Distributed Graph-Based Clustering for Network Intrusion Detection

Corbin McNeill
Computer Science Department
Wheaton College, IL
Wheaton, Illinois 60187
corbin.mcneill@my.wheaton.edu

Enyue Lu
Department of Computer Science
Salisbury University
Salisbury, Maryland 21801
ealu@salisbury.edu

Matthias Gobbert
University of Maryland, Baltimore County
Department of Mathematics and Statistics
Baltimore, Maryland 21250
gobbert@umbc.edu

***Abstract*—In order to process large volumes of network traffic data and quickly detect intrusions, we use parallel computation frameworks for Network Intrusion Detection Systems (NIDS). Additionally, we model network data as graphs to highlight the interconnected nature of network traffic, and apply unsupervised graph-based clustering to flag anomalies as network intrusions. We particularly examine the effectiveness of barycentric clustering on graph network traffic models for intrusion detection. The clustering algorithms have been implemented in Hadoop MapReduce for the purpose of rapid intrusion detection. Furthermore, k-nearest neighbor graphs (kNNGs) are used to optimize the clustering process. We find that across various data sets, unsupervised graph based clustering is able to exceed a 92% intrusion detection accuracy and that kNNGs can effectively optimize the network traffic graphs for larger data sets.**

I. Introduction

Network Intrusion Detection Systems (NIDS) have been used in order to monitor network traffic. One type of intrusion detection is anomaly detection, which uses unsupervised methods to detect anomalous network intrusions.

Methods of graph-based anomaly detection have been explored and applied to network intrusion detection; however, no parallel computation approaches yet exist. We examine the effectiveness of graph-based clustering in Hadoop MapReduce for network intrusion detection.

II. Network Data

The KDD-99 cup's test data set was used for all conducted experiments. The KDD-99 data set simulates Local Area Netowrk (LAN) traffic [1], and contains over 310,000 network records, a majority of which are network intrusions [2].

In order to perform unsupervised anomaly detection on the test data set, intrusions must be statistical anomalies within the data set. Many Denial of Service (DOS) intrusions were remove by filtering the data set to only include records of network connections that made successful logins.

The filtered data set contained 53,645 records of which roughly 20% were network intrusions. All experiments used a random sampling of records from this filtered data set.

III. Graph Creation

In order to model network traffic, we construct a similarity graph. In this weighted undirected graph, vertices represent network connection records. Edge weights are outputs of a similarity functions representing the similarity (or lack of similarity) between the two records that edges connect. For the KDD-99 filtered data set, edge weights were calculated by the radial basis function measure:

$$w_{i,j} = \frac{1}{exp(\|\vec{v_i} - \vec{v_j}\|^2)}$$

where $\vec{v_x}$ is the data vector associated with the x^{th} network record represented as the x^{th} graph vertex [1]. This similarity measure is inspired by the Euclidean distance between points in space. All continuous network record features are scaled to the range $[0, 1]$ and symbolic features are assigned a distance of 0 if identical, otherwise 1. By this calculation, network records with high similarity will be connected by an edge with a higher weight.

This graph model can employ different edge densities ranging from a complete graph to a sparse graph.

A. Complete Graph

The complete graph approach creates $\frac{(|V|)(|V|-1)}{2}$ edges, and any graph clustering algorithm which accounts for all edges will therefore have a running time of at least $O(|V|^2)$, where $|V|$ is the number of network records. This makes such an approach inconvenient for processing large numbers of network records.

Complete graphs can be created by the following algorithm in MapReduce.

Algorithm 1 Full Graph Creation

```
1: procedure MAP($v_1$)                              ▷ match vertices
2:     vertices ← all vertices from distributed cache
3:     for $v_2$ in vertices do
4:         if $v_1 > v_2$ then
5:             emit: $(v_1, v_2)$
6: procedure REDUCE($v_1, list\{v_{adj}\}$)           ▷ calculate weight
7:     for $v_2$ in $v_{adj}$ do
8:         emit: $((v_1, v_2), weight(v_1, v_2))$
```

B. K Nearest Neighbor Graph

K Nearest Neighbor Graphs (kNNGs) can serve an approximation of complete graphs for graph clustering [3]. A kNNG is a graph where edges exist only between every vertex and its k nearest neighbors, where k is some constant. This allows the graph to store only edges which will have a significant effect during clustering. This approach will create graphs that contain no more than $k|V|$ edges.

KNNGs were formed from fully connected graphs using two additional MapReduce jobs. The first *map* function bins every edge by each vertex. The first *reduce* emits each vertex's k highest weighted edges. The second *map* function is the identity function. The second *reduce* function removes duplicate edges.

Table I shows the extent to which kNNGs reduce the number of edges for graph clustering algorithms to consider.

Graph Edges Remaining						
Iterations	0	1	2	3	4	5
Full Graph	497k	348k	322k	300k	282k	277k
kNNG	32k	20k	13k	9k	5k	3k

TABLE I
THE NUMBER OF EDGES REMAINING AFTER VARIOUS BARYCENTRIC CLUSTERING ITERATIONS. $|V| = 1000$ AND $k = 50$.

IV. CLUSTERING

After similarity graphs were formed, barycentric clustering was used to partition network records into various clusters. Barycentric clustering is an algorithm for clustering undirected graphs motivated by the physical concept of vertices, connected by springs, iteratively affecting each others positions. Barycentric clustering identifies edges with abnormally low edge weights and removes them [4]. After enough iterations, multiple connected components will separate. These remaining connected components partition the network records into closely related clusters. Barycentric clustering can be implemented in Hadoop MapReduce to distribute storage and computation [5].

V. INTRUSION CLASSIFICATION

Once clusters have been formed, classification algorithms are used to determine which clusters contain intrusive network records and which clusters contain normal network records.

An optimal classification strategy appeared to be classifying every vertex record within the largest connected component as *non-intrusive* and classifying all other vertex records as *intrusive*.

VI. DETECTION ACCURACY

The effectiveness of graph-based clustering and intrusion classification can be evaluated via detection accuracy. Detection accuracy is calculated as:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP is the number of true positives, FN is the number false negatives, etc. In general terms, accuracy is the percent of properly classified records.

Fig. 1 shows the detection accuracy of different data sets during graph clustering. Table II highlights the detection accuracy of kNNGs after 2 iterations and full graphs after 5 iterations. The selection of these numbers of iterations, respectively yielded the highest detection accuracies.

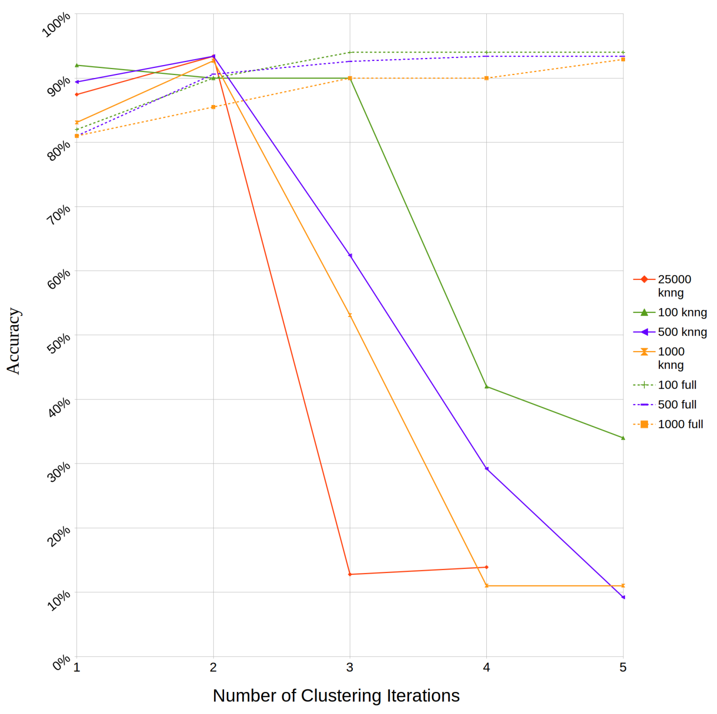

Fig. 1. This graph shows the detection accuracy after various iterations of barycentric clustering for both full graphs and kNNGs with $k = 50$.

Detection Accuracy				
Data Set Size	100	500	1,000	25,000
Full Graph @ 5th iteration	94%	93.4%	92.9%	N/A
kNNG @ 2nd iteration	90%	93.4%	92.7%	93.4%

TABLE II
THE DETECTION ACCURACY FOR VARIOUS DATA SETS. $k = 50$.

VII. CONCLUSION

We find that graph-based clustering algorithms serve as an effective method for unsupervised network intrusion detection with detection accuracy consistently above 92%. Additionally graph creation and graph clustering can be distributed via Hadoop MapReduce and optimized with kNNGs in order to quickly detect network intrusions. KNNG optimizations appeared effective for larger data sets.

ACKNOWLEDGMENT

This work is funded by the National Science Foundation under the Research Experience for Undergraduates Program.

REFERENCES

[1] H. Alene, "Graph based clustering for anomaly detection in ip networks," 2011.
[2] S. D. Bay, D. F. Kibler, M. J. Pazzani, and P. Smyth, "The uci kdd archive of large data sets for data mining research and experimentation," 2000.
[3] C. Rosenberger and L. Brun, "Similarity-based matching for face authentication."
[4] J. Cohen, "Barycentric graph clustering," 2008.
[5] ——, "Graph twiddling in a mapreduce world," 2009.