Automated Design of Random Dynamic Graph Models for Enterprise Computer Network Applications

Aaron Scott Pope
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri
aaron.pope@mst.edu

Daniel R. Tauritz
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri
dtauritz@acm.org

Chris Rawlings
Los Alamos National Laboratory
Los Alamos, New Mexico
crawlings@lanl.gov

ABSTRACT

Dynamic graphs are an essential tool for representing a wide variety of concepts that change over time. In the case of static graph representations, random graph models are often useful for analyzing and predicting the characteristics of a given network. Even though random dynamic graph models are a trending research topic, the field is still relatively unexplored. The selection of available models is limited and manually developing a model for a new application can be difficult and time-consuming. This work leverages hyper-heuristic techniques to automate the design of novel random dynamic graph models. A genetic programming approach is used to evolve custom heuristics that emulate the behavior of real-world dynamic networks.

CCS CONCEPTS

• **Mathematics of computing → Random graphs**; • **Theory of computation → Dynamic graph algorithms**; • **Software and its engineering → Genetic programming**.

KEYWORDS

Random graph models, dynamic graphs, genetic programming

ACM Reference Format:
Aaron Scott Pope, Daniel R. Tauritz, and Chris Rawlings. 2019. Automated Design of Random Dynamic Graph Models for Enterprise Computer Network Applications. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3322049

1 INTRODUCTION

Random graph models are an invaluable tool for studying and anticipating the development of networks in a wide variety of applications [3]. More specifically, random dynamic graph models capture the behavior of a network that changes over time as vertices and edges are added and removed. Random graph modeling for dynamic applications is an active research area, but the field is still relatively young. The proper selection of a random graph model is critical for accuracy of the representation, but the options for dynamic versions of these models are currently limited.

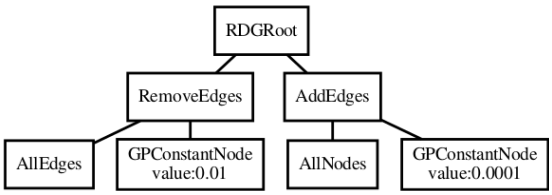


Figure 1: Example random graph heuristic parse tree.

Previous work has demonstrated the potential of using generative hyper-heuristic searches to automate the design of random graph models [1, 4] However, these approaches have only been applied to static graph models. This work investigates the use of genetic programming (GP) to evolve novel graph update heuristics that accurately mimic the behavior of a target dynamic network. Results are presented from two applications that involve modeling real-world computer network activity.

2 METHODOLOGY

A population of graph update algorithms is evolved to mimic the behavior of a set of dynamic input graphs. Solutions are represented using strongly typed Koza-style parse trees. See Figure 1 for an example parse tree representation of a basic graph update heuristic.

Solutions are evaluated by executing them on a set of input graphs and comparing the output to the target graph for the next time step. Four objective values measure the similarity of the evolved and target behaviors. Degree centrality (**DC**) measures the similarity in the distribution of vertex degree values. The edge addition (**EA**) and edge removal (**ER**) objectives measure the similarity in the number of edges added or removed, respectively, at each time step. Finally, the size difference (**SD**) objective compares the overall change in the number of edges at each time step. The **DC**, **EA**, and **ER** objectives are compared using the p-value from Kolmogorov-Smirnov tests, which are maximized when the two samples have similar distributions. **SD** is the absolute percent error in the number of edges averaged over each time step and subtracted from one to convert it to a maximization objective.

To mitigate overfitting, objective values are scaled to the score achieved by comparing multiple instances of the target graph against each other. The formula for this scaling is $\mathbf{O} = 1 - \frac{|\mathrm{O_T} - \mathrm{O_E}|}{\mathrm{O_T}}$ where $\mathbf{O}$ is an objective in $\{\mathbf{DC}, \mathbf{EA}, \mathbf{ER}, \mathbf{SD}\}$, $\mathrm{O_T}$ is the value for that objective achieved by the target dynamic graph evaluated against itself,

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3322049

and $\mathbf{O_E}$ is the value for that objective achieved by the evolved model. The absolute difference penalizes solutions that have inflated objective values as a result of being overfitted on the test cases they were evaluated against. Final objective values are averaged over a configurable number of test cases to measure the robustness of the evolved graph model.

The primitive set used was initially inspired by previous work evolving static random graph models [1, 4]. Terminal operations include ephemeral constants (e.g., numbers, booleans), graph metrics (e.g., average degree), and graph elements (i.e., nodes and edges). Function primitives include mathematical and logical operations (e.g., multiply, and) control flow operations (e.g., conditional branching, operation sequences), collection manipulation (e.g., node list concatenation), and graph operations (e.g., add/remove edges).

3 EXPERIMENT

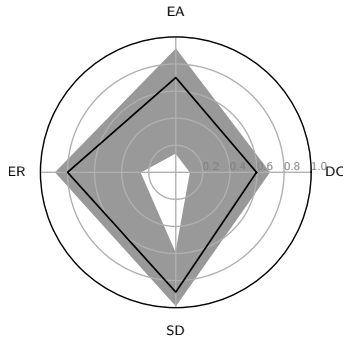
This work was applied to modeling the dynamic network behavior of two applications taken from data collected on the computer network at Los Alamos National Laboratory (LANL) [5] The first application involves communication between devices on the network in the form of NetFlow sessions. The second represents authentication events that occur when an account is used to access a computer via another, such as a remote desktop session. A static graph is generated for six minute increments during normal business hours (7am to 5pm) that contains an edge connecting two computer vertices if traffic or authentication is observed between those computers during that time window. To keep the evaluation time manageable for this proof-of-concept, the resulting graphs are reduced to activity between the most active 1000 computers. These static graphs are combined to produce a dynamic graph with 100 time steps for each of the 50 highest activity days. The down-selection in terms of days is done to remove non-business days, such as weekends and holidays, and provide a more consistent target for the evolutionary process to model. During solution evaluation, a subset of these days is chosen randomly without replacement to generate test cases. The evolution in this work used the Non-dominated Sorting Genetic Algorithm (NSGA-II) [2] with a population size of 50, a 50% of choosing recombination or mutation, 30 test cases per evaluation, and terminated after 10000 evaluations.

4 RESULTS

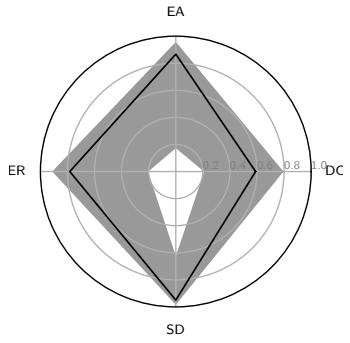
Figure 2 shows the range of objective values achieved by the final population of solutions for both applications. The shaded region indicates the range of values (minimum and maximum) and the black line shows the median. Since these values are scaled to the objective values achieved by evaluating the target against itself, objective values closer to one indicate more accurate modeling. For both of these applications, the **SD** objective appears to be the easiest to optimize. The **DC** objective seems to be the hardest; this could be the result of lacking operations that are specifically aware of the graph's degree distribution.

5 CONCLUSION

Random graph models are an invaluable tool for a variety of applications. When modeling a dynamic concept with a random graph, the appropriate model must be selected for an accurate representation.



(a) NetFlow



(b) Authentication

Figure 2: Range of objective values achieved by the final population of solutions from an example run for each application. The shaded region indicates the range between the minimum and maximum values and the black line shows the median.

However, the quality and variety of appropriate dynamic models is limited. Accurate models for new applications can be manually developed, but this process can be difficult and time-consuming. This work investigated the potential of hyper-heuristics for automating the design of generative models for random dynamic graphs. Results on enterprise network applications demonstrate that the approach has potential for accurately modeling real-world phenomenon.

REFERENCES

[1] Alexander Bailey, Mario Ventresca, and Beatrice Ombuki-Berman. 2014. Genetic Programming for the Automatic Inference of Graph Models for Complex Networks. *IEEE Transactions on Evolutionary Computation* 18, 3 (2014), 405–419.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197. https://doi.org/10.1109/4235.996017

[3] Petter Holme and Jari Saramäki. 2012. Temporal networks. *Physics Reports* 519, 3 (2012), 97–125. https://doi.org/10.1016/j.physrep.2012.03.001

[4] Aaron S. Pope, Daniel R. Tauritz, and Alexander D. Kent. 2016. Evolving Random Graph Generators: A Case for Increased Algorithmic Primitive Granularity. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1–8. https://doi.org/10.1109/SSCI.2016.7849929

[5] Melissa J. M. Turcotte, Alexander D. Kent, and Curtis Hash. 2018. *Unified Host and Network Data Set.* World Scientific, Chapter Chapter 1, 1–22. https://doi.org/10.1142/9781786345646_001 arXiv:https://www.worldscientific.com/doi/pdf/10.1142/9781786345646_001