Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

SIMULATION OF BIPARTITE OR DIRECTED GRAPHS WITH PRESCRIBED DEGREE SEQUENCES USING MAXIMUM ENTROPY PROBABILITIES

Paul Glasserman

Graduate School of Business
Columbia University
403 Uris Hall
New York, 10027, USA

Enrique Lelo de Larrea

Department of IE&OR
Columbia University
500 West 120th Street
New York, 10027, USA

ABSTRACT

We propose an algorithm for simulating bipartite or directed graphs with given degree sequences, motivated by the study of financial networks with partial information. Our algorithm sequentially computes certain "maximum entropy" matrices, and uses the entries of these matrices to assign probabilities to edges between nodes. We prove the correctness of the algorithm, showing that it always returns a valid graph and that it generates all valid graphs with positive probability. We illustrate the algorithm in an example of an inter-bank network.

1 INTRODUCTION

We consider the problem of simulating bipartite or directed graphs with given degree sequences. Our investigation is motivated by the analysis of network models of interconnectedness in the financial system. Following the global financial crisis of 2007-2009, network models have received growing attention as a framework for the study of contagion. We are particularly interested in two types of networks:

Asset-Firm networks: These are bipartite graphs in which one set of nodes corresponds to investment firms and the other nodes represent financial assets. An edge between two nodes means that the indicated firm owns the indicated asset. A drop in value for one asset may force a firm that holds that asset to sell other assets, driving down their prices, in turn creating losses for other firms holding those assets. In this setting, shocks spread through the financial system by moving back and forth between assets and the firms that own them. This type of model is studied in, for example, Chen et al. (2014), Caccioli et al. (2014), Squartini et al. (2017) and Section 11 of Glasserman and Young (2016).

Inter-bank networks: These are directed graphs in which each node represents a bank. A directed edge from one node to another indicates a payment obligation from one bank to another. If a bank defaults, it fails to meet its payment obligations to other banks, potentially causing those banks to fail, creating a cascade of failures. The framework of Eisenberg and Noe (2001) has spawned a large literature on these types of models, as surveyed in Glasserman and Young (2016).

In studying these types of networks, we have at best partial information. For example, we may know the total amount a bank has borrowed from and lent to other banks, without knowing the amount it has borrowed from or lent to any individual bank. Moreover, in studying the financial stability implications of features of a network, we are interested in exploring all networks consistent with those observable features, and not just one particular configuration.

Simulation is a natural tool to address both issues. The goal is to simulate random instances of networks that are consistent with the available information. Here we focus on the case in which we know the degrees of all nodes but not which nodes are connected by edges. In an asset-firm network, this corresponds to knowing how many assets are held by each firm and how many firms hold each asset, without knowing which assets are held by which firms. In an inter-bank network, the assumption is that we know the number of banks against which each bank has payment obligations or claims, without knowing which pairs of banks have transacted with each other. Given a set of features (node degrees) we would like to sample uniformly from the set of graphs that have those features. Given such an algorithm, one can then estimate a measure of performance or risk, conditional on the features, by averaging results over multiple random instances.

The problem of incomplete information in inter-bank networks has commonly been addressed by filling in the missing information with a configuration that is, in some sense, the most likely one, given the available information. See Section 12.2 of Glasserman and Young (2016) for a discussion of these methods and Anand et al. (2018) for an empirical comparison. Simulation methods have received less attention but include the detailed network construction method of Hałaj and Kok (2013) and the Bayesian method of Gandy and Veraart (2017). Rather than simulate uniformly, Gandy and Veraart (2017) assume prior information is available on the probabilities of individual edges, and they draw samples consistent with this information using a Markov Chain Monte Carlo (MCMC) approach.

Outside the setting of financial networks, the problem of generating random graphs with given degree sequences has been studied in various other settings. A first natural approach for this problem is the *pairing model* (also known as the *configuration model*). Fix n vertices and degree sequences $d_1, d_2, \dots, d_n$ (with $\sum_i^n d_i$ even). The pairing model creates d_i copies of each vertex and then selects pairs at random, until all of the vertices have been "matched". The result is a graph (that may contain loops and multiple edges) with the desired degree sequence. Wormald (1999) describes in detail the properties of this method. Because of the possibility of loops and multiple edges, this method is not guaranteed to produce valid samples in the sense we verify for our method.

A second method uses an MCMC algorithm. It starts with a graph with the desired degree sequence. At each iteration, it selects two edges, say $\{a,b\}$, $\{c,d\}$, with a,b,c,d all different. It then replaces these edges with either $\{a,c\}$, $\{b,d\}$ or $\{a,d\}$, $\{b,c\}$. The resulting graph maintains the correct degree sequence. This method is called the *switching method* in Wormald (1999) and *local rewiring algorithm (LRA)* in Squartini and Garlaschelli (2011). See also Blitzstein and Diaconis (2010). Some drawbacks of this method are that the samples produced are not independent and that it can can be computationally expensive since the chain must be "run" until stationarity is reached.

Blitzstein and Diaconis (2010) propose another algorithm for this problem using a graphicality condition. As opposed to the two previous methods, their algorithm does not sample graphs uniformly at random. They use importance sampling weights to fix this issue. However, for the numerical example they provide, the weights seem to be skewed to the right, compromising the quality of the importance sampling estimation.

Our approach builds on work of Barvinok (2010) on random 0–1 matrices with given row and column sums; these matrices can be interpreted as the adjacency matrices of bipartite or directed graphs with given degree sequences. Barvinok (2010) proved that there is a connection between the uniform distribution over these matrices and a certain "maximum entropy" matrix. The same type of matrix is used by Squartini and Garlaschelli (2011) in a numerical approximation without simulation. Since our objective is to simulate graphs uniformly, it is natural to exploit this connection. We use a sequence of such matrices to define probabilities for individual edges.

We prove the correctness of our method, by which we mean that it always returns a valid graph and it reaches every feasible graph with positive probability. The method does not sample feasible graphs uniformly, but we show that expectations with respect to uniform sampling can be estimated by weighting samples appropriately. The weights are produced as a simple byproduct of the sampling algorithm. We expect our use of the maximum entropy matrix to produce well-behaved weights, meaning that their distribution is not highly skewed; we illustrate their performance through an example. Computing maximum entropy matrices is the main computational demand of our algorithm. We briefly discuss potential speed-ups, which are the subject of ongoing work.

2 RANDOM GRAPHS AND RANDOM ADJACENCY MATRICES

We represent bipartite or directed graphs using 0–1 matrices. These matrices can be interpreted as the adjacency matrices of the underlying graph. Following the notation of Barvinok (2010), let $R = (r_1, r_2, \dots, r_m)$ and $C = (c_1, c_2, \dots, c_n)$ be the sum of the rows and columns of the matrix respectively. For bipartite graphs, R and C represent the degree sequences of the first and second disjoint sets of vertices. On the other hand, for directed graphs, R and C represent the out-degrees and in-degrees of the vertices. Since every edge of the graph contributes one unit to both R and C, it is necessary to have $\sum_{i=1}^m r_i = \sum_{j=1}^n c_j$. Furthermore, we clearly must have $0 < r_i \leq n$ and $0 < c_j \leq m$. For the case of directed graphs, we must impose $m = n$.

Suppose also that we wish to force the exclusion of some edges of the graph. For this purpose, let $W = (w_{ij})$ with $w_{ij} \in \{0,1\}$ be a *pattern* matrix. If a particular edge is to be excluded, we set the corresponding w_{ij} to 0 and, otherwise, to 1. Observe that in the directed graph case, the elements of the diagonal correspond to loops. If we wish to exclude graphs that contain loops, we can do so by making the diagonal elements $w_{ii} = 0$.

Given the degree sequences R and C, and the pattern matrix W, we now define the set of 0–1 matrices with such degrees and assigned zeros.

Definition 1 $\Sigma(R,C;W)$ is the set of all $m \times n$ matrices $M = (m_{ij})$ such that $\sum_{j=1}^n m_{ij} = r_i$ for all i, $\sum_{i=1}^m m_{ij} = c_j$ for all j, $m_{ij} \in \{0,1\}$, and $m_{ij} = 0$ if $w_{ij} = 0$. If $w_{ij} = 1$ for all i and j we denote it $\Sigma(R,C)$.

We also need to relax the integrality constraint of $\Sigma(R,C;W)$ and work with the following polytope.

Definition 2 $\mathscr{P}(R,C;W)$ is the set of all $m \times n$ matrices $X = (x_{ij})$ such that $\sum_{j=1}^n x_{ij} = r_i$ for all i, $\sum_{i=1}^m x_{ij} = c_j$ for all j, $x_{ij} \in [0,1]$, and $x_{ij} = 0$ if $w_{ij} = 0$. If $w_{ij} = 1$ for all i and j we denote it $\mathscr{P}(R,C)$.

We interpret the x_{ij} as the probability that the edge (i,j) is present in the graph. The row and column sum constraints on X imply that a graph sampled with these probabilities has the correct *expected* degree sequences. We need to ensure that the correct degree sequence holds with probability one.

We now introduce a *maximum entropy* problem which will be of great relevance for the algorithm.

2.1 Maximum Entropy Problem and its Dual

For $x \in [0,1]$ we define the *entropy function* as

$$h(x) = x \log\left(\frac{1}{x}\right) + (1-x)\log\left(\frac{1}{1-x}\right).$$

It is a strictly concave function with $h(0) = h(1) = 0$. The derivative is $h'(x) = \log(1-x) - \log(x)$ and therefore $h'(0^+) = \infty$ and $h'(1^-) = -\infty$. For X in $\mathscr{P}(R,C)$ we define the *entropy function* as

$$H(X) = \sum_{ij} h(x_{ij}).$$

Since H is a strictly concave function it attains a unique maximum at some $Z = Z(R,C;W)$. That is, the *maximum entropy matrix* Z is the solution to

$$\begin{aligned} \underset{X}{\text{maximize}} \quad & H(X) \\ \text{subject to} \quad & X \in \mathscr{P}(R,C;W). \end{aligned} \tag{1}$$

Observe that the number of variables of the problem is $m \times n$, which may be large for certain applications. Therefore, in practice, it is more convenient to work with the dual problem which is

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & F(x,y) = \left(\prod_{i=1}^m x_i^{-r_i}\right)\left(\prod_{j=1}^n y_j^{-c_j}\right)\left(\prod_{ij}(1 + w_{ij}x_i y_j)\right) \\ \text{subject to} \quad & x > 0,\ y > 0. \end{aligned} \tag{2}$$

As in Barvinok (2010), to solve the problem in Equation 2 we change the variables to $s_i = \log(x_i)$ and $t_j = \log(y_j)$ and solve instead the unconstrained convex problem

$$\underset{s,t}{\text{minimize}} \quad G(s,t) = -\sum_{i=1}^{m} r_i s_i - \sum_{j=1}^{n} c_j t_j + \sum_{ij} \log\left(1 + w_{ij} e^{s_i + t_j}\right). \tag{3}$$

This problem has only $m+n$ variables. We then recover Z by setting

$$z_{ij} = \frac{e^{s_i+t_j}}{1+e^{s_i+t_j}},$$

whenever $w_{ij} = 1$ and $z_{ij} = 0$ otherwise.

3 SIMULATION ALGORITHM FOR BIPARTITE OR DIRECTED GRAPHS

Given degree sequences R and C, the objective is to generate a random matrix $M \in \Sigma(R,C)$, assuming that this set is non-empty. Our algorithm is described in Algorithm 1.

Algorithm 1 Graph simulation using maximum entropy probabilities.

```
1:  input: $R$ and $C$
2:  verify $\Sigma(R,C)$ non-empty
3:  initialize $M \leftarrow 0$, $W \leftarrow W_0$, and $p_M \leftarrow 1$
4:  set $\hat{R} \leftarrow R$ and $\hat{C} \leftarrow C$
5:  compute $Z\left(\hat{R},\hat{C};W\right)$
6:  for all $(i,j)$ do
7:      set $m_{ij} \leftarrow 1$ with probability $z_{ij}$
8:      if $m_{ij} = 1$ then
9:          set $p_M \leftarrow p_M * z_{ij}$
10:         set $\hat{r}_i \leftarrow \hat{r}_i - 1$
11:         set $\hat{c}_j \leftarrow \hat{c}_j - 1$
12:     else
13:         set $p_M \leftarrow p_M * (1 - z_{ij})$
14:     end if
15:     set $w_{ij} \leftarrow 0$
16:     compute $Z\left(\hat{R},\hat{C};W\right)$
17: end for
18: return $M$ and $p_M$
```

Observe that in line 2, the algorithm verifies that $\Sigma(R,C)$ has at least one element. This can be done quickly using the criterion given by the Gale-Ryser theorem (see, for instance, Gale (1957)). It suffices to check that $\sum_{i=1}^{m} r_i = \sum_{j=1}^{n} c_j$ and that $\sum_{j=1}^{k} c_j \leq \sum_{i=1}^{m} \min(r_i, k)$, for all $k = 1, 2, \ldots, n$.

The algorithm starts with an empty graph $M = 0$ and an initial pattern $W = W_0$. In the case of bipartite graphs and directed graphs when loops are permitted, $W_0 = (w_{ij}^0)$ has all entries equal to 1. If we exclude loops, then we set diagonal $w_{ii}^0 = 0$. The algorithm then sequentially adds edges at random using the maximum entropy matrix Z. At each step, if edge (i,j) is added to M, we reduce the corresponding r_i and c_j by one. If the edge is not added, we leave the degree sequences intact. In either case, we update the pattern by setting $w_{ij} = 0$. This will reduce the polytope of feasible matrices to a subset where the presence of edge (i,j) is never allowed. Finally, we update Z with the new degree sequences and pattern. At the end, we return the matrix M. In the next section, we show that this matrix is always in $\Sigma(R,C)$. The algorithm also computes the probability p_M of generating this particular matrix M. Observe from lines 9 and 13 that

$$p_M = \prod_{ij} \left(m_{ij} z_{ij} + (1 - m_{ij})(1 - z_{ij}) \right).$$

In this expression, the z_{ij} do not belong to the same maximum entropy matrix. Each one is an element of its corresponding matrix Z computed in line 16.

3.1 Illustration of the Algorithm

To better explain our algorithm, we present a simple example. Consider the bipartite graphs with $m = n = 3$ and $R = C = (1,1,2)$. It is easy to see that, in this case, there are five graphs consistent with the constraints, so $|\Sigma(R,C)| = 5$. Figure 1 presents one realization of Algorithm 1.

(1) $M = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $p_M = 1$ $Z = \begin{pmatrix} \mathbf{0.22} & 0.22 & 0.57 \\ 0.22 & 0.22 & 0.57 \\ 0.57 & 0.57 & 0.86 \end{pmatrix}$

(2) $M = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $p_M = 0.78$ $Z = \begin{pmatrix} 0 & \mathbf{0.32} & 0.68 \\ 0.32 & 0.18 & 0.5 \\ 0.68 & 0.5 & 0.82 \end{pmatrix}$

(3) $M = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $p_M = 0.25$ $Z = \begin{pmatrix} 0 & 0 & \mathbf{0} \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$

(4) $M = \begin{pmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} \end{pmatrix}$ $p_M = \mathbf{0.25}$ $Z = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

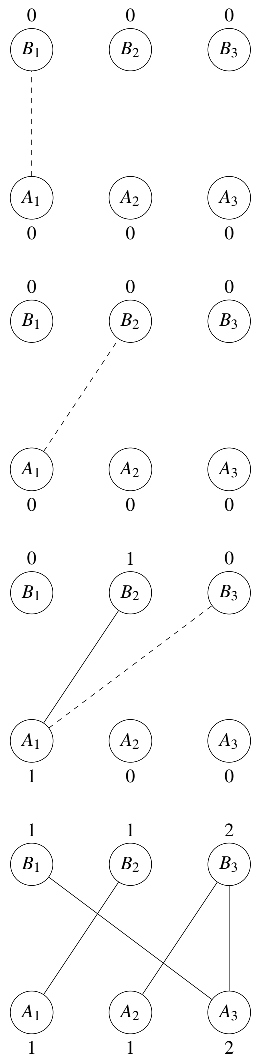

Figure 1: Realization of Algorithm 1 with $m = n = 3$ and $R = C = (1,1,2)$.

In Step 1 of Figure 1, we start with the empty graph $M = 0$ and a temporary graph probability of $p_M = 1$. We compute the maximum entropy matrix Z and focus on the first potential edge (connecting A_1 and B_1). To decide if the edge will be generated, we generate a Bernoulli random variable X_1 with probability $p = z_{11} = 0.22$. In this particular realization, $X_1 = 0$ and thus the edge is not generated.

In Step 2, we update p_M to be $p_M \leftarrow p_M * (1 - z_{11})$ since the previous edge was not generated. We compute Z and consider the next edge (connecting A_1 and B_2). We simulate a Bernoulli X_2 with probability $p = z_{12} = 0.32$. This time $X_2 = 1$ and the edge is generated.

In Step 3, the graph M now contains the edge generated in the previous step and we update p_M to be $p_M \leftarrow p_M * z_{12}$, since the previous edge was generated. After computing Z, observe that all its entries are either 0 or 1. This means that, given the previous edges that we have decided to either include or not, there is only one possible graph left that satisfies the degree sequences. The edges that need to be added to M are the ones where $z_{ij} = 1$.

Step 4 presents the output graph M and its corresponding probability of being generated by Algorithm 1. Notice that $p_M = 0.25 \neq 0.2 = 1/|\Sigma(R,C)|$. We conclude that, in general, the algorithm does not sample uniformly from $\Sigma(R,C)$. However, if uniform samples are required, it is possible to use importance sampling weights as discussed in Section 4.3.

Observe that in Step 1, instead of using only z_{11}, we could have used the entire Z to generate all of the edges at the same time. This alternative method would produce a random graph with the correct *expected* degree sequences. However, and contrasting with Algorithm 1, it is not true that this graph will satisfy the degree constraints with probability 1.

4 PROPERTIES OF THE ALGORITHM

We prove that Algorithm 1 is correct and that it reaches every feasible graph with positive probability.

4.1 Termination and Correctness

We first show that Algorithm 1 always terminates, that is, it never gets stuck in line 16, and that it is correct, meaning that the output M has the desired degree sequences. For this, we first show that the 0–1 matrices with prescribed degree sequences form the vertices of the relaxed polytope $\mathscr{P}(R,C;W)$.

Lemma 1 Suppose $\mathscr{P}(R,C;W)$ is non-empty. Then $\Sigma(R,C;W)$ is the set of vertices of $\mathscr{P}(R,C;W)$.

The proof may be found in Appendix A. An immediate corollary to the previous lemma is that every matrix in $\mathscr{P}(R,C;W)$ can be represented as a convex combination of matrices in $\Sigma(R,C;W)$.

Corollary 1 If $X \in \mathscr{P}(R,C;W)$ then there exist $q > 0$ integer, $\lambda_1, \lambda_2, \dots, \lambda_q$ with $\lambda_k > 0$ for all k and $\sum_{k=1}^{q} \lambda_k = 1$ and there exist $M^1, M^2, \dots, M^q$ in $\Sigma(R,C;W)$ such that $X = \sum_{k=1}^{q} \lambda_k M^k$.

The termination and correctness of Algorithm 1 is established in the next theorem.

Theorem 1 If $\Sigma(R,C)$ is non-empty then Algorithm 1 always terminates with an M in $\Sigma(R,C)$.

The proof may be found in Appendix B.

4.2 Reachability

We now show that Algorithm 1 can reach any matrix $\tilde{M}$ in $\Sigma(R,C)$. First, we show a result that illustrates the fact that the maximum entropy matrix lies in the interior of the polytope if possible.

Lemma 2 Suppose $\mathscr{P}(R,C;W)$ is non-empty. Fix (i,j) such that $w_{ij} = 1$ and let $\delta \in \{0,1\}$. Then $z_{ij} = \delta$ if and only if for all X in $\mathscr{P}(R,C;W)$ we have $x_{ij} = \delta$.

The proof may be found in Appendix C. We can now show that any 0–1 matrix M with prescribed degree sequences is reachable by the algorithm.

Theorem 2 If $\tilde{M} \in \Sigma(R,C)$ then the probability that $\tilde{M}$ is generated by Algorithm 1 is positive.

Proof. Given $\tilde{M} \in \Sigma(R,C)$, denote A_k as the event that the algorithm generates a matrix M that matches $\tilde{M}$ on the first k edges visited, $k = 0, 1, \dots, m \times n$. Since $A_{m\times n}$ is the event that $M = \tilde{M}$, it suffices to show that $\mathrm{P}(A_{m\times n}) > 0$. We show this by induction on k. Trivially, $\mathrm{P}(A_0) = 1$. Suppose $\mathrm{P}(A_{k-1}) > 0$. This implies that there exists a realization of the algorithm in which $M_{i_l j_l} = \tilde{M}_{i_l j_l}$ for $l = 1, 2 \dots, k-1$. Suppose that at iteration k, edge (i_k, j_k) is picked. If $\tilde{M}_{i_k j_k} = 0$, then, by Lemma 2, $z_{i_k j_k} < 1$. This implies that this realization will have $M_{i_k j_k} = \tilde{M}_{i_k j_k}$ with probability $1 - z_{i_k j_k} > 0$. If $\tilde{M}_{i_k j_k} = 1$, then, again by Lemma 2, $z_{i_k j_k} > 0$. This implies that this realization will have $M_{i_k j_k} = \tilde{M}_{i_k j_k}$ with probability $z_{i_k j_k} > 0$. In either case, we have found a realization of the algorithm in which $M_{i_l j_l} = \tilde{M}_{i_l j_l}$ for $l = 1, 2 \ldots, k$, that is, $\mathrm{P}(A_k) > 0$. By induction, we have $\mathrm{P}(A_{m \times n}) > 0$ and the result follows. $\square$

4.3 Non-Uniformity and Importance Weights

Consider the problem of sampling random graphs uniformly from $\Sigma(R,C)$. In this case, Algorithm 1 cannot be used directly since the sampling it produces is non-uniform. For example, in Section 3.1 we saw that the graph generated had $p_M \neq 1/|\Sigma(R,C)|$, which is the probability of every graph under the uniform distribution. However, we can use importance sampling to correct the non-uniformity. The importance weight for a given matrix M is

$$w_M = \frac{1}{|\Sigma(R,C)| p_M} \propto \frac{1}{p_M}.$$

Suppose that we are interested in estimating $\theta = \mathrm{E}[f(M)]$, where f is a function of interest and the expectation is with respect to the uniform distribution in $\Sigma(R,C)$. We can use Algorithm 1 to simulate a random sample of graphs $M_1, M_2, \ldots, M_q$. Using the importance weights, we obtain the estimator $\hat{\theta} = q^{-1} \sum_{i=1}^{q} w_{M_i} f(M_i)$. However, the weight w_M depends on the quantity $|\Sigma(R,C)|$, which in general is unknown, so the estimator $\hat{\theta}$ cannot be computed. Blitzstein and Diaconis (2010) address this problem by using instead the estimator

$$\tilde{\theta} = \frac{\sum_{i=1}^{q} w_{M_i} f(M_i)}{\sum_{j=1}^{q} w_{M_j}}. \tag{4}$$

This estimator can be computed explicitly because the constant factor $1/|\Sigma(R,C)|$ cancels. As a ratio estimator, $\tilde{\theta}$ is biased, but it converges with probability 1 to θ as the number of samples q grows.

5 EXAMPLE OF AN INTER-BANK NETWORK

To test Algorithm 1, we borrow an example from Table 2 of Gandy and Veraart (2017), drawn from results of the European Banking Authority's stress test. In this table, they present a network of 11 German banks and their mean out-degrees under two Erdős-Rényi models. To construct the degree sequences, we round the mean out-degrees to the nearest integer. The results, denoted $R_{0.5}$ and $R_{0.9}$, for each of the two models can be seen in Table 1. Finally, we assume that the mean in-degrees are the same as the mean out-degrees and we set $C_{0.5} = R_{0.5}$ and $C_{0.9} = R_{0.9}$.

Table 1: Rounded mean out-degrees for two Erdős-Rényi models.

Bank	$R_{0.5}$	$R_{0.9}$	Bank	$R_{0.5}$	$R_{0.9}$
DE020	6	9	DE028	5	8
DE019	6	9	DE027	4	8
DE021	6	9	DE024	4	8
DE022	5	9	DE023	3	7
DE018	5	9	DE025	2	6
DE017	5	9			

Given these degree sequences, we used Algorithm 1 (with no loops allowed) to generate 5,000 random graphs for each model. On average, it took approximately 1.02 and 1.34 seconds to draw one random graph from models $R_{0.5}$ and $R_{0.9}$ respectively. The implementation was written in Python and ran on a 2.9 GHz MacBook Pro. One of these graphs corresponding to the model $R_{0.5}$ may be seen in Figure 2.

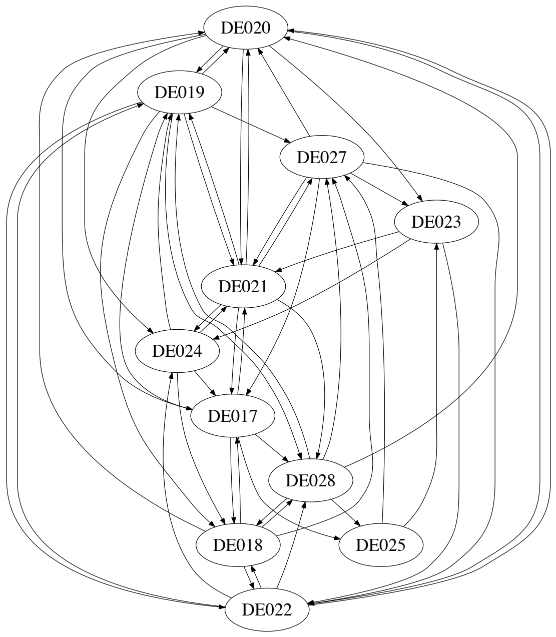

Figure 2: Random graph in $\Sigma(R_{0.5}, C_{0.5})$.

We also computed the importance weights (up to a multiplicative constant) for these random samples. Figure 3 presents a histogram for the weights of model $R_{0.9}$ and model $R_{0.9}$. In both cases, the distribution does not seem to be extremely skewed, meaning that there are no graphs with large weights that would dominate in importance sampling. This observation can be confirmed numerically by using the diagnostic proposed in Chatterjee and Diaconis (2018). We define $\kappa_q = \max_{i=1,2,\dots,q} w_{M_i} / \sum_{i=1}^{q} w_{M_i}$. The importance weights are considered to be well behaved if κ_q is smaller than a certain threshold, for instance 0.01. That is indeed the case in our example where $\kappa_q = 0.00081$ for model $R_{0.5}$ and $\kappa_q = 0.00075$ for model $R_{0.9}$.

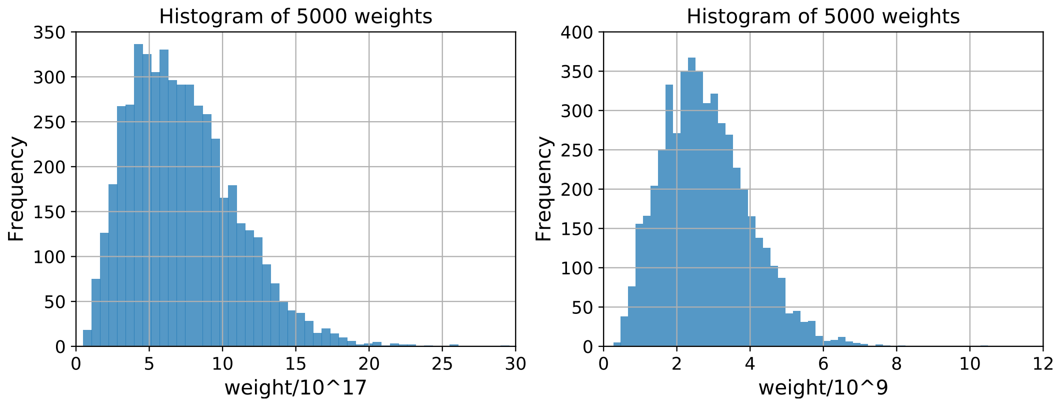

Figure 3: Histograms of weights for model $R_{0.5}$ (left) and model $R_{0.9}$ (right).

As discussed in Glasserman and Young (2016), given a random instance of an inter-bank network, one can evaluate various measures of systemic risk and then average over multiple draws. Taking a weighted average with the importance sampling weights w_M approximates expectations over a uniform distribution of inter-bank networks that are compatible with the observed degrees.

For example, for bank $i = 1, 2, \ldots, n$ consider the *clustering* measure f_i defined as

$$f_i = \sum_{j=1}^{n} \sum_{k>j}^{n} \left(m_{ji} m_{ki} \sum_{l \neq i}^{n} m_{jl} m_{kl} \right).$$

Recall that $m_{ij} = 1$ if bank i has a financial obligation with bank j and $m_{ij} = 0$ otherwise. Then f_i counts, for each pair of banks that owe money to bank i, how many other banks they both owe money to. Intuitively, a high value of f_i would indicate that the joint failure of banks owing money to bank i would have a worse impact on the entire inter-bank network. We estimated f_i using the importance sampling estimator in Equation 4. The results in Table 2 provide measures of the centrality of the banks estimated from their in-degrees.

Table 2: Estimation of f_i using importance sampling.

Bank	$R_{0.5}$	$R_{0.9}$	Bank	$R_{0.5}$	$R_{0.9}$
DE020	20.67	186.44	DE028	15.48	153.01
DE019	20.75	186.46	DE027	15.41	152.91
DE021	20.76	186.39	DE024	9.85	153.17
DE022	15.48	186.14	DE023	4.99	118.52
DE018	15.45	186.39	DE025	1.61	85.45
DE017	15.36	186.45			

6 ONGOING WORK

In current work, we are investigating enhancements to Algorithm 1, including the following:

1. *Quick update of the maximum entropy matrix Z.* The computation of line 16 is the most computationally expensive operation of the algorithm, since it involves solving the convex optimization problem of Equation 3. Therefore, finding a way to quickly update Z (even approximately) using its value in the previous iteration could potentially lead to a considerable speed-up in runtime.
2. *Order in which the edges are visited.* Our current implementation of Algorithm 1 visits the edges in a deterministic order. However, it is possible to consider adapted random orders, probably depending on the current value of Z. The order could affect the distribution of the importance weights. It could also lead to a speed-up in runtime if it reaches a Z with 0–1 entries in an earlier stage of the algorithm, such as we saw in Section 3.1.

7 CONCLUSION

We have presented an algorithm to generate random bipartite or directed graphs with a determined degree sequence. The algorithm relies mainly on the sequential computation of the maximum entropy matrix. The entries of this matrix determine the individual edge probabilities that the algorithm uses. We proved that the algorithm always finishes and that the output is indeed a graph with the desired degree sequences. We also proved that, given a degree sequence, every graph that satisfies is generated by our algorithm with positive probability. Although our procedure does not simulate graphs uniformly, we can use importance sampling weights to compute expectations under the uniform distribution. The application to an inter-bank network produced weights with a reasonable distribution, that is, not extremely skewed. Developing an update rule for the maximum entropy matrix and exploring adaptive orders in which the algorithm visits the edges could lead to potential speed-ups.

A PROOF OF LEMMA 1

Before proving this lemma, we recall the concept of *total unimodularity*.

Definition 3 An integer matrix A is said to be totally unimodular (TUM) if every square submatrix of A has determinant equal to 0, 1 or -1.

We also recall the definition of the well-known *transportation polytope*, which we denote $\mathscr{T}(R,C)$.

Definition 4 $\mathscr{T}(R,C)$ is the set of all $m\times n$ matrices $X=(x_{ij})$ such that $\sum_{j=1}^{n} x_{ij}=r_i$ for all i, $\sum_{i=1}^{m} x_{ij}=c_j$ for all j, and $x_{ij}\geq 0$ for all i and j.

With these definitions in hand, we proceed to prove the lemma.

Proof. Suppose that X is not a vertex of $\mathscr{P}(R,C;W)$. Then, it can be written as a convex combination of two different elements of $\mathscr{P}(R,C;W)$. This implies that there is an $x_{ij}\in(0,1)$, which means that X cannot be in $\Sigma(R,C;W)$. This proves that $\Sigma(R,C;W)$ is a subset of the vertices of $\mathscr{P}(R,C;W)$. To establish the other inclusion, observe that the polytope $\mathscr{P}(R,C;W)$ is simply the transportation polytope $\mathscr{T}(R,C)$ with the additional constraints $x_{ij}\leq 1$ for all i and j, and $x_{ij}=0$ whenever $w_{ij}=0$. Let A be the constraint matrix associated with $\mathscr{T}(R,C)$. By adding a slack variable y_{ij} for each (i,j) where $w_{ij}=1$ we can rewrite $\mathscr{P}(R,C;W)$ as the set of vectors z such that $z\geq 0$ and $Bz=d$, where

$$B=\begin{pmatrix}\tilde{A} & 0\\ I & I\end{pmatrix},\quad z=\begin{pmatrix}x\\ y\end{pmatrix},\quad d=\begin{pmatrix}\tilde{d}\\ 1\end{pmatrix},\quad \tilde{d}=\begin{pmatrix}R^\top\\ C^\top\end{pmatrix},$$

and $\tilde{A}$ consists of the columns of A corresponding to the (i,j) such that $w_{ij}=1$. It follows from Theorem 13.3 of Papadimitriou and Steiglitz (1998) that $\tilde{A}$ is TUM and therefore, it is not hard to see (for instance by induction) that B is also TUM. Finally, by Theorem 13.1 of Papadimitriou and Steiglitz (1998) we conclude that $\mathscr{P}(R,C;W)$ has integral vertices. In other words, the vertices of $\mathscr{P}(R,C;W)$ are a subset of $\Sigma(R,C;W)$. □

B PROOF OF THEOREM 1

Before proving the theorem, we introduce some helpful notation. For an integer vector x we define x^{-i} as

$$x_k^{-i}=\begin{cases}x_k, & k\neq i\\ x_i-1, & k=i,\end{cases}$$

and similarly for an integer matrix $A=(a_{ij})$ we define $A^{-ij}=(a_{kl}^{-ij})$ as

$$a_{kl}^{-ij}=\begin{cases}a_{kl}, & (k,l)\neq(i,j)\\ a_{ij}-1, & (k,l)=(i,j).\end{cases}$$

The next lemma establishes a relationship between the maximum entropy matrix and the existence of 0–1 matrices in a reduced polytope.

Lemma 3 Suppose $\Sigma(R,C;W)$ is non-empty. Fix (i,j) such that $w_{ij}=1$. Then

(a) If $z_{ij}<1$ then $\Sigma(R,C;W^{-ij})$ is non-empty.
(b) If $z_{ij}>0$ then $\Sigma(R^{-i},C^{-j};W^{-ij})$ is non-empty.

Proof. Consider (i,j) such that $w_{ij}=1$. Since $Z\in\mathscr{P}(R,C;W)$, by Corollary 1 we can write $Z=\sum_{k=1}^{p}\lambda_k M^k$, with $M^k\in\Sigma(R,C;W)$. In particular $z_{ij}=\sum_{k=1}^{p}\lambda_k m_{ij}^k$. If $z_{ij}<1$ then there exists k such that $m_{ij}^k=0$. That is, $M^k\in\Sigma(R,C;W^{-ij})$. This proves (a). Similarly if $z_{ij}>0$ then there exists k such that $m_{ij}^k=1$. That is, $(M^k)^{-ij}\in\Sigma(R^{-i},C^{-j};W^{-ij})$. This proves (b). □

The previous lemma is key to guarantee that line 16 of Algorithm 1 is always feasible. It tells us that if the maximum entropy probability $z_{ij} > 0$ then, the algorithm can generate the corresponding edge with positive probability and the polytope with the reduced degree sequences will be feasible for the next step. The logic for when $z_{ij} < 1$ is analogous.

We can now proceed and prove the termination and correctness of Algorithm 1.

Proof. First, we prove that the algorithm runs to completion, that is, line 16 is always feasible. Since we assume that $\Sigma(R,C)$ is non-empty, this implies that $\mathscr{P}(R,C)$ is non-empty and thus the problem in Equation 1 is feasible. That is, we can initially compute $Z(R,C;W)$. Now, suppose that at any iteration of the algorithm we have that $\mathscr{P}\left(\hat{R},\hat{C};W\right)$ is non-empty and that we have computed $Z\left(\hat{R},\hat{C};W\right)$. The algorithm then picks (i,j). If $z_{ij} = 0$, then we set $w_{ij} = 0$. By Lemma 3, $\Sigma\left(\hat{R},\hat{C},W^{-ij}\right)$ is non-empty and we can compute $Z\left(\hat{R},\hat{C};W^{-ij}\right)$. If $z_{ij} = 1$, then we set $m_{ij} = 1$, $w_{ij} = 0$ and reduce the degree sequences. Again, by Lemma 3, $\Sigma\left(\hat{R}^{-i},\hat{C}^{-j},W^{-ij}\right)$ is non-empty and we can compute $Z\left(\hat{R}^{-i},\hat{C}^{-j};W^{-ij}\right)$. Finally, if $0 < z_{ij} < 1$, Lemma 3 guarantees that both $\Sigma\left(\hat{R}^{-i},\hat{C}^{-j},W^{-ij}\right)$ and $\Sigma\left(\hat{R},\hat{C},W^{-ij}\right)$ are non-empty and we can compute $Z\left(\hat{R}^{-i},\hat{C}^{-j},W^{-ij}\right)$ if $m_{ij} = 1$ or $Z\left(\hat{R},\hat{C},W^{-ij}\right)$ otherwise. By induction, at the end of each step we have that $\Sigma\left(\hat{R},\hat{C};W\right)$ is non-empty and we can proceed to the next step. Since the number of pairs (i,j) is finite, the algorithm terminates. Finally, we prove that the output $M \in \Sigma(R,C)$. To see this, observe that at each iteration of the algorithm we set $w_{i,j} = 0$ and we update $m_{ij} = 1$ if and only if we reduce the degree sequences. This means that at the end of each iteration we have that if $X \in \mathscr{P}\left(\hat{R},\hat{C};W\right)$, then $M + X \in \mathscr{P}(R,C)$. At the end of the algorithm, we have that $W = 0$ and $\mathscr{P}\left(\hat{R},\hat{C};W\right)$ is non-empty. In particular, it follows that $\hat{R} = \hat{C} = 0$ and that $X = 0$ is the only point in $\mathscr{P}\left(\hat{R},\hat{C};W\right)$. This implies that $M \in \mathscr{P}(R,C)$ and, since $m_{ij} \in \{0,1\}$, we conclude that $M \in \Sigma(R,C)$. □

C PROOF OF LEMMA 2

Proof. The if part is trivial since $Z \in \mathscr{P}(R,C;W)$. For the only if part, suppose that $z_{ij} = \delta$ and that there exists X such that $x_{ij} \neq \delta$. Let $\lambda \in (0,1)$ and consider $Y = (1-\lambda)Z + \lambda X = Z + \lambda(X - Z)$. It is clear that $Y \in \mathscr{P}(R,C;W)$. We then have

$$\begin{aligned}
H(Y) - H(Z) &= \sum_{kl}(h(y_{kl}) - h(z_{kl})) \\
&= \sum_{kl:z_{kl}\in\{0,1\}}(h(y_{kl}) - h(z_{kl})) + \sum_{kl:0<z_{kl}<1}(h(y_{kl}) - h(z_{kl})) \\
&\geq h(y_{ij}) - h(z_{ij}) + \sum_{kl:0<z_{kl}<1}(h(y_{kl}) - h(z_{kl})) \\
&= h'(\xi_{ij}^{\lambda})\lambda(x_{ij} - \delta) + \sum_{kl:0<z_{kl}<1}(h'(z_{kl})\lambda(x_{ij} - z_{ij})) + o(\lambda),
\end{aligned}$$

where the inequality is due to the fact that $h(x) \geq 0$ if $x \in [0,1]$ and $h(x) = 0$ if $x \in \{0,1\}$, and ξ_{ij}^{λ} is a point between z_{ij} and y_{ij}. Since $x_{ij} \neq z_{ij} = \delta$, it follows that $h'(\xi_{ij}^{\lambda})\lambda(x_{ij} - \delta) > 0$ for all $\lambda \in (0,1)$. Also, since $Y \to Z$ as $\lambda \to 0$, we have $h'(\xi_{ij}^{\lambda})\lambda \to 0$ as $\lambda \to 0$. Finally, observe that $|h'(\xi_{ij}^{\lambda})| \to \infty$ as $\lambda \to 0$. With these observations, we conclude that

$$H(Y) - H(Z) \geq f(\lambda) + o(f(\lambda)),$$

with $f(\lambda) > 0$ for all $\lambda \in (0,1)$ and $f(\lambda) \to 0$ as $\lambda \to 0$. This means that there exists $\lambda > 0$ such that $H(Y) > H(Z)$ which contradicts the fact that Z is optimal for the problem in Equation 1. Therefore, we must have $x_{ij} = \delta$ for all X in $\mathscr{P}(R,C;W)$. □

REFERENCES

Anand, K., I. van Lelyveld, Á. Banai, S. Friedrich, R. Garratt, G. Haaj, J. Fique, I. Hansen, S. Martínez Jaramillo, H. Lee, J. L. Molina-Borboa, S. Nobili, S. Rajan, D. Salakhova, T. C. Silva, L. Silvestri, and S. R. Stancato de Souza. 2018. "The Missing Links: A Global Study on Uncovering Financial Network Structures from Partial Data". *Journal of Financial Stability* 35:107–119.

Barvinok, A. 2010. "On the Number of Matrices and a Random Matrix with Prescribed Row and Column Sums and 0–1 Entries". *Advances in Mathematics* 224(1):316–339.

Blitzstein, J., and P. Diaconis. 2010. "A Sequential Importance Sampling Algorithm for Generating Random Graphs with Prescribed Degrees". *Internet Mathematics* 6(4):489–522.

Caccioli, F., M. Shrestha, C. Moore, and J. D. Farmer. 2014. "Stability Analysis of Financial Contagion Due to Overlapping Portfolios". *Journal of Banking & Finance* 46:233–245.

Chatterjee, S., and P. Diaconis. 2018. "The Sample Size Required in Importance Sampling". *The Annals of Applied Probability* 28(2):1099–1135.

Chen, C., G. Iyengar, and C. C. Moallemi. 2014. *Asset-Based Contagion Models for Systemic Risk*. Columbia Business School. https://www8.gsb.columbia.edu/researcharchive/articles/25467.

Eisenberg, L., and T. H. Noe. 2001. "Systemic Risk in Financial Systems". *Management Science* 47(2):236–249.

Gale, D. 1957. "A Theorem on Flows on Networks". *Pacific Journal of Mathematics* 7(2):1073–1082.

Gandy, A., and L. A. M. Veraart. 2017. "A Bayesian Methodology for Systemic Risk Assessment in Financial Networks". *Management Science* 63(12):4428–4446.

Glasserman, P., and H. P. Young. 2016. "Contagion in Financial Networks". *Journal of Economic Literature* 54(3):779–831.

Hałaj, G., and C. Kok. 2013. "Assessing Interbank Contagion Using Simulated Networks". *Computational Management Science* 10(2):157–186.

Papadimitriou, C. H., and K. Steiglitz. 1998. *Combinatorial Optimization: Algorithms and Complexity*. Mineola, New York: Dover Publications, Inc.

Squartini, T., A. Almog, G. Caldarelli, I. van Lelyveld, D. Garlaschelli, and G. Cimini. 2017. "Enhanced Capital-Asset Pricing Model for the Reconstruction of Bipartite Financial Networks". *Physical Review E* 96:032315.

Squartini, T., and D. Garlaschelli. 2011. "Analytical Maximum-Likelihood Method to Detect Patterns in Real Networks". *New Journal of Physics* 13(8):083001.

Wormald, N. C. 1999. "Models of Random Regular Graphs". In *Surveys in Combinatorics, 1999*, edited by J. D. Lamb and D. A. Preece, London Mathematical Society Lecture Note Series, 239–298. Cambridge University Press.

AUTHOR BIOGRAPHIES

PAUL GLASSERMAN is the Jack R. Anderson Professor of Business at Columbia University. He is author of the book *Monte Carlo Methods in Financial Engineering*, which was awarded the 2006 INFORMS Lanchester Prize, and he is a past recipient of the I-Sim Outstanding Simulation Publication Award. Since 2011, he has also worked part-time at the Office of Financial Research in the U.S. Treasury Department on problems of financial stability, which motivated this work. His email address is pg20@columbia.edu.

ENRIQUE LELO DE LARREA is a PhD student in Operations Research at Columbia University. Before joining Columbia, he worked during two years as a credit risk analyst at BBVA Bancomer. He holds a double Bachelor's degree in Applied Mathematics and Actuarial Science from ITAM and a Master's degree in Operations Research from Columbia. His research interests include stochastic simulation, applied probability and financial engineering. His email address is enrique.lelodelarrea@columbia.edu.