Generating Networks by Learning Hyperedge Replacement Grammars

A Dissertation

Submitted to the Graduate School

of the University of Notre Dame

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Salvador Aguiñaga

Tim Weninger, Director

Graduate Program in Computer Science and Engineering

Notre Dame, Indiana

November 2017

Generating Networks by Learning Hyperedge Replacement Grammars

Abstract

by

Salvador Aguiñaga

Network modeling is critical and central to the study of complex systems. Modeling enables researchers to examine emergent behavior and related phenomena, a mechanism we still know little about and one arising from the milieu of interacting patterns at the local level. These complex systems range from the global economy, the neuroscience of the brain, weather, protein folding molecular interactions, to the Internet. Evaluating network models on their ability to learn the underlying features automatically is integral to algorithm development in many areas of computational science.

Here we describe methods and develop algorithms that extend and evaluate hyperedge replacement grammars, a formalism in formal language theory. We detail extensions for model-inference on real-world networks and graph generation. Discovering patterns involved in system behavior to build models for real-world systems that preserve many of the network properties during the generation step is the central focus of this work. Growing similar structures at various scale is also crucial to the evolution of the scientific tools required in today's information landscape. Experimental results demonstrate that hyperedge replacement grammars offer a new way to learn network features that facilitate compelling graphical structure generation that advances network science in areas of modeling and network analysis.

To my brilliant and beautiful wife, Kristin and to my darling children, Izzy, Adri, and Vivi.

CONTENTS

FIGURES

TABLES

ACKNOWLEDGMENTS

First, I am indebted to my advisor, Tim Weninger, one of the most positive, generous, and extraordinary individuals I have ever met. I consider myself lucky for the opportunity to work with him and all he has taught me. I am grateful for his support and generous advice for the last four years. The work presented in this thesis would not have been possible without him.

Special thanks go out to Joyce Yeats, Ginny Watterson, M. D. McNally, and Diane Wordinger in the Department of Computer Science and Engineering. Joyce has always guided me and helped navigate all aspects of the graduate program from day one. Their help and dedication to graduate students' success are invaluable. I would like to thank iCENSA. My interaction with the students and professors in iCENSA has had an impact on how I see the world of research and science. I would like to extend another especial thanks to Ms. Jasmin Botello for making iCENSA a great place to work. iCENSA would not be the great place it is without her. Her enthusiasm and commitment to both, professors and students, makes us feel that we belong and are part of an exceptional community of researchers.

I would like to thank every one of my collaborators. They are a fantastic group of talented individuals. I would like to acknowledge Nitesh Chawla and Aastha Nigam, Corey Pennycuff, Cindy Wang who is now a graduate student at CMU. To David Chiang, I would like to extend my thanks for his guidance, and for challenging me and offering insights that have shaped my work. Before working with Tim, Chris Poellabauer supported me and worked with me on a pair of excited projects. I am grateful to have gotten to know him and his students. Chris made feel at home my first two years of graduate school, and I am thankful for his enthusiasm and guidance during my most challenging years of graduate school. I would like to thank and acknowledge my thesis committee for their support and advice: James Evans, University of Chicago, Nitesh Chawla, and David Chiang.

Next, I would like to acknowledge the entire Weninger Lab especially my lab mates: Maria Glenski, Corey Pennycuff, and Baoxu (Dash) Shi. I am especially thankful to Dash for engaging with me on many exciting conversations and for allowing me and my family to be part of a special day: Jessie and Dash's wedding day. The DSG-ND group is an incredible collection of individuals; it has been my honor to know them and worked with them.

Lastly and most importantly, I would like to thank my love, my best friend, and the smartest person I know, my beautiful wife, Kristin. Her belief in me, her support, advice, and her endurance for putting up with me especially during the most trying days shows me how fortunate I am. All that I have accomplished during my tenure as a graduate student I owe it all to her and the love of my beautiful children: Isabella, Adrian, and Violetta. Their zest, curiosity, love, and affection kept me going. I am indeed a lucky man.

CHAPTER 1

TREE DECOMPOSITION

1.1 Introduction

The underlying mechanism of how social networks grow or, more generally, how real-world networks grow or evolve is a core area of research in computer science, mathematics, engineering, and physics. A recently introduced generative network modeling framework, Hyperedge Replacement Grammars (HRG), explores network structure to derive a set of replacement rules. These rules are also known as graph rewriting rules or as will be used throughout this work, production rules. These rules represent the grammars of the graph in question. Applying these recursively, we grow graphs with characteristic properties similar to the input graph. The rules derived are in response to the variable-elimination algorithm used in the decomposition step. The research question of interest is, what is the diversity of the productions rules we end up with in choosing the variable elimination algorithm during the decomposition step? We are motivated to explore this question because we want to examine if the rules are invariant to the choice of algorithms in this critical step. The original implementation of the HRG graph model yields TD characterized by relatively low treewidth (tw). In graph theory, $tw$is the size of the largest vertex set in a tree decomposition of a given graph. Vertex sets are also characterized as cliques, thus twcomes or is computed from the size of the largest clique in a chordal completion of the graph.

A key assumption in the tree decomposition step is computing tree decompositions with low (or close to optimal) tw. The motivation behind low $tw$is that with structures such as these trees, we can solve problems in probabilistic inference, constraint satisfaction, and matrix decomposition [8, 12, 12]. Unfolding how these structures are computed as a function of tree decomposition heuristics will be explored to gain better understanding on the structural patterns that underlie community and other network properties inherent to a diverse set of networks such as social, biological, and collaboration networks.

1.1.1 Core problem definition

We study the problem of graph decomposition. We examine patterns graph topology found in the resulting clique-trees (CT). The purpose is to infer a model for the given graph or a class of graphs. We have described how HRG takes the output of a tree decomposition and construct production rules, a process analogous to the productions derived in context-free grammars [2]. Formally, for any static graph, *i.e.*, a graphical representation of any complex system, let $G = (V, E)$ be the graph observed and $H = (X, E)$ its hypergraph. The graph $H^* = (X, E)$ is obtained through stochastic application of the production rules. Graph generation is split into two procedures, inference of HRG production rules and stochastic application of the graph grammars. The inference step may be visualized as follows, $H \xrightarrow{TD(H,\ell)} CT \xrightarrow{HRG} PR$ and the graph generation step $\mathcal{G}_{HRG}(PR, \nu) \rightarrow H^*$, where ν is the desired graph order (number of vertices), a$\mathcal{G}$ is the hyperedge replacement graph grammar, and ℓ is the variable elimination algorithm that characterizes the resulting clique-tree (CT). Synthetic graphs H^* reflect many of the network properties inherent in the reference or input graph. We interchangeably use the terms *reference graph* and *input graph* throughout this work when referring to a real-world graph.

Earlier work utilized the maximum cardinality search variable elimination algorithm described by Tarjan and Yannakakis [22] in the tree decomposition step. Thus, one of the open questions in the HRG graph model is how the choice of tree decomposition algorithm biases or affects the topology of the graph-fragment patterns used in the productions inferred. We answer this question and leverage multiple tree decompositions to hone in on patterns that might be invariant to tree decomposition and may offer insight into the fundamental patterns that contribute the global structure of a graphs.

We are interested in examining how the choice of ℓ contributes to the resulting characteristics of the CT and thus to the resulting graph grammar. Construction of the graph grammar is biased on the choice of variable elimination algorithm.

1.1.2 Outline of this work

In Section 1.2 we examine the background literature. In Section 1.4 we outline specific research questions and describe experiments and methods used to help answer them. Details of the results and a brief discussion on what conclusions we can draw are presented in Section 1.5 and conclude by looking ahead to some future direction in 1.5.2.

1.2 Background information and related work

A great deal of work has focused on applications of the algorithms that underlies tree decomposition. The concept of tree decomposition is also known by other other names depending on the specific branch of computer science and they include clique-trees, cluster graphs, and junction trees. This work examines the use of tree decomposition heuristics for deriving HRG's production rules characterizing the features of the graph model.

Existing algorithms for constructing TDs range from those shown to be practically intractable due to their complexity to those characterized as computational for employing heuristic approaches whose deviation from optimality is guaranteed. Shoikhet and Geiger [21] developed a computational (*i.e.*, practical) algorithm, Quick-

Tree, that can optimally triangulate graphs in a reasonable amount of time. This algorithm saw usefulness in problems in Bayesian inference and clique-tree inference algorithms. Gogate and Dechter [6] developed QuickBB, a branch and bound (BB) algorithm for computing the *tw*of undirected graphs. BB search algorithms fuel algorithm design trends in discrete and combinatorial optimization.

Jones *et al.*, describe very similar work where they extracted grammars using tree decomposition. Their work focuses on a class of tree decomposition called edge-mapped TD [10]. This tree decomposition is extended to use a topological sort that produces clique trees like those in HRG. However, they evaluate the model in contrast to other forms of the tree decomposition yielding linear trees. They rely on a measure of perplexity (akin to the entropy of text) to evaluate the production rules. Their work does not generate graphs using the derived rules and their approach is tested only on a series of graphs that have an average of 15 vertices as well as edges.

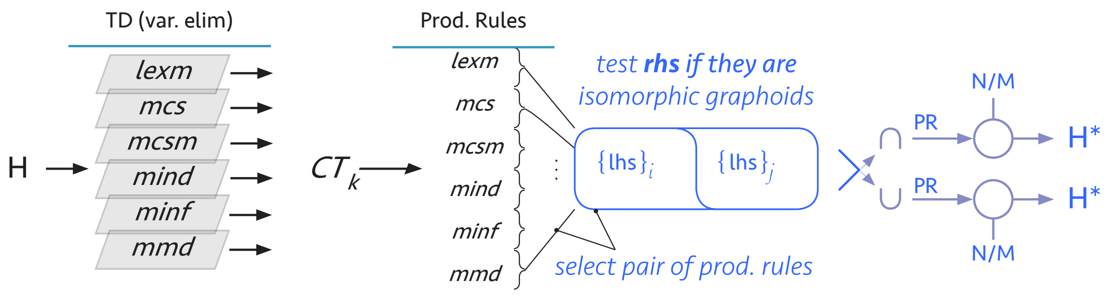


Figure 1.1. Proposed methodology: Tree Decomposition, clique-tree (CT), Production Rules (PR)

1.2.1 Tree Decomposition Influence on HRG

The influence of tree decomposition on graph grammars is explored to shed light on the bias or the assumptions inherent in the underlying algorithms. All graphs can be decomposed (though not uniquely) into a clique-tree, also known as a tree decomposition, junction tree, join tree, intersection tree, or cluster graph. clique-trees are found through a simple ordering the vertices in the input put graph. Computing a perfect ordering ensures graph chordality and finding this perfect ordering relies on maximum cardinality search or lexicographical BFS (lexBFS) algorithms. Minimal vertex separators is another algorithm for computing clique-trees are examined in this paper.

Constant factor approximation algorithms typically used for treewidth problems include triangulation heuristic algorithms such as min-degree, max-cardinality search, min fill, lexBFS, and min-vertex separators, we expand on these next with more detail. Treewidth (tw) is defined as a property for clique trees or cluster graphs, measuring the ''tree-likeness" of the reference graph.

1.3 Methods and proposed work

1.3.1 Tree Decomposition and Variable Elimination

The question of interest is how does tree decomposition contributes, affects, or biases the productions rules derived from the input graph? To address this we focus on deriving rules using variations of tree decomposition. In the decomposition tree step, a Adcock *et al.* [1] have developed tree decomposition that relies on different variable elimination ordering algorithms.

A battery of variable elimination heuristics will be used in the tree decomposition step. These heuristics include maximum cardinality search (MCS), minimum triangulation of H (MCS-M) [3], lexicographic search with minimal ordering (LEX M) [20],

TABLE 1.1

Summary of variable elimination algorithms used in this study.

VE Heuristic	Description
MCS	Maximum cardinality search is a simple heuristic that works well on chordal graphs.
MCSM	Minimum triangulation extension to MCS.
Min Degree	Minimum degree is a well known general-purpose ordering scheme and is widely used in sparse matrix computation.
Min Fill	Minimum fill consists of greedy node elimination with the fewest edges are added breaking ties arbitrarily.
Lex-M	Derived from lexicographic breadth-first search for minimal triangulation.
MMD	Multiple minimum degree

minimum fill-in (MINF), Minimum degree (MIND), and Multiple Minimum Degree (MMD) [15]. These heuristics an others are described by Kemazi and Poole [11]. However, it must be noted that these heuristics do not add up the complete list of available algorithms [4, 18], but we choose this for this work.

The baseline tree decomposition variable elimination algorithm allowing or facilitating transformation of an input graph into clique-tree is MCS. In our earlier work, the HRG model uses Tarjan and Yanakakis' MCS algorithm. We examine and evaluate clique trees from the heuristics in Table 1.1. We select a diverse set of real world networks. Table 1.2 shows some of the properties of these graphs exploring others as well in detail next.

Answering the core question explored in this work requires us to examine some properties of the clique trees. We start with (tw), a graph feature or property measuring how “treelike” a tree is.

1.3.2 Datasets

A number of empirical networks, otherwise known as real-world networks, are used to evaluate the different heuristics. We are interested in variable elimination and their effect on graph grammars. These networks vary in the number of nodes and edges. We consider small and large networks push the limits of tree decomposition, but more importantly we want a wide range and diverse set of graph grammars (graph fragments) to understand how the POE contributes to the network model. These networks are also characterized by other properties that we hope HRG is able to model accurately.

Small Network Datasets. Highland tribes (Gama) is a social network of tribes of the Gahuku-Gama alliance structure found in Eastern Central Highlands of New Guinea. Zachary’s Karate club (Karate) is a snapshot of a university karate club studied by Wayne Zachary. All real-world networks are public datasets [13, 14].

Larger Network Datasets. *EuroRoad* is an infrastructure network of Europe’s roads where nodes are cities and edges represent the road that connects them. *CollegeMsg* is a network of private message between college students using an online social network at the University of California, Irvine. Table 1.2 illustrates the size of the graphs used in the experimental section.

1.4 Experiments and Results

One of they ways we can inspect how the choice of variable elimination algorithm affects the production rules is to generate synthetic graphs and use a graph alignment

TABLE 1.2

Real-world networks

Datasets	nodes (n)	edges (m)	Avg. degree ($\bar{k}$)
Tribes (Gama)	16	58	7.25
Les Miserables	77	254	6.60
Hypertext	113	2196	38.87
arenas-jazz	198	2742	27.70
Contact	274	2124	15.50
email-Eu-core	309	1938	12.54
Infectious	410	2765	13.49
EuroRoad	1174	1417	2.41
College Msg	1899	13838	14.57

test to measure graphlet distance between graphs. Figure 1.2 shows that half the datasets show no significant difference in their GCD score. Complete GCD results are presented in Tab. 1.5.

1.4.1 Evaluation of Tree Decompositions

Methods of evaluating tree decomposition include tw. Derived productions rules undergo a isomorphic test to find subsets used to grow synthetic graphs. The resulting graph get the bulk of the metrics. These graphs will be evaluated based on degree distribution, clustering coefficients, hop-plot, and graphlet correlation distance (GCD).

One of our first goals is to evaluate HRG's variable elimination method of the resulting CT. Each dataset is transformed into a clique tree, the tree is binarized, only them we derive the production rules. The tree decomposition is obtained using

TABLE 1.3

tw as a function of variable elimination algorithm

	Treewidth (tw)						
Dataset	mcs* μ (σ)	mcs	lexm	mcsm	mind	minf	mmd
LesMis	9 (0)	11	9	11	9	9	9
contact	40	42	43	43	50	40	40
arenas-jazz	59 (0)	88	77	81	104	59	73
pdzbase	6	9	12	13	6	6	6
ucforum	126	326	361	341	282	276	279
Hypertext	76 (0)	80	89	89	76	76	76
Infectious	39 (0)	65	56	128	42	40	49
emailEuCore	34 (0)	41	46	45	35	34	35
EuroRoad	6.6 (2.6)	42	30	48	19	16	16
College Msg	87.6 (20.3)	459	602	543	404	394	403

the tool INDDGO [1] while each variable elimination heuristic is specified as one of the arguments in the execution.

1.4.2 Treewidth

Table 1.3 shows the (tw) that results from tree decomposition using different variable elimination algorithms. The first column shows the tw computed using the HRG model. We learn that our implementation of *mcs* is among the variable elimination algorithms that yield clique-trees with the smallest tw and that the next best algorithm is is $minf$. The baseline algorithm for *TD*is a Python implementation

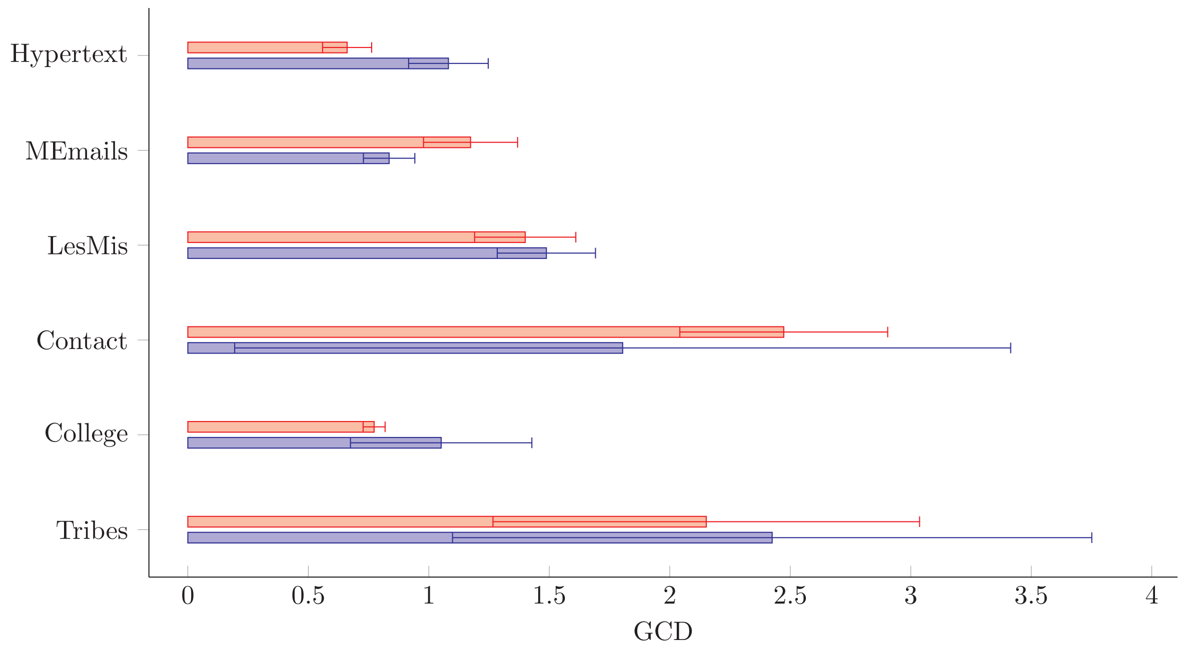


Figure 1.2. Graphlet correlation distance on graphs generated using productions derived from clique-trees computed with 🟦 MCS (baseline) and 🟧 MinF.

of QuickBB, which relies on *mcs*. All other elimination orderings will be computed using INDDGO [7].

1.4.3 Production Rules

Production rules are characterized by by the number of non-terminal nodes in the left-hand side (LHS) of the rule.A matrix showing the percentage of overlap between different.

1.4.4 Isomorphic Overlap

We are interested in patterns found in the various production rules. Each set of production rules set are independently derived. However, production rules still share

TABLE 1.4

Production rules overlap: email-Eu-core

	mcs	mind	mcsm	lexm	mmd	minf
mcs	-	0.182	0.182	0.161	0.194	0.186
mind		-	0.207	0.159	0.220	0.209
mcsm			-	0.180	0.201	0.191
lexm				-	0.157	0.155
mmd					-	0.184
minf						-

Figure 1.3. Isomorphic overlap via Jaccard Similarity

dependence on the source graph even if they are derived using different TD schemes. We examine graph structure as found in RHS rules. We use both approximate and complete isomorphic tests. We select rules that have the same number of terminal objects in the LHS. We compute Jaccard similarity to assess the ratio of isomorphic rules over a range of TD. Triangle heat dots show the Jaccard similarity between any pair of production rules. This result implies that certain variable elimination methods share an affinity. We intend to exploit this affinity to test if the intersection of isomorphic graph-fragments (multi-graph instances of the RHS of production) yields a smaller set of production rules that generates graphs.

1.4.5 Isomorphic graph-fragments

A subset of production rules from the concatenation of rules derived using several different tree decompositions is possible using isomorphic graph-fragment tests. This approach lets us hone in on automatically produced or derived rules. Some graph-fragments are observed to be found or produced by different tree decomposition algorithms. This collection of PHRG rules lead to productions that can generating graphs using a stochastic application of the rules. Fig 1.5 show the number of rules in a concatenated rules set (blue) and the subset resulting from isomorphic patterns. These smaller subset of production rules captures sufficient information to generate synthetic graphs. In Fig. 1.5 we show both the rules subset from the isomorphic test in contrast to the union of the rules for real-world networks and for random (Barabasi-Albert) graphs.

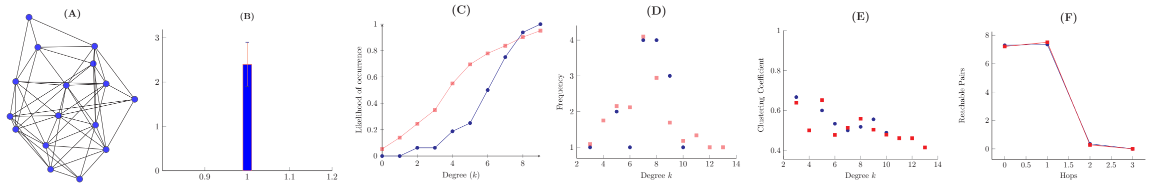


Figure 1.4. Degree distribution. (A) Highland tribes graph, (B) GCD, (C) Degree CDF, (D) Degree distribution, (E) Clustering coefficients, (F) Hop-plot

1.4.6 Generalization to Classes of Graphs

Formative patterns playing a key role in how networks grow might be found to be consistent across similar class of networks. And through a range of networks

Figure 1.5. Union and set intersection of the production rules derived using different variable elimination methods.

generated by the Barabasi-Albert random graph model we probe the usefulness of patterns found through a battery of tree decomposition algorithms. We want to examine if the production rules inferred from these artificially generated graphs hold the power to characterize the degree distribution in a single class of well defined graphs.

We generate random graphs of various sizes using the Barabasi-Albert model and we process each graph. In other words, we infer as set of production rules using the same set of tree decomposition algorithms used throughout our experimental setup. Furthermore, we hone in on RHS graph-fragments that are isomorphic regardless of the TD algorithm and test if this set holds sufficient information to generate similar graphs.

1.5 Conclusions

Generative network models using hyperedge replacement grammars are power tool in network science. The HRG graph model preserves many of the input graph's network properties such as degree distribution and others. One open question on the HRG generative model is how the choice of tree decomposition affects the inferred model. We examine a battery of tree decomposition algorithms to examine the production rules and find an answer to this question. Our initial experimental results highlight two important findings: not all variable elimination algorithms yield clique trees, and of those variable elimination algorithms that yield clique trees not all derived production rules fire, *i.e.*, can generate graphs.

TABLE 1.5

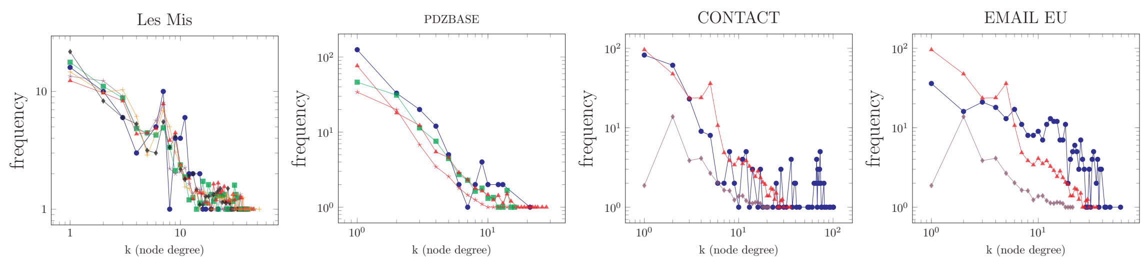
Clique-trees using variable elimination heuristics

	Graphlet Correlation Distance $\mu(\sigma)$					
Dataset	mcs	lexm	mind	mcsm	minf	mmd
LesMis	1.46 (0.16)	1.6 (0.16)	1.7 (0.2)	1.8 (0.13)	1.4 (0.15)	1.774 (0.280)
ucforum	0.985 (0.093)	1.030 (0.098)	1.198 (0.093)	0.948 (0.113)	1.242 (0.107)	1.002 (0.106)
Hypertext	0.707 (0.137)	0.582 (0.112)	0.606 (0.143)	0.576 (0.137)	0.661 (0.154)	0.653 (0.177)
Infectious	0.760 (0.060)	0.889 (0.083)	0.824 (0.060)	0.684 (0.060)	0.857 (0.0613)	0.763 (0.085)
jazz	1.2 (0.1)	0.98 (0.03)	1.2 (1.6)	0.75 (0.04)	1.11 (0.09)	0.977 (0.083)
pdzbase	1.8 (1.3)	2.2 (1.5)	4.0 (.40)	3.2 (1.3)	3.8 (0.7)	2.7 (0.27)
emailEuCore	3.0456 (0.118)	2.554 (0.100)	2.552 (0.248)	2.491 (0.151)	2.862 (0.269)	2.679 (0.267)
EuroRoad	1.633 (1.059)	2.300 (1.237)	2.820 (1.231)	2.728 (1.239)	3.003 (1.186)	3.224 (1.102)
College Msg		3.276 (0.275)	0.835 (0.036)	0.605 (0.047)	0.773 (0.059)	0.773 (0.059)

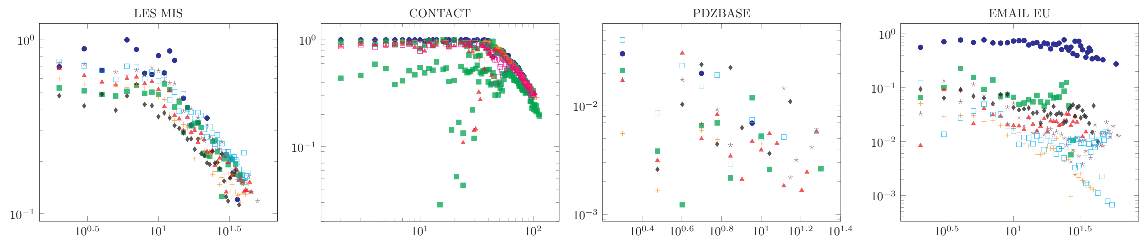
Where the derived production rules do fire, we explore the intersection of rules. If we further examine where the rules, or graph fragments, intersect we test if these subsets are smaller than any one individual rules set and test the set's ability to fire. When this small set of production rules can fire, we test the set to grow graphs. Our conclusion is that there are latent graph fragments with structure that when combined are compact enough to grow graphs. Using multiple variable elimination algorithms during the tree decomposition step flushes out these graph fragments. This finding hints to the need for a optimal method of finding a more compact set of production rules.

1.5.1 Limitations

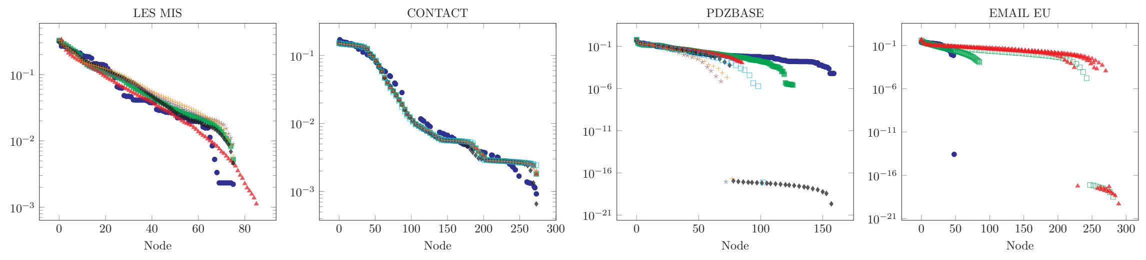
The challenges in this approach is the run-time increase required for each variable elimination algorithm. This naive approach does not offer a guarantee that we will end up production rules sets guaranteed to fire. The second limitation concerns scalability. An increase in the number of times we perform a tree decomposition given



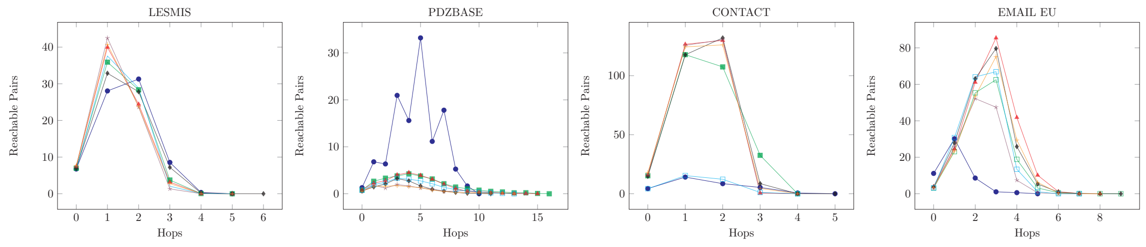
(a) Degree distribution.



(b) Clustering Coefficients.



(c) Eigenvector Centrality



(d) Hop Plot.

Figure 1.6. Network properties

different variable elimination algorithms increases the complexity of this intermediate step. Processing large graphs increases run-time, but we can not guarantee the tree decomposition will yield viable production rules. The size of the graph is address in this work by sampling from the reference graph. These limitations, again hint at the need for new approaches combining smart graph sampling and optimization to an ensemble of variable elimination yielding richer rules compared to the naive approach.

1.5.2 Future Direction

Potential directions for this work include looking into the use of perplexity, a measure of entropy of a subgraph, to evaluate derived HRG models. Especially as we improve on the on the naive HRG implementation. Another future direction for this work may explore deeper an ensemble tree decomposition that relies on finding a fast and practical variable elimination methodology that guarantees both, a smaller subset of rules and rules that can fire. These future directions may lead to the development of new tools for natural language processing and graph mining applications.

CHAPTER 2

CONCLUSIONS AND FUTURE DIRECTIONS

The underlying mechanisms in complex systems, *i.e.*, the local milieu of microstructures and their interactions which researchers still know little about, are critical to our understanding of the world around us. Uncovering universal patterns in areas as diverse as viral videos, the economy, or the Internet, is a monumental challenge. Understanding the properties of these patterns is critical to discovering useful tools to harness data to find novel solutions across a diverse range of domains such as medicine, national security, and global policy.

This work explores and evaluates principled techniques that learn the LEGO-like building blocks of real-world networks. We achieve this by exploiting the overlapping techniques where graph theory meets formal language theory. Building on these ideas, pioneered by my collaborators Tim Weninger and David Chiang, we examine graph decomposition, grammar extraction, inference, and analysis of network patterns. This relationship allows for a Hyperedge Replacement Grammar (HRG) to be extracted from any simple graph without loss of information, but more importantly with the power to generate new graphs modeled on the input or reference graph.

2.1 Conclusion

We focus on advancing HRG, a graph rewriting formalism, to a generative network model for a diverse set of real world graphs. This model learns the building blocks of simple networks and leverages its generating power to grow graphs that exhibit properties of interest found in the reference graph. The specific themes covered include model inference and stochastic graph generation, HRG fixed-size graph generation, measures of model resilience, and model bias resulting from the use of different tree decomposition algorithms. We expand on these themes below in more detail.

- **HRG Growing graphs.** Initial implementation derives a HRG, a set of production rules. Chapter **??** described the initial implementation of HRG as a graph model which shows the principled approach to transforming graphs into trees, deriving the graph grammar, and growing graphs using a stochastic growth algorithm.

- **Probabilistic HRG.** Improvements to the HRG model where the order (or size, in terms of the number of nodes) is specified. Chapter **??**, described a new procedure for growing graphs to address a limitation in the original implementation. Although the generated graphs have the same mean size as the original graph, the variance was too high to be useful. We use dynamic programming to sample a graph with specified size to illustrate this concept.

- **Infinity mirror tests model degeneration.** A new metric of model robustness examined model degeneration in Chapter **??**: infinity mirror test. This procedure examined any graph generator's output and repeatedly fits a model to the new output to see how well the model captures sufficient features to reconstruct graphs. We see that after a few recursive runs some of the existing graph models tend to degenerate. Some models quickly loose the necessary features the model relies on to generate graphs with many of properties in the original graph.

- **Graph grammar bias.** We explore the core of the tree decomposition step, a critical step during model inference. We explore how the production rules are biased during tree decomposition, by examining the algorithms involved in transforming the input graph into tree-like graphs. Chapter 1 shows an examination of various tree decomposition algorithms. We evaluate productions rules from various variable elimination algorithms. We do this by studying the rules between sets. Moreover, where we have viable rules, we grow graphs and evaluate them on the quality of the synthetic graphs the rules produce against the input (or reference) graph.

This work has expanded into other projects. Two collaborations have extended the HRG graph model into the exploration of temporal (dynamic) graphs and latent variables to improve the model.

2.1.1 Collaborations

2.1.1.1 Temporal graph grammars

Let's consider the behavior of travelers at airport. Modeling the incoming and outgoing flights and the movement of passengers is best represented by considering the temporal dynamics of the interactions or the insertion or deletion of nodes. We could potentially model these dynamics of how passengers move about a fixed space, although we have not yet explored this. We could also incorporate temporal behavior of the nodes and edges into our model requiring the HRG model to examine the time-stamps of the edges. This project considered growing graphical structures but did not plan for node deletions. We presented our findings at the Mining and Learning with Graphs Workshop [17] earlier this year.

2.1.1.2 Improving the HRG using latent variables

Our objective in this paper addresses one of the limitations of the HRG model. The graph grammar encodes only enough information to ensure that the result is well-formed. A critical step in deriving the instructions for building new graphs is the tree-decomposition step, transforming the input graph into the tree-like structure. We expected the productions used for graph generation to come from the top, middle, and bottom of the tree-graph. However, in the generation step, the process of selecting the rules is probabilistic relative to rule frequency within the grammar. Our primary focus is how to provide context for the rules on when they should fire.

My collaborators Xinyi Wang *et al.* proposed a mechanism that corrects for the problem described above. By encoding context in latent variables, we discover we can build a better model. We provide more details in a paper submission to an artificial intelligence conference, which is currently under review.

2.2 Vision and Future Work

By developing algorithms for graph generation models, we understand a bit better the underlying mechanism playing a significant role in network growth. Understanding local patterns of large networks can lead to high impact applications, novel mathematical abstractions, and to the development of sorely needed tools for the advancement of field today. There are three areas for future direction described in the following section.

2.2.1 Analytic Methods for the Network Properties of HRG Graphs

Limitations in the model surface as we explore improvements, extensions, and optimization. One critical weakness is the inability to show a theoretical guarantee of analytical results. Network properties such as degree distribution, diameter, and other spectral properties of the graphs could be explored to determine if grammar patterns exist proving that the generated structures exhibit multinomial degree distribution and or multinomial eigenvector distribution as is the case with Kronecker graphs. Moreover, we have not examined the likelihood of guaranteeing the grammars from connected graphs. Can we ensure that any generated graph, including extrapolated graphs, are connected? Analytically exploring the properties of HRG graphs such as connectivity is an important and exciting task that might yield surprising results. Other considerations include faster optimization of the graph generation step.

2.2.2 Systems: Exploring network

Development of null models will continue to be a vital application of the HRG graph model as others have done including Chung-Lu and Kronecker Product graph models. Exploiting the richness of the grammars for use in machine learning offers new paths for practical applications of HRG. Graph grammars provide a mechanism for building a repository of production rules that can fire to form or grow neural network architectures for well defined deep learning tasks. Designing a recurrent neural network controller that optimizes the selection of the rules to use during the graph generation step should recursively yield better architecture over time. Training deep neural networks is difficult, tedious and time-consuming. Methods able to automatically discover high-performing network architectures which are qualitatively better than hand-crafted solutions can enhance deep learning tasks for high-performance scientific computing in novel ways.

2.2.3 Applications: Graph Engines

In combinatorial optimization tasks, approximating algorithms offer solutions to otherwise NP-hard or NP-Complete problems, if applied to a decision problem. A pyramid algorithm would be one example. In human and computer vision literature, these algorithms are well utilized, but only more recently have been applied to problem-solving tasks. Multi-resolution graph pyramids, where the bottom of the pyramid contains the entire graph and successive layers have compressed (or contracted) graph information reduces the size of the input as we climb up the pyramid and reach a point where a combinatorial solution to the problem is feasible [5, 9, 16, 19]. How graphs contract from one pyramid level to the next depends on the class to which the network belongs. Our HRG model offers two potential applications in pyramid graph algorithms. First, to explore using hyperedge replacement for layer-to-layer contraction. Second, to investigate extracting rules set for selecting contraction algorithms according to the graph class in question.

BIBLIOGRAPHY

1. A. B. Adcock, B. D. Sullivan, and M. W. Mahoney. Tree decompositions and social graphs. *Internet Mathematics*, (just-accepted), 2016.

2. S. Aguinaga, R. Palacios, D. Chiang, and T. Weninger. Growing graphs from hyperedge replacement grammars. In *CIKM*. ACM, 2016.

3. A. Berry, J. R. Blair, P. Heggernes, and B. W. Peyton. Maximum cardinality search for computing minimal triangulations of graphs. *Algorithmica*, 39(4):287–298, 2004.

4. A. Berry, R. Pogorelcnik, and G. Simonet. Organizing the atoms of the clique separator decomposition into an atom tree. *Discrete Applied Mathematics*, 177: 1–13, 2014.

5. M. Bister, J. Cornelis, and A. Rosenfeld. A critical view of pyramid segmentation algorithms. *Pattern Recognition Letters*, 11(9):605–617, 1990.

6. V. Gogate and R. Dechter. A complete anytime algorithm for treewidth. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 201–208. AUAI Press, 2004.

7. C. Groër, B. D. Sullivan, and D. Weerapurage. Inddgo: Integrated network decomposition & dynamic programming for graph optimization. *ORNL/TM-2012*, 176, 2012.

8. H. Guo and W. Hsu. A survey of algorithms for real-time bayesian network inference. In *AAAI/KDD/UAI02 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*. Edmonton, Canada, 2002.

9. Y. Haxhimusa, W. G. Kropatsch, Z. Pizlo, A. Ion, and A. Lehrbaum. Approximating tsp solution by mst based graph pyramid. *GbRPR*, 4538:295–306, 2007.

10. B. K. Jones, S. Goldwater, and M. Johnson. Modeling graph languages with grammars extracted via tree decompositions. In *Proceedings of the 11th International Conference on Finite State Methods and Natural Language Processing*, pages 54–62, 2013.

11. S. M. Kazemi and D. Poole. Elimination ordering in lifted first-order probabilistic inference. In *AAAI*, pages 863–870, 2014.

12. A. M. Koster, S. P. van Hoesel, and A. W. Kolen. Solving partial constraint satisfaction problems with tree decomposition. *Networks*, 40(3):170–180, 2002.

13. J. Kunegis. Konect: the koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1343–1350. ACM, 2013.

14. J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

15. J. W. Liu. Modification of the minimum-degree algorithm by multiple elimination. *ACM Transactions on Mathematical Software (TOMS)*, 11(2):141–153, 1985.

16. S. F. Mousavi, M. Safayani, and A. Mirzaei. Graph pyramid embedding in vector space. In *Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on*, pages 146–151. IEEE, 2014.

17. C. Pennycu, S. Aguinaga, and T. Weninger. A temporal tree decomposition for generating temporal graphs. 2017.

18. N. Peyrard, S. De Givry, A. Franc, S. Robin, R. Sabbadin, T. Schiex, and M. Vignes. Exact and approximate inference in graphical models: variable elimination and beyond. *arXiv preprint arXiv:1506.08544*, 2015.

19. Z. Pizlo and Z. Li. Pyramid algorithms as models of human cognition. In *Electronic Imaging 2003*, pages 1–12. International Society for Optics and Photonics, 2003.

20. D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.

21. K. Shoikhet and D. Geiger. A practical algorithm for finding optimal triangulations. In *AAAI/IAAI*, pages 185–190, 1997.

22. R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on computing*, 13(3):566–579, 1984.

This document was prepared & typeset with pdfLaTeX*, and formatted with* NDdiss2$_\varepsilon$ *classfile (v3.2013[2013/04/16]) provided by Sameer Vijay and updated by Megan Patnott.*