1 Introduction

Teasing out interesting relationships buried within volumes of data is one of the most fundamental challenges in data science research. Increasingly, researchers and practitioners are interested in understanding how individual pieces of information are organized and interact in order to discover the fundamental principles that underlie complex physical or social phenomena. Indeed the most pivotal moments in the development of a scientific field are centered on discoveries about the structure of some phenomena [1]. For example, chemists have found that many chemical interactions are the result of the underlying structural properties of interactions between elements [2, 3]. Biologists have agreed that tree structures are useful when organizing the evolutionary history of life [4, 5], sociologists find that triadic closure underlies community development [6, 7], and neuroscientists have found small world dynamics within neurons in the brain [8, 9].

Graphs offer a natural formalism to capture this idea, with nodes (or vertices) representing the individual entities and edges (or links) describing the relationships in the system. Arguably, the most prescient task in the study of such systems is the identification, extraction, and representation of the small substructures that, in aggregate, describe the underlying phenomenon encoded by the graph. Furthermore, this opens the door for researchers to design tools for tasks like anomaly/fraud detection [10, 11], and anonymizing social network data [12]. Researchers, over the years, have opted for a two-pronged approach to tackle these challenges—subgraph mining and graph generators.

1.1 Subgraph Mining

Rooted in data mining and knowledge discovery, subgraph mining methods are efficient and scalable algorithms for traditional frequent itemset mining on graphs [13, 14]. Frequent graph patterns are subgraphs that are found from a single large graph or a collection of many smaller graphs. A subgraph is deemed to be frequent if it appears more than some user-specified support threshold. Being descriptive models, frequent subgraphs are useful in characterizing graphs and can be used for clustering, classification, or other discriminative tasks.

Unfortunately, these methods suffer from a so-called combinatorial explosion problem wherein the search space grows exponentially with the pattern size [15]. This causes computational headaches, and can also return a massive result set that hinders real-world applicability. Recent work that heuristically mines graphs for prominent or representative subgraphs have been developed in response, but are still limited by their choice of heuristic [16–19]. Alternatively, researchers characterize a network by counting small subgraphs called graphlets and therefore forfeit any chance of finding larger, more interesting structures [20–22]. Overcoming these limitations will require a principled approach that discovers the structures within graphs and is the first research objective of the proposed work.

1.2 Graph Generators

Graph generator models, like frequent subgraph mining, also find distinguishing characteristics of networks, but go one step further by generating new graphs that resemble the original
They do so by mimicking local graph properties like the counts of frequent subgraphs, as well as some global graph properties like the degree distribution, clustering coefficient, community structure, and assortativity.

**Parametric Graph Models.** Early graph generators, like the random graph of Erdős and Rényi [23], the small world network of Watts and Strogatz [24], the scale-free graph of Albert and Barabási [25] and its variants [26–28], or the more recent LFR benchmark graph generators [29] did not learn a model from a graph directly. Instead, they had parameters that could be tuned to generate graphs with certain desirable properties like scale-free degree distributions and community structure.

More recently, the Block Two-level Erdős-Rényi (BTER) model models real-world graphs as a scale-free collection of dense Erdős-Rényi graphs [30]. Graph generation involves two phases—in the first phase, nodes are divided into blocks based on their degrees in the input graph, and then each block is modeled as a dense Erdős-Rényi graph. Then in phase two, edges are established across blocks to ensure that degree sequences match up with that of the original. Therefore, it achieves two important goals—local clustering and respecting the degree sequence of the original graph. BTER, however, fails to capture higher-order structures and degenerates in graphs with homogenous degree sequences (like grids).

In general, this exercise of fine-tuning the model parameters to generate graphs that are topologically faithful to an input graph is taxing and often hard to achieve. Thus, researchers sought to develop smarter graph models which would focus on actually learning the topological properties of any input graph while minimizing the need for manual tinkering of parameters.

**The Configuration Model.** The first of the new generation of graph models was the Chung-Lu model, also known as the configuration model [31], which did not require any input parameters from the user. It creates a new graph by randomly rewiring edges based on the degree sequence of the original graph. Even though the degree sequence of the generated graph exactly matches that of the original, it often fails to incorporate higher-order structures like triangles, cliques, cores, and communities observed in the original graph. Researchers have attempted to fix these flaws by proposing improvements like incorporating assortativity and clustering [30, 32–34], but despite this, these modifications failed to result in significant improvements. Thus, incorporating higher-order topological in the learning process of a model is critical. To this effect, researchers from various domains started proposing models to do just that.

**Stochastic Block Models.** Stochastic Block Models (SBMs) first find blocks or communities of nodes in the graph, usually by an MCMC search, and then forming a block matrix that encodes the block-to-block connectivity patterns [35, 36]. For generating a graph from a block matrix, we need to provide either the counts of nodes in each block or the relative size of each block and the total number of nodes. Then, the generator creates an Erdős-Rényi graph inside each block and random bipartite graphs across communities. They have been extended to handle edge-weighted [37], bipartite [38], temporal [39], and hierarchical networks [40]. The block connectivity matrix is indicative of the connection patterns observed in the actual network, whether it is assortative, disassortative, hierarchical, or core-periphery.

**Exponential Random Graphs.** Sociologists have developed a paradigm called Exponential Random Graph Models (ERGMs), also known as the $p^*$ model [41]. ERGMs first learn a distribution of graphs having a prescribed set of patterns like edges, non-edges, and transitivity, for example, from the input graph, and then sample a new graph from that space. So, ERGMs learn a model based on the specific patterns found in the input graph. Thus the standard ERGM is also not only good at generating new graphs, but the learned model can also be informative about the nature of the underlying graph, albeit through the lens of only a handful of small
structures, e.g., edges, triangles, 4-cliques [42]. It does, however, suffer from issues that limit its applicability—all possible patterns must be pre-defined, model fitting oftentimes degenerates, and the time complexity is exponential in the number and the size of the patterns.

**Kronecker Graphs.** Kronecker models aim to learn a $k \times k$ initiator matrix $I$ (usually $k = 2$) from the input graph, and model the adjacency matrix of the generated graph as repeated Kronecker products $I \otimes \cdots \otimes I$ [43, 44]. Kronecker graphs have desirable topological properties like scale-free degree distributions and small diameter. The initiator matrix, like the block matrix for SBMs, is indicative of the overall connectivity patterns of the graph [45]. The matrix can be learned either by doing a stochastic gradient descent [44] or by solving a system of linear equations [46]. While the generation process is fast and can be parallelized [47], the learning process suffers from computational bottlenecks, preventing them from scaling to large networks.

**Graph Neural Networks.** Recent advances in graph neural networks have produced graph generators based on recurrent neural networks [48], variational auto-encoders [49–52], transformers [53], and generative adversarial networks [54] each of which have their advantages and disadvantages. Graph auto-encoders (GraphAE, GraphVAE) first learn node embeddings of the input graph by message passing and then construct a new graph by doing an inner-product of the latent space and passing it through an activation function like the sigmoid function. NetGAN trains a Generative Adversarial Network (GAN) to generate and distinguish between real and synthetic random walks over the input graph and then builds a new graph from a set of random walks produced by the generator after training is completed. GraphRNN decomposes the process of graph generation into two separate RNNs - one for generating a sequence of nodes, and the other for the sequences of edges. However, due to the sequential nature of computation and the dependence over node orderings, they fail to scale beyond small graphs [55]. We have found that most, if not all, of these methods are sensitive to the initial train-test-validation split and require a large number of training examples. Furthermore, they do not provide a compact and interpretable model for graphs because the models learn millions of parameters.

**Graph Grammars.** Research into graph grammars started in the 1960s as a natural generalization of string grammars, starting with tree replacement grammars and tree pushdown automata [56, 57]. Over the years, researchers have proposed new formalisms and extensions that expanded this idea to handle arbitrary graphs. Broadly speaking, there are two classes of graph grammars—node (see Figure 1(E)) and edge or hyperedge replacement grammars (see Figure 1(C))—depending on their re-writing strategies. Nevertheless, until recently, the production rules were required to be hand-crafted by domain experts, which limited their applicability. The ability to learn the rewriting rules from a graph without any prior knowledge or assumptions lets us create simple, flexible, faithful, and interpretable generative models [58–60]. The research described in this proposal continues the recent line of work to fill this void. Furthermore, it allows us to ask incisive questions about the extraction, inference, and analysis of network patterns in a mathematically elegant and principled way.

Table 1 classifies the various graph models discussed above into different classes depending on their interpretability, scalability, and generation quality. We observe that SBM and CNRG appear to be balanced in terms of scalability and interpretability. On the surface, Hierarchical SBM and CNRG share the same philosophies—they both look to uncover and leverage the multi-level structures observed in real-world networks, as well as utilizing the Minimum Description Length(MDL) principle to learn compact yet faithful models. CNRG appears to be more flexible than nested SBMs, especially when it comes to generating topologically faithful graphs. This is because the CNRG grammar rules stores local boundary information (via boundary degrees), which lets it preserve local connectivity patterns. This feature is absent in hierarchical SBMs, and
Figure 1: (A) An example graph $H$ with 9 nodes and 16 edges. (B) One possible tree decomposition $T$ of $H$, with $\eta$'s denoting the bags of nodes. (C) Hyperedge Replacement Grammar (HRG) production rules derived for $H$ using $T$. The LHS represents hyperedges (drawn as open diamonds), and the RHS represents hypergraphs. External nodes are drawn as filled circles. (D) One possible dendrogram $D$ of $H$ obtained from a hierarchical clustering algorithm. $\eta$'s represent the internal nodes of $D$. (E) Clustering-based Node Replacement Grammar (CNRG) rules for $H$ learned from $D$. The LHS is a single non-terminal node drawn as a square labeled with size $\omega$ (drawn inside the node). The RHS is a subgraph with non-terminal nodes drawn as squares and labeled (illustrated inside the node), terminal nodes labeled with the number of boundary edges (drawn above the node). $S$ denotes the starting hyperedge/non-terminal.

The rewiring process during graph generation is entirely random.

1.3 Thesis Statement

The research described in this proposal makes significant progress in filling the void in current research to find scalable, and interpretable graph modeling methods by leveraging the new-found link between formal language theory, graph theory, and data mining. Furthermore, the proposed work will bridge the gap between subgraph mining and graph generation to create a new suite of models and tools that can not only create informative models of real-world data, but also generate, extrapolate, and infer new graphs in a precise, principled way. To support this goal, the following objectives will be accomplished:

**Objective 1:** Precise and complete structure discovery, including extraction, sampling, and robust evaluation protocols will be developed and vetted.

**Objective 2:** Principled graph generation will be demonstrated and studied using the discovered structures on static and evolving data.

**Objective 3:** An analysis of the discovered structures and their relationships to real-world
Proposition Overview

In the following sections, we provide an overview of the work done so far, highlighting the progress we made to support the claims made in the previous sections.

In Subsection 2.1, we will describe the current work in node replacement grammars, mainly focusing on extraction and generation, partially fulfilling objectives 1 and 2.

Future work is described in Section 3, as we strive to make the grammar induction process more stable and flexible. In addition to that, we will propose a new stress test called the Infinity Mirror Test for graph models, which would help uncover hidden biases, and test their robustness at the same time. This will help us fulfill all the objectives mentioned earlier.

Finally, we will conclude the proposal in Section 4 and provide a timeline for the completion of the future works so proposed.

2 Previous Work

Work unrelated to the thesis. We have previously authored two papers that introduced new community detection algorithms for complex networks [61, 62]. In the first paper [62], we used a geometric $t$-spanner [63] to identify bridge edges—edges that participate in a disproportionate amount of shortest paths in the graph—which essentially span across clusters. Removing these edges reveal the underlying community structure, with different communities often appearing as individual components.

In the second paper [61], we proposed a mixture of breadth-first and depth-first traversals across a graph to identify densely interconnected regions (clusters) [61], similar to how the OPTICS algorithm works for spatial data [64]. During the traversals, we utilize some clever data structures to grow clusters around seed nodes. Additionally, we have contributed chapters on Spectral Community Detection, and the NetworkX graph library to two NSF compendium reports edited by Dr. Peter Kogge [65, 66].

2.1 Synchronous Hyperedge Replacement Graph Grammars

The work presented in this section was performed in collaboration with Corey Pennycuff, Catalina Vajiac, David Chiang, and Tim Weninger and was published as a full paper in 2018 in the International Conference on Graph Transformations, 2018 [67].

We present a method to extract synchronous grammar rules from a temporal graph. We find that the synchronous probabilistic hyperedge replacement grammar (PSHRG), with RHSs containing synchronized source- and target-PHRGs, can clearly and succinctly represent the graph dynamics found in the graph process. We also find that the source-PHRG grammar extracted from the graph can be used to parse a graph. The parse creates a rule-firing ordering that, when applied to the target-PHRG, can generate graphs that are predictive of the future growth of the graph.
The PSHRG model is currently limited in its applicability due to the computational complexity of the graph parser. We are confident that future work in graph parsers will enable PSHRGs to model much larger temporal graphs. The limitation of graph parsing, however, does not affect the ability of PSHRGs to extract and encode the dynamic properties of the graph. As a result, we expect that PSHRGs may be used to discover previously unknown graph dynamics from sizeable real-world networks. But we do not consider this opportunity in the proposed work.

2.2 Towards Interpretable Graph Modeling with Vertex Replacement Grammars

The work presented in this section was performed in collaboration with Justus Hibshman and Tim Weninger and was published as a full paper in 2019 in the IEEE International Conference on Big Data, 2019 [60].

In this work we developed an model called BUGGE: the Bottom-Up Graph Grammar Extractor, which extracts grammar rules that represent interpretable substructures from large graph data sets. It uses the Minimum Description Length (MDL) heuristic to find interesting structural patterns in directed graphs. Using synthetic data sets we explored the expressivity of these grammars and showed that they clearly articulated the specific dynamics that generated the synthetic data. On real-world data sets, we further explored the more frequent and most interesting (from an information-theoretic point of view) rules and found that they clearly represent meaningful substructures that may be useful to domain experts. This level of expressivity and interpretability is needed in many fields with large and complex graph data.

2.3 Modeling Graphs with Vertex Replacement Grammars

The work presented in this section was performed in collaboration with Justus Hibshman and Tim Weninger and was published as a full paper in 2019 in the IEEE International Conference on Data Mining, 2019 [59].

Recent work at the intersection of formal language theory and graph theory has explored the use of graph grammars for graph modeling. Existing graph grammar formalisms, like Hyper-edge Replacement Grammars, can only operate on small tree-like graphs.

With this goal in mind, the present work describes Clustering-based Node Replacement Grammars (CNRG), which is a variant of vertex replacement grammar, which contains graphical rewriting rules that can match and replace graph fragments similar to how a context-free grammar (CFG) rewrites characters in a string. These graph fragments represent a concise description of the network building blocks and the instructions about how the graph is pieced together.

**Clustering-based Node Replacement Grammars (CNRGs).** A CNRG is a 4-tuple $G = (\Sigma, \Delta, P, S)$ where $\Sigma$ is the alphabet of node labels; $\Delta \subseteq \Sigma$ is the alphabet of terminal node labels; $P$ is a finite set of productions rules of the form $X \rightarrow (R, f)$, where $X$ is the LHS consisting of a nonterminal node (i.e., $X \in \Sigma \setminus \Delta$) with a size $\omega$, and the tuple $(R, f)$ represent the RHS, where $R$ is a labeled multigraph with terminal and possibly nonterminal nodes, and $f \in \mathbb{Z}^+$ is the frequency of the rule, i.e., the number of times the rule appears in the grammar, and $S$ is the starting graph which is a non-terminal of size 0. This formulation is similar to node label controlled (NLC) grammar [68], except that the CNRG used in the present work does not keep track of specific rewiring conditions. Instead, every internal node in $R$ is labeled by the number of boundary edges to which it was adjacent to the original graph. The sum of the boundary degrees is, therefore, equivalent to $\omega$, which is also equivalent to the label of the LHS.
Extracting a CNRG

First, we compute a dendrogram $D$ of $H$ using a hierarchical clustering algorithm such as the Leiden method [69], or hierarchical spectral $k$-means [70].

The CNRG extraction algorithm on a graph $H$ involves executing the following three steps in order until the graph $H$ is empty. An example of this process is shown in Fig. 2.

1. **Subtree selection** - we compute a dendrogram $D$ from a multigraph $H$. From $D$ we select internal tree nodes $\eta_i$ with at most $\mu$ leaf nodes, each corresponding to a set of leaf nodes $V_{\eta_i}$ and CNRG rule $r_{\eta_i}$. Of those $\eta_i$'s, we use the MDL heuristic to pick an $\eta^*$ which minimizes the description length of both the rule $r_{\eta^*}$ and the updated graph $H'$ obtained by compressing $H$ with the said rule. $V_{\eta^*}$ is compressed into a non-terminal node $X$ with size $\omega$ equal to the number of boundary edges spanning between $V_{\eta^*}$ and $V \setminus V_{\eta^*}$.

2. **Rule construction** - $X$ becomes the LHS of the rule. The RHS graph $R$ is the subgraph induced by $V_{\eta}$ on $H$. Additionally, we store the boundary degree of nodes in $R$, signifying the number of boundary edges each node takes part in. For our purposes, we do not keep track of the connection instructions. The rule is added to the grammar $G$ if it is a new rule. Otherwise, the frequency of the existing rule is updated.

3. **Updation** - we update the graph $H$ by (a) removing the subgraph induced by $V_{\eta^*}$ on $H$, and (b) connecting $X$ to the rest of the graph with the boundary edges. The modified graph is set as the current graph $H$ in the next step. The dendrogram $D$ is updated accordingly by replacing the subtree rooted at $\eta^*$ by $X$.

In the final iteration, the LHS is set to $S$, the starting non-terminal of size 0.

---

**Figure 2:** One iteration of the rule extraction process. (A) Example graph $H$ with 9 nodes and 16 edges. (B) A dendrogram $D$ of $H$ with internal nodes as $\eta_i$. The internal node $\eta_6$ is selected. (C) The current graph with highlighted nodes $\{c, d, e\}$. The 5 boundary edges are in red. (D) The RHS is a non-terminal of size 5 corresponding to the 5 boundary edges in (C). The boundary degrees are drawn above the internal nodes. (E) Updated graph with $\{c, d, e\}$ replaced by the non-terminal of size 5. (F) Updated dendrogram with subtree rooted at $\eta_6$ replaced by the new non-terminal.

**Figure 3:** CNRG rules obtained from the dendrogram in Fig. 2(B) with $\mu = 4$. 
Graph Generation

The grammar $G$ encodes information about the original graph $H$ in a way that can be used to generate new graphs. How similar are these newly generated graphs to the original graph? Do they contain similar structures and similar global properties? In this section, we describe how to repeatedly apply rules to generate these graphs.

We use a stochastic graph generating process to generate graphs. Simply put, this process repeatedly replaces nonterminal nodes with the RHSs of production rules until no nonterminals remain.

Formally, a new graph $H_0$ starts with $S$, a single nonterminal node labeled with 0. From the current graph, we randomly select a nonterminal and probabilistically (according to each rule’s frequency) select a rule from $G$ with an LHS matching the label $w$ of the selected nonterminal node. We remove the nonterminal node from $H_0$, which breaks exactly $w$ edges. Next, we introduce the RHS subgraph to the overall graph randomly rewiring broken edges respecting the boundary degrees of the newly introduced nodes. For example, a node with a boundary degree of 3 expects to be connected with exactly 3 randomly chosen broken edges. This careful but random rewiring helps preserve the topological features of the original network. After the RHS rule is applied, the new graph $\hat{H}$ will be larger and may have additional nonterminal nodes. We set $H_0 = \hat{H}$ and repeat this process until no more nonterminals remain.

An example of this generation process is shown in Fig. 4 using the rules from Fig. 3(A). We begin with $[0]$ and apply $r_1$ to generate a multigraph with two nonterminal nodes and two edges. Next, we (randomly) select the nonterminal on the right and replace it with $r_2$ containing four terminal nodes and 6 new edges. There is one remaining nonterminal, which is replaced with $r_3$ containing two terminal nodes, one nonterminal node, and 5 edges. Finally, the last nonterminal node is replaced with $r_4$ containing three terminal nodes and three edges. The edges are rewired to satisfy the boundary degrees, and we see that $\hat{H} = H$. In this way, the graph generation algorithm creates new graphs. The previous example conveniently picked rules that would lead to an isomorphic copy of the original graph; however, a stochastic application of rules and random rewiring of broken edges is likely to generate various graph configurations.

Main Results and Discussions

Through experiments on real-world networks taken from SNAP and KONECT, we show that the CNRG model is both compact and faithful to the original graph when compared to other graph models. To compare the model size, we adopted a modified version of the Minimum
Description Length (MDL) encoding used by Cook et al. [71]. For assessing the generative power of CNRGs, we use Graphlet Correlation Distance (GCD) [72]. Fig. 5 shows the comparison plots.

A potentially significant benefit from the CNRG model stems from its ability to directly encode local substructures and patterns in the RHSs of the grammar rules. Forward applications of CNRGs may allow scientists to identify previously unknown patterns in graph datasets representing critical natural or physical phenomena. Further investigation into the nature of the extracted rules and their meaning (if any) is a top priority.

3 Proposed Work

The research goal of this proposal is to study, develop, characterize, and evaluate techniques that use graph grammar formalisms to discover and understand the structure and growth of real-world networks in a principled way.

To achieve precise and complete structure discovery of a real-world network, two essential requirements must be met within a single system:

i. The building blocks, i.e., small subgraphs, that comprise any real world network must be efficiently and exactly captured in the model, and

ii. The model must represent the local and global patterns that reside in the data.

The first requirement overcomes limitations that are found in state-of-the-art graph mining algorithms. By extracting an CNRG from the graph’s dendrogram, the model will capture all the necessary graph building blocks. The second requirement is met by the relationship between graphs and context-free string grammars. Because of the significance of the proposed work, there is a significant amount of research that needs to be done. Among the many possibilities, the three future objectives detailed in this proposal were chosen because they have the most potential for broad impact and open the door for the widest follow-up work.

First, we study the stability of the CNRG extraction algorithm as this directly affects the
quality of the extracted grammar rules. Having a deterministic rule extractor opens the door for
a comparison of grammar rules across different classes of graphs. We then proceed to extend the
Vertex Replacement Grammar formalism to handle multi-layer graphs, a first step to handling
the class of attributed networks. Finally, we study the robustness of existing graph generative
models using a novel stress test called the Infinity Mirror Test, which helps uncover hidden biases
in such models.

3.1 Studying the stability of CNRG grammars

A Clustering-based Node Replacement Grammar (CNRG) can represent any graph structure,
but not uniquely. That is, many different dendrograms can represent a single graph, and a
the same dendrogram may represent many different graphs [73]. Production rules are directly
extracted from the dendrogram, so it is essential to understand how the choice of clustering
algorithm and the shape of the dendrogram affects the grammar.

Designing graph models that are stable for a fixed input graph(s) and parameters remains a
challenge today. In other words, multiple instances of a graph model (say SBMs) on the same
input graph(s) and input parameters, should yield (nearly) identical model parameters. Note
that this does not impose a constraint on the space of generated graphs - since the same model
can often produce a family of random graphs.

Having stable graph models allows us to compare the different model parameters across
different domains and classes. Now, we discuss one of the critical challenges we to face to
make the CNRG graph grammar extraction process stable. Although CNRGs [59] are much
more stable than HRGs [58], the grammar extraction process is sensitive to the dendrogram of
the input graph. A different dendrogram can often lead to a similar but non-identical set of
grammar rules.

Popular graph clustering algorithms like Leiden [69], Louvain [74], Infomap [75], or even
spectral clustering [70], are unstable since the solution space is non-convex and the lack of a clear
global maximum [76]. To counter this problem, researchers have proposed consensus clustering
methods, which aggregate multiple flat cluster assignments into one aggregated clustering
[77–79]. Hierarchical clustering techniques usually inherit all the disadvantages of flat clustering
algorithms when it comes to stability. Consensus hierarchical clustering usually involves aggregating
individual dendrograms or trees into a representative dendrogram [73, 80, 81]. They usually
have to impose additional constraints like the choice of distance metrics. These constraints
seem to work well for spatial data; they are not as suitable for graph data.

The goal of this task is to understand the relationship between a hierarchical graph clustering
algorithm, its dendrogram, and the extracted CNRG. The choice of the clustering algorithm, the
shape of the dendrogram, and the extracted CNRG will be rigorously investigated using tree
distance metrics, graph alignment techniques, and standard statistical analysis. Even though the
resulting dendrograms may prove to be of different shapes, the extracted CNRG may still be
stable because the node labels are not copied into the grammar. It is, therefore, possible, even
likely, that different dendrograms will still produce very similar CNRGs.

No matter the outcome, further exciting questions can be asked and answered. Because of
the CNRG-to-graph relationship, if the extracted CNRGs are indeed vastly different, then the
production rules that do overlap will be uniquely informative about the nature of the data. If
the extracted CNRGs are similar, then the extracted CNRG will be uniquely representative of the
data.

Evaluation Plan We possess hundreds of graph datasets originating from public online re-
sources like SNAP, KONECT, and the UCI Repository, as well as several hierarchical graph clus-
tering algorithms that can be used in experimental tests [69, 70, 74, 75]. To answer questions 
about stability, a principled notion of tree and grammar similarity is required. Many principled 
metrics exist for tree similarity [82], but we will need to make some adaptations to account for 
items unique to a dendrogram. Comparing different grammars is one area where results from 
formal language theory may be helpful. Unfortunately, the problem of determining whether 
two different grammars produce the same string is undecidable [83], which means that the exact 
comparison between CNRGs is undecidable. Nevertheless, just as in formal language theory, 
many approximate similarity methods exist for CFGs that can be adapted for CNRGs [84].

Here the central theme of this proposal is evident: *we will be able to adapt and leverage ideas and approaches from computational and formal language theory to solve difficult challenges in graph mining and network analysis.*

### 3.2 Multi-Layer Vertex Replacement Graph Grammars

So far, in this proposal, we have restricted ourselves to homogeneous networks, where all the 
nodes and edges are identical. In practice, we often deal with networks where individual nodes 
and edges have associate attributes. Edges in those cases usually encode a specific connectivity 
pattern across a set of nodes. We call such networks heterogeneous, and while researchers have 
proposed methods to model such graphs, it remains an open area of research [85–88].

Graph grammars, due to their somewhat simple design, should allow us to adapt previously 
used formalisms to a heterogeneous setting. However, care must be exercised while doing so 
since it can lead to overfitting the input graph leading to an exponential number of graph rewrit-
ing rules. As the starting point, we will study a specific class of heterogeneous graphs known 
as multi-layer graphs [89–91]. The airline network can be thought of as a multi-layer network, 
with nodes representing airports, edges representing non-stop flights by an airline, and where 
each airline has its own layer. Temporal networks can be thought of as multi-layer networks 
with different time steps represented as different layers. More generally, though, the layers can 
represent different contexts, and a node $u$ in layer $\ell_1$ can be connected to a node $v$ in any layer 
$\ell_2$. So, the edges can be classified into two categories— intra-layer and inter-layer— depending on 
whether their endpoints lie on the same or different layers.

Multi-layer graphs are usually represented with a pair of tensors, each encoding the intra-
layer and inter-layer connectivity patterns, respectively. This idea can be extended to learn a pair 
of inter and intra-layer graph grammars. However, while doing so, we should look to preserve 
the node correspondence as well as attribute correlations across layers, and not treat the layers 
as independent networks.

**Evaluation Plan** We will adopt a methodology of similar rigor used in the previous papers to 
study multi-layer graph grammars. By using a combination of synthetic and real-world multi-
layer graphs obtained from the SNAP, ICON, and Konect databases, we will test out the accuracy 
of the proposed model(s). This can be achieved by comparing the different distributions associated 
with multi-layer networks as well as studying communities and other mesoscale structures [92].

### 3.3 The Infinity Mirror Test for Graph Generators

Next, we propose a stress test for evaluating graph models. This test iteratively and repeat-
edly fits a model to itself, exaggerating models’ implicit biases.

Graph models extract meaningful features $\Theta$ from a graph $G$ and are commonly evaluated 
by generating a new graph $\tilde{G}$. Early models, like the Erdős-Rényi and Watts-Strogatz models,
depend on preset parameters to define the model. More recent approaches aim to learn $\Theta$ directly from $G$ to generate $\tilde{G}$.

The Chung-Lu model, for example, generates $\tilde{G}$ by randomly rewiring edges based on $G$’s degree sequence [31]. The Stochastic Block Model (SBM) divides the graph $G$ into blocks and generates new graphs $\tilde{G}$ respecting the connectivity patterns within and across blocks [35]. Graph grammars extract a list of node or hyperedge replacement rules from $G$ and generate $\tilde{G}$ by applying these grammar rewriting rules [58, 59]. Recently, graph neural network architectures have also gained popularity for graph modeling. For example, Graph Variational Auto-Encoders (GVAE) learn a latent node representation from $G$ and then sample a new graph $\tilde{G}$ from the latent space [50]. NetGAN learns a Generative Adversarial Network (GAN) on both real and synthetic random walks over $G$ and then builds $\tilde{G}$ from a set of plausible random walks [54].

Each of these models has its advantages, but each model may induce modeling biases that standard evaluation metrics are unable to demonstrate. In the present work, we describe the Infinity Mirror Test to determine the robustness of various graph models. We characterize the robustness of a graph generator by its ability to learn and regenerate the same model repeatedly. The infinity mirror test is designed to that end.

**Infinity Mirror Test**

Named after a common toy, which uses parallel mirrors to produce infinitely-repeating reflections, this methodology trains a model $\mathcal{M}$ on an input graph $G_0$ and generates a new graph $\tilde{G}$ and repeats this process iteratively on $\tilde{G}$ for a total of $k$ generations, obtaining a sequence of graphs $\langle G_1, G_2, \cdots, G_k \rangle$. This process is illustrated in Figure 6.

The infinity mirror methodology produces a sequence of generated graphs, each graph prediction based on a model of the previous prediction. Like repeatedly compressing a JPEG image, we expect that graphs generated later in the sequence will eventually degenerate. However, much can be learned about hidden biases or assumptions in the model by examining the sequence of graphs before degeneracy occurs.

We illustrate some initial results in Figure 7 using three synthetic input graphs with easily-identifiable visual structure $G_0 = \{a 10 \times 10$ grid graph, a 25-node ring of 4-cliques, and a synthetic graph with 3 well-defined communities$\}$.

**Methodology and Evaluation Plan**

We consider the following graph models $\mathcal{M} = \{\text{Chung-Lu, degree-corrected SBM (DC-SBM), Hyperedge Replacement Grammars (HRG), Clustering-based Node Replacement Grammars (CNRG), GVAE, and NetGAN}\}$.

For each combination of input graph $G_0$ and generative model $\mathcal{M}$, we generate 50 independent chains of length $k = 20$ and compute the DELTACon score [93] comparing $G_0$ and $G_k$. We select the chain resulting in the median DELTACon score, and illustrate $G_1, G_5,$ and $G_{20}$ (i.e., the 1st, 5th, and 20th generations) in Figure 7 if they exist.
Preliminary Results and Discussion

These initial results show that CNRG can capture and maintain specific structures from the input graph. On the ring of cliques, we see perfect replication, as well as on the first generation of community, with some marginal degradation as the number of repetitions increases. SBM performs similarly-well on such structured input. Neither model works particularly well on grids; CNRG introduces triangles, while SBM creates nodes of degree 1.

HRG fails for grids and does not appear to perform well on the other two graph types. The Chung-Lu model mimics the degree distribution of the input as we would expect, but fails to capture the network structure. We can see long chains formed from the grid input by the last generation, and the output of the other two graph types do not resemble the original graph. We also see that Chung-Lu fails to preserve the connectivity of the graph, as shown by the disconnected components in the last generation of each column.

GraphVAE results in overly dense graphs regardless of the input, which obfuscates any topological structure the input graph had. NetGAN tends to produce sparse, tree-like graphs. This becomes a problem down the line. The way that NetGAN computes its train/test/validation split relies on finding a spanning tree on the input graph; when the model is recursively applied to its increasingly tree-like outputs, it will eventually fail to find a satisfactory split, causing the
model to fail to fit.

The infinity mirror test we proposed exposes biases in models that might otherwise go unnoticed. Future work will determine how best to use this methodology to analyze biases and errors in graph models.

4 Summary

Uncovering hidden relationships buried within large volumes of data is one of the most fundamental challenges in data mining research. By observing the world through a combination of lenses of graph theory, formal language theory, and data mining, we can gain a greater understanding and appreciation of the wealth of data around us. The research described in this proposal makes significant inroads to that effect.

We have described novel methods that help us understand how we can study and design the next generation of graph generative models—ones that are flexible, interpretable, and scalable—by using graph grammars. Furthermore, by doing so, we lay the groundwork for future work in this space, some of which are highlighted in Section 3. This includes expanding current formalisms to study attributed networks, which adds a great deal of complexity to the analysis. Finally, we propose a novel stress test for graph models that exaggerates and reveals the latent biases of a model. This will help the researchers in the community gain insight into the inner workings of such models while also guiding future researchers in designing better methods.

Timeline

<table>
<thead>
<tr>
<th>Activity</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infinity Mirror</td>
<td>Now - May 2020</td>
</tr>
<tr>
<td>Stable Graph Grammars</td>
<td>Aug - Dec 2020</td>
</tr>
<tr>
<td>Multi-layer Graph Grammars</td>
<td>Jan - Dec 2021</td>
</tr>
<tr>
<td>Defend</td>
<td>Spring 2022</td>
</tr>
</tbody>
</table>
References


[16] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *ICDM*, pages 721–724. IEEE, 2002.


