Vertex Censored Stochastic Kronecker Product Graphs

Adam Guetz

December 10, 2010

Abstract

Stochastic Kronecker Product Graphs are an interesting and useful class of Generative Network Models. They can be fitted using a fast Maximum Likelihood Estimator and reproduce many important statistical properties commonly found in large real-world networks. However, they suffer from an unfortunate drawback: the need to pad the Stochastic Kronecker Product Graph with isolated vertices. To address this issue, we propose the use of a *Vertex Censored* Stochastic Kronecker Product Graph model. Using a Sequential Importance Sampling scheme, we demonstrate that this class of models can be fit in similar amount of computation time and give significant improvements in fit as measured by Maximum Likelihood.

An interesting family of generative models for graphs is *Stochastic Kronecker Product Graphs*[8]. What makes this interesting is that Kronecker graphs seem to be appropriate for modeling many real world networks, as they can mimic several properties observed to be prevalent. For example, they are shown to have *multinomial degree distribution*, which with properly chosen parameters can produce heavy-tails. Give each vertex in the initiator graph self-loops, and Kronecker graphs can be made to have (small) constant diameter. Eigenvalues can be shown to be related to the degree distribution, and can be made heavy tailed. As noted in [5], Kronecker Product Graphs typically encapsulate *core-peripherery* type networks due to the recursive nature of the Kronecker Product Graph formulation. Many social, informational, and biological networks are known to have *dense cores*, making Stochastic Kronecker Product Graph networks appropriate for modeling. And perhaps most importantly, Stochastic Kronecker Product Graphs can be fit using a *linear* ($O(E)$) time algorithm via a Maximum Likelihood Estimation algorithm called KronFit developed by Leskovec et al [6].

As defined in [6], Kronecker Product Graphs start with an *initiator matrix* $K_k \in \mathcal{R}^{N_1 \times N_1}$, we then form the Kronecker Product Graph adjacency matrix

$$K_k = \otimes^k K_1 \tag{1}$$

where $\otimes$ is the *matrix Kronecker product*. See [5] for definitions and many interesting properties. As a consequence of this construction, the Kronecker

Product Graph matrix K_k has N_1^k rows and columns. To make this into a graphical model, we interpret the matrix entry $K_k(i,j)$ to represent the probability of an edge between vertices i and j. We then place an edge between each pair of vertices (i,j) according to an independent Bernoulli trial with parameter $K_k(i,j)$. This model is called a *Stochastic Kronecker Product Graph.* Note that this formulation holds the *Erdös-Rényi* $G(n,p)$ model on N_1^k vertices as a special case.

A primary difficulty inherent in Stochastic Kronecker Product Graph (Stochastic Kronecker Product Graph) models is the fact that Kronecker product graphs have numbers of vertices that are composite in the sizes of the generator graphs. In most basic model, where we choose a single initiator graph with N_0 nodes, this means that we need $n = N_0^k$ for some integer k. Clearly, most real-world graphs don't fit this descriptions. In order to compensate for this difficulty, the authors of [5] suggest padding the real-world graphs with empty vertices, and computing the likelihood of this enhanced network under the Stochastic Kronecker Product Graph model. However, the large number of padded vertices distorts the likelihood computations, as we are essentially evaluating the likelihood for a much larger graph instead of the graph size appropriate for the data. Assuming n is uniformly distributed between $N_0^{k-1} < n \leq N_0^k$, this would give us $(1-1/N_0)/2$ expected fraction of padded vertices, with a maximum $1-1/N_0$ fraction padded in the worst case. The worst case is attained if the original graph contains $N_0^k + 1$ vertices.

Some potential problems that padding with empty vertices creates as follow. First, a large number of isolated vertices skews the degree distribution of the Stochastic Kronecker Product Graph with respect to the original matrix. This is perhaps not problematic for graphs with power-law or heavy-tailed degree distributions that would expect most vertices to be incident to very few edges. However, there are many important classes of networks that are not power-law, such as regular graphs, and padding these networks with a large number of isolated vertices can destroy much of the structure.

Second, adding a large number of padded vertices means that we are computing the likelihood of a potentially much larger graph. This has the potential effect of greatly deflating the Maximum Likelihood Estimate for the graph. While many of the interesting and important statistical features of the original graph may still be reproduced, formulating models that have high Maximimum Likelihood Estimates is important in many settings such as Bayesian statistics and Model Selection. Further, since padding is more likely to occur with larger initiator matrices, smaller initiator matrices are often prefered in the Stochastic Kronecker Product Graph model, leaving out potentially useful larger initiators. The fact that $N_1 = 2$ is often the best or close to the best choice for initiator matrix dimension is noted in [5], and makes sense since the number of vertices in a is much likelier to be close to a power of 2 than for any other $N_1 > 2$.

Finally, it is desirable to be able to model 'missing data' problems where we know that we don't have access to the full network. This often takes the form of "link prediction" (missing edges), but also includes many cases where we don't have access to the full vertex set. Vertex Censored models can be naturally

adapted to addressing these types of problems.

1 Stochastic Kronecker Product Graph model

In this section we briefly review the Stochastic Kronecker Product Graph model and the KronFit algorithm used to compute Maximum Likelihood Estimates.

1.1 Likelihood under Stochastic Kronecker Product Graph model

In general, the likelihood of a graph $G(V, E)$ under K_k can be computed as

$$\mathrm{P}(G|K_k) = \prod_{(u,v)\in E} K_k(u,v) \prod_{(u,v)\notin E} (1 - K_k(u,v)). \tag{2}$$

Instead of probability, it is usually easier to work in terms of the log-likelihood:

$$l(K_k|G) = \sum_{(u,v)\in E} \log K_k(u,v) + \sum_{(u,v)\notin E} \log(1 - K_k(u,v)) \tag{3}$$

$$= \sum_{u,v\in V} \log(1 - K_k(u,v)) + \sum_{(u,v)\in E} [\log K_k(u,v) - \log(1 - K_k(u,v))] \tag{4}$$

$$= l_e(K_k|G) + l_E(K_k|G), \tag{5}$$

where l_e denotes the log-likelihood of the *empty* graph, and l_E is the *edge correction* to the log-likelihood.

Due to the recursive structure of Kronecker products, for each vertex u we can associate a vector

$$u = (u_1, u_2, \ldots, u_k), \tag{6}$$

where $u_i \in \{1, \ldots, N_1\}$, and the probability of an edge between two vertices v and u is

$$K_k(u,v) = \prod_{i=1}^{k} K_1(u_i, v_i), \tag{7}$$

The log-likelihood of an empty graph is

$$l_e(K_1(u,v)) = \sum_{u,v} \log(1 - K_p(u,v)). \tag{8}$$

Leskovec et al [5] use the Taylor series approximation to (8),

$$l_e(K_1(u,v)) \approx -\left(\sum_{i=1}^{N_1}\sum_{j=1}^{N_1} K_1(i,j)\right)^k - \frac{1}{2}\left(\sum_{i=1}^{N_1}\sum_{j=1}^{N_1} K_1(i,j)^2\right)^k - \ldots \tag{9}$$

which means that $l_e(K_1(u,v))$ can be computed in constant time. To compute the log-likelihood of the full graph, we can then apply the edge correction no the empty graph, adding the log-likelihoods for each edge in G and removing the corresponding 'no-edge' contributions. This means that the total log-likelihood for G under K_p can be approximately computed in $O(E)$ time.

1.2 Sampling Over the Space of Permutations

As stated in the previous section, it is assumed that we know the approprtiate mapping between vertices in G and vertices in K_k. However, this is generally not the case, and we need to account for this fact by summing over all possible permutations. We can state the conditional probability of a mapping π as

$$\mathrm{P}(\pi|G,K_k) = \frac{\mathrm{P}(G|\pi,K_k)\mathrm{P}(\pi|K_k)}{\mathrm{P}(G|K_k)}. \tag{10}$$

Finding π that maximizes this quantity is similar to the Linear Ordering Problem (LOP) [7, 4], which is NP-hard. To sample from this distribution, Leskovec et al[6] suggest a Metropolis-Hastings Markov Chain Monte Carlo algorithm which has stationary distribution (10). As a base chain they use the "vertex switch" Markov Chain, also known as the "random-to-random" shuffle. Other Markov Chains on permutations are possible; see [2] for discussion and analysis.

The advantage of using a "local" Markov Chain such as random-to-random is that it allows us to use "local" updates to the likelihood. Given the likelihood under π, to compute an update to π' we write

$$\begin{aligned}\delta_{\pi,\pi'} &= \log \mathrm{P}(G|\pi',K_k) - \log \mathrm{P}(G|\pi,K_k) &(11)\\ &= l(K_k^{\pi'}) - l(K_k^{\pi}) &(12)\end{aligned}$$

Under the Stochastic Kronecker Product Graph model, the log-likelihood of the empty graph is the same under both permutations, so this simplifies to

$$\begin{aligned}\delta_{\pi,\pi'} &= l_E(K_k^{\pi'}) - l_E(K_k^{\pi}) &(13)\\ &= \sum_{(\pi'(u),\pi'(v))\in E} [\log K_k(\pi'(u),\pi'(v)) - \log(1-K_k(\pi'(u),\pi'(v)))]\\ &- \sum_{(\pi(u),\pi(v))\in E} [\log K_k(\pi(u),\pi(v)) - \log(1-K_k(\pi(u),\pi(v)))]\,, &(14)\end{aligned}$$

which by combining terms can be computed in constant time for sparse graphs [5].

1.3 Computing Gradients

Computation of gradients follows the same process as log-likelihood computations. To compute $\nabla_i l_(K_k)$, we compute an empty graph component and then apply an edge correction. Approximation to the empty graph gradient proceeds by differentiating the log-likelihood of the Taylor series approximation (9) with respect to parameter i. This process is repeated for each of the N_1 parameters.

2 Vertex Censored Stochastic Kronecker Product Graph model

In this section, we specify the Vertex Censored Stochastic Kronecker Product Graph model and present an algorithm to compute Maximum Likelihoods. The primary motivation for this class of models is to compensate for the likelihood distortions introduced by vertex padding in Stochastic Kronecker Product Graph. Vertex Censored models can be thought of as generating networks by some well-defined additive process, then "dropping" a subset of vertices. Equivalently, we can assume the dropped subset is simply hidden to us, or "censored". Vertex Censored Network Models allow us to make more valid comparisons across Stochastic Kronecker Product Graphs with different sized initiator matrices. Censored data is a common problem in the statistical literature; a previous application of censoring to network data in an unrelated problem can be found in Thomas and Blitzstein [9]. A key component of our algorithm uses *Sequential Importance Sampling* to estimate the empty-graph log-likelihood as well as its gradient.

The basic model can be stated quite simply. Suppose that for our graph $G(V, E)$, $n = |V| < N_1^k$. Instead of "padding" G with isolated vertices, we define an injective mapping $\sigma : V \mapsto V_K$. We can then compute the log-likelihood under the censored model as

$$l^\sigma(K_k) = \sum_{u,v \in V} \big[I_{\{(u,v)\in E\}} \log(K_k(\sigma(u), \sigma(v)) + I_{\{(u,v)\notin E\}} \log(1 - K_k(\sigma(u), \sigma(v))\big], \tag{15}$$

which, as in Stochastic Kronecker Product Graph, gives rise to the empty-graph log-likelihoods $l_e^\sigma(K_k)$ and edge corrections $l_E^\sigma(K_k)$.

2.1 Computing Likelihoods

Note, however, that under the decomposition (15) we lose the recursive structure that allows us to easily compute l_e^σ using the Taylor series expansion (9). To compensate for this, we first represent l_e as an expectation,

$$l_e(K_k) = n^2 \mathrm{E}[\log(1 - K_k(u, v))], \tag{16}$$

where u and v are chosen uniformly at random. Using this same form, we can define the censored version as:

$$l_e^\sigma(K_k) = n^2 \mathrm{E}[I_{\{\sigma^{-1}(u),\sigma^{-1}(v)\in V\}} \log(K_k(\sigma(u), \sigma(v)))] \tag{17}$$

We will attempt to estimate (16) using *Importance Sampling*. There are many references on Importance Sampling, an excellent overall book treatment is Asmussen and Glynn [3]. The basic idea is instead of sampling from a distribution f, we can choose some biased distribution g to give lower variance,

$$\mathrm{E}[\log(1 - K_k(u, v))] = \mathrm{E}[\log(1 - K_k(\hat{u}, \hat{v}))\frac{f(\hat{u}, \hat{v})}{g(\hat{u}, \hat{v})}. \tag{18}$$

In this case, we have $u, v \sim f$, where f is the uniform distribution, and $\tilde{u}, \tilde{v} \sim g$, where g is the importance distribution that we need to choose. Choosing g is difficult, but in general a good guideline is the fact that choosing $g \propto |\log K_k(u,v)| f(u,v)$ results in a *zero-variance* estimator, for which a single draw will give the exact answer. Obviously, choosing this optimal importance distribution for non-trivial problems is typically impossible. However, in this case, the first order Taylor series approximation suggests that using

$$g(\hat{u}, \hat{v}) = \frac{K_k(\hat{u}, \hat{v})}{\sum_{u,v} K_k(u,v)} \tag{19}$$

will be close to optimal. We can simulate from g sequentially, drawing each element of u and v as

$$g_i(\hat{u}_i, \hat{v}_i) = \frac{K_1(\hat{u}_i, \hat{v}_i)}{\sum_{u_i, v_i} K_1(u_i, v_i)}, \tag{20}$$

then combining these as $g(\hat{u}, \hat{v}) = \prod_{i=1}^{k} g_i(\hat{u}_i, \hat{v}_i)$. Computing the gradient of the log-likelihood follows the same pattern, although it is not as straightforward choosing a near optimal importance distribution. In practice, choosing the same importance distribution for the gradient as the log-likelihood appears to give satisfactory results.

2.2 Sampling Permutations

Sampling of permutations proceeds as in the Stochastic Kronecker Product Graph model. We can run the random-to-random Metropolis-Hastings algorithm over σ with the same effect. One problem with this formulation is that flipping the ordering of a 'visible' vertex with a 'censored' vertex requires 'dense' operations to compute $\delta_{\sigma,\sigma}$, as this operation changes the empty-graph log-likelihood $l_e^{\sigma}(K_k)$. To compute this update approximately, we can use an Importance Sampling scheme as above, but only estimating the empty graph likelihood of the individual vertices switched. This can be done quickly and accurately, however this will introduce some 'stochastic drift' over time due to adding of many Monte Carlo estimators with independent errors. This necessitates occasionally re-approximating the full likelihood.

An alternative is for most steps to run a Markov chain that only switches mapped vertices of V, i.e. permuting σ instead of permuting the whole space. Since the empty-graph log-likelihoods haven't changed, this only requires a constant-time udpate as in the Stochastic Kronecker Product Graph model. We could then more infrequently do a fuller step that switches a mapped vertex with an unmapped vertex. We could also simply make the assumption that all of the unmapped vertices belong to a known fixed subset, such as all of the lowest expected degree vertices in K_k, or by matching the expected degree of vertices in K_k to vertex degrees in G.

2.3 Choosing Censored Vertices

As noted in [5], uniformly dropping vertices alters the expected degree distribution of the Stochastic Kronecker Product Graph. However, there is no real reason to assume that the censored data is chosen uniformly; indeed in many contexts it may make more sense to assume the probability of "seeing" a vertex is a function of the number of edges incident to it. Choosing the probability of censoring a vertex as inversely proportional to degree would for example preserve a power-law degree distribution. We may choose also choose different arbitrary rules for dropping vertices, such as dropping the vertices with smallest degree. This last approach is quite similar to the "vertex padding" approach of [6], but has the advantage of producing a higher MLE because we're censoring instead of padding. This is the approach that we take in the numerical examples below.

3 Numerical Examples

The primary network we studied is the AS-ROUTEVIEWS network references in [6]. This network has $n = 6474$ vertices and $m = 26467$ edges. To test the importance sampling scheme, we computed $l_e(K_k)$ for the uncensored Stochastic Kronecker Product Graph model using both the Taylor series approximation (9) and the SIS approximation. Figure 1 shows the results for the $k = 2$ model using the optimal parameters. To compute the following graph, we ran the standard uniform 'vanilla' Monte Carlo scheme with 100 samples 100 times and did the same with the SIS estimator. We then plot the histograms and compare with the Taylor series approximation. Note the extreme reduction in variance; the standard estimate from the SIS samples is $\mu_{SIS} = -24650 \pm 60$, for the vanilla MC estimate $\mu_{MC} = -21614 \pm 17495$, where the 'true' value from the Taylor series expansion is $\mu_{ts} = -24644$. In general, vanilla MC is quite poor, whereas computing the SIS with something like $10n$ samples gives about 5 digits of accuracy.

For gradient computations, however, using $10n$ samples only gives about 2 or 3 digits of accuracy. This is much better than the vanilla MC estimator, but as a result the algorithm had a difficult time converging. More thought needs to be put into the design of the importance sampler for gradient computations.

However, one nice feature of the Vertex Censored model is that we can take the same parameters generated by the standard Stochastic Kronecker Product Graph model with vertex padding and est simply "censor" the padded vertices. While this doesn't give the Maximum Likelihood Estimator for the Vertex Censored model, it does give a lower bound on the MLE (or upper bounds on the negative log-likelihood). Note the big improvements in MLE for size 4 and 5 initiator matrices in Figure 2. Again, these are *upper bounds* on the negative log-likelihood for the VCStochastic Kronecker Product Graph model. This shows that the padded vertices do indeed have a noticable effect, and that more model parameters give a higher likelihood.

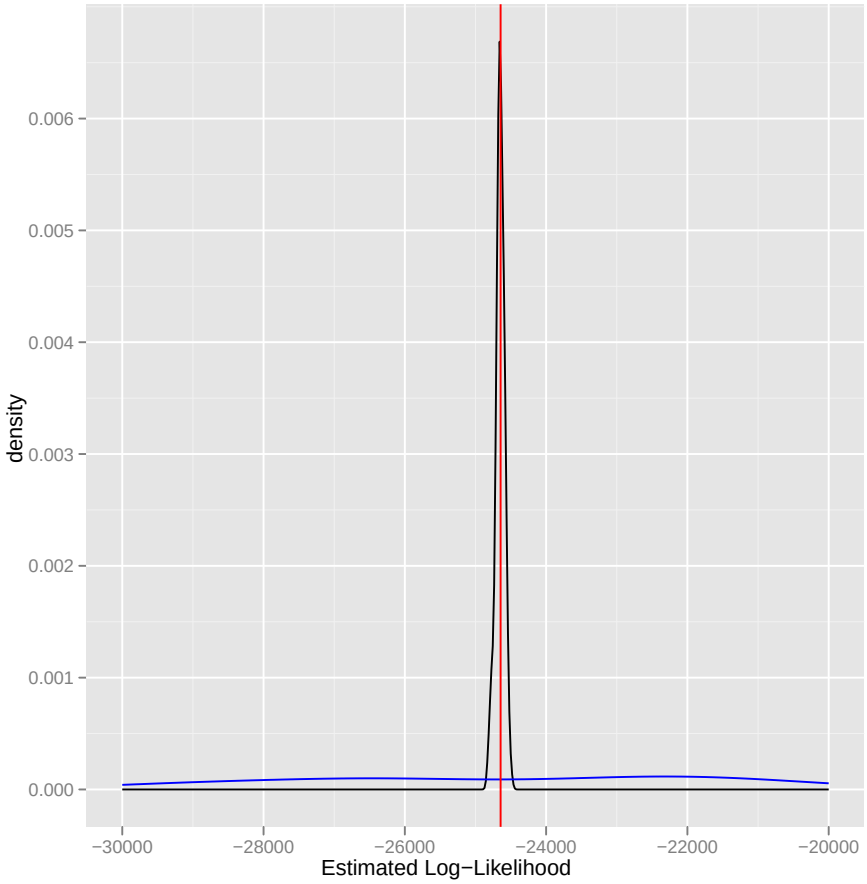

Figure 1: Performance of Vanilla and SIS Monte Carlo simulations on AS-ROUTEVIEWS graph for $N_1 = 2$.

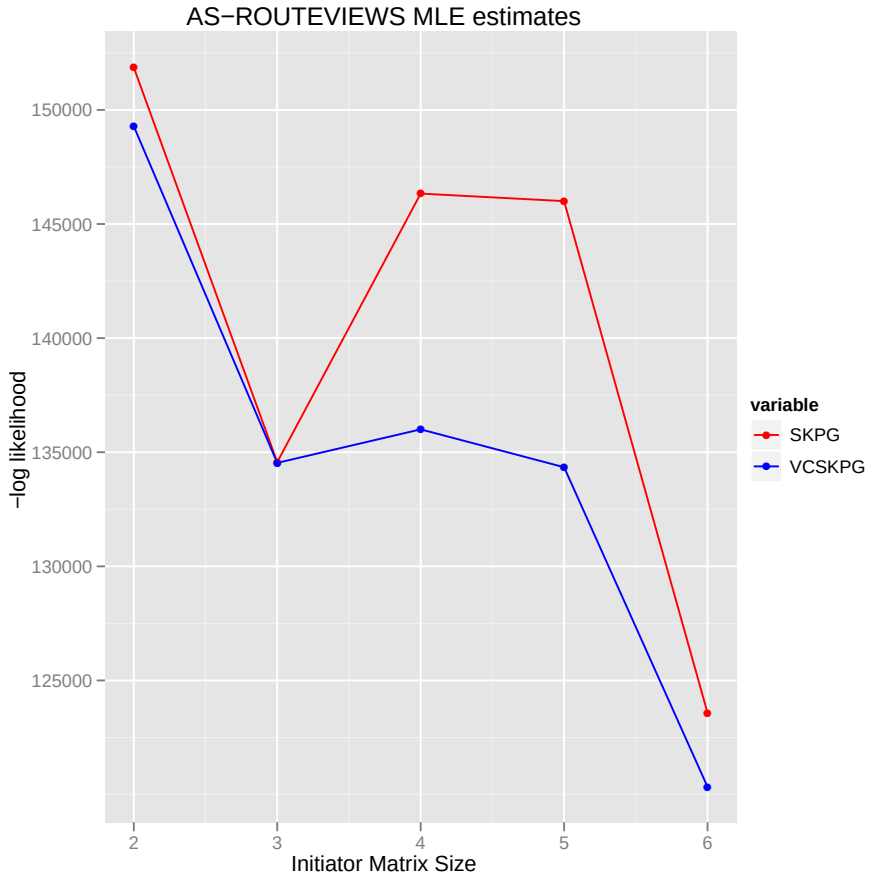

Figure 2: Comparison of SKPG and VCSKPG models for AS-ROUTEVIEWS graph

3.1 Implementation

All code is written in C++, analysis in R. The main routine to compute Maximum Likelihood Estimates was modified from the KronFit implementation in the publically available C++ library SNAP[1]. We call our version VCKronFit (Vertex Censored KronFit).

4 Conclusions and Future Work

There is much more to be done on this interesting class of models.

- Many more numerical studies should be performed.
- More carefully choosing a importance distribution for the gradient should be possible and produce better results.
- Gradient descent, while simple and easy to implement, often has poor convergence properties. Other numerical optimation algorithms could be tried, particularly something like BFGS.
- Mixture models of Kronecker products are feasible, relatively easy to implement using current algorithms, and may be of some interest.
- The convergence properties of the Permuation Sampling and Gradient Descent algorithms for Stochastic Kronecker Product Graphs are to my knowledge poorly understood. A better understanding of these properties are necessary in order to show the existence of FPRAS for this problem.

References

[1] SNAP graph library, snap.stanford.edu. Accessed December 2010.

[2] D. Aldous and P. Diaconis. Shuffling cards and stopping times. *American Mathematical Monthly*, 93(5):333–348, 1986.

[3] S. Asmussen, P.W. Glynn, and SpringerLink (Online service. *Stochastic Simulation: Algorithms and Analysis*. Springer, 2007.

[4] A. Blum, G. Konjevod, R. Ravi, and S. Vempala. Semi-definite relaxations for minimum bandwidth and other vertex-ordering problems. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 100–105. ACM, 1998.

[5] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *The Journal of Machine Learning Research*, 11:985–1042, 2010.

[6] J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *Proceedings of the 24th international conference on Machine learning*, page 504. ACM, 2007.

[7] A. Newman. Cuts and orderings: On semidefinite relaxations for the linear ordering problem. *Approximation, Randomization, and Combinatorial Optimization*, pages 195–206, 2004.

[8] MEJ Newman and EA Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences*, 104(23):9564, 2007.

[9] A.C. Thomas and J.K. Blitzstein. The Effect of Censoring Out-Degree On Network Inferences. Unpublished manuscript, 2009.