Estimating Graph Parameters using Graph Grammars

Sourav Mukherjee SOURAV1@UMBC.EDU
Tim Oates OATES@CS.UMBC.EDU
Department of Computer Science and Electrical Engineering,
University of Maryland Baltimore County,
1000 Hilltop Circle, Baltimore 21250, Maryland, USA.

Keywords: graph grammars, grammar induction, graph mining, relational data mining

Abstract

Stochastic graph grammars are probabilistic models suitable for modeling relation data, complex organic molecules, social networks, and various other data distributions. In this paper, we demonstrate that such grammars can be used to reveal useful information about the underlying distribution. In particular, we demonstrate techniques for estimating the expected number of nodes, the expected number of edges, and the expected value of the average node degree, in a graph sampled from the distribution. These estimation techniques use the underlying grammar, which is assumed to be known, and hence do not require sampling. Experimental results indicate that our estimation techniques are reasonably accurate. Further, we present a characterization of grammars that can generate graphs that are not connected. Thus, this paper shows that once a stochastic graph grammar for a distribution of graphs has been learned, it can be used to answer several interesting and useful queries about the distribution itself, without requiring sampling.

1. Introduction

Stochastic graph grammars have recently emerged as viable probabilistic models for various types of data, such as relational data, organic molecules, social networks and so on. Such grammars compactly represent probability distributions over graphs. Stochastic graph grammars have a hierarchical, recursive structure that is amenable to interpretation by domain experts. Further, they can be used to determine the likelihood of generating a particular sample (*inference*), and also for the task of generating new samples from the distribution(*sampling*). In this paper, however, we show that the utility of graph grammars extends beyond inference and sampling, and that a graph grammar can be used to extract useful information about the distribution it represents. Given the stochastic grammar for a distribution of graphs, we present techniques that take the grammar as input, and find answers to the following interesting questions about the underlying distribution:

- What is the expected number of nodes in a graph sampled from this distribution?
- What is the expected number of edges?
- What is the expected degree of a node?
- What can we infer about the connectivity of a graph sampled from this distribution?

We show that these questions can be answered using a stochastic graph grammar, without resorting to any sampling process.

The rest of the paper is organized as follows. Section 2 presents motivation for our work. Section 3 formally defines the terms and concepts we shall use throughout this paper. Section 4 presents a brief survey of related work, and highlights the contributions of this paper. In Section 5, we investigate the answers to the questions posed above. Section 6 gives experimental results. Finally, Section 7 concludes and points out future directions.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. Motivation

In this section, we provide a few real life scenarios where it is important to be able to extract useful information from distributions of graphs. We look at two application domains: chemical databases and social networks.

In bioinformatics, the properties of large and complex organic molecules are often studied. These molecules can be represented as graphs, where labeled vertices denote atoms and an edge represents the presence of a chemical bond between two atoms (Dehaspe et al., 1998). Organic molecules often have heirarchical structure, in which groups of atoms, called functional groups, behave as a single entity. A functional group, in turn, can be composed of smaller functional groups. The recursive, hierarchical structure of a graph grammar makes it ideal for describing families of such organic molecules. Given a family of organic molecules (e.g. proteins), it might be interesting to investigate certain properties of that class of molecules, such as, the expected size of a molecule belonging to that class. In this paper, we show that such parameters can be evaluated if a graph grammar for the class of molecules is known.

In recent years, social networks such as Facebook, Orkut, and LinkedIn have become immensely popular. Social networks can be represented as graphs, where nodes represent members of the network, and edges represent acquaintance between members. These networks usually have hierarchical and recursive structures. Given a specific class of social networks (e.g. university communities), we might be interested in knowing how many friends a member is expected to have on average, how many members are expected to be in the network, whether it is possible for the network to have isolated pockets, and so on. In this paper, we show that these questions can be answered if the underlying graph grammar is known.

In the next section, we formally define the concepts related to stochastic graph grammars, that will aid in our presentation.

3. Preliminaries

In this section, we review the basic definitions and concepts pertaining to stochastic graph grammars. The presentation will follow that given in (Oates et al., 2003). Throughout this discussion, we will use the notation $G = (V, E)$ to refer to a graph with V being the set of vertices, and E being the set of edges.

Definition 3.1. *Let $G = (V, E)$ be a graph. A* hyperedge *is an ordered subset of its vertices V. Alternatively, a hyperedge of degree n can be thought of as a mapping $H : \{1, 2, ..., n\} \to V$.*

A hypergraph *is a graph that can, in addition to edges, also have hyperedges.*

Definition 3.2. *A* (hyperedge replacement) stochastic context-free graph grammar *(SCFGG) is defined as a tuple (S, N, T, P, p) where:*

- *N is the set of non-terminal symbols,*
- *T is the set of terminal symbols, disjoint from N,*
- *$S \in N$ is a special non-terminal called the start symbol,*
- *P is a set of productions,*
- *p is a probability function defined on the set of productions, such that the sum of the probabilities of all productions with the same left hand side equals* 1.

In a hyperedge replacement SCFGG, terminals are used to denote graphs without hyperedges, while non-terminals are used to label hyperedges. A production is an ordered pair (H, α), written as $H \to \alpha$, where H is a non-terminal and α is a hypergraph.

A SCFGG can be viewed as a generative model: we start with the start symbol S, and at each step we replace any non-terminal H with a graph α such that there is a production $H \to \alpha$. This process is continued until we arrive at a graph that has no non-terminal symbols. When a hyperedge H in a graph G is replaced using the production $H \to \alpha$, G is called the host graph, and α is called the subgraph. Finally, the probability of such a derivation is defined as the product of the probabilities of its productions.

The term *embedding rules* is used to describe how the subgraph is placed in the host-graph while replacing a non-terminal. In general, embedding rules for HR grammars tend to be much simpler than those for node-replacement grammars. Let the host-graph G have a hyperedge H of degree n. Then the vertices constituting H must be labeled as $1', 2', ..., n'$. Also, for every production $H \to \alpha$, α must have n vertices labeled $1, 2, ..., n$. When H is replaced by α, node j in the subgraph must be glued to node j' in the host-graph. This is the only embedding rule this paper will assume.

As an illustrative example, we consider the simple grammar shown in Figure 1, which is a grammar for all

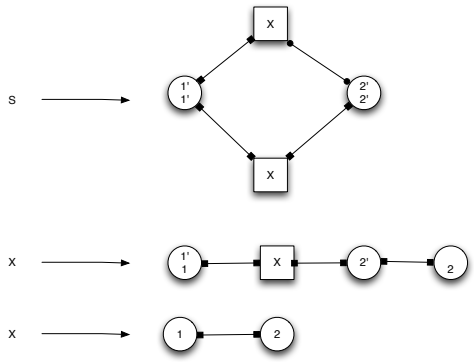

Figure 1. A grammar for generating simple cycles. Note that non terminal hyperedges are labeled with rectangles, with the non-terminal symbols written inside them. Also, the vertices of the hyperedge are enumerated using primed integers: 1', 2'. Finally, the subgraphs have vertices labeled 1,2 indicating how they will be embedded in the host-graph.

simple cycles. We shall use this grammar to illustrate our estimation techniques in later sections.

In the next section, we briefly survey work done in graph grammars and related areas.

4. Related Work

In this section, we review the literature relevant to stochastic graph grammars.

Since a graph depicts a set with a relation defined on it, stochastic graph grammars are important as tools in relational data mining. In recent years, the problem of mining information from relational data by constructing probabilistic models for the underlying distribution has received a lot of attention. Markov Logic Networks (MLN) (Richardson & Domingos, 2006) have emerged as a popular approach to the problem. Our approach, involving stochastic graph grammars, differs from MLNs in the following respects:

1. As mentioned in (Richardson & Domingos, 2006), an MLN can be constructed to represent any distribution over relational data, as long as the domain is finite. Since the schema of such data is also fixed, the probability distribution is necessarily over a finite set. Although the authors comment that it might be possible to extend MLNs to infinite domains, their work entirely deals with MLNs in finite domains. On the other hand, even simple stochastic graph grammars (such as the one in Figure 1) can represent infinite families of graphs.

2. Inference using MLNs typically involves finding the probability that a certain first order logic (FOL) formula is true, given that another FOL formula is also true (Richardson & Domingos, 2006). In the context of stochastic graph grammars, inference conventionally means computing the probability that a graph was generated by a given grammar. However, in this paper, we show that we can use a stochastic graph grammar to infer useful parameters of the distribution itself.

3. An MLN, together with a set of constants, defines a Markov network (Richardson & Domingos, 2006), whose size increases exponentially with the number of constants. Stochastic graph grammars, on the other hand, tend to be compact models for the underlying distribution, which do not have such combinatorial explosion in size.

We now turn to the problem of learning grammars from training data. In the context of string grammars, Lari and Young (Lari & Young, 1990) have presented an algorithm for learning the production probabilities, given the productions and training examples. The problem of learning the structure (the productions) has also been addressed, both for hidden Markov models (HMMs) (Bell et al., 1990; Ron et al., 1994; Stolcke & Omohundro, 1994) that can learn only regular grammars, and for the more general context free grammars (Cook et al., 1976; Stolcke & Omohundro, 1994).

Now we turn to the problem of learning graph grammars from training data. The theory of graph grammars, along with several applications, has been surveyed in (Rozenberg, 1997). Oates et al. (Oates et al., 2003) have proposed an algorithm, PEGG (Parameter Estimation in Graph Grammars), for estimating the production probabilities of a stochastic context free graph grammar, given its productions. Algorithms for learning the structure of graph grammars include SubdueGL (Jonyer et al.,) for deterministic graph grammars, and its extension to stochastic grammars presented in (Doshi et al., 2002). Both these algorithms are for node-replacement graph grammars. One of the early approaches to learning deterministic hyperedge-replacement (HR) graph grammars was presented by Jeltsch and Kerowski (Jeltsch & Kreowski, 1993). However, to the best of our knowledge, no satisfactory solution exists to the problem of learning stochastic HR graph grammars. Although one of our research goals is to formulate such an algorithm, that is not the concern of this paper.

In (Stolcke & Segal, 1994), Stolcke et al. have demon-

strated that a stochastic string grammar may be used to extract useful information about the underlying distribution of strings. In this paper, we adopt a similar approach, and demonstrate that the utility of stochastic graph grammars extends beyond inference and sampling. We show that it is possible to use a stochastic graph grammar to unravel useful and interesting information about the underlying distribution of graphs. In particular, we show that:

1. Given a graph grammar, we can estimate the expected number of vertices and edges that a graph generated from any non-terminal in that grammar will have.
2. Using the above, we can estimate the average degree of a vertex.
3. Given a graph grammar, we can infer whether that grammar can generate graphs that are not connected.

In the next section, we shall demonstrate how these estimations can be made based on a stochastic graph grammar.

5. Estimating Graph Parameters

In this section, we present techniques for estimating the expected number of nodes and edges in a graph sampled from a distribution given the grammar for the distribution. We also present a characterization of graph grammars that can generate graphs that are not connected (in this paper, we shall refer to such graphs as *unconnected* graphs).

Before presenting these techniques, we define some notation that aid our presentation.

5.1. Notation

Given a grammar G, let Z be a nonterminal in the grammar, such that there are N_Z production rules with Z on the left hand side, with probabilities $p_{Z,1}, p_{Z,2}, ..., p_{Z,N_Z}$, satisfying $\sum_{j=1}^{N_Z} p_{Z,j} = 1$. Let the j^{th} such production be of the form $Z \rightarrow \alpha_j$ where α_j is a graph with $v_{Z,j}$ vertices, $a_{Z,j}$ edges, and $h_{Z,j}$ hyperedges, labeled $Z_{j,1}, Z_{j,2}, ..., Z_{j,h_{Z,j}}$. Note that these non-terminals do not have to be all distinct; they may even be the same as Z. Finally, let D_Z denote the degree of the non-terminal Z. We will express our estimates in terms of these symbols.

We now consider the problem of estimating the expected number of nodes in any graph generated by a grammar.

5.2. Expected Number of Nodes

For any non-terminal Z, let n_Z represent the expected number of nodes in any graph obtained by expanding Z.

As an illustrative example, we refer to the simple grammar shown in Figure 1, which is a grammar for all simple cycles.

Let us assume that the three productions have probabilities given by p_S, $p_{X,1}$ and $p_{X,2} = 1 - p_{X,1}$ respectively.

Given the grammar in Figure 1, we can make the following observations (explained later):

$$\begin{aligned} n_S &= p_S(2 + (n_X - 2) + (n_X - 2)) && (1)\\ &= p_S(2n_X - 2) && (2)\\ n_X &= p_{X,1}(3 + n_X - 2) + p_{X,2}(2) && (3)\\ &= p_{X,1}(n_X + 1) + p_{X,2}(2) && (4) \end{aligned}$$

To understand how these equations arise, let us consider Equation 3. Non-terminal X in Figure 1 can expand in two ways, corresponding to the second and the third productions in the grammar. The probability terms $p_{X,1}$ and $p_{X,2}$ in Equation 3 correspond to these two scenarios. If X expands according to the third production in Figure 1, then the resulting graph will have 2 nodes; this explains the term $p_{X,2}(2)$ in the equation. However, if X expands according to the second production in Equation 1, then the resulting graph will have 3 nodes, plus the nodes arising due to the expansion of the embedded X in the right hand side of the production. When the embedded X is expanded, two nodes in the resulting subgraph (labeled 1 and 2) will be glued to the two nodes labeled 1' and 2' in the host-graph. This explains the term $p_{X,1}(3 + n_X - 2)$ in Equation 3. We can similarly explain Equation 1.

To generalize this result to arbitrary grammars, refer to the notation defined in Section 5.1. The equation for n_Z, the expected number of nodes in any graph obtained by expanding Z, is given by:

$$n_Z = \sum_{j=1}^{N_Z} p_{Z,j}(v_{Z,j} + \sum_{k=1}^{h_{Z,j}} (n_{Z_{j,k}} - D_{Z_{j,k}})) \quad (5)$$

To see how Equations 1 and 3 follow from Equation 5,consider the non-terminal S. The number of productions with S on the left-hand side is $N_S = 1$, with probability $p_{S,1} = p_S = 1$. In the right-hand side of

this production, there are $v_{S,1} = 2$ nodes, and $h_{S,1} = 2$ hyperedges, viz. $S_{1,1} = X, S_{1,2} = X$. Inserting these values into Equation 5 yeilds:

$$n_S = p_S(2 + (n_X - 2) + (n_X - 2))$$

which is exactly Equation 1.

For X, the number of productions is $N_X = 2$ with probabilities $p_{X,1}, p_{X,2}$. The first production has $v_{X,1} = 3$ nodes and $h_{X,1} = 1$ hyperedge, namely $X_{1,1} = X$. The second production has $v_{X,2} = 2$ nodes and no hyperedge, $h_{X,2} = 0$. Setting these values into Equation 5 yeilds:

$$n_X = p_{X,1}(3 + n_X - 2) + p_{X,2}(2)$$

which is exactly Equation 3.

Thus we see that for each non-terminal Z in the grammar, we will have a single linear equation, leading to a system to linear-equations with the same number of equations as the number of non-terminals.

In the next section, we use the same approach to estimate the expected number of edges in graphs obtained by expanding any non-terminal Z.

5.3. Expected Number of Edges

In this section, we develop a system of linear equations for estimating the expected number of edges in a graph obtained from any non-terminal in the grammar. As in the case of the expected number of nodes, we first explain the technique using the example grammar in Figure 1. Then, we generalize the idea to arbitrary graph grammars. We define the following notation: for any non-terminal Z, let e_Z be the expected number of edges in any graph obtained by expanding Z.

The problem of estimating the expected number of edges is different from that of estimating the expected number of nodes, in that unlike nodes, edges are not glued together when a subgraph is embedded inside a host-graph. Thus, we get the following set of equations for the grammar in Figure 1:

$$\begin{aligned} e_S &= p_S(e_X + e_X) && (6)\\ &= p_S(2e_X) && (7)\\ e_X &= p_{X,1}(1 + e_X) + p_{X,2}(1) && (8) \end{aligned}$$

To understand how these equations arise, consider Equation 8. The non-terminal X can expand in two different ways, as seen from Figure 1. If it expands according to the third producion, the number of edges is 1. This explains the $p_{X,2}(1)$ term in Equation 8. However, if it expands according to the second production, then the resulting graph has one edge, plus the number of edges obtained by expanding the embedded X. This explains the $p_{X,1}(1 + e_X)$ term in Equation 8. Equation 7 may be explained similarly.

Now, we generalize this idea to arbitrary graph grammars. Referring to the symbols defined in Section 5.1, the equation for e_Z, the expected number of edges in any graph obtained by expanding Z, is given by:

$$e_Z = \sum_{j=1}^{N_Z} p_{Z,j}(a_{Z,j} + \sum_{k=1}^{h_{Z,j}} e_{Z_{j,k}}) \quad (9)$$

Once again, we see that for each non-terminal Z in the grammar, we will have a single linear equation, leading to a system to linear-equations with the same number of equations as the number of non-terminals.

In the next section, we will show how to use these results, to estimate the average degree of any node, in a graph obtained from any non-terminal in the graph.

5.4. Expected Degree of a Node

In this section, we present two techniques for estimating the average node degree of a graph generated from a given grammar. The first technique, which we call the *Naïve Degree Estimator*, is simpler to implement and understand; the second method, which we call the *Linear Degree Estimator* is slightly more involved, but much more accurate. We present an emperical comparison of these two techniques later.

5.4.1. Naïve Degree Estimator

The average degree $\bar{d}$ of a node in a graph $G = (V, E)$ is defined as

$$\bar{d} = \frac{1}{|V|} \sum_{v \in V} d(v) \quad (10)$$

We also know that (West, 2001)

$$\bar{d} = \frac{2|E|}{|V|} \quad (11)$$

We will refer to this result as the Handshaking Lemma.

Given a non-terminal Z, let $\bar{d}_Z$ denote the expected value of the average degree of a node, of any graph obtained from Z. Then, we can estimate $\bar{d}_Z$ as:

$$\bar{d}_Z \approx \frac{2e_Z}{n_Z} \tag{12}$$

Of course, Equation 12 is only an estimate. The estimation is exact only if the number of nodes and the number of edges are independant random variables, which, in general, is not the case. This fact limits the accuracy of the Naïve Degree Estimator technique, as reflected in the experimental results. We next present the second, more involved, degree estimation technique, which also turns out to be a more accurate one.

5.4.2. Linear Degree Estimator

In this section, we introduce a technique that reduces the degree estimation problem to solving a linear system of equations, analogous to those developed for estimating the expected node and edge counts. Let $G = (V, E)$ be a graph, and let $\{V_1, ..., V_m\}$ be a partition of the vertices V. Further, let for $1 \leq i \leq m$, let G_i be the subgraph of G induced by V_i. Finally, for any graph X, let $\bar{d}(X)$ denote the average degree of a node in X. It is easy to see that:

$$\bar{d}(G) = \frac{\sum_{i=1}^{m} |V(G_i)|\bar{d}(G_i)|}{|V(G)|} \tag{13}$$

We now make the following observations:

1. For a randomly generated graph, we may not know the node counts $|V(G_i)|$ in advance. However, we can estimate these counts by the *expected* node counts, using the results given in Section 5.2.
2. If we consider the process of deriving a graph from a grammar, the subgraphs obtained by expanding the non-terminal hyperedges provide a decomposition of the vertex set, except that the decomposition is not pairwise disjoint. In particular, there may be nodes that belong to more than one such subgraph. However, the total degree of such a node would be the sum of contributions from each of the subgraphs the node belongs to.

We modify Equation 13, in view of the above observations, to arrive at the following result.

Let, for a non-terminal Z, $\bar{d}_Z$ indicate the expected average node degree of any graph derived from the non-terminal symbol Z. Recall that the average is computed over all nodes in a graph, and the expectation is computed over the distribution of the graphs. Once again, we refer to the symbols defined in Section 5.1.

The expected number of nodes in the graph α_j is given by

$$n_{Z,j} = \sum_{k=1}^{h_{Z,j}} (n_{Z_{j,k}} - D_{Z_{j,k}}) + v_{Z,j} \tag{14}$$

Let us number the vertices in α_j as $1, 2, ..., v_{Z,j}$ and let for vertex $l (1 \leq l \leq v_{Z,j})$, a_l be the number of terminal edges incident on that vertex. Then the expression for the expected average number of nodes is given by:

$$\bar{d}_Z - \sum_{j=1}^{N_Z} \sum_{k=1}^{h_{Z,j}} \frac{p_{Z,j}}{n_{Z,j}} \bar{d}_{Z_{j,k}} = \sum_{j=1}^{N_z} \sum_{l=1}^{v_{Z,j}} \frac{p_{Z,j}}{n_{Z,j}} a_l \tag{15}$$

Thus, we get a linear equation for every non-terminal Z in the grammar. By solving this linear system, we can arrive at an estimate of the expected average node degree.

In the next section, we characterize grammars that can produce unconnected graphs.

5.5. Connectivity

In this section, we present conditions a grammar must satisfy in order to generate an unconnected graph (*necessary conditions*), and also conditions that guarantee that a grammar will generate an unconnected graph (*sufficient conditions*). We begin with a definition.

Definition 5.1. *Consider a hyper-graph (a graph that has edges as well as non-terminal hyperedges) g. The* meta-graph *induced by g, denoted by $M(g)$, is a graph obtained by adding a new node X for every hyperedge X in g; further, if n is any node in g, then there is an edge between X and n in $M(g)$ if and only if n is a vertex in the hyperedge X in g.*

Note that if g has no hyperedges, then $g = M(g)$. We now state our theorem on connectivity. [1]

Theorem 5.1. *Let G be a graph grammar. A necessary condition for G to be able to generate an unconnected graph is that G must have a production $X \rightarrow \alpha$, where $M(\alpha)$ is unconnected. Further, G* must *generate an unconnected graph if either G has a production of the form $S \rightarrow \alpha$ where $M(\alpha)$ is unconneceted, or,*

[1] Proof is omitted here due to lack of space.

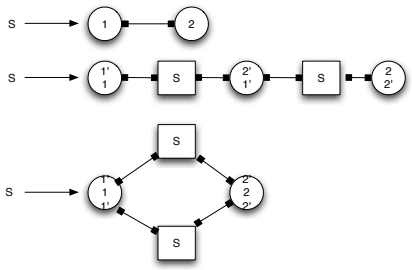

Figure 2. The series-parallel grammar.

G has a production of the form $X \to \alpha$ where $M(\alpha)$ is unconnected, and at least one connected component in $M(\alpha)$ has no gluing nodes in it.

In the next section, we present experimental results to validate our theoretical formulations.

6. Experimental Results

In this section, we present experimental results that validate our theoretical formulation. We have used two grammars, refered to as *series-parallel* (see Figure 2) and $a^n b^n c^n$ (see Figure 3). The probabilities of the productions are set as follows. For the series-parallel grammar (Figure 2) ,we let the probabilities of the first production be $p_{terminal}$, and those of the other two productions each be $(1 - p_{terminal})/2$. We vary $p_{terminal}$ from 0.60 to 1.0. For the $a^n b^n c^n$ grammar (Figure 3), let the probabilities of the second and the fourth productions each be $p_{terminal}$, and let the probabilities of the first and the third productions each be $1 - p_{terminal}$. We vary $p_{terminal}$ from 0.60 to 1.0.

For each grammar and for each setting of production probabilities, we first estimate the expected number of nodes, the expected number of edges, and the expected average node degree, using the above techniques. Then, we generate 10,000 independant samples from each grammar and compute the sample means of node count, edge count, and average node degree.

The results are presented in Table 1 for the series-parallel grammar, and in Table 2 for the $a^n b^n c^n$ grammar.

Remarks The results in Table 1 and Table 2 indicate that the estimates obtained using our technique are fairly accurate. The tables also show that as $p_{terminal}$ increases towards 1.0, the estimates get progressively closer to the corresponding sample means. This can be explained by the fact that when terminal productions have a higher probability, longer derivations become less probable, and hence, the variances in the parameters being measured (such as the number of nodes in a graph) become smaller.

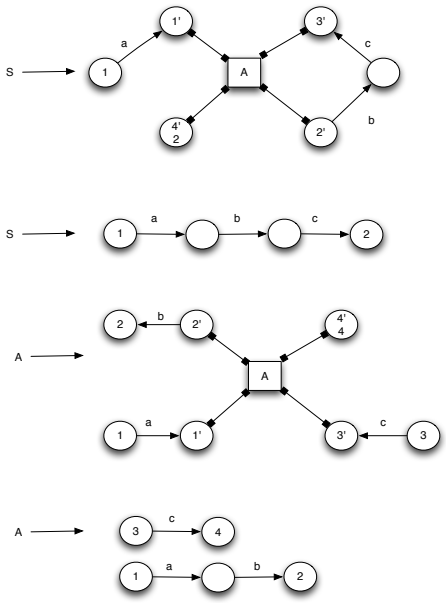

Figure 3. The grammar $a^n b^n c^n$. Note that this grammar contains directed labeled terminal edges.

Another observation we may make from the tables is that the Linear Degree Estimator is much more accurate than the Naïve Degree Estimator, as had been predicted in Section 5.4.1.

7. Conclusion

Graph grammars are useful probabilistic models for distributions over graphs because they are compact, hierarchical, and amenable to interpretation by domain experts. However, in this paper, we have demonstrated that the utility of graph grammars goes beyond elucidation of structure and generation of samples. We have presented grammar-based techniques to estimate the expected number of nodes, the expected number of edges, and the expected average node degree in a graph generated by the grammar. We have also presented a characterization of grammars that can produce graphs that are not connected.

Future directions include exploring the characterization of grammars the generate planar graphs, and applying these results to real-life domains such as relational databases, organic molecules, and social networks.

Table 1. Experimental results for the Series-Parallel grammar. Estimates using our algorithm are compared against sample means from a set of 10,000 samples. The symbols $n_S, e_S, \bar{d}_S^{\text{NAIVE}}, \bar{d}_S^{\text{Linear}}$ stand for the expected number of nodes, the expected number of edges, the expected average node degree by the Naïve estimator, and the expected average node degree by the Linear degree estimator, respectively.

Terminal Production Probability	n_S	Sample Mean	e_S	Sample Mean	$\bar{d}_S^{\text{Naive}}$	Sample Mean	$\bar{d}_S^{\text{Linear}}$	Sample Mean
0.60	2.99	2.9428	6.99	2.9869	4.6667	1.5572	1.3043	1.5607
0.65	2.5833	2.5932	4.5	2.2158	3.4839	1.4224	1.3067	1.4286
0.70	2.375	2.3768	3.25	1.7764	2.7368	1.3351	1.2706	1.3295
0.75	2.25	2.2562	2.5	1.5042	2.22	1.2465	1.2209	1.2477
0.80	2.1667	2.1619	1.9999	1.337	1.8462	1.1799	1.1691	1.1814
0.85	2.1071	2.1049	1.6428	1.2152	1.1559	1.1263	1.12	1.1303
0.90	2.0625	2.0612	1.375	1.1261	1.3333	1.0778	1.0753	1.0761
0.95	2.0278	2.0269	1.1667	1.0559	1.1507	1.0354	1.0354	1.0354
1.00	2.0	2.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 2. Experimental results for the $a^n b^n c^n$ grammar. Estimates using our algorithm are compared against sample means from a set of 10,000 samples. The symbols $n_S, e_S, \bar{d}_S^{\text{NAIVE}}, \bar{d}_S^{\text{Linear}}$ stand for the expected number of nodes, the expected number of edges, the expected average node degree by the Naïve estimator, and the expected average node degree by the Linear degree estimator, respectively.

Terminal Production Probability	n_S	Sample Mean	e_S	Sample Mean	$\bar{d}_S^{\text{Naive}}$	Sample Mean	$\bar{d}_S^{\text{Linear}}$	Sample Mean
0.60	6	6.0073	5.6667	5.0214	1.8889	1.6046	1.5815	1.6017
0.65	5.6154	5.5882	5.1538	4.6305	1.8356	1.5873	1.5722	1.5885
0.70	5.2857	5.2807	4.7143	4.2609	1.7838	1.5730	1.5625	1.5724
0.75	5.0	4.9999	4.33	3.9855	1.733	1.5584	1.5525	1.5594
0.80	4.75	4.7131	4.0	3.756	1.6842	1.5470	1.5423	1.5426
0.85	4.5294	4.5523	3.7059	3.5265	1.6364	1.5354	1.5319	1.5349
0.90	4.3333	4.333	3.4444	3.3273	1.5897	1.5219	1.5213	1.5219
0.95	4.1579	4.153	3.2105	3.1485	1.5443	1.5116	1.5107	1.5099
1.0	4.0	4.0	3.0	3.0	1.5	1.5	1.5	1.5

References

Bell, T. C., Cleary, J. G., & Witten, I. H. (1990). *Text compression*. Englewood Cliffs, NJ, USA: Prentice Hall.

Cook, C. M., Rosenfeld, A., & Aronson, A. R. (1976). Grammatical inference by hill climbing. *infsci*, *10*, 59–80.

Dehaspe, L., Toivonen, H., & King, R. D. (1998). Finding frequent substructures in chemical compounds. *KDD* (pp. 30–36).

Doshi, S., Huang, F., & Oates, T. (2002). Inferring the structure of graph grammar from data. *Proceedings of the International Conference on Knowledge Based Computer Systems (KBCS)*.

Jeltsch, E., & Kreowski, H.-J. (1993). Grammatical inference based on hyperedge replacement. *Grammatical Inference: Theory, Applications and Alternatives; 1st International Colloquium* (pp. 7/1–7/6). The Institution of Electrical Engineers.

Jonyer, I., Holder, L. B., & Cook, D. J. MDL-based context-free graph grammar induction. .

Lari, K., & Young, S. (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, *4*, 35–36.

Oates, T., Doshi, S., & Huang, F. (2003). Estimating maximum likelihood parameters for stochastic context-free graph grammars. *Proceedings of the 13th International Conference on Inductive Logic Programming* (pp. 281–298). Springer-Verlag.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107 – 136.

Ron, D., Singer, Y., & Tishby, N. (1994). The power of amnesia. *Advances in Neural Information Processing Systems* (pp. 176–183). Morgan Kaufmann Publishers, Inc.

Rozenberg, G. (Ed.). (1997). *Handbook of graph grammars and computing by graph transformations, volume 1: Foundations*. World Scientific.

Stolcke, A., & Omohundro, S. (1994). Inducing probabilistic grammars by bayesian model merging. *Proceedings of the Second International ICGI Colloqium on Grammatical Inference and Applications* (pp. 106–118). Berlin: Springer Verlag.

Stolcke, A., & Segal, J. (1994). Precise N-gram probabilities from stochastic context-free grammars. *ACL* (pp. 74–79).

West, D. B. (2001). *Introduction to graph theory (2nd edition)*. Upper Saddle River): (Prenctice Hall.