Analysis of Synthetic Graph Generation Methods for Directed Network Graphs

Spencer Callicott
Distributed Analytics and Security Institute, Mississippi State University
Mississippi State, Mississippi
spencer@dasi.msstate.edu

ABSTRACT

Historically, scientific experiments have been conducted to generate scale-free network graphs based on structure. Metrics used to measure veracity ensure the integrity of a scale-free algorithm given a seed. However, studies do not explore the performance benefits or drawbacks of specific algorithms running on Apache Spark and GraphX. Recognizing the lack of performance benchmarks demands ensuring accuracy through experimenting. This study will utilize the Stochastic Kronecker Graph model to synthetically generate graphs given a seed graph.

CCS CONCEPTS

• **Theory of computation** → **Graph algorithms analysis**; • **Computing methodologies** → *Modeling and simulation*;

KEYWORDS

Stochastic Kronecker, Apache Spark, Cyber-Security, Benchmark

1 INTRODUCTION

The purpose of this study is to analyze the performance and accuracy of an implementation of Stochastic Kronecker Expansion in Apache Spark for use with Directed Network Graphs. We propose that due to the nature of the algorithm, the synthetically generated graphs will maintain the properties of their seed graphs by mirroring the structure and distribution of nodes and edges. Also, the algorithm will scale well across a cluster, allowing for faster performance given more computational power. Seed data will be gathered from the publicly available Swedish Defense Research Network Capture Dataset [1].

2 METHODS

Stochastic Kronecker allows for rapid expansion of a small matrix-seed to a large synthetic graph [3]. Accuracy is achieved from the probability values of the seed matrix that determine when and how to add additional edges to a graph. Both veracity and performance measurements were taken of the experiments to ensure that the synthetic graph mirrors the structure of the seed and that the algorithm correctly scales across a cluster of computing nodes. The veracity metrics tested were a degree distribution comparison as well as the euclidean distance measurement of degree distribution and the overall PageRank veracity.

3 RESULTS

Test results were gathered for ranges of 10 to 60 compute nodes in increments of 10 while generating a graph with up to 10^{10} total edges.

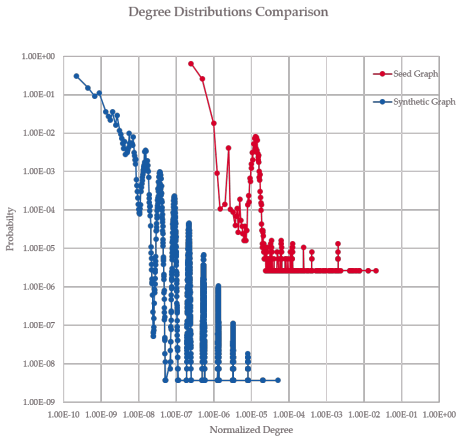

Figure 1: Probability of Degree normalized by graph size.

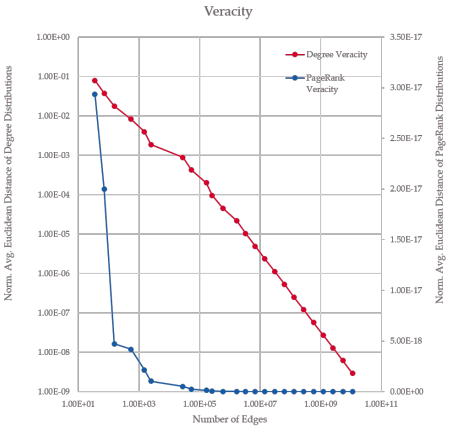

Figure 2: Euclidean Distance of Degree Distribution and PageRank calculations.

3.1 Veracity

Synthetic Kronecker creates graphs of magnitudes larger than the original seed graph retained the properties of the seed graph accurately as shown by the probability of each graph's normalized degree (Fig. 1).

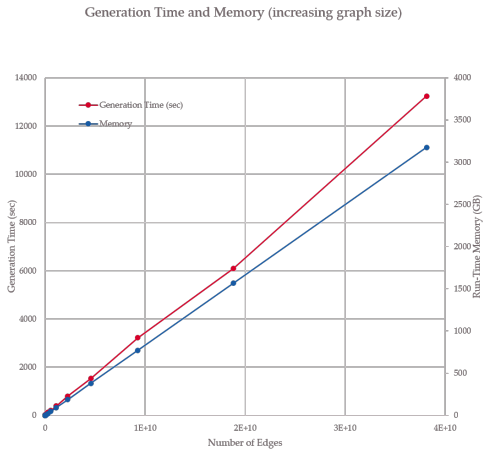

Figure 3: Scalability of Graph Size vs. Time and Memory.

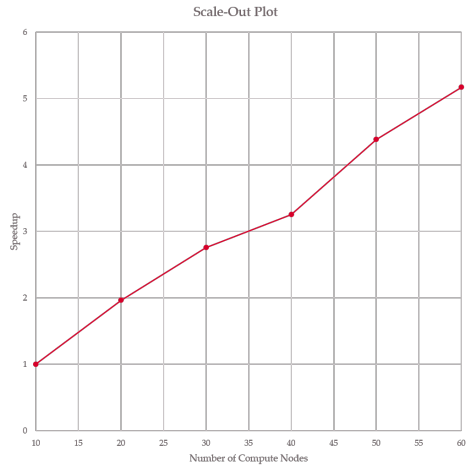

Figure 4: Node scalability of constant graph size.

Also, both the normalized Euclidian Distance and the PageRank distance between the graphs decreased as graph size increased as shown by (Fig. 2). The inaccuracy at lower graph sizes is a result of the Kronecker algorithm not having enough space to correctly match the structure of the seed graph.

3.2 Performance

Generation time scaled linearly with the number of edges generated. Total memory expended for the graph also scaled linearly. Both of these results show that the implementation was able to achieve O(E) time complexity, where E is the number of edges as shown by (Fig. 3).

Also, Kronecker scaled well across the cluster since the speedup factor increased when adding new nodes to the compute cluster as shown by (Fig. 4).

4 CONCLUSION

Overall, the Spark Stochastic Kronecker graph generation algorithm scales well across a cluster and is able to produce large synthetic graphs that are highly accurate. Potential for further application exists in analyzing the performance characteristics of the algorithm closer to potentially identify system bottlenecks on specific architectures. Other scale-free graph algorithms could also be tested against this implementation to find the fastest directed graph generation algorithm. Seed graphs containing network data with known-malicious activity could also be expanded using Kronecker in order to train and test malicious activity classification systems. The code used for these experiments is available as freeware under the GNU GPL-3.0 license [2].

REFERENCES

[1] Swedish Defence Research Agency. 2011. *Network Traffic Dataset.* ftp://download.iwlab.foi.se/dataset/smia2011/Network_traffic/.
[2] Cordio S. Callicott S. Lewis J. Rui J. Iannucci, S. 2011. *Cyber Security Benchmark.* https://github.com/msstate-dasi/csb/.
[3] Chakrabarti D. Kleinberg J. Faloutsos C. Leskovec, J. and Z. Ghahramani. 2010. Kronecker graphs: An approach to modeling networks. *The Journal of Machine Learning Research* 11 (2010), 985–1042.