larger pattern of behavior among traders. It is also interesting to note that increasing the *mst* to 55% results in a different AND smaller normative pattern, as shown in Figure 11.

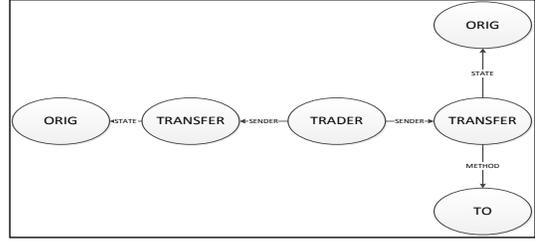


Figure 11. Best substructure for Traders with *mst*= 55%.

Based upon the size of the substructures discovered, the use of an *mst* of 50% appears to be a reasonable choice. In addition, SUBDUE, which does not use a minimum support threshold, also discovers the same substructure as shown in Figure 9, where an *mst* of 50% was used by the frequent subgraph miner GASTON. Similar results using GASTON and SUBDUE are observed in graphs that represent employees at other levels. It should also be noted that both approaches (with an *mst* of 50% for GASTON) were able to discover the normative patterns in less than 4 seconds.

6. ANOMALY DETECTION

In order to further assess the differences in frequency-based and compression-based approaches to graph mining, we will now look at a technique for anomaly detection that relies on the discovery of normative patterns in order to detect anomalies. While frequency-based approaches offer the potential of faster pattern discovery, there is the question of what effect differences in the discovered patterns have on the detected anomalies.

In order to test our ability to detect anomalies using a frequent subgraph miner, we chose to implement the GBAD algorithms within GASTON, because GBAD is not limited to certain graph types (e.g., bipartite graphs) or certain domains (e.g., intrusion detection), and has been well documented [5]. GBAD has already been implemented within the SUBDUE graph-based knowledge discovery approach [3][9][11].

There are three general *categories of anomalies*: insertions, modifications and deletions. Insertions constitute the presence of an unexpected vertex or edge in an instance of the normative pattern. Modifications would consist of an unexpected label on a vertex or edge in the normative pattern. Deletions would constitute the unexpected absence of a vertex or edge from the normative pattern. GBAD consists of three algorithms, where each algorithm is intended to discover one of the corresponding possible graph-based anomaly categories. The reader should refer to [5] for a more detailed description of GBAD.

For this work, we implemented all three GBAD algorithms into the GASTON framework, which we will call GBAD-FSM. None of the algorithms had to be modified to fit into this FSM, with the only significant difference being the generation of the normative pattern. In previous work, we implemented the GBAD algorithms into a compression-based framework, which we will now call GBAD-MDL. In GBAD-MDL, the normative pattern (that provides the greatest amount of compression) is discovered and saved for the GBAD algorithms to use, along with all of the instances of the subgraph. However, GASTON by default reports all subgraphs that meet its minimum support threshold, no matter what the size of the subgraph. So, we modified GBAD-FSM to output the *n* best substructures that maximize (frequency * size). In addition, we saved all instances of the best substructure, so as

to avoid overlapping of potential subgraphs as well as to know which vertices and edges were part of the normative subgraphs.

6.1 Synthetic Experiments

For the following experiments, the graphs are created with the following attributes:

- 100 to 32,000 transactions with the number of vertices and edges ranging from 5,000 to 424,000,
- 80 - 1500 unique labels,
- A minimum support threshold of 50% (for discovering the normative pattern),
- An anomalous threshold of 7% (i.e., up to 7% difference is acceptable as a potential anomaly).

In terms of what constitutes as an “anomaly” for these experiments, we will define anomalies in graph-based data as any substructure in a graph that is part of (or attached to or missing from) a *normative pattern*. Formally, the definition is as follows.

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 .

Using the same graphs as before, where t is the number of transactions in a graph, we randomly created anomalies of varying sizes, where some substructures are more anomalous than others.

On experiments involving *anomalous deletions*, as shown in Figure 12, the percentage of anomalies discovered is higher for the GBAD-FSM approach than the GBAD-MDL approach (where a value of 1.0 means all anomalies were discovered).

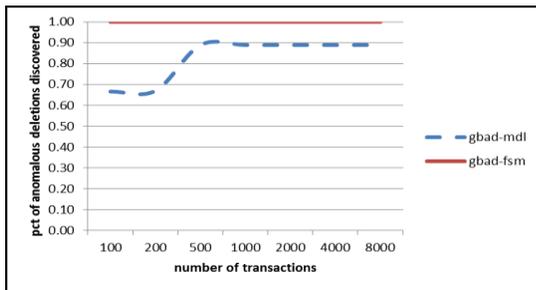


Figure 12. Percentage of anomalous deletions discovered.

In terms of false positives, neither approach reports any false positives (i.e., substructures that are more anomalous than the most anomalous substructure), except for a couple of cases when using the GBAD-FSM approach on a graph consisting of 16,000 transactions (~848,000 vertices and edges). In this example, some random “noise” in the graph (i.e., structure that is unexpected and at a low frequency) is picked up as equally anomalous to the randomly injected anomalies.

On experiments involving *anomalous insertions*, both approaches discover all of the injected anomalies, except for the GBAD-MDL run on graphs of 32,000 transactions (~1,696,000 vertices and edges). However, in terms of false positives, the GBAD-MDL approach reports many other substructures as equally anomalous when the graphs are smaller, as shown in Figure 13. In these cases, the noise that is prevalent in the smaller graphs is repeated

in the large graphs to the point that their frequency is no longer as anomalous as the targeted anomalies.

Note that the GBAD-FSM implementation does not report any false positives. The reason for this is being investigated, as the algorithms for determining what substructures are anomalous are identical between the two approaches, so clearly the compression approach is affecting GBAD-MDL’s accuracy.

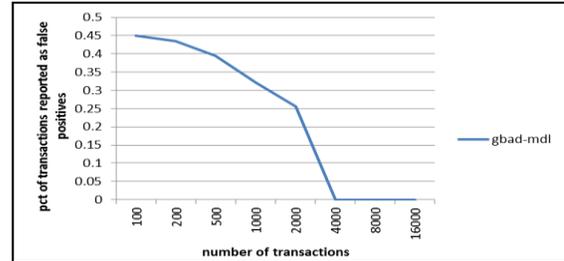


Figure 13. Percentage of false positives running GBAD-MDL.

On experiments involving *anomalous modifications*, the results are very similar between the GBAD-MDL and GBAD-FSM. Both approaches are able to discover anywhere from 33% to 100% of the targeted anomalies, and in all cases where the anomaly is not discovered, it is a case of an anomalous modification being made to a non-normative vertex or edge (due to the randomness of the graph generation). Since the definition of anomaly that is used by the GBAD approach assumes that an anomaly is a small modification to a normative pattern, this behavior is to be expected. False positives are also at a minimum, as the GBAD-FSM approach does not report any false positives, and the GBAD-MDL approach only reports a few false positives (less than 10% of the transactions) when the number of transactions is less than 2000. Again, structural modifications as a result of “noise” are more predominant when the graph is smaller.

Figure 14 shows the running times of the experiments on graphs with anomalous deletions. Clearly the GBAD-FSM result is faster when the graph reaches 1000 transactions or more, with almost an exponential growth in the running times for GBAD-MDL. Runs of GBAD-MDL on graphs of over 8,000 transactions (~424,000 vertices/edges) were aborted after 24 hours, and not expected to finish in a reasonable amount of time for these experiments (or most real-world scenarios). Similar behavior is observed for detecting anomalous modifications and insertions.

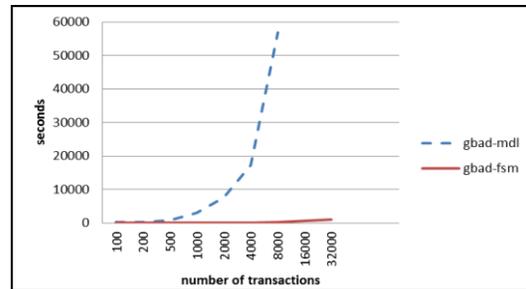


Figure 14. Running times for discovering anomalous deletions.

6.2 Accuracy

In the previous section, we arbitrarily chose an anomalous threshold of 0.07, targeting substructures whose differences from the normative pattern did not exceed more than 7% change. However, as with most anomaly detection approaches, one can

adjust thresholds to achieve different results. A decrease in the threshold may reduce the number of potential anomalies, while an increase in the threshold may not only result in more anomalies being discovered, but also may introduce more false positives. To analyze the effects of the anomalous threshold on both of these approaches, we will measure accuracy based on different thresholds of 0.05, 0.10, 0.15 and 0.20.

For these experiments, we generated 30 synthetic graphs with random anomalies (i.e., deletion, insertion and modification) for a total of 90 graphs, and evaluated both approaches using the different thresholds. Figure 15 shows the accuracy for both the GBAD-FSM and GBAD-MDL approaches on the discovery of anomalous deletions and anomalous modifications. While the GBAD-MDL approach is slightly more accurate in terms of detecting *anomalous deletions*, particularly at the higher thresholds, that is due to the aggressive pruning that takes places with the GBAD-FSM approach. As was documented earlier, the GBAD-FSM is significantly faster, much of which is due to the canonical tree structure that is known to be faster than the compression approach. However, because of the pruning, the accuracy is diminished slightly.

For the GBAD-FSM approach on the 30 graphs that contained *anomalous modifications*, the randomness of the anomalous modifications, plus an anomalous threshold above what is needed for small modifications, clearly affects the accuracy. Where the modification occurs can influence the amount of change in a substructure – in some cases, effectively cutting the substructure in half. In addition, the increase in the threshold allows for more false positives (and false negatives) to be reported. For the GBAD-MDL approach on these graphs, while there are fluctuations in terms of the number of false positives and false negatives, the accuracy is fairly consistent. Threshold 0.1 produces the most anomalies, while thresholds above and below this value result in more false results. This behavior is similar between the two approaches, with a slightly better accuracy using the GBAD-MDL approach.

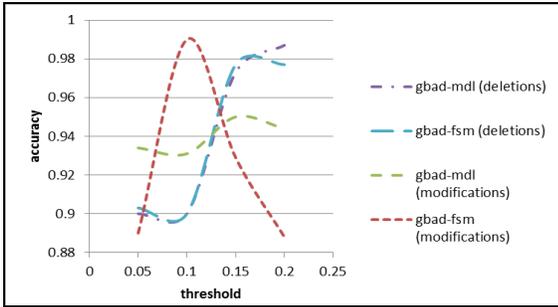


Figure 15. Accuracy for detecting deletions and modifications.

For *anomalous insertions*, the accuracy using the GBAD-FSM approach is 94.3. For the GBAD-MDL approach the accuracy is 98.3%. Again, due to an aggressive pruning strategy associated with maintaining the canonical tree for frequent subgraph mining, some anomalies are not discovered, pushing non-targeted anomalies to a false-positive state.

In summary, the accuracy of detecting anomalous deletions is fairly consistent across both approaches, while the accuracy of detecting anomalous insertions fluctuates, specifically with the GBAD-FSM approach, which appears more susceptible to the location of the inserted anomaly relative to the normative pattern.

6.3 Real-World Example

Using shipping data obtained from Customs Border and Protection (<http://www.cbp.gov/>), we are able to create a graph-based representation of cargo information where containers, ports, financial information, shippers, importers, etc., are represented as labeled vertices, and labeled edges convey their relationships. Figure 16 shows a portion of the graph that we used on our experiments. We injected different types of known (reported) anomalies into the data, allowing us to analyze the performance of these two approaches on real-world data at significant volumes.

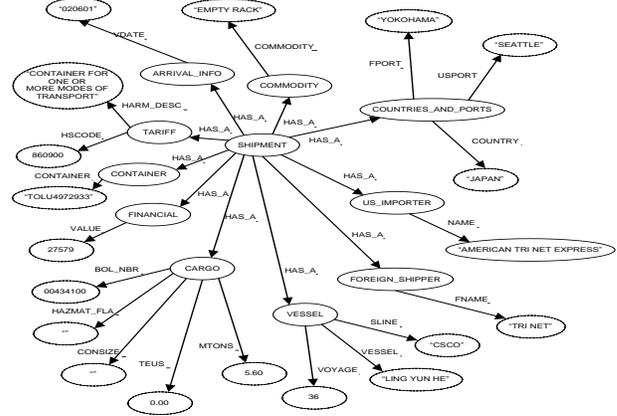


Figure 16. Example graph substructure of a cargo shipment.

As reported in [5], among 1000 shipments, where each shipment consists of ~60 vertices and edges, the GBAD-MDL approach is able to successfully discover the anomalous structure associated with the existence of an extra edge and vertex (in this case, the vessel traversed an extra port – an unexpected deviation from the normal pattern). In addition, the GBAD-MDL approach reports the substructure as anomalous due to the missing information – in this instance, the financial information that was removed. Similar to GBAD-MDL, GBAD-FSM discovers the anomalous instance. In addition, we experimented on cargo shipments with anomalous modifications, and both approaches report the most anomalous substructure as the one containing a different country of origin – one that deviated from the normal pattern. It should also be noted that the GBAD-FSM approach was significantly faster at discovering the anomalies. The GBAD-MDL approach took 22,523 seconds to discover all of the anomalies, whereas GBAD-FSM only took 15 seconds.

7. CONCLUSIONS AND FUTURE WORK

We have compared a frequency-based and compression-based approach to mining normative and anomalous patterns in graphs. While there are some advantages to the SUBDUE compression-based approach with its multiple features and best substructure instance generation, the running-times clearly favor a frequent subgraph miner like GASTON where an appropriate minimum support threshold is used. In terms of accuracy, both approaches were able (when given enough time) to discover the same normative patterns. In the future, experiments with larger graphs (> 1 million nodes) will need to be examined, as that is a known issue with many data mining approaches, as well as different sizes of normative patterns. Also, a more thorough analysis of the effects of noise in a graph (a known issue in real-world data such as TCP/IP networks), is warranted. We also plan on investigating the inclusion of efficiency mechanisms from GASTON into SUBDUE. In addition, in the process of executing these

experiments, we discovered several topological properties that affect the performance of these approaches. First, for the GBAD-FSM approach, the size of a transaction (i.e., the larger the frequent subgraph) affects the algorithm's running times. Also, in order to speed up the GBAD-FSM approach, overlapping substructures are not considered, which can be an issue with the number of reported anomalous substructures. And, both approaches are affected by the connectivity of the substructures, and the number of unique labels. Only connected substructures are considered for the normative patterns, and the greater the number of unique labels, the longer the algorithms take to execute. Finally, both of these approaches are based upon the GBAD definition of an anomaly [5]. There are other potential definitions of what constitutes an anomaly, and they should be examined in terms of their effect on performing anomaly detection.

8. ACKNOWLEDGMENTS

This material is based upon work supported by the DHS under Contract No. HSHQDC-10-C-00212. Any opinions, findings and conclusions expressed in this material are those of the author(s) and do not necessarily reflect the views of the DHS.

9. REFERENCES

- [1] Akoglu, L., Mcglohon, M. and Faloutsos, C. *OddBall: Spotting Anomalies in Weighted Graphs*, Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), June 23, 2010.
- [2] Chandola, V., Banerjee, A. and Kumar, V., *Anomaly Detection: A Survey*, Technical Report TR 07-017, University of Minnesota, August 15, 2007.
- [3] Cook, D. and Holder, L. *Graph-based data mining*. IEEE Intelligent Systems 15(2), 32-41, 2000.
- [4] Diesner, J. and Carley, K. *Exploration of Communication Networks from the Enron Email Corpus*, Computational and Mathematical Organization Theory, 11 (3), p. 201-228, 2005.
- [5] Eberle, W. and Holder, L. *Anomaly Detection in Data Represented as Graphs*. Intelligent Data Analysis, An International Journal, Volume 11(6), 2007.
- [6] Eberle, W. and Holder, L. *Analyzing Catalano/Vidro Social Structure Using GBAD*. VAST 2008 Challenge Track, VisWeek. October, 2008.
- [7] Eberle, W., Holder, L. and Graves, J. *Detecting Employee Leaks Using Badge and Network IP Traffic*. IEEE Symposium on Visual Analytics Science and Technology. October 2009.
- [8] Eberle, W. and Holder, L. *Mining for Insider Threats in Business Transactions and Processes*. Computational Intelligence in Data Mining, IEEE Symposium Series on Computational Intelligence. April 2009.
- [9] Holder, L., Cook, D. and Djoko, *Substructure Discovery in the SUBDUE System*, AAAI Workshop on Knowledge Discovery in Databases, 1994.
- [10] Huang, Z. *Link Prediction Based on Graph Topology: The Predictive Value of the Generalized Clustering Coefficient*, Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 20 August, 2006.
- [11] Ketkar, N., Holder, L. and Cook, D. *Qualitative Comparison of Graph-based and Logic-based Multi-Relational Data Mining: A Case Study*, *Proceedings of the ACM KDD Workshop on Multi-Relational Data Mining*, August 2005.
- [12] Noble, C. and Cook, D. *Graph-Based Anomaly Detection*. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 631-636, 2003.
- [13] 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, pp. 74-81, 2005.
- [14] Sun et al. *Relevance search and anomaly detection in bipartite graphs*. SIGKDD Explorations 7(2),p.48-55, 2005.
- [15] Wan, X., Milios, E., Janssen, J. and Kalyaniwalla, N. *Link-Based Event Detection in Email Communication Networks*, ACM Symposium on Applied Computing, 2009.
- [16] S. Nijssen and J. Kok, "A Quickstart in Frequent Structure Mining Can Make a Difference," *International Conference on Knowledge Discovery and Data Mining*, SIGKDD, pp. 647-652, 2004.
- [17] X. Yan and J. Han, "gSpan: Graph-Based Substructure Pattern Mining," *Proceedings of International Conference on Data Mining*, ICDM, pp. 51-58, 2002.
- [18] M. Kuramochi and G. Karypis, "An Efficient Algorithm for Discovering Frequent Subgraphs," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1038-1051, 2004.
- [19] J. Huan, W. Wang and J. Prins, "SPIN: Mining Maximal Frequent Subgraphs from Graph Databases," *Knowledge Discovery and Data Mining (KDD)*, 2004.
- [20] Z. Zeng, J. Wang, J., L. Zhou and G. Karypis, "Coherent closed quasi-clique discovery from large dense graph database," *Conference on Knowledge Discovery and Data Mining*, SIGKDD, 797-802, 2006.
- [21] E. Gudes, S. Shimony and N. Vanetik, "Discovering Frequent Graph Patterns Using Disjoint Paths," *IEEE Transactions on Knowledge and Data Engineering*, 18(11):1441-1456, 2006.
- [22] L. Thomas, S. Valluri and K. Karlapalem, "MARGIN: Maximal Frequent Subgraph Mining," *Sixth International Conference on Data Mining (ICDM)*, 109-1101, 2006.
- [23] J. Sun, P. You, S. Papadimitriou and C. Faloutsos, "GraphScope: Parameter-free Mining of Large Time-evolving Graphs," *KDD 2007*, August 12-15, 2007.
- [24] D. Cook and L. Holder, "Graph-based data mining," *IEEE Intelligent Systems* 15(2), 32-41, 2000.
- [25] H. Tong, S. Papadimitriou, P. Yu and C. Faloutsos "Proximity Tracking on Time-Evolving Bipartite Graphs," *SIAM Data Mining*, 2008, Atlanta, GA, April 24-26, 2008.
- [26] Lin, S. & Chalupsky, H. (2003). Unsupervised Link Discovery in Multi-relational Data via Rarity Analysis. *Proceedings of 3rd IEEE ICDM Intl. Conf. on Data Mining*. pp. 171-178.
- [27] Chakrabarti, D. (2004). AutoPart: Parameter-Free Graph Partitioning and Outlier Detection. *PKDD, 8th European Conference on Principles/Practices of KDD*. pp. 112-124.
- [28] Rattigan, M. & Jensen, D. (2005). The case for anomalous link discovery. *ACM SIGKDD Exploration News*. 7(2):41-47.