1 Introduction

1.1 Random Geometric Graphs

Random geometric graphs (RGGs) are a simple model for networks that form according to the proximity of their vertices in space. They are discussed in detail in Penrose [1], but briefly, an RGG $G_{n,r}$ over $\{1, \dots, n\}$ can be constructed by choosing n points $U_1, \dots, U_n$ uniformly from the unit cube in a metric space with arbitrary dimension, and then adding edges $\{i, j\}$ when $\|U_i - U_j\| \leq r$. These graphs are useful for studying real-world networks whose connections arise due to the physical proximity of the nodes. One is displayed in Figure 1.

RGGs have been used to study the spread of computer viruses on ad-hoc computer networks [2], which are networks automatically formed by a new class of devices that connect via short-range signals like WiFi and Bluetooth. One can imagine a class of similar applications, such as the study of highway networks or the electrical grid, or of any other sort of network that forms based on the physical space in which it is situated.

1.2 Generalizing RGGs

A simple generalization of RGGs would be to remove the assumption that edges form whenever nodes are closer than the threshold r. More generally, we can define a generalized RGG $G_{n,f}$, parametrized by a function $f : \mathbb{R}_0^+ \to [0, 1]$, again over the random points $U_1, \dots, U_n$, where edges $\{i, j\}$ appear with probability $f(\|U_i - U_j\|)$. This model is strictly more general than the one above, since a $G_{n,r}$ graph can be produced by choosing $f(x) = [\![x \leq r]\!]$, where the notation $[\![\textit{condition}]\!]$, used throughout, indicates the function that is 1 when *condition* is true and 0 otherwise. This generalization allows for graphs whose edges form with arbitrary probability according to the distance between two nodes: maybe the're more likely to form when nodes are farther away, or according to some probability distribution. Essentially any network that forms only according to distances in physical space can be produced using this model.

But we can imagine an even more general form, abstracting over the idea of space. What if nodes connected not according to their geometric distance from each other, but rather according to their separation in a graph? The topology of a graph might be able to carry information about more general structures than metric spaces. In general, such a model might be able to describe connections that happen inside graphs: for example, it might be able to predict networks of 'likes' or 'shares' that happen inside social networks, or predict which members of a social network might be likely to 'friend' or 'follow' each other given the current structure of the social network. One could also imagine applications for recommender systems: given a bipartite structure of users and, say, movies they have watched, what users are likely to watch similar movies in the future? Or, maybe such a model could predict which members of a nation's legislature might be likely to collaborate on bills, given the structure of the legislature's political hierarchy. In the following sections, we give a formalization of such a model and study some of its properties.

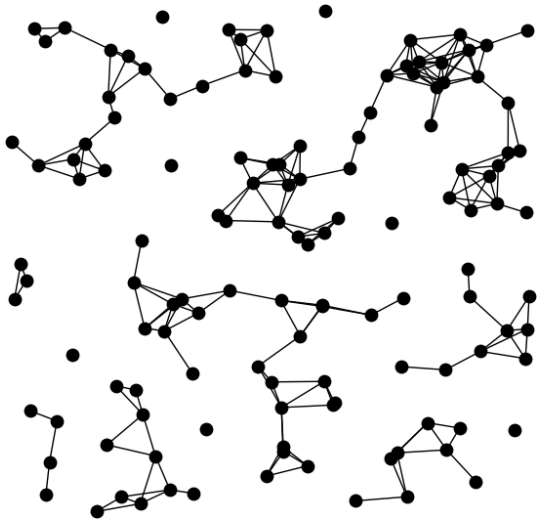

Figure 1: An RGG generated with $n = 128$, $r = 0.1$.

2 Latent Structure Random Graphs

2.1 Notation

Jumping off of the above generalized RGGs, we propose a formalization for random graphs produced using a probabilistic transformation of a latent graph structure, which will be referred to as Latent Structure Random Graphs (LSRGs). Formally, an LSRG $G_{L,f}$ is parametrized by a latent graph L and a function $f : \mathbb{Z}_0^+ \to [0, 1]$. We will let $G_{L,f}$ have the same vertices as L, but edges (i, j) in $G_{L,f}$ will each appear independently with probability $f(\sigma_L(i, j))$, where σ_L is the natural distance metric associated with L. In the following remarks, we will assume that L and $G_{L,f}$ are simple graphs, possibly with self-loops, for ease of analysis, although generalizations to directed graphs with weighted edges should be apparent.

We will refer to the normalized degree distribution of L as $p = (p_0, p_1, \dots)$, with mean degree c. It will also be helpful to name the normalized neighbor-degree distribution $r = (r_0, r_1, \dots)$, indicating the distribution of the degrees of vertices encountered after randomly choosing an edge and following it to one of its endpoints. Let c_r indicate the mean neighbor-degree. When referring to $G_{L,f}$, let p', c', r', and c'_r denote the corresponding entities. Note that we may at times use f_m to denote what should properly be written $f(m)$, for brevity.

2.2 Simple Case

To get familiarized with this model, it will be helpful to visualize a simple case, where f is the point mass at 2, e.g. $f(m) = [\![m = 2]\!]$. The following figure will show the sometimes surprising results of the LSRG model for various latent structures L.

Even for such simple f and L, the model produces some interesting results that interact closely with the underlying latent structure. For graphs of very low order (A-C), it produces the edge complement of the latent graph. For cycle graphs (D-E), it produces an isomorphic

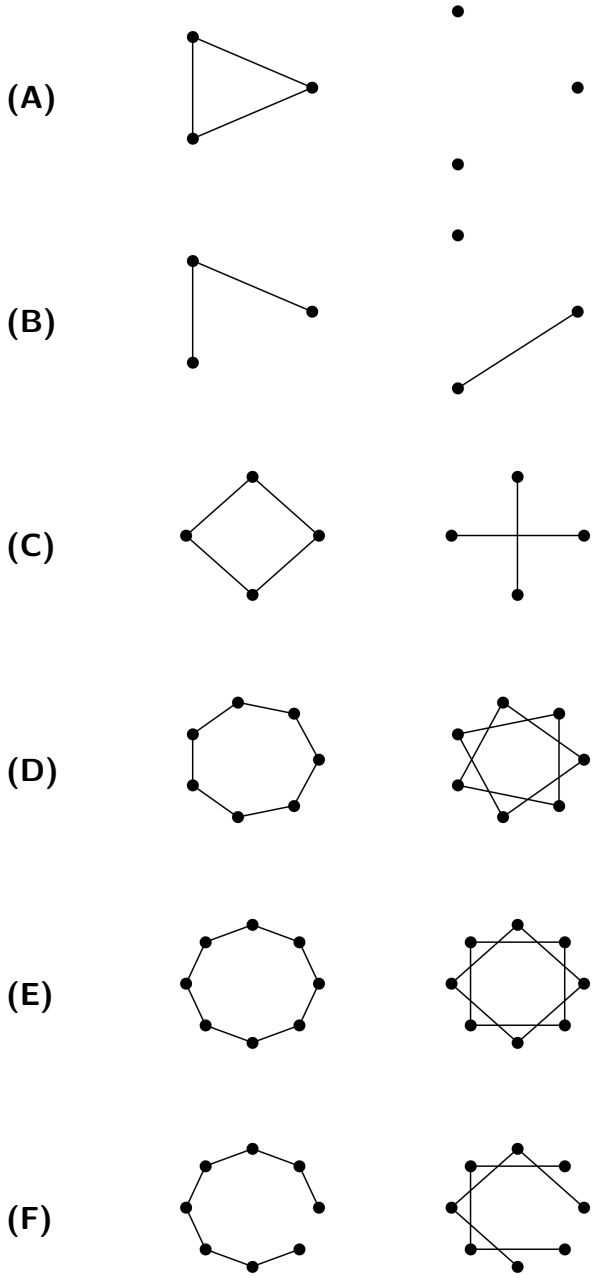

Figure 2: The result of the LSRG model with $f(m) = [\![m = 2]\!]$ on some small graphs. Latent graphs appear on the left, and their corresponding LSRGs are on the right.

cycle when the number of vertices is odd (D), or two isomorphic cycles when the number of vertices is even. For path graphs (F), it 'unzips' the path into two separate paths. From these examples, we can see that this model is capable of producing the k-fuzz [3] of L by choosing $f(m) = [\![1 \leq m \leq k]\!]$, and that in general, it produces a probabilistic variant on the idea of the k-fuzz, allowing for more general structures to form and building in randomness. We will continue with the simple $f(m) = [\![m = 2]\!]$ case in the next section to get a feel for how LSRGs behave with more complex latent graphs.

3 Motivating Example

3.1 Preferential Attachment Graphs

Since this model is meant to describe phenomena arising on real-world networks, it makes sense to study it using a latent graph with real-world structure. Two candidates are the preferential attachment model proposed by Barabási and Albert [4] and D. J. de Solla Price's cumulative advantage model [5]. Both of these models attempt to explain the growth of real-world networks in terms of 'rich get richer' phenomena, and have proposed as suitable for the study of social networks, technological networks, and others. A more in depth comparison of the two appears in Newman [6], but for now, it might make sense to begin with the simpler preferential attachment model and see if it can be used productively.

A preferential attachment graph $G_{n,c}$ is recursively grown, starting from some inital graph, and adding new nodes one at a time. Each new node gets c edges, which connect to already existing nodes with probability proportional to their degree. This process results in graphs with the so-called 'scale-free' property of having a degree distribution with a power-law tail, and intuitively exposes a possible mechanism for the presence of that sort of scaling in real-world systems. We will consider the f discussed above, and attempt to see what properties we can compute for $G_{L,f}$ when L is a tree generated by preferential attachment.

3.2 Simple Case, Revisited

We would like to study the case where L is a preferential attachment graph with $c = 1$, which naturally produces recursive trees, in the limit as n grows. Finding properties of LSRGs produced from such L in the special case $f(m) = [\![m = 2]\!]$ reduces to characterizing the grandchildren of a given node when that node is considered the root of the tree. For example, the mean degree c' of $G_{L,f}$ is simply the expected number of grandchildren of a random node in L.

To find c', we will need to use certain facts about L. Preferential attachment graphs are well studied, and in particular, results for the degree distribution p in the limit of large n have been obtained, taking the form $p_k \sim k^{-3}$ in the limit of large k, or more precisely

$$p_k = \frac{2c(c+1)}{k(k+1)(k+2)} \ ,$$

as shown by Bollobás et. al. [7]. When $c = 1$, that comes out to

$$p_k = \frac{4}{k(k+1)(k+2)} \ .$$

Additionally, Fotouhi and Rabbat [8] have given an expression for the normalized conditional degree distribution $p(\ell \mid k)$, which denotes the probability that a random neighbor of a node with degree k will have degree ℓ, which in the $c = 1$ case is given by

$$p(\ell \mid k) = \frac{k+2}{k\ell(\ell+1)} - \frac{6}{k\ell}\frac{\binom{k+\ell-2}{\ell-1}}{\binom{k+\ell+2}{\ell}} \ .$$

Since the degree of a node in $G_{L,f}$ is its number of grandchildren in L, we ought to be able to use the above to solve for $G_{L,f}$'s degree distribution. To find that distribution p', we can use the law of total probability:

$$\begin{aligned} p'_i &= \mathbb{P}(\text{random } v \in V(L) \text{ has } i \text{ granchildren}) \\ &= \sum_{k=0}^{\infty} p(i \mid k) p_k \end{aligned}$$

This procedure could be carried out in practice for real-world observed graphs, but it is clearly not a general enough method for use in most cases. Even in this case, where f's support is very small, evaluating this sum analytically would be hard, and won't generalize to different f. So, from this example we can infer that we have a need for better tools.

We can also get a hint that the preferential attachment model might not be the best one to use by attempting to calculate the expected conditional degree. In the following, let D_i be the random variable denoting the degree of vertex i in $G_{L,f}$, and d_i be i's degree in L. Then we can say

$$\mathbb{E}[D_i \mid d_i = k] = k \sum_{\ell=0}^{\infty} (\ell \, p(\ell \mid k) - 1) \ ,$$

because once we know $d_i = k$, D_i will be given by the the degrees of i's k neighbors. But, substituting our expression for $p(\ell \mid k)$, we can see immediately that this sum diverges because of $p(\ell \mid k)$'s heavy tails. In other words, in the limit of large n, there might exist a node that will connect to a substantial fraction of the rest of the graph. In Fig. 3, this effect is shown on a small graph. The dense center of the preferential attachment tree, filled with nodes whose degrees are in the heavy tail of its degree distribution, becomes a tangled web of edges since each has so many neighbors of neighbors.

The problem is that the Barabási and Albert model produces graphs with too-heavy tails, which cannot be adjusted. Luckily, Price's cumulative advantage model provides a way to adjust the decay rate of its asymptotic degree distribution, while still retaining scale-free structure.

3.3 Cumulative Advantage Graphs

In what follows, we will make use of a slightly modified version of Price's cumulative advantage model. The main differences are that we will erase edge directions once the graph is grown, and that the out-degree of nodes is fixed instead of random. To construct such a graph $G_{n,\alpha,c}$, after choosing a smoothing parameter α and an out-degree c, and starting from

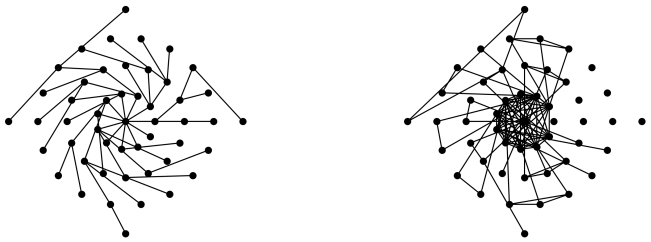

Figure 3: An LSRG $G_{L,f}$ (right) where $f(m) = [\![m = 2]\!]$ and L (left) is a tree generated by preferential attachment $n = 50$.

some initial graph, recursively add nodes. After a node has been added, add edges from it to c other nodes, where the other nodes are selected independently with probability

$$\frac{d_i^{(\text{in})} + \alpha}{n(\bar{d}^{(\text{in})} + \alpha)} ,$$

which is the add-α smoothed version of the Barabási-Albert process. Once the graph is constructed, erase edge directions. Again, setting $c = 1$ produces trees.

These graphs are also relatively well studied. As Price showed, in the limit of large n and when $c = 1$, their in-degree distribution looks like

$$p_k = \frac{\alpha + 1}{2\alpha + 1} \prod_{j=1}^{k-1} \frac{\alpha - 1 + j}{2\alpha + 1 + j} ,$$

which looks like $p_k \sim k^{-2-\alpha}$ as k grows. However, there seems to be no expression like for the conditional degree distribution, like there was above. Without that information, we will have to use new tools, which are developed in the next section.

4 Degree Properties and Generating Functions

4.1 Molloy and Reed Random Graphs

Since we have no information about the conditional degree distribution of Price's model graphs, it makes sense to assume that for any neighbors $i, j \in V(L)$, their degrees are independent. This setup is equivalent to assuming that we are working with graphs selected uniformly at random from all graphs with L's degree sequence, or in other words, the generalized random graphs studied by Molloy and Reed [9, 10]. These graphs are as random as possible given their degree sequence, meaning that edges appear independently once it is known. Our procedure will be to generate Price trees with $\alpha = 2$, to get a degree distribution whose tails decay sufficiently quickly, and then to generate a Molloy and Reed random

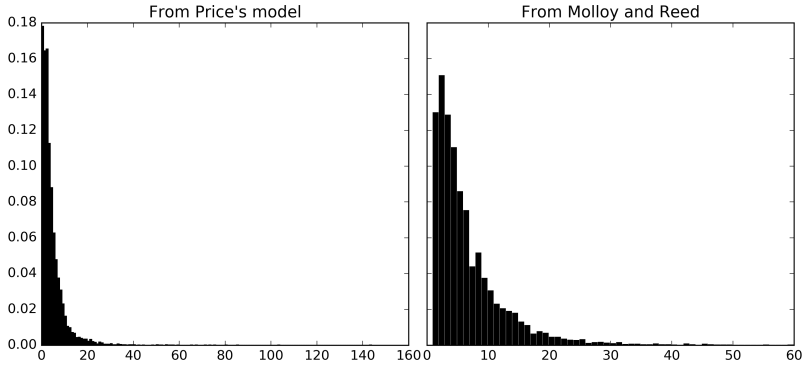

Figure 4: Degree distributions for $G_{L,f}$ where $f = [\![m = 2]\!]$. On the left, L is a tree generated with Price's model. On the right, L is the giant component of a Molloy and Reed random graph generated from the Price tree's degree sequence. Both have $n = 10,000$. Because it is only the giant component, there are no zero-degree nodes and significantly fewer low-degree nodes, but the middle sections of these two pmfs are comparable.

graph over the tree's degree sequence. To keep the graph connected and aid analysis, we will take just the giant component from that random graph, and use it as the latent graph L to be fed into our model. Figure 4 compares the degree distributions of the $G_{L,f}$ that result when doing this procedure and using the tree from Price's model without going through the Molloy and Reed graph.

4.2 Probability Generating Functions and Branching Processes

In the Molloy and Reed setting, many useful tools become available, among them the formalization of branching processes and probability generating functions, for instance as they are used in Newman *et. al.* [11] in the study of cluster growth. In that paper, the authors study the random variable representing the size of the component in which a random node resides. Let Z be the random variable describing this quantity, and let v be the random node in question. The authors study component size by imagining component growth as a branching process starting at v. Let Z_m refer to the (random) number of children in the mth generation of the branching process. Then, he zeroth generation of the branching process always has one node, namely v, giving $Z_0 = 1$. The first generation consists of v's neighbors. Since v is a random node, that means that Z_1 is distributed according to the pmf p. The second generation consists of Z_1 branches, each of which will produce a number of children drawn from the graph's neighbor-degree pmf, which we called r. In the Molloy and Reed setting, r can be shown to be given by

$$r_k = \frac{k+1}{c} p_{k+1} \ .$$

Subsequent generations of the branching process also have the number of children in each of their branches drawn from r. Our goal is to study the Z_m and see if we can figure out how many neighbors v will have in $G_{L,f}$ given $f(m)$ and Z_m.

To do so, we will make some use of generating functions. Here, we will introduce some of the properties of generating functions that were the most useful in this work, but we direct interested readers to Wilf's thorough text [12]. For any pmf $a = (a_0, a_1, \dots)$, we say that a is 'generated' by the function

$$g_a(s) = \sum_{i=0}^{\infty} a_k s^k \ ,$$

because we can take derivatives to find that the probability a_k is given by

$$a_k = \frac{1}{k!} g_a^{(k)}(0) \ ,$$

where $g_a^{(k)}(s)$ denotes the kth derivative of g_a with respect to s. We will also sometimes say that g_a generates random variables A distributed according to a, and refer to g_a as g_A. Note also that since a is a pmf, $g_a(1) = 1$. Additionally, taking a single derivative and evaluating at $s = 1$ gives

$$g_a'(s)|_{s=1} = \sum_{k=1}^{\infty} k a_k s^{k-1} \Bigg|_{s=1} = \sum_{k=0}^{\infty} k a_k \ ,$$

which is exactly the expected value of a random variable distributed according to a. The last and most interesting property of pgfs that we will use here is that for arbitrarily distributed and independent random variables N and $X_1, X_2, \dots \sim_{\text{iid}}$, where N is generated by g_N and the X_i are all generated by the same pgf g_X, the random variable

$$S = \sum_{i=1}^{N} X_i$$

is generated by the composition of g_N and g_X, e.g.

$$g_S(S) = g_N(g_X(S)) \ .$$

In the following, we will use $\circ$ to denote function composition, and let g_a^n be $g_a \circ g_a \circ \cdots \circ g_a$, n times. For convenience, we will say that g_a^0 is the identity function, which turns out to generate the point mass pmf centered at 1.

With this notation in place, we can elaborate on the study of component growth presented in [11]. It will be useful for us to understand what the generating functions g_{Z_m} of the generations Z_m are in order to understand the structure of our latent structure graphs. First, note that since there is always one node in the zeroth generation, it is simply generated by the identity function $g_{Z_0}(s) = s$. Since $Z_1 \sim$ pmf p, $g_{Z_1} = g_p$. From this point on, the size of a future generation Z_m is determined by Z_{m-1} branches, each of which independently realizes some children according to r. In other words, if $N_1, N_2, \dots$ are independent random variables distributed according to r, then

$$Z_m = \sum_{i=1}^{Z_{m-1}} N_i.$$

So, we can use the property of generating functions described above, which gives the recurrence relation

$$g_{Z_m} = g_{Z_{m-1}} \circ g_r.$$

This relation is easily solved given the initial conditions above, yielding

$$g_{Z_m} = g_p \circ g_r^{m-1}.$$

Here, we can compute the expected number of nodes in any generation, $\mathbb{E}[Z_m] = g'_{Z_m}(1)$, starting with the first few, after which a pattern will become apparent.

$$\begin{aligned} g'_{Z_1}(s) &= g'_p(1) = c \\ g'_{Z_2}(s) &= g'_p(g_r(1))g'_r(1) = c_r c \\ g'_{Z_3}(s) &= g'_p(g_r^2(1))g'_r(g_r(1))g'_r(1) = c_r^2 c \end{aligned}$$

In general, we can see that the chain rule for differentiation will produce

$$\mathbb{E}[Z_m] = c_r^{m-1} c \ ,$$

which for large m will always either go to zero or diverge to infinity, depending on the value of c_r. At this point, we have laid the groundwork for further investigation of LSRGs.

4.3 Degree Properties of LSRGs

We can think of a random node v's degree D_v in $G_{L,f}$ similarly to how we think of the growth of v's component, in terms of a branching process. The analogy works because v might have an edge to each of the nodes in the mth generation of the branching process, independently with probability $f(m)$. So, removing the randomness associated with f, D_v is completely determined by the Z_m.

Let $D_{v,m}$ be the random variable indicating how many neighbors v gets in $G_{L,f}$ from the mth generation of the branching process, or in other words, the number of v's neighbors in $G_{L,f}$ that were distance m from v in L. There will be Z_m candidates, each of which is accepted or rejected according to a coin toss with probability $f(m)$. So, we can describe $D_{v,m}$ by introducing new random variables for the possible edges $E_{m,1}, E_{m,2}, \ldots \sim_{\text{iid}} \text{Ber}(f(m))$, which gives

$$D_{v,m} = \sum_{i=0}^{Z_m} E_{m,i}.$$

We can use our composition trick again to obtain

$$g_{D_{v,m}} = g_{Z_m} \circ g_{E_m},$$

where g_{E_m} is the well known generating function for Bernoulli random variables

$$g_{E_m}(s) = 1 - f(m) + f(m)s \ .$$

At this point, even without solving for g_{D_v}, we can already estimate its expected value. Applying linearity of expectation and using the pgf expression for expected value, we can write

$$\begin{aligned}\mathbb{E}[D_v] &= \sum_{m=0}^{\infty} \mathbb{E}[D_{v,m}] \\ &= \sum_{m=0}^{\infty} g'_{D_{v,m}}(1) \\ &= \sum_{m=0}^{\infty} g'_{Z_m}(g'_{E_m}(1)) g'_{E_m}(1)\end{aligned}$$

Here, pause to note that $g'_{E_m}(1) = f(m)$, which leaves

$$\begin{aligned}\mathbb{E}[D_v] &= \sum_{m=0}^{\infty} g'_{Z_m}(1) f(m) \\ &= \sum_{m=0}^{\infty} c_r^m c f(m) \\ &= c \sum_{m=0}^{\infty} c_r^{m-1} f(m) \\ &= \frac{c}{c_r} g_f(c_r) \ .\end{aligned}$$

Note that g_f is not a probability generating function, but just the regular generating function for the sequence $f = (f(0), f(1), \dots)$. This brief expression says a good deal about the behavior of our model, in large part because it shows what kind of f will be 'well-behaved' for a given L, using the well-developed theory of convergence and infinite series. For $c_r < 1$, f does not have to decay so quickly, but for $c_r > 1$, which is the situation we are interested in, we need f to drop off very fast indeed. We are concerned with the $c_r > 1$ case because it corresponds to a supercritical branching process for component growth, and we are studying the giant component of a graph, which indicates that the process would have to be supercritical. Note that c_r can be expressed simply in terms of generating functions, with $c_r = \frac{g''_p(1)}{g'_p(1)}$.

5 Degree Properties with Convolution Approach

5.1 Branching as Convolution

We begin this alternate approach with a reexpression of the branching process above in terms of convolutions of random variables. Recall that $Z_0, Z_1, \dots$ indicate the random quantities of nodes at each level of the branching process, and that $Z_0 = 1$, Z_1 is drawn from p, and that for $m > 1$, Z_m is built from Z_{m-1} iid samples from r. Let $z_0, z_1, \dots$ be the pmfs for these random variables. We can observe that z_0 is the point pmf at 1, and that $z_1 = p$. From

these initial conditions, z_m can be specified recursively, given z_{m-1} and for $m > 1$ as follows, using the law of total probability:

$$z_m(i) = \sum_{j=0}^{\infty} \mathbb{P}(Z_m = i \mid Z_{m-1} = j)\mathbb{P}(Z_{m-1} = j) \ .$$

But recall that given $Z_{m-1} = k$, Z_m is the sum of j random variables $R_1, \ldots, R_j \sim_{\text{iid}} r$. So, we can calculate the pmf of the sum of these random variables by the j-fold convolution of their pmf r. Let $r^j(i)$ denote the probability that this sum equals i. Then we can write

$$z_m(i) = \sum_{j=0}^{\infty} r^j(i) z_{m-1}(j) \ ,$$

after substituting $z_{m-1}(j)$ for $\mathbb{P}(Z_{m-1} = j)$. For convenience, let r^0 be the point pmf at zero. This will help in simulation, where r^0 might arise as the pmf of how many kids a generation might have given that it just had none, which should always give zero again.

Having the pmfs z_m is helpful, but even better would be to have a pmf for the total $Z = \sum_m Z_m$. To do so, it will be helpful to observe that when calculating $\mathbb{P}(Z = k)$, only the first k generations matter. Z_0 is always 1, and if a Z_m is ever 0, the branching process has to stop. That means that to accumulate $Z = k$, it will take at most k generations, summing from Z_0 out to Z_{k-1}, as long as $Z_k = 0$. It will help to introduce some notation. Let

$$T_k = \sum_{m=0}^{k} Z_m$$

indicate the total size of the branching process out to generation k, and define

$$\tau_m(i, j) = \mathbb{P}(Z_m = i, T_m = j) \ .$$

Then, by the argument above, we can write

$$\mathbb{P}(Z = k) = \mathbb{P}(Z_k = 0, T_k = k) = \tau_k(0, k) \ .$$

With this formalization in place, we can go back to LSRGs with some idea of where we will be heading.

5.2 LSRGs with Convolution

As we observed earlier, given that $Z_m = j$, $D_{v,m}$ becomes a sum of j iid Bernoulli random variables, each with success probability $f_m \equiv f(m)$. In other words, the following conditional pmf is that of a Binomial random variable:

$$\mathbb{P}(D_{v,m} = k \mid Z_m = j) = \binom{j}{k} f_m^k (1 - f_m)^{j-k} \ .$$

This relation is helpful, because we have the pmfs for each Z_m from above. That means that we can use the law of total probability again, to define the new sequence of pmfs

$$\begin{aligned} d_m(k) &\equiv \mathbb{P}(D_m = k) \\ &= \sum_{j=0}^{\infty} \mathbb{P}(D_{v,m} = k \mid Z_m = j)\mathbb{P}(Z_m = j) \\ &= \sum_{j=0}^{\infty} \binom{j}{k} f_m^k (1 - f_m)^{j-k} u_m(j) \ . \end{aligned}$$

At this point, we can already produce a naive estimate for the expected mean degree $\mathbb{E}[D_v]$, by ignoring cross-generation effects and summing over the expectations of each of these pmfs. Mean degrees calculated with this estimate appear in Fig. 7.

Less naively, with these results in place, we can begin to calculate the pmf

$$d(k) \equiv \mathbb{P}(D_v = k) \ .$$

Building off of the random variables T_m defined above, define new random variables $U_0, U_1, \ldots$ like

$$U_m = \sum_{k=0}^{m} \sum_{\ell=1}^{Z_k} E_{m,\ell} \ ,$$

where $E_{m,\ell} \sim_{\text{iid}} \text{Bernoulli}(f_m)$ as above. Also, let

$$\omega_m(i,j) = \mathbb{P}(Z_m = i, U_m = j) \ .$$

Thanks to the Markovity of Z_{m+1} given Z_m, this expression yields the following recurrence for ω_{m+1}, by the law of total probability:

$$\begin{aligned} \omega_{m+1}(i,j) &= \mathbb{P}(Z_{m+1} = i, U_{m+1} = j) \\ &= \sum_{k=0}^{\infty} \sum_{\ell=0}^{\infty} \mathbb{P}(Z_{m+1} = i, U_{m+1} = j \mid Z_m = k, U_m = \ell)\, \mathbb{P}(Z_m = k, U_m = \ell) \\ &= \sum_{k=0}^{\infty} \sum_{\ell=0}^{\infty} \mathbb{P}(U_{m+1} = j \mid U_m = \ell, Z_{m+1} = i)\, \mathbb{P}(Z_{m+1} = i \mid Z_m = k)\, \omega_m(k,\ell) \\ &= \sum_{k=0}^{\infty} \sum_{\ell=0}^{\infty} \mathbb{P}(U_{m+1} = j \mid Z_{m+1} = i)\, r^{\ell}(j)\, \omega_m(k,\ell) \\ &= \sum_{k=0}^{\infty} \sum_{\ell=0}^{\infty} \binom{i}{j} f_m^i (1 - f_m)^{j-i}\, r^{\ell}(j)\, \omega_m(k,\ell) \ . \end{aligned}$$

To justify the above, note that the third and fourth steps follows from the second by substituting ω_m for the joint distribution of Z_m and U_m, and by breaking the joint distribution of Z_{m+1}, U_{m+1} when conditioned on Z_m, U_m up using the fact that U_{m+1} only depends on Z_{m+1} and Z_{m+1} only depends on Z_m. We use the result from above that the distribution of Z_{m+1} given that $Z_m = \ell$ is r^{ℓ}, and that U_{m+1} given Z_{m+1} is Binomial.

Note that we have a basis from which each ω_m can be constructed. We know that $\omega_0(i,j) = \mathbb{P}(Z_0 = i, U_0 = j)$ can only be nonzero at $i = 1$ and $j = 0$ or $j = 1$, since the zeroth generation of the branching process always has $Z_0 = 1$, leaving U_0 Bernoulli with success probability f_m. So we have:

$$\omega_0(i,j) = \begin{cases} 1 - f_m & (i,j) = (0,0) \\ f_m & (i,j) = (0,1) \\ 0 & \text{otherwise} \end{cases} .$$

Plugging this into the recurrence above, any ω_m can be obtained.

This expression is nice to have, but it is not so clear how to get from it to $d(k)$. In the last section, we were able to derive the clean expression $\mathbb{P}(Z = k) = \tau_k(0,k)$ based on the insight that the branching process can take a maximum k generations, with the last one extinct, to produce k total children. In the current setting, however, the branching process can take as long as it likes to produce $D_v = k$, because each generation only has some probability of contributing neighbors to v in $G_{L,f}$.

However, we do know that as long as the branching process continues, each generation has some nonzero probability of contributing to D_v, as long as f's support goes out far enough. Or, in other words, $\mathbb{P}(D_{v,m} > 0 \mid Z_m > 0) > 0$ whenever $f(m) > 0$. So, $\mathbb{P}(U_m = k)$ should be a (not strictly) monotone increasing quantity in m. Since it is bounded from above, we know it has a limit, so we can say

$$d(k) = \lim_{m \to \infty} u_m(k) ,$$

if we define u_m to be the marginal distribution

$$u_m(k) \equiv \sum_{\ell=0}^{\infty} \mathbb{P}(U_m = k, Z_m = \ell) = \sum_{\ell=0}^{\infty} \omega_m(k,\ell) .$$

6 Results From Simulation

6.1 Notes on Simulation

The simulations below were carried out in Python, making use of the scientific Python stack [13] and the NetworkX [14] package. All code used to produce the figures in this text has been released on GitHub [15], including the main function for creating LSRGs, as well as some fast routines for making Price's model trees and Molloy and Reed random graphs, and for finding shortest pairwise distances in trees using Tarjan's offline lowest common ancestors algorithm [16, 17]. Unfortunately, that optimization is not available for finding pairwise distances in general graphs, which means that making LSRGs from Molloy and Reed random graphs tends to be slow.

When computing the infinite sums above, especially when computing the estimates for the mean degree and the degree distribution for a given real graph, not the infinite graph imagined by the branching process intuition, our routines tend to only carry those sums out to the diameter of the graph, or to the end of f's support, whichever is least. The intuition

for this decision is that LSRGs generated from these graphs will not have pairs of nodes with distance greater than the diameter, so such pairs ought not to contribute to estimated degree statistics. Also, if f only has support going from zero to some number k, the LSRG will not care about nodes separated further k steps in the latent graph. We will later use this fact to help in cases where algorithms' complexity scales poorly with the graph's diameter.

6.2 Figures and Discussion

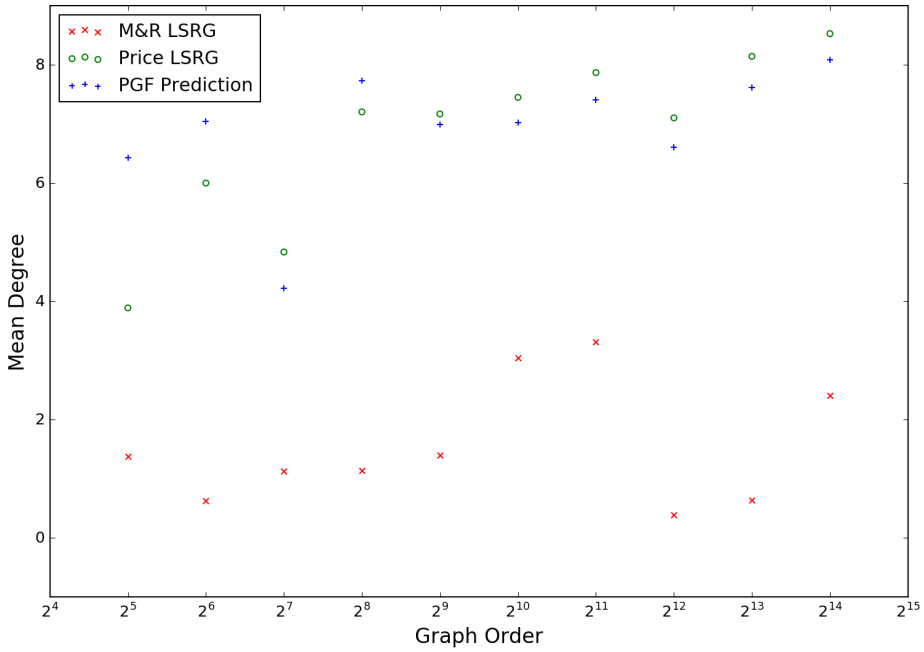

Figure 5: Observed mean degrees for LSRGs produced using both latent Price trees and latent Molloy and Reed graphs targeting those trees, plotted with the PGF estimate for mean degree discussed in 4.3. Both LSRGs are generated using $f = (0, 0.5, 0.25, 0.125, 0, 0, \dots)$.

Fig. 5 is both encouraging and discouraging. On the one hand, it does seem that the PGF estimate for mean degree $\frac{c}{c_r} g_f(c_r)$ does converge with the mean degrees observed in representative Molloy and Reed LSRGs. On the other hand, the predictions are wildy inaccurate when it comes to the Price LSRGs, whose mean degree is consistently smaller. This disparity suggests that the distance properties of the Price latent graph diverge significantly from those of the Molloy and Reed latent graph, which is confirmed in Fig. 6. There is a range of literature discussing these sorts of structural differences, including [18, 19, 20].

Studying the distance matrix for the LSRG generated from the price tree, though, we do observe some interesting effects. The general characteristic of smaller distances in the upper left (nodes added earlier in the Price process) is maintained, but there are also many unreachable nodes. In more detail, a white matrix pixel indicates infinite distance, but notice that they always form lines which meet on the main diagonal. Each pair of white bands that crosses on the main diagonal indicates an unreachable node. Comparing this matrix with its

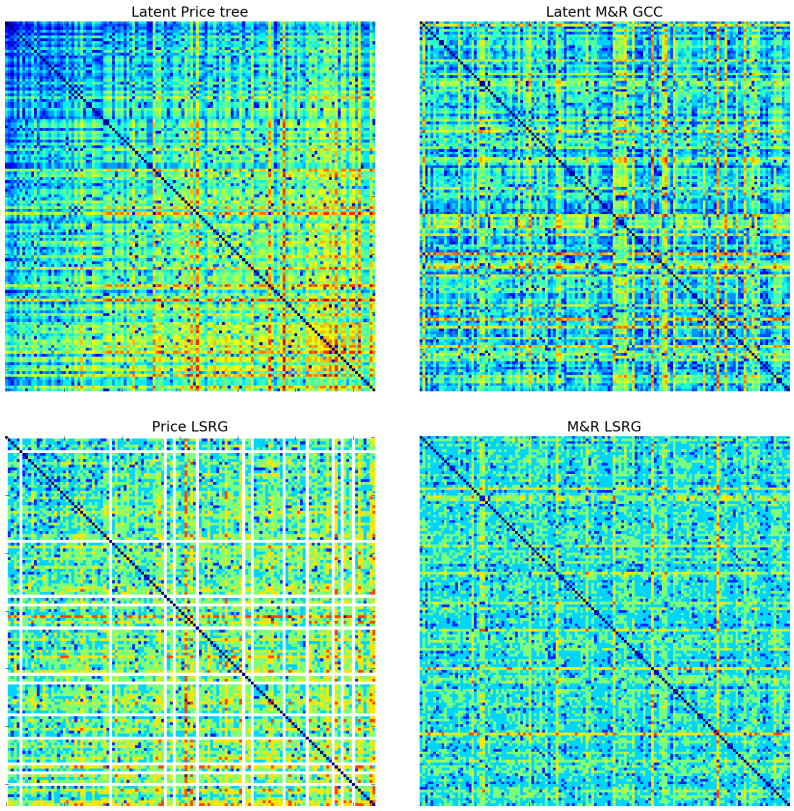

Figure 6: Distance matrices for latent graphs (top row; Price tree left, Molloy and Reed giant component right) and LSRGS generated from those latent graphs (bottom row). Cooler colors represent shorter pairwise distances, hotter colors represent longer ones, and white bands indicate unreachable nodes. All graphs have order $n = 128$.

Molloy and Reed counterpart, we can observe that it seems much more structured, much less 'mixed,' and that it has a much higher variance over distances. So, as should be expected, it seems that the Molloy and Reed process erases a lot of the community or hierarchy structure that was present in the Price tree. That said, some of the patterns that were present in the Molloy and Reed latent graph are maintained in its LSRG, looking similar but 'fuzzier' after the transformation.

Fig. 7 shows the naive estimate for mean degree using convolutions, which seems not to converge like the pgf estimate did. The difference indicates that intergenerational dependence is a significant factor, and that ignoring it like this naive estimator does is not the best approach. It might be that the u_m estimator will fix this problem, but that will be left for future work. Across estimators, it should be noted that they only make sense as the graph gets large, and when f's support does not approach the fringes of the graph, where the branching process intuition breaks down.

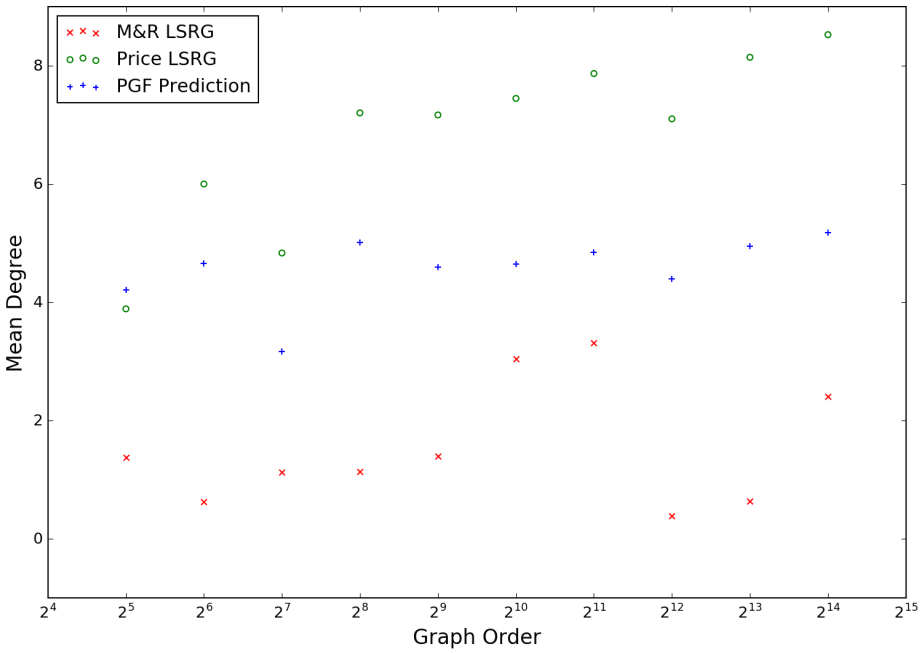

Figure 7: Observed and predicted mean degrees for the same exact graphs as in 6, this time using the naive estimate for mean degree given in the convolution section above.

References

[1] M. Penrose, "Random geometric graphs," *Oxford Studies in Probability*, vol. 5, 2003.

[2] M. Nekovee, "Worm epidemics in wireless ad hoc networks," *New Journal of Physics*, vol. 9, no. 6, p. 189, 2007.

[3] P. G. Doyle and J. L. Snell, "Random walks and electric networks," *Carus Mathematical Monographs*, vol. 22, 1984.

[4] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, oct 1999.

[5] D. J. de Solla Price, "Networks of scientific papers," *Science*, vol. 149, pp. 510–515, jul 1965.

[6] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, pp. 167–256, jan 2003.

[7] B. Bollobás, O. Riordan, J. Spencer, and G. Tusnády, "The degree sequence of a scale-free random graph process," *Random Structures & Algorithms*, vol. 18, pp. 279–290, apr 2001.

[8] B. Fotouhi and M. G. Rabbat, "Degree correlation in scale-free graphs," *The European Physical Journal B*, vol. 86, dec 2013.

[9] M. Molloy and B. Reed, "A critical point for random graphs with a given degree sequence," *Random Structures & Algorithms*, vol. 6, pp. 161–180, mar 1995.

[10] M. Molloy and B. Reed, "The size of the giant component of a random graph with a given degree sequence," *Combinatorics, Probability and Computing*, vol. 7, pp. 295–305, sep 1998.

[11] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Physical Review E*, vol. 64, jul 2001.

[12] H. S. Wilf, *generatingfunctionology: Third Edition*. A. K. Peters/CRC Press, 2005.

[13] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online; accessed 2017-02-01].

[14] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference* (G. Varoquaux, T. Vaught, and J. Millman, eds.), (Pasadena, CA USA), pp. 11 – 15, 2008.

[15] C. Windolf, "Latent structure random graphs." Source code for figures appearing in this paper, available at `https://github.com/cwindolf/capstone`.

[16] R. E. Tarjan, "Applications of path compression on balanced trees," *Journal of the ACM*, vol. 26, pp. 690–715, oct 1979.

[17] H. N. Gabow and R. E. Tarjan, "A linear-time algorithm for a special case of disjoint set union," in *Proceedings of the fifteenth annual ACM symposium on Theory of computing - STOC 1983*, Association for Computing Machinery (ACM), 1983.

[18] J. H. H. Grisi-Filho, R. Ossada, F. Ferreira, and M. Amaku, "Scale-free networks with the same degree distribution: Different structural properties," *Physics Research International*, vol. 2013, pp. 1–9, 2013.

[19] S. Dommers, R. van der Hofstad, and G. Hooghiemstra, "Diameters in preferential attachment models," *Journal of Statistical Physics*, vol. 139, pp. 72–107, jan 2010.

[20] R. van der Hofstad, G. Hooghiemstra, and D. Znamenski, "Distances in random graphs with finite mean and infinite variance degrees," feb 2005.