Internet Mathematics Vol. 6, No. 4: 489–522

A Sequential Importance Sampling Algorithm for Generating Random Graphs with Prescribed Degrees

Joseph Blitzstein and Persi Diaconis

Abstract. Random graphs with given degrees are a natural next step in complexity beyond the Erdős–Rényi model, yet the degree constraint greatly complicates simulation and estimation. We use an extension of a combinatorial characterization due to Erdős and Gallai to develop a sequential algorithm for generating a random labeled graph with a given degree sequence. The algorithm is easy to implement and allows for surprisingly efficient sequential importance sampling. The resulting probabilities are easily computed on the fly, allowing the user to reweight estimators appropriately, in contrast to some ad hoc approaches that generate graphs with the desired degrees but with completely unknown probabilities. Applications are given, including simulating an ecological network and estimating the number of graphs with a given degree sequence.

1. Introduction

Random graphs with given degrees have recently attracted great interest as a model for many real-world complex networks, allowing for heterogeneity between vertices that is unavailable in the Erdős–Rényi model. The main result of this paper is a new sequential importance-sampling algorithm for generating random

ISSN: 1542-7951 print

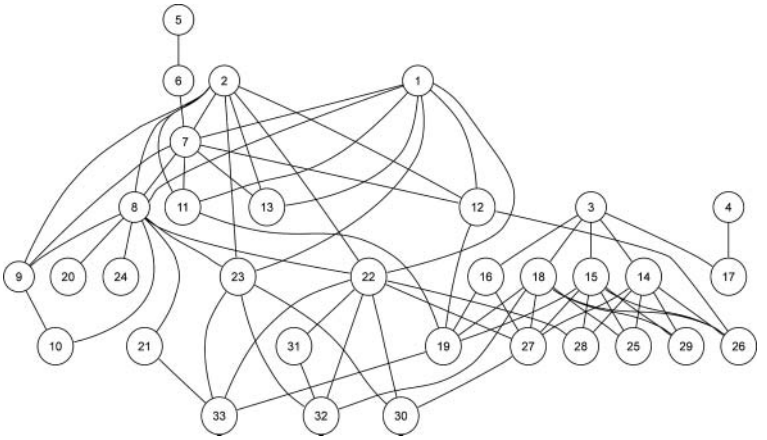

Figure 1. Food web for the Chesapeake Bay ecosystem in summer.

graphs with a given degree sequence. The idea is to build up the graph sequentially (one edge at a time), where each choice has a known probability so that importance sampling can be used to estimate quantities of interest.

Most previously studied algorithms for this problem either sometimes get stuck or produce loops or multiple edges in the output, requiring frequent restarts. Often for such algorithms, the probability of a restart being needed on a trial rapidly approaches 1 as the degree parameters grow, resulting in an enormous number of trials being needed on average to obtain even one graph. A major advantage of our algorithm is that it never gets stuck. This is achieved using an extension of the Erdős–Gallai characterization, which is explained in Section 2, and a carefully chosen order of edge selection. Throughout, we are concerned with generating simple graphs, i.e., no loops or multiple edges are allowed (the problem becomes considerably easier if loops and multiple edges are allowed).

For example, the graph in Figure 1 is the observed food web of 33 types of organism (such as bacteria, oysters, and catfish) in the Chesapeake Bay during the summer. The data are from [Baird and Ulanowicz 89] and are available online [Ulanowicz 05]. Each vertex represents one of the 33 types of organism, and an edge between two vertices indicates that one preys upon the other (see Section 10 for further details about the data and discussion of directed vs. undirected graphs).

The degree sequence of the above graph is

$$d = (7, 8, 5, 1, 1, 2, 8, 10, 4, 2, 4, 5, 3, 6, 7, 3, 2, 7, 6, 1, 2, 9, 6, 1, 3, 4, 6, 3, 3, 3, 2, 4, 4).$$

Applying importance sampling as explained in Section 9, with 100,000 trials, gave $(1.533 \pm 0.008) \times 10^{57}$ as the estimated number of graphs with the same degree sequence d.

A natural way to test properties of the food web is to condition on the degree sequence, generate a large number of random graphs with the same degree sequence, and then see how the actual food web compares. See Section 10 for details of such a conditional test for this example, using 6000 random graphs with degree sequence d generated by our algorithm. An R implementation is available on the first author's web page or by e-mail request. Other applications of our algorithm (based on an earlier version of this work) are given in [Beichl and Cloteaux 08], which studies how closely connected to each other high-degree vertices are, and [Olding and Wolfe 09], which discusses hypothesis testing for network structure (where Erdős–Rényi would be the simplest null model, but usually we want to be able to incorporate some heterogeneity in the vertices).

Section 3 reviews several previous algorithms for our problem. Section 4 presents our algorithm, and Section 5 gives a proof that the algorithm works. The algorithm relies on an extension of the Erdős–Gallai theorem to handle the case in which certain edges are *forced* (required to be used); this is discussed in Section 6. This is followed in Section 7 by some estimates of the running time.

The probability of a given output for our algorithm is explicitly computable. The output distribution is generally nonuniform, but the random graphs produced can be used to simulate a general distribution via importance sampling. These ideas are discussed in Section 8, which describes sequential importance sampling and its practical and theoretical performance (many open problems remain about rigorous bounds on the variance). Applications to approximately enumerating graphs with a given degree sequence are then given in Section 9. Lastly, in Section 10 we describe an exponential family in which the degrees are sufficient statistics, and use the algorithm to test this model on the food web example.

2. Graphical Sequences

Definition 2.1. A finite sequence $(d_1, \ldots, d_n)$ of nonnegative integers $n \geq 1$ is called *graphical* if there is a labeled simple graph (no loops or multiple edges) with vertex set $\{1, \ldots, n\}$ in which vertex i has degree d_i. Such a graph is called a *realization* of the degree sequence $(d_1, \ldots, d_n)$.

There are many well-known efficient tests of whether a sequence is graphical, e.g., [Mahadev and Peled 95] lists eight equivalent necessary and

sufficient conditions. The most famous such criterion appears in [Erdős and Gallai 60]:

Theorem 2.2. (Erodös–Gallai.) *Let $d_1 \geq d_2 \geq \cdots \geq d_n$ be nonnegative integers with $\sum_{i=1}^{n} d_i$ even. Then $d = (d_1, \ldots, d_n)$ is graphical if and only if*

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(k, d_i) \quad \text{for each } k \in \{1, \ldots, n\}.$$

The necessity of these conditions is easy to see: for any set S of k vertices in a realization of d, there are at most $\binom{k}{2}$ "internal" edges within S, and for each vertex $v \notin S$, there are at most $\min(k, \deg(v))$ edges from v into S. Note that the Erdős–Gallai conditions can be checked using $\Theta(n)$ arithmetic operations and comparisons and $\Theta(n)$ space, since we can compute and cache the n partial sums of the degrees and for each k, the largest i with $\min(k, d_i) = k$ (if any exists), and there are n inequalities to check. Also, d needs to be sorted initially if not already given in that form. This can be done in $O(n \log n)$ time using a standard sorting algorithm.

Instead of having to test all of the inequalities in the Erdős–Gallai conditions, it is often convenient to have a recursive test. A particularly simple recursive test has been found [Havel 55, Hakimi 62]. We include a proof for motivation and future use.

Theorem 2.3. (Havel–Hakimi.) *Let d be a proposed degree sequence of length $n \geq 2$ and let i be a coordinate with $d_i > 0$. If d does not have at least d_i positive entries other than i, then d is not graphical. Assume that there are at least d_i positive entries other than at i. Let $\tilde{d}$ be the degree sequence of length $n-1$ obtained from d by deleting coordinate i and subtracting 1 from the d_i coordinates in d of highest degree, aside from i itself. Then d is graphical if and only if $\tilde{d}$ is graphical. Moreover, if d is graphical, then it has a realization in which vertex i is joined to any choice of the d_i highest-degree vertices other than i.*

Proof. If d does not have at least d_i positive entries other than i, then there are not enough vertices to attach i to, so d is not graphical. So assume that there are at least d_i positive entries aside from i. It is immediate that if $\tilde{d}$ is graphical, then d is graphical: take a realization of $\tilde{d}$ (with labels $(1, 2, \ldots, i-1, i+1, \ldots, n)$), introduce a new vertex labeled i, and join i to the d_i vertices whose degrees had 1 subtracted from them.

Conversely, assume that d is graphical, and let G be a realization of d. Let $h_1, \ldots, h_{d_i}$ be a choice of the d_i highest-degree vertices other than vertex i (so $\deg(h_j) \geq \deg(v)$ for all $v \notin \{i, h_1, \ldots, h_{d_i}\}$). We are done if vertex i is already joined to all of the h_j, since then we can delete vertex i and its edges to obtain a realization of $\tilde{d}$. So assume that there are vertices v and w (not equal to i) such that $w = h_j$ for some j, v is not equal to any of $h_1, \ldots, h_{d_i}$, and i is adjacent to v but not to w.

If $\deg(v) = \deg(w)$, we can interchange vertices v and w without affecting any degrees. So assume that $\deg(v) < \deg(w)$. Then there is a vertex $x \neq v$ joined by an edge to w but not to v. Perform a *switching* by adding the edges $\{i, w\}, \{x, v\}$ and deleting the edges $\{i, v\}, \{w, x\}$. This does not affect the degrees, so we still have a realization of d. Repeating this if necessary, we can obtain a realization of d in which vertex i is joined to the d_i highest-degree vertices other than i itself. □

The Havel–Hakimi theorem thus gives a recursive test for whether d is graphical: apply the theorem repeatedly until either the theorem reports that the sequence is not graphical (if there are not enough vertices available to connect to some vertex) or the sequence becomes the zero vector (in which case d is graphical). In practice, this recursive test runs very quickly, since there are at most n iterations, each consisting in setting a component i to 0 and subtracting 1 from d_i components. The algorithm also needs to find the highest degrees at each stage, which can be done by initially sorting d (in time $O(n \log n)$) and then maintaining the list in sorted order.

Note that when d is graphical, the recursive application of Havel–Hakimi constructs a realization of d, by adding at each stage the edges corresponding to the change in the d vector. This is a simple algorithm for generating a deterministic realization of d. In the next section, we survey previous algorithms for obtaining random realizations of d.

3. Previous Algorithms

Several methods have been developed for generating random graphs with given degrees, given degree distributions, or given expected degrees (see [Chung and Lu 2002b, Chung and Lu 2002a] for the latter), each with its own strengths and limitations. Here we focus on graphs with given degrees; note, though, that if we wish to obtain a given degree distribution or given expected degrees, we may first sample a random degree sequence with the desired properties, and then (if graphical) use this as input to our algorithm. [Arratia and Liggett 05] gives the

asymptotic probability that an i.i.d. degree sequence is graphical. In particular, the asymptotic probability is 1/2 if the distribution has finite mean and is not supported on only even degrees or on only odd degrees.

There are two main categories of such algorithms: random matchings methods, and Markov chain Monte Carlo methods. (An interesting algorithm not fitting in either of these categories can be found in [Tinhofer 79, Tinhofer 90]. Tinhofer's approach involves choosing random adjacency lists combined with an accept/reject procedure, but this algorithm has resisted theoretical analysis.)

3.1. Pairing Model Algorithms

The *pairing model* (also known as the *configuration model* or *stubs model*) generates random multigraphs using random matchings: imagine "stubs" emanating from each vertex, and randomly tying together pairs of stubs to form edges. This algorithm has been rediscovered many times (see [Wormald 99, Bollobás 80, Bender and Canfield 78]). The resulting multigraph is not uniformly distributed, but conditional on it being a simple graph; it is uniformly distributed over graphs with the given degrees. Unfortunately, as d increases, the probability of having loops or multiple edges approaches 1 very rapidly. Bender and Canfield showed that for d-regular graphs,

$$P(\text{simple}) \sim e^{(1-d^2)/4} \quad \text{as } n \to \infty.$$

To obtain more simple graphs, an obvious (but hard to analyze) modification is to forbid any pairing that induces a loop or multiple edges, as studied in [Steger and Wormald 99]. This can easily get stuck. Steger and Wormald showed that for d-regular graphs, the probability of getting stuck is small for $d = o((n/(\log^3 n)^{1/11})$, and this bound in improved in [Kim and Vu 04] to $d = o(n^{1/3}/\log^2 n)$. For more general degrees, little is known about the probability of getting stuck or about the algorithm's (nonuniform) distribution on graphs.

3.2. Markov Chain Monte Carlo Algorithms

Several approaches using Markov chain Monte Carlo (MCMC) have been proposed, often using the same *switching move* as in the proof of the Havel–Hakimi theorem: from a graph G, pick two random edges $\{x, y\}$ and $\{u, v\}$ uniformly with x, y, u, v distinct such that $\{x, u\}$ and $\{y, v\}$ are not edges, and then add the edges $\{x, u\}, \{y, v\}$ and delete the edges $\{x, y\}, \{u, v\}$. [Cooper et al. 07] studies the mixing time of this chain for regular graphs. The corresponding random walk on adjacency matrices is a well-known random walk on tables (see [Diaconis and

Gangolli 95, Diaconis and Sturmfels 98]). For regular graphs, another MCMC algorithm was given in [Jerrum and Sinclair 90], based on perfect matchings in a specially constructed graph.

For all of these chains, theoretical analysis is currently available only under severe restrictions on the degrees (such as assuming that they are all equal), and even then the mixing times are bounded by high-degree polynomials, which are unlikely to be useful in practice for knowing how long to run the chain.

In short, we do not know any previously published algorithm that both works well in practice for the counting and estimation problems discussed here and has established theoretical properties for a wide range of degree sequences. Thus, we use *characterizations* to avoid getting stuck and *sequential importance sampling* to account for the probability distribution on graphs.

4. The Sequential Algorithm

We now present our sequential algorithm and describe some of its features. In this discussion, degrees should be thought of as *residual degrees* unless otherwise specified (how many edges still need to be chosen for each vertex); in particular, we will start with the degree sequence d and add edges until the degree sequence is reduced to 0. Since we will be frequently adding or subtracting 1 at certain coordinates of degree sequences, we introduce some notation for this operation.

Notation. 4.1. For any vector $d = (d_1, \ldots, d_n)$ and distinct $i_1, \ldots, i_k \in \{1, \ldots, n\}$, define $\oplus_{i_1,\ldots,i_k} d$ to be the vector obtained from d by adding 1 at each of the coordinates $i_1, \ldots, i_k$, and define $\ominus_{i_1,\ldots,i_k}$ analogously:

$$(\oplus_{i_1,\ldots,i_k} d)_i = \begin{cases} d_i + 1 & \text{for } i \in \{i_1, \ldots, i_k\}, \\ d_i & \text{otherwise,} \end{cases}$$

and

$$(\ominus_{i_1,\ldots,i_k} d)_i = \begin{cases} d_i - 1 & \text{for } i \in \{i_1, \ldots, i_k\}, \\ d_i & \text{otherwise.} \end{cases}$$

For example, $\oplus_{1,2}(3,2,2,1) = (4,3,2,1)$ and $\ominus_{1,2}(3,2,2,1) = (2,1,2,1)$. With this notation, our sequential algorithm can be stated compactly; see Algorithm 1.

For example, suppose that the starting sequence is $(3,2,2,2,1)$. The algorithm starts by choosing which vertex to join vertex 5 to, using the candidate list $\{1,2,3,4\}$. Say it chooses 2. The new degree sequence is $(3,1,2,2,0)$. The degree

Algorithm 1: Sequential algorithm for random graph with given degrees.

Input: a graphical degree sequence $(d_1, \ldots, d_n)$.

1. Let E be an empty list of edges.
2. If $d = 0$, terminate with output E.
3. Choose the least i with d_i a minimal positive entry.
4. Compute candidate list $J = \{j \neq i : \{i, j\} \notin E \text{ and } \ominus_{i,j}\, d \text{ is graphical}\}$.
5. Pick $j \in J$ with probability proportional to its degree in d.
6. Add the edge $\{i, j\}$ to E and update d to $\ominus_{i,j} d$.
7. Repeat steps 4–6 until the degree of i is 0.
8. Return to step 2.

Output: E.

of vertex 5 is now 0, so the algorithm continues with vertex 2, etc. One possible sequence of degree sequences is

$$(3,2,2,2,1) \to (3,1,2,2,0) \to (2,0,2,2,0) \to (1,0,2,1,0) \to (0,0,1,1,0) \\ \to (0,0,0,0,0),$$

corresponding to the graph with edge set $\{\{5,2\},\{2,1\},\{1,4\},\{1,3\},\{3,4\}\}$. As another example, we show in Figure 2 the first two of 6000 graphs generated using our algorithm applied to the degree sequence of the food web of Figure 1. Each took about 13 seconds to generate (on a 1.33-GHz PowerBook). Qualitatively, they appear to be more spread out and less hierarchical than the actual food web. We discuss this more in Section 10, comparing test statistics of the actual graph with those of the random graphs.

Algorithm 1 always terminates in a realization of $(d_1, \ldots, d_n)$. The output of the algorithm is not uniformly distributed over all realizations of $(d_1, \ldots, d_n)$ in general, but every realization of $(d_1, \ldots, d_n)$ has positive probability. importance-sampling techniques can then be used to compute expected values with respect to the uniform distribution (or a different desired distribution), as described in Section 8. For the simpler problem of generating random labeled *trees* with given degrees, the analogous algorithm (with suitable selection probabilities) turns out to be *exactly* uniform.

We now make remarks on the specific steps and some ways of speeding up the implementation of the algorithm.

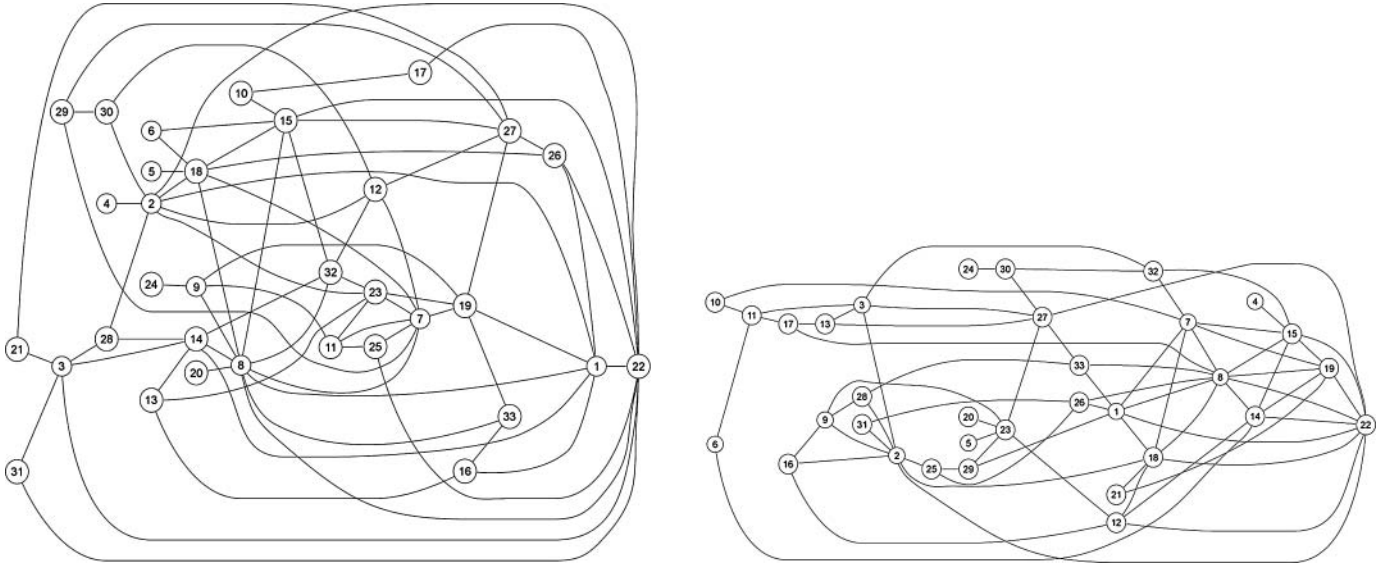

Figure 2. Random graphs with food web degrees.

1. In step 4, any test for graphicality can be used; Erdős–Gallai is particularly easy to implement and runs quickly.

2. It follows from Theorem 5.1 that a candidate at a later stage is also a candidate at an earlier stage, within the same choice of i from step 3. Thus, in step 4 it suffices to test candidates from the previous stage if that stage had the same i.

3. Let $m = |\{j : d_j \geq j - 1\}|$ be the *corrected Durfee number* of d. A beautiful result (see [Mahadev and Peled 95, Theorem 3.4.1]) is that d is graphical if and only if it satisfies the first m Erdős–Gallai inequalities. In many cases, the corrected Durfee number m is much less than n, greatly reducing the number of inequalities to check.

4. In step 5, any probability distribution p on J with $p(j) > 0$ for all j can be used. An interesting problem here is to find a distribution p that makes the output as close to uniform as possible. In our empirical tests, choosing a candidate with probability proportional to its degree was significantly better than choosing a candidate uniformly (see Section 8). But it remains open to prove some sort of optimality here.

5. An alternative to step 4 is to pick j with $\{i, j\} \notin E$ randomly and accept it if $\ominus_{i,j} d$ is graphical; otherwise, pick again without replacement. This approach runs faster, but has the disadvantage that it becomes very difficult to compute the probabilities discussed in Section 8, which are required for importance sampling.

5. Proof that the Algorithm Works

The theorem below guarantees that the algorithm never gets stuck, by showing that at least one candidate vertex always exists.

Theorem 5.1. *Let $d = (d_1, \ldots, d_n)$ be a graphical degree sequence with $d_i > 0$, arranged so that $d_n = \min\{d_1, \ldots, d_n\}$. Let $d = d^{(0)}, d^{(1)}, d^{(2)}, \ldots, d^{(j)} = \tilde{d}$ be graphical degree sequences of length n (for some $j \geq 1$) such that $d^{(i)}$ is obtained from $d^{(i-1)}$ by subtracting 1 at coordinate n and at another coordinate v_i not previously changed. That is,*

$$d^{(i)} = \ominus_{n,v_i} d^{(i-1)} \quad \text{for } i \in \{1, \ldots, j\},$$

where $n, v_1, \ldots, v_j$ are distinct. Then d has a realization containing all of the edges $\{n, v_1\}, \ldots, \{n, v_j\}$, and $\tilde{d}$ has a realization containing none of these edges.

Proof. The desired realization of d immediately yields that of $\tilde{d}$, by deleting the edges $\{n, v_1\}, \ldots, \{n, v_j\}$, and conversely, the desired realization of $\tilde{d}$ immediately gives that of d. Note that $j \leq d_n$, since the degree of vertex n in $d^{(j)}$ is $d_n - j$. We use a backward induction on j, over $1 \leq j \leq d_n$, by showing that the claim is true for $j = d_n$ and that if it is true for $j + 1$, then it is true for j.

First assume $j = d_n$, and let G_j be a realization of the degree sequence $d^{(j)}$. Note that vertex n has degree 0 in G_j. Adding edges in G_j from vertex n to each v_i, $1 \leq i \leq j$, yields a graph with degree sequence $d^{(0)}$ containing the desired edges. Now assume that the result holds for $j + 1$, and prove it for some fixed j with $1 \leq j \leq d_n - 1$.

Call the vertices $v_1, \ldots, v_j$ *touched* vertices, and the remaining vertices other than vertex n *untouched*. The proof hinges on whether we can find a realization of $d^{(j)}$ such that vertex n is adjacent to an untouched vertex.

Suppose that $d^{(j)}$ has a realization G_j containing an edge $\{n, x\}$, with x an untouched vertex. Deleting the edge $\{n, x\}$ yields a graph G_{j+1} with degree sequence $d^{(j+1)}$ of the form in the statement of the theorem, with $j + 1$ in place of j and $v_{j+1} = x$. The inductive hypothesis then implies that $d^{(0)}$ has a realization containing the edges $\{n, v_1\}, \ldots, \{n, v_{j+1}\}$. So it suffices to show that $d^{(j)}$ has a realization with an edge $\{n, x\}$, with x an untouched vertex.

Let T consist of the j touched vertices, and decompose $T = A \cup B$, where a touched vertex x is in A iff $\{n, x\}$ is an edge in some realization of $d^{(j)}$, and $B = T \setminus A$. Here B may be empty, but we can assume that A is nonempty, since otherwise any realization of $d^{(j)}$ has an edge $\{n, x\}$ with x untouched. Let $|A| = a$ and $|B| = b$ (so $a + b = j$).

Let $x \in A$ and let y be an untouched vertex or $y \in B$. Consider a realization of $d^{(j)}$ containing the edge $\{n, x\}$. Note that if the degrees of x and y are equal in $d^{(j)}$, then they can be interchanged without affecting any degrees, and then $\{n, y\}$ is an edge (which contradicts the definition of B if $y \in B$, and gives us the desired edge if y is untouched). If the degree of x is less than that of y, then we can perform a switching as in the proof of the Havel–Hakimi theorem (pick a vertex $w \neq x$ adjacent to y but not adjacent to x, add edges $\{n, y\}, \{x, w\}$, and delete edges $\{n, x\}, \{y, w\}$), again producing a realization with the edge $\{n, y\}$. So assume that the degree of x is strictly greater than the degree of y in $d^{(j)}$ for each $x \in A$ and y that is either untouched or in B. Then the vertices in A have the a highest degrees (excluding n itself) in $d^{(j)}$.

Let d' be the degree sequence with $d'_n = d_n - b$, $d'_y = d_y - 1$ for $y \in B$, and $d'_y = d_y$ otherwise. Note that $\tilde{d}_i \leq d'_i \leq d_i$ for all i, with equality on the left for $i \in B$ and equality on the right for $i \in A$. Also, the assumption $d_n = \min\{d_1, \ldots, d_n\}$ implies that $d'_n = \min\{d'_1, \ldots, d'_n\}$ and $d_n^{(j)} = \min\{d_1^{(j)}, \ldots, d_n^{(j)}\}$. We claim that

d' is graphical. Assuming that d' is graphical, we can then complete the proof as follows. Note that the vertices in A have the a highest degrees in d' (excluding n itself), since this is true for $d^{(j)}$ and in passing from $d^{(j)}$ to d', these degrees are increased by 1, while all other degrees aside from that of vertex n are unchanged. So by the Havel–Hakimi theorem, d' has a realization containing all of the edges $\{n, x\}$, $x \in A$ (since $a \leq d'_n = d_n - b$ because $j < d_n$). Deleting these edges yields a realization $G^{(j)}$ of $d^{(j)}$ containing none of these edges. By definition of B, $G^{(j)}$ also does not contain any edge $\{n, y\}$ with $y \in B$. Thus, $G^{(j)}$ is as desired. So it suffices to prove that d' is graphical.

To show that d' is graphical, we verify the Erdős–Gallai conditions. For $k = n$, since d is graphical, we have

$$\sum_{i=1}^{n} d'_i \leq \sum_{i=1}^{n} d_i \leq n(n-1).$$

Assume $k < n$, and let $I \subseteq \{1, \ldots, n-1\}$ be an index set for the k largest degrees of d'. If $k \leq d'_n$, then $k \leq d'_i \leq d_i$ for all i, and we have

$$\begin{aligned}\sum_{i \in I} d'_i &\leq \sum_{i \in I} d_i \leq k(k-1) + \sum_{i \notin I} \min\{k, d_i\} = k(k-1) + \sum_{i \notin I} k \\ &= k(k-1) + \sum_{i \notin I} \min\{k, d'_i\}.\end{aligned}$$

So assume $k > d'_n$ (which implies $k > \tilde{d}_n$). Then

$$\sum_{i \in I} d'_i = a' + \sum_{i \in I} \tilde{d}_i,$$

where $a' \leq a$, since d' and $\tilde{d}$ differ only on $A \cup \{n\}$. Since $\tilde{d}$ is graphical,

$$\begin{aligned}a' + \sum_{i \in I} \tilde{d}_i &\leq a' + k(k-1) + \sum_{i \notin I} \min\{k, \tilde{d}_i\} \\ &= a' + k(k-1) + \tilde{d}_n + \sum_{i \notin I, i \neq n} \min\{k, \tilde{d}_i\} \\ &\leq a' + k(k-1) + \tilde{d}_n + \sum_{i \notin I, i \neq n} \min\{k, d'_i\} \\ &= a' + k(k-1) + \tilde{d}_n - d'_n + \sum_{i \notin I} \min\{k, d'_i\} \\ &\leq k(k-1) + \sum_{i \notin I} \min\{k, d'_i\},\end{aligned}$$

where the last inequality follows from $a' + \tilde{d}_n - d'_n = a' + (d_n - j) - (d_n - b) = a' - a \leq 0$. Hence, d' is graphical. □

Using the above theorem, we can now prove that the algorithm never gets stuck.

Corollary 5.2. *Given a graphical sequence $d = (d_1, \ldots, d_n)$ as input, Algorithm 1 terminates with a realization of d. Every realization of d occurs with positive probability.*

Proof. We use induction on the number of nonzero entries in the input vector d. If $d = 0$, the algorithm terminates immediately with empty edge set, which is obviously the only realization of the zero vector. Suppose $d \neq 0$ and the claim is true for all input vectors with fewer nonzero entries than d. Let i be the smallest index in d with minimal positive degree. There is at least one candidate vertex j to connect i to, since if $\{i, j\}$ is an edge in a realization of d, then deleting this edge shows that the sequence $d^{(1)}$ obtained by subtracting 1 at coordinates i and j is graphical.

Suppose that the algorithm has chosen edges $\{i, v_1\}, \ldots, \{i, v_j\}$ with corresponding degree sequences $d = d^{(0)}, d^{(1)}, \ldots, d^{(j)}$, where $j \geq 1$ and $d_i^{(j)} = d_i - j > 0$. Omitting any zeros in d and permuting each sequence to put vertex i at coordinate n, Theorem 5.1 implies that $d^{(j)}$ has a realization G_j using none of the edges $\{i, v_1\}, \ldots, \{i, v_j\}$. Then G_j has an edge $\{i, x\}$ with $d_x = d_x^{(j)}$, and $\{i, x\}$ is an allowable choice for the next edge. Therefore, the algorithm can always extend the list of degree sequences $d = d^{(0)}, \ldots, d^{(j)}$ until the degree at i is $d_i - j = 0$. Let $\{i, v_1\}, \ldots \{i, v_j\}$ be the edges selected (with $d_i - j = 0$). Note that if $\{i, w_1\}, \ldots, \{i, w_j\}$ are the edges incident with i in a realization of G, then these edges are chosen with positive probability (as seen by deleting these edges one by one in any order).

The algorithm then proceeds by picking a minimal positive entry in $d^{(j)}$ (if any remains). By the inductive hypothesis, running the algorithm on input vector $d^{(j)}$ terminates with a realization of $d^{(j)}$. Thus, the algorithm applied to d terminates with edge set $E = \{\{i, v_1\}, \ldots, \{i, v_j\}\} \cup \tilde{E}$, where $\tilde{E}$ is an output edge set of the algorithm applied to $d^{(j)}$. No edges in $\tilde{E}$ involve vertex i, so E is a realization of d. Again by the inductive hypothesis, every realization of $d^{(j)}$ is chosen with positive probability, and it follows that every realization of d is chosen with positive probability. □

Remark 5.3. Many seemingly similar algorithms frequently get stuck, in the sense that even though graphicality is maintained by the choices at each stage, one can reach a sequence from which one would be forced to restart (or allow loops or multiple edges). For example, the above result is false if the "minimal positive

degrees" rule is dropped. For a counterexample, consider the degree sequences $(1,1,2,2,5,3), (1,1,2,2,4,2), (0,1,2,2,4,1)$ and note that there is no realization of the sequence $(1,1,2,2,5,3)$ containing the edge $\{1,6\}$.

6. Forced Sets of Edges

Theorem 5.1 is also related to the problem of finding a realization of a graph that requires or forbids certain edges. To make this precise, we introduce the notion of a forced set of edges.

Definition 6.1. Let d be a graphical degree sequence. A set F of pairs $\{i,j\}$ with $i,j \in \{1,\dots,n\}$ is *forced* for d if for every realization $G=(V,E)$ of d, $F \cap E \neq \varnothing$. If a singleton $\{e_1\}$ is forced for d, we will say that e_1 is a forced edge for d.

The one-step case ($j=1$) in Theorem 5.1 gives a criterion for an edge to be forced. Indeed, for this case the assumption about the minimum is not needed.

Proposition 6.2. *Let d be a graphical degree sequence and $i,j \in \{1,\dots,n\}$ with $i \neq j$. Then $\{i,j\}$ is a forced edge for d if and only if $\oplus_{i,j} d$ is not graphical.*

Proof. Suppose that $\{i,j\}$ is not forced for d. Adding the edge $\{i,j\}$ to a realization of d yields a realization of $\oplus_{i,j} d$. Conversely, suppose that $\{i,j\}$ is forced for d. Arguing as in the proof of Theorem 5.1, we see that i and j must have greater degrees in d than any other vertex. Suppose (for contradiction) that $\oplus_{i,j} d$ is graphical. Then Havel–Hakimi gives a realization of $\oplus_{i,j} d$ that uses the edge $\{i,j\}$. Deleting this edge gives a realization of d not containing the edge $\{i,j\}$, contradicting it being forced for d. □

Beyond this one-step case, an additional assumption is needed for analogous results on forced sets, as shown by the counterexample after the proof of Theorem 5.1. Much of the proof consisted in showing that the set of all "touched" vertices is not a forced set for $\tilde{d}$. This immediately yields the following result.

Corollary 6.3. *Let d be a graphical degree sequence and let $d' = \oplus_i^k \oplus_{j_1,\dots,j_k} d$ for some distinct vertices $i, j_1, \dots, j_k$. Suppose that $d'_i = \min\{d'_1, \dots, d'_n\}$. Then the set of edges $\{\{i,j_1\},\dots,\{i,j_k\}\}$ is forced for d if and only if d' is not graphical.*

We may also want to know whether there is a realization of d containing a certain list of desired edges. This leads to the following notion.

Definition 6.4. Let d be a graphical degree sequence. A set S of pairs $\{i, j\}$ with $i, j \in \{1, \ldots, n\}$ is *simultaneously allowable* for d if d has a realization G with every element of S an edge in G. If S is a simultaneously allowable singleton, we call it an *allowable edge*.

Results on forced sets imply dual results for simultaneously allowable sets and vice versa, by adding or deleting the appropriate edges from a realization. For example, Proposition 6.2 implies the following.

Corollary 6.5. *Let d be a graphical degree sequence and $i, j \in \{1, \ldots, n\}$ with $i \neq j$. Then $\{i, j\}$ is an allowable edge for d if and only if $\ominus_{i,j} d$ is graphical.*

Similarly, the dual result to Corollary 6.3 is the following (which is also an easy consequence of Theorem 5.1).

Corollary 6.6. *Let d be a graphical degree sequence and let $\tilde{d} = \ominus_i^k \ominus_{j_1,\ldots,j_k} d$ for some distinct vertices $i, j_1, \ldots, j_k$. Suppose that $d_i = \min\{d_1, \ldots, d_n\}$. Then $\{\{i, j_1\}, \ldots, \{i, j_k\}\}$ is simultaneously allowable for d if and only if $\tilde{d}$ is graphical.*

7. Running Time

In this section, we examine the running time of the algorithm. Let $d = (d_1, \ldots, d_n)$ be the input. Since no restarts are needed, the algorithm has a fixed, bounded worst-case running time. Each time a candidate list is generated, the algorithm performs $O(n)$ easy arithmetic operations (adding or subtracting 1) and tests for graphicality $O(n)$ times. Each test for graphicality can be done in $O(n)$ time by Erdős–Gallai, giving a total worst-case $O(n^2)$ running time each time a candidate list is generated (a sorted degree sequence also needs to be maintained, but the total time needed for this is dominated by the $O(n^2)$ time we already have).

Since a candidate list is generated each time an edge is selected, there are $\frac{1}{2}\sum_{i=1}^n d_i$ candidate lists to generate. Additionally, the algorithm sometimes needs to locate the smallest index of a minimal nonzero entry, but we are already assuming that we are maintaining a sorted degree sequence. The overall worst-case running time is then $O(n^2 \sum_{i=1}^n d_i)$. For d-regular graphs, this becomes a worst case of $O(n^3 d)$ time. Note that an algorithm that requires restarts has an unbounded worst-case running time.

8. Importance Sampling

The random graphs generated by our algorithm are not distributed uniformly, but by design it is easy to use importance sampling to estimate expected values and probabilities for any desired distribution, including uniform. Importance sampling allows us to reweight samples from a distribution available to us, called the *trial distribution*, to obtain estimates with respect to the desired *target distribution*. Choosing a good trial distribution is often a difficult problem; one approach to this, sequential importance sampling, involves building up the trial distribution recursively as a product, with each factor conditioned on the previous choices.

Section 8.1 reviews some previous applications of importance sampling and explains how they are related to the current work. Section 8.2 then shows how sequential importance sampling works in the context of random graphs produced by our algorithm, and Section 8.3 discusses some practical and theoretical results on the variances of such estimators, and associated open problems.

For general background on Monte Carlo computations and importance sampling, we recommend [Hammersley and Handscomb 65, Liu 01, Fishman 66]. Sequential importance sampling is developed in [Liu and Chen 98, Doucet et al. 01]. Important variations that could be coupled with the present algorithms include use of control variates [Owen and Zhou 00] to bound the variance and adaptive importance sampling as in [Rubinstein and Kroese 04].

8.1. Some Previous Applications

Sequential importance sampling was used in [Snijders 91] to sample from the set of all zero–one tables with given row and column sums. The table was built up from left to right, one column at a time; this algorithm could get stuck midway through, forcing it to backtrack or restart.

In [Chen et al. 05], the authors introduced the crucial idea of using a combinatorial test to permit only partial fillings that can be completed to full tables. Using this idea, they gave efficient importance-sampling algorithms for zero–one tables and (two-way) contingency tables. For zero–one tables, the combinatorial test is the Gale–Ryser theorem. For contingency tables, the test is simpler: the sum of the row sums and the sum of the column sums must agree. In [Admiraal and Handcock 08], the authors give simulations and software for this, as a tool for generating *bipartite* graphs; here we are concerned with general graphs.

For multiway contingency tables, a sequential importance-sampling algorithm is developed in [Chen et al. 06]. A completely different set of mathematical tools turns out to be useful in this case, including Gröbner bases, Markov bases, and

toric ideals. [Chen 07] gives importance-sampling algorithms for tables in which certain positions are constrained to be 0. In such problems, there is an appealing interplay between combinatorial and algebraic theorems and importance sampling.

Our graph algorithm produces symmetric zero–one tables with trace 0, where the combinatorial test is the Erdős–Gallai characterization. Rather than simplifying the problem, here symmetry completely changes the problem, since it is an additional constraint, requiring different characterizations and a different scheme for filling in the table. For both zero–one tables and graphs, refinements to the combinatorial theorems were needed to ensure that partially filled-in tables could be completed and sequential probabilities computed.

These problems fall under a general program: how can one convert a characterization of a combinatorial structure into an efficient generating algorithm? A refinement of the characterization is often needed, for use with testing whether a partially determined structure can be completed. [Blitzstein 06] develops more algorithms of this nature, including algorithms for generating connected graphs, digraphs, and tournaments.

8.2. Importance Sampling of Graphs

In applying importance sampling with our algorithm, some care is needed because it is possible to generate the same graph in different orders, with different corresponding probabilities. For example, consider the graphical sequence $d = (4, 3, 2, 3, 2)$. The graph with edges

$$\{1,3\}, \{2,3\}, \{2,4\}, \{1,2\}, \{1,5\}, \{1,4\}, \{4,5\}$$

can be generated by the algorithm in any of eight orders. For example, there is the order just given, and there is the order

$$\{2,3\}, \{1,3\}, \{2,4\}, \{1,2\}, \{1,4\}, \{1,5\}, \{4,5\}.$$

The probability of the former is $1/20$, and that of the latter is $3/40$, even though the two sequences correspond to the same graph. This makes it more difficult to directly compute the probability of generating a specific graph, but as shown below, importance sampling can still be used with appropriate weights.

Fix a graphical sequence d of length n as input to the algorithm. We first introduce some notation to clearly distinguish between a graph and a list of edges.

Definition 8.1. Let $\mathcal{G}_{n,d}$ be the set of all realizations of d and let $\mathcal{Y}_{n,d}$ be the set of all possible sequences of edges output by the algorithm. For any sequence

$Y \in \mathcal{Y}_{n,d}$ of edges, let Graph(Y) be the corresponding graph in $\mathcal{G}_{n,d}$ (with the listed edges and vertex set $\{1, \ldots, n\}$). We call $Y, Y' \in \mathcal{Y}_{n,d}$ *equivalent* if *Graph*(Y') = Graph(Y). Let $c(Y)$ be the number of $Y' \in \mathcal{Y}_{n,d}$ with Graph(Y') = Graph(Y).

The equivalence relation defined above partitions $\mathcal{Y}_{n,d}$ into equivalence classes. There is an obvious one-to-one correspondence between the equivalence classes and $\mathcal{G}_{n,d}$, and the size of the class of Y is $c(Y)$. Note that $c(Y') = c(Y)$ if Y' is equivalent to Y. The number $c(Y)$ is easy to compute as a product of factorials:

Proposition 8.2. *Let $Y \in \mathcal{Y}_{n,d}$ and let $i_1, i_2, \ldots, i_m$ be the vertices chosen in the iterations of step 3 of the algorithm in an instance in which Y is output (so the algorithm gives i_1 edges until its degree goes to 0, and then does the same for i_2, etc.). Let $a_1, \ldots, a_m$ be the degrees of $i_1, \ldots, i_m$ respectively, at the time when each was chosen. Then*

$$c(Y) = \prod_{k=1}^{m} a_k! \,.$$

Proof. Let Y' be equivalent to Y. Note from step 3 of the algorithm that Y' has the same vertex choices $i_1, i_2, \ldots, i_m$ as Y. We can decompose Y and Y' into blocks corresponding to $i_1, \ldots, i_m$. Within each block, Y' and Y must have the same set of edges, possibly in a permuted order. Conversely, Theorem 5.1 implies that any permutation that independently permutes the edges within each block of the sequence Y yields a sequence Y'' in $\mathcal{Y}_{n,d}$. Clearly, any such Y'' is equivalent to Y. □

A main goal in designing our algorithm, in addition to not letting it get stuck, is to have a simple formula for $c(Y)$. In many seemingly similar algorithms, it is difficult to find an analogue of the above formula for $c(Y)$, making it much more difficult to use importance sampling efficiently. Before explaining how importance sampling works in this context, a little more notation is needed.

Notation. 8.3. For $Y \in \mathcal{Y}_{n,d}$, write $\sigma(Y)$ for the probability that the algorithm produces the sequence Y. Given a function f on $\mathcal{G}_{n,d}$, write $\hat{f}$ for the induced function on the larger space $\mathcal{Y}_{n,d}$, with $\hat{f}(Y) = f(\text{Graph}(Y))$.

For Y the output of a run of the algorithm, $\sigma(Y)$ can easily be computed sequentially along the way. Each time the algorithm chooses an edge, have it record the probability with which it was chosen (conditioned on all of the previously

chosen edges), namely, its degree divided by the sum of the degrees of all candidates at that stage. Multiplying these probabilities gives the probability $\sigma(Y)$ of the algorithm producing Y.

We now show how to perform importance sampling with the algorithm, adjusting for having a proposal distribution σ distributed on $\mathcal{Y}_{n,d}$ rather than on $\mathcal{G}_{n,d}$.

Proposition 8.4. *Let π be a probability distribution on $\mathcal{G}_{n,d}$ and let G be a random graph drawn according to π. Let Y be a sequence of edges distributed according to σ. Then*

$$E\left(\frac{\hat{\pi}(Y)}{c(Y)\sigma(Y)}\hat{f}(Y)\right) = Ef(G).$$

In particular, for $Y_1, \ldots, Y_N$ the output sequences of N independent runs of the algorithm,

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N}\frac{\hat{\pi}(Y_i)}{c(Y_i)\sigma(Y_i)}\hat{f}(Y_i)$$

is an unbiased estimator of $Ef(G)$.

Proof. Let Y be an output of the algorithm. We can compute a sum over $\mathcal{Y}_{n,d}$ by first summing within each equivalence class and then summing over all equivalence classes:

$$\begin{aligned} E\left(\frac{\hat{\pi}(Y)}{c(Y)\sigma(Y)}\hat{f}(Y)\right) &= \sum_{y\in\mathcal{Y}_{n,d}}\frac{\hat{f}(y)\hat{\pi}(y)}{c(y)\sigma(y)}\sigma(y) = \sum_{y\in\mathcal{Y}_{n,d}}\frac{\hat{f}(y)\hat{\pi}(y)}{c(y)} \\ &= \sum_{G\in\mathcal{G}_{n,d}}\sum_{y:\mathrm{Graph}(y)=G}\frac{\hat{f}(y)\hat{\pi}(y)}{c(y)} = \sum_{G\in\mathcal{G}_{n,d}} f(G)\pi(G) \\ &= Ef(G), \end{aligned}$$

where the second-to-last equality holds because on the set $C(G) = \{y : \mathrm{Graph(y)} = G\}$, we have $\hat{f}(y) = f(G)$, $\hat{\pi}(y) = \pi(G)$, and $c(y) = |C(G)|$. □

Taking π to be uniform and f to be a constant function allows us to estimate the number of graphs with degree sequence d; this is explored in the next section. The ratios

$$W_i = \frac{\hat{\pi}(Y_i)}{c(Y_i)\sigma(Y_i)}$$

are called *importance weights*. A crucial feature of our algorithm is that the quantity $c(Y_i)\sigma(Y_i)$ can easily be computed on the fly, as described above. By

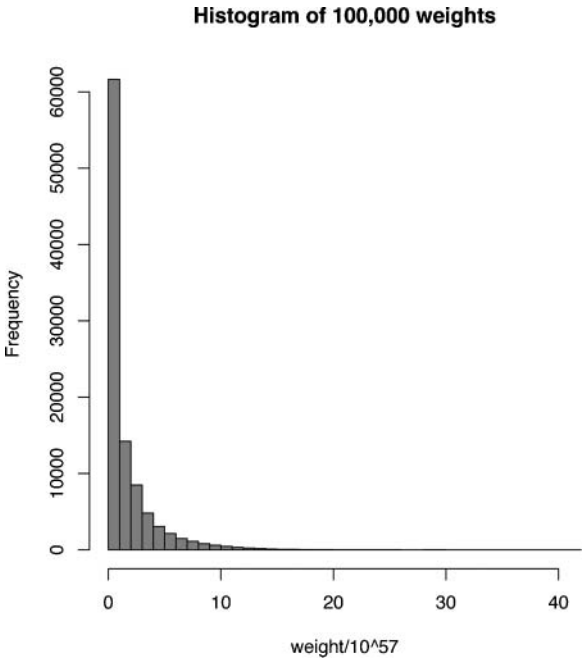

Figure 3. Histogram of 6000 importance weights for the Chesapeake food web.

taking f to be a constant function, we see that $EW_i = 1$, so the average sum of the importance weights in $\hat{\mu}$ is N. Another estimator of $Ef(G)$ is then

$$\tilde{\mu} = \frac{1}{\sum_{i=1}^{N} W_i} \sum_{i=1}^{N} W_i \hat{f}(Y_i).$$

This estimator is biased but has the crucial advantage that the importance weights need be known only up to a constant. Moreover, the variance is typically lower for this estimator, and the law of large numbers ensures consistency.

A major factor in the performance of importance sampling is how much variation there is in the importance weights. Let π be the uniform distribution on $\mathcal{G}_{n,d}$. We plot in Figure 3 a histogram of importance weights for 6000 trials with d the degree sequence of the food web of Figure 1. The weights shown are scaled by dividing by 10^{52} and omitting the constant $\hat{\pi}(Y)$. The weights vary greatly from a minimum of 2.9×10^{52} to a maximum of 2.9×10^{58}, but most are between 1.2×10^{56} and 1.9×10^{57}. The ratio of maximum to median is 52, making the largest few weights influential but not completely dominant.

8.3. What about the Variance?

Obtaining good bounds on the variances of importance-sampling estimators is notoriously difficult, and there is often a wide gulf between theory and practice. Tantalizingly, most examples for which good bounds are currently available are simple or extreme, and far from any typical-looking data one would encounter. In some cases, when the proposal distribution is not good enough, the empirical estimates of the variance are misleading; this is analogous to MCMC sometimes getting stuck in a local mode, and not "knowing" about the rest of the space in any reasonable amount of time. See [Bassetti and Diaconis 05] for some examples explicitly comparing the efficiencies of importance sampling and MCMC.

As an extreme example, consider d of the form $d = (1, \ldots, 1, k)$ with $k \geq 2$, which is the graph analogue of an example in the context of the algorithm in [Chen et al. 05] for generating zero–one tables with given row and column sums. [Bezáková et al. 06] shows that in certain examples with all but one row sum and all but one column sum equal to 1, exponentially many trials are needed. [Blitzstein 06] gives explicit variance calculations for $d = (1, \ldots, 1, k)$ with $l + 2k$ vertices of degree 1. When $l = 1$ and $k = 100$, the relative standard deviation is 2.3×10^{12}, so more than 10^{24} graphs would be needed to obtain an acceptably low standard deviation. Such behavior does not seem characteristic of cases in which the degrees do not behave as wildly, but rigorous bounds are elusive except in simple cases.

In [Bayati et al. 10], the authors analyze a closely related example for graphs with several $\sqrt{n}$ terms in place of the single k, showing that for the Steger–Wormald algorithm the resulting distribution is extremely far from uniform. They then improve on the Steger–Wormald algorithm by reweighting the edge selection process, and use concentration inequalities to show that their algorithm gives approximately uniform graphs when the maximum degree is $O(m^{1/4-\epsilon})$ (for $\epsilon > 0$, with m the number of edges). Their algorithm does not use importance sampling directly, however, though the improved weights suggest possible improvements that can be made in designing a good proposal distribution for importance sampling.

[Blanchet 09] introduces a framework for converting questions about the efficiency of importance-sampling algorithms to questions about rare-event simulation. In particular, for a form of the contingency-table algorithm discussed earlier, Blanchet derives a bound that essentially states that asymptotically, the estimates are accurate with high probability under the conditions that $\max_j c_j = o(s^{1/2})$, $\sum_j c_j^2 = O(s)$, and the r_j are bounded, where r_j and c_j are row and column sums (respectively) and $s = \sum_j c_j$. These constraints are quite restrictive, and it is not yet clear to what extent they can be relaxed, nor whether

his framework can be extended to give bounds for algorithms using characterizations such as Gale–Ryser and Erdős–Gallai. Much further work is needed both for importance sampling and MCMC approaches to such problems, to narrow the gulf between simulation results and theoretical bounds on variances and mixing times respectively, and to give useful guidance as to which method is likely to be more reliable and efficient in a given applied problem. Of course, one way to investigate the questions about the distribution of the estimator is to try it out on a variety of examples where the exact answer is available. We begin this job in the following section.

9. Estimating the Number of Graphs

To estimate the number $|\mathcal{G}_{n,d}|$ of realizations of d, let π be uniform on $\mathcal{G}_{n,d}$ and take f to be the constant function $f(G) = |\mathcal{G}_{n,d}|$. By Proposition 8.4,

$$E\left(\frac{1}{c(Y)\sigma(Y)}\right) = |\mathcal{G}_{n,d}|.$$

Asymptotic formulas for $|\mathcal{G}_{n,d}|$ are available for regular and some nonregular degree sequences (see [Bender and Canfield 78, McKay and Wormald 90, McKay and Wormald 91, Barvinok and Hartigan 10]), but there are few nonasymptotic closed-form expressions.

For the food web example of Figure 1, the estimated size of $\mathcal{G}_{n,d}$ was $(1.51 \pm 0.03) \times 10^{57}$ using 6000 trials. The asymptotic formulas are not of much use in such an example, with a fixed n of moderate size (here $n = 33$).

As an application and test of this method, we estimated the number of labeled 3-regular graphs on n vertices. The exact values for all even $n \leq 24$ are available as Sequence A002829 in Sloane's wonderful "On-Line Encyclopedia of Integer Sequences" [Sloane 05], and can be computed in general with a messy recurrence in [Goulden and Jackson 04].

For $n = 4$, there is only one labeled 3-regular graph on n vertices, the complete graph K_4. Comfortingly, the algorithm does give 1 as its estimate for this case. In general, a degree sequence with exactly one realization is called a *threshold sequence* (see [Mahadev and Peled 95]), and it is easy to see that the algorithm gives 1 as the estimate for any threshold sequence.

Table 1 gives the estimators $\hat{\mu}$ obtained by the trials for all even n between 6 and 24, along with the number of trials, the correct value μ, and the percent error. The number after each $\pm$ indicates the estimated standard error.

For each of these degree sequences, the coefficient of variation (ratio of standard deviation to mean) was approximately 0.4, ranging between 0.39 and 0.43.

n	runs	$\hat{\mu}$	μ	% error
6	500	71.1 ± 1.2	70	1.57
8	500	18964 ± 365	19355	2.06
10	500	$(1.126 \pm 0.021) \times 10^7$	1.118×10^7	0.72
12	500	$(1.153 \pm 0.022) \times 10^{10}$	1.156×10^{10}	0.26
14	500	$(1.914 \pm 0.036) \times 10^{13}$	1.951×10^{13}	1.93
16	500	$(5.122 \pm 0.093) \times 10^{16}$	5.026×10^{16}	1.91
18	500	$(1.893 \pm 0.034) \times 10^{20}$	1.877×10^{20}	0.85
20	500	$(9.674 \pm 0.17) \times 10^{23}$	9.763×10^{23}	0.92
22	500	$(6.842 \pm 0.12) \times 10^{27}$	6.840×10^{27}	0.029
24	500	$(6.411 \pm 0.11) \times 10^{31}$	6.287×10^{31}	1.97

Table 1. Estimators $\hat{\mu}$ obtained by the trials for all even n between 6 and 24, along with the number of trials, the correct value μ, and the percent error. The number after each $\pm$ indicates the estimated standard error.

A measure of the efficiency of an importance sampling scheme is the *effective sample size*, as given in [Kong et al. 94]. The effective sample size approximates the number of i.i.d. samples from the target distribution required to obtain the same standard error as the importance samples. The estimated effective sample sizes for these examples, computed using the coefficients of variation, range between 422 and 434 for 500 runs of the algorithm.

As a comparison between choosing candidates uniformly and choosing candidates with probability proportional to their degrees, we generated 50 estimators from each algorithm, with each based on 100 runs of the algorithm applied to the 3-regular degree sequence with $n = 10$. The true value is $11180820 \approx 1.118 \times 10^7$. The mean of the estimators for uniform candidates was 1.137×10^7 (an error of 1.7%), while that of the degree-based selection was 1.121×10^7 (an error of 0.25%).

For a highly nonregular example, we tested the algorithm with the graphical degree sequence $d = (5, 6, 1, \ldots, 1)$ with eleven 1's. To count the number of labeled graphs with this degree sequence, note that there are $\binom{11}{5} = 462$ such graphs with vertex 1 not joined to vertex 2 by an edge (these graphs look like two separate stars), and there are $\binom{11}{4}\binom{7}{5} = 6930$ such graphs with an edge between vertices 1 and 2 (these look like two joined stars with an isolated edge left over). Thus, the total number of realizations of d is 7392.

Running 500 trials of the algorithm gave the estimate 7176 ± 587, an error of 3%. The algorithm with uniform selection of candidate gave the terrible estimate

of 3702 with 500 trials, indicating the importance of choosing a good distribution on the candidate vertices.

10. An Exponential Family Model

For a labeled graph G with n vertices, let $d_i(G)$ be the degree of vertex i. In the preceding sections, we have kept d_i fixed. In this section, we allow d_i to be random by putting an exponential family model on labeled graphs with n vertices. In this model, the d_i are used as energies or sufficient statistics.

Formally, define a probability measure P_β on the space of all graphs on n vertices by

$$P_\beta(G) = Z^{-1} \exp\left(-\sum_{i=1}^{n} \beta_i d_i(G) \right),$$

where Z is a normalizing constant (depending on β). The real parameters $\beta_1, \ldots, \beta_n$ are chosen to achieve given expected degrees. This model appears explicitly in [Park and Newman 04] using the tools and language of statistical mechanics.

Such models are standard fare in statistics and statistical mechanics. They are extremely widely used in social network analysis (where they are called p^* or ERGMs (exponential random graph models)), having been introduced in this context in [Holland and Leinhardt 81] in the social network context (see also [Anderson et al. 99, Frank and Strauss 86, Newman 03, Robins et al. 07]) as well as [Snijders 02, Snijders et al. 06, Strauss 86]). These models are also increasingly being used in economics (see the recent book [Jackson 08]).

[Holland and Leinhardt 81] gives iterative algorithms for the maximum likelihood estimators, and [Snijders 02] discusses MCMC methods for such models. For the degree-based models considered here, [Chatterjee et al. 11] gives a provably efficient iterative algorithm for computing the MLE of the β_i, and uses this to study graph limits.

One standard motivation for using the probability measure P_β is that this model gives the maximum entropy distribution on graphs with a given expected degree sequence (see [Lauritzen 88] for further discussion of this). In contrast to most other exponential family graph models, the normalizing constant is available here in closed form. Furthermore, there is an easy method of sampling exactly from P_β, as shown by the following easily checked result.

Lemma 10.1. *Fix real parameters $\beta_1, \ldots, \beta_n$. Let Y_{ij} be independent binary random variables for $1 \le i < j \le n$, with*

$$P(Y_{ij} = 1) = \frac{e^{-(\beta_i + \beta_j)}}{1 + e^{-(\beta_i + \beta_j)}} = 1 - P(Y_{ij} = 0).$$

Form a random graph G by creating an edge between i and j if and only if $Y_{ij} = 1$. Then G is distributed according to P_β, with

$$Z = \prod_{1 \le i < j \le n} \left(1 + e^{-(\beta_i + \beta_j)}\right).$$

Note that putting $\beta_i = 0$ results in the uniform distribution on graphs. Also, it follows from Lemma 10.1 that by choosing $\beta_i = \beta$ for all i, for

$$\beta = -\frac{1}{2} \log\left(\frac{p}{1-p}\right),$$

we recover the classical Erdős–Rényi model.

Another model with given expected degrees and edges chosen independently is given in [Chung and Lu 2002b, Chung and Lu 2002a], where an edge between i and j is created with probability $w_i w_j / \sum_k w_k$ (including the case $i = j$), with w_i the desired expected degree of vertex i and $w_i^2 < \sum_k w_k$ for all i. This has the advantage that it is immediate that the expected degree of vertex i is w_i, without any parameter estimation required. The model we are considering here makes it more difficult to choose the β_i, but it yields the maximum entropy distribution, makes the degrees sufficient statistics, and does not require the use of loops. If loops are desired in the exponential model, they may easily be added by allowing $i = j$ in Lemma 10.1. We would like to understand better the precise relationship between the distribution obtained from the Chung and Lu model and the maximum entropy distribution of the exponential family model.

Remark 10.2. The formula for the normalizing constant is equivalent to the identity

$$\prod_{1 \le i < j \le n} (1 + x_i x_j) = \sum_G \prod_{i=1}^{n} x_i^{d_i(G)},$$

where the sum on the right is over all graphs G on vertices $1, \ldots, n$. This identity is closely related to the following symmetric function identity (which is a consequence of Weyl's identity for root systems; see [Macdonald 95, Section I.5, Exercise 9]):

$$\prod_{1 \le i < j \le n} (1 + x_i x_j) = \sum_\lambda s_\lambda,$$

where s_λ is the Schur function corresponding to a partition λ, and the sum on the right ranges over all partitions λ with Frobenius notation of the form $(\alpha_1 - 1 \cdots \alpha_r - 1 \mid \alpha_1 \cdots \alpha_r)$, with $\alpha_1 \leq n-1$. We hope to find (and use) a stochastic interpretation for this Schur function expansion.

The model given by P_β is quite rich; it has n real parameters. Our algorithm can be applied to assess the adequacy of using degrees as sufficient statistics (and not having higher-order terms such as triangle counts). In some problems, a larger exponential family is used (e.g., with parameters corresponding to the number of triangles or other patterns of interest), and the β_j are considered nuisance parameters. A natural approach to eliminating these nuisance parameters is to condition on the degrees, and our algorithm can be used to do this conditional inference.

Given a graph G, we may test adequacy of the P_β model as follows, using the fact that the conditional distribution of a graph G given its degrees $d_1(G), \ldots, d_n(G)$ is uniform over all graphs with this degree sequence. First choose any test statistic T (such as the number of triangles, the diameter, the number of components, or the size of the largest component), and compute $T(G)$. Then generate a large number of uniform graphs with the same degrees as G, and compare the observed value of $T(G)$ against the distribution of T obtained from the sampled random graphs. A closely related approach to testing such a model is to embed it in a larger family and then test whether the new parameters are 0. Indeed, it is shown in [Lehmann and Romano 05, Section 4.4] that our proposed test is optimal (UMP unbiased) in the exponential family model extended by adding $T(G)$ to the sufficient statistic $(d_1(G), \ldots, d_n(G))$.

As an example, we return now to the Chesapeake Bay food web shown in Figure 1. We represent this as an undirected graph for this example, though it is clearly more natural to use a directed graph: it matters whether x eats y or y eats x (especially to x and y). The undirected analysis still reveals information about the connectivity and common substructures in the food web, while analogous algorithms for directed graphs can also be developed (see [Blitzstein 06] for related characterizations and associated algorithms). The blue crab, which is cannibalistic, is represented by vertex 19; we have omitted the loop at vertex 19, not for any moral reason but because we are considering simple graphs.

The graph has 33 vertices, including at least one of every degree from 1 to 10, as illustrated in Figure 4; the degrees vary widely (and do not resemble a power law).

As a first test statistic, we computed the *clustering coefficient* of each graph, which is a measure of transitivity. There are a few different definitions; here we will use the original definition of [Watts and Strogatz 98]: for each vertex v of

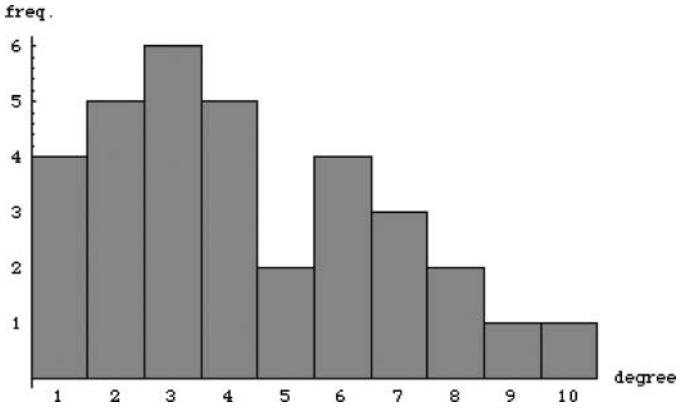

Figure 4. Frequencies of degrees in the Chesapeake food web.

degree $d_v \geq 2$, let C_v be the proportion of edges present between neighbors of v out of the $\binom{d_v}{2}$ possible edges. Put $C_v = 0$ if $d_v < 2$. The clustering coefficient of a graph is then defined to be the average of C_v over all vertices of the graph.

Using the estimator $\tilde{\mu}$ of Section 8.2, the estimated average clustering coefficient for a graph with the same degrees as the food web is 0.157. A histogram of the estimated distribution is shown in Figure 5. The actual clustering coefficient is 0.176, slightly above the mean. Thus, the clustering coefficient is consistent with what the model predicts.

In an attempt to explain and quantify the observation that the real food web is more compact and hierarchical than the vast majority of the random graphs, we examined cycles in the graphs. Specifically, we counted the number of k-cycles in the real food web and the first 1000 random graphs, for $3 \leq k \leq 6$. The cycles

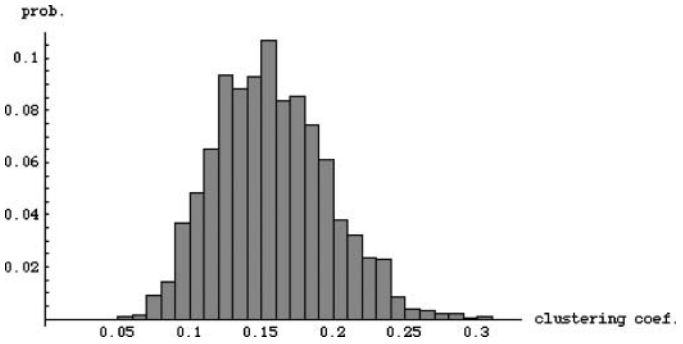

Figure 5. Histogram of clustering coefficients; the real food web value is 0.176.

are treated as unoriented subgraphs of the graph, i.e., a cycle $x \to y \to z \to x$ is considered the same cycle as $y \to z \to x \to y$ and $x \to z \to y \to x$.

The enumeration was done by recursively counting the number of simple paths from v to v, summing over all v, and dividing by $2k$ (since each cycle can be started at any of its k vertices and traversed in either direction). Histograms of the numbers of k-cycles are shown in Figure 6.

For 3-cycles (triangles), the actual value of 18 is extremely close to the estimated mean, which is 19. This is not surprising, especially since the clustering coefficient is closely related to the number of triangles. For 4-cycles, though, the actual value of 119 is nearly double the estimated mean of 60. In fact, the number of 4-cycles in the actual food web is larger than the number of 4-cycles in *any* of the 1000 random graphs tested! Explaining this of course requires looking at the directed graph, but the undirected graph was very helpful in detecting this phenomenon in the first place. Inspecting the corresponding directed subgraphs, two forms are prevalent: (1) x and y both eat z and w and (2) x eats y and z, while y and z eat w. Interestingly, it is observed in [Milo et al. 02] that pattern (2) is extremely common in all seven of the food webs they study. They call a pattern that occurs much more often in a real network than in corresponding random networks a "network motif" (see also [Itzkovitz et al. 03]). Finding network motifs can reveal structure in the network that would be missed by taking a highly degree-centric point of view.

In generating random graphs for comparison purposes, Milo et al. use two algorithms. First, they use a form of the switchings Markov chain discussed in Section 3.2 (adapted for directed graphs), which is not known to be rapidly mixing for general degree sequences. Second, they use a variant of the pairing model, modified from an algorithm in [Newman 01]. Their algorithm can get stuck and has nonuniform output, as pointed out in [King 04]. Our algorithm also has a nonuniform output distribution, but never gets stuck and makes it easy to estimate with respect to the uniform distribution.

Returning to the cycle results, for 5-cycles the actual value is 153, which is significantly lower than the estimated mean of 191. It is at the fifth percentile of the estimated distribution. For 6-cycles, the actual value of 582 is close to the estimated mean of 595. A biological interpretation would be welcome for why 4-cycles are extremely common and 5-cycles are rare, while 6-cycles are close to the middle of the distribution.

10.0.1. A Note on Software. Graphs were drawn using Graphviz [Ellson et al. 03]. The implementation of the algorithm and importance-sampling computations presented here were done using R [R Core Team 05] and Mathematica

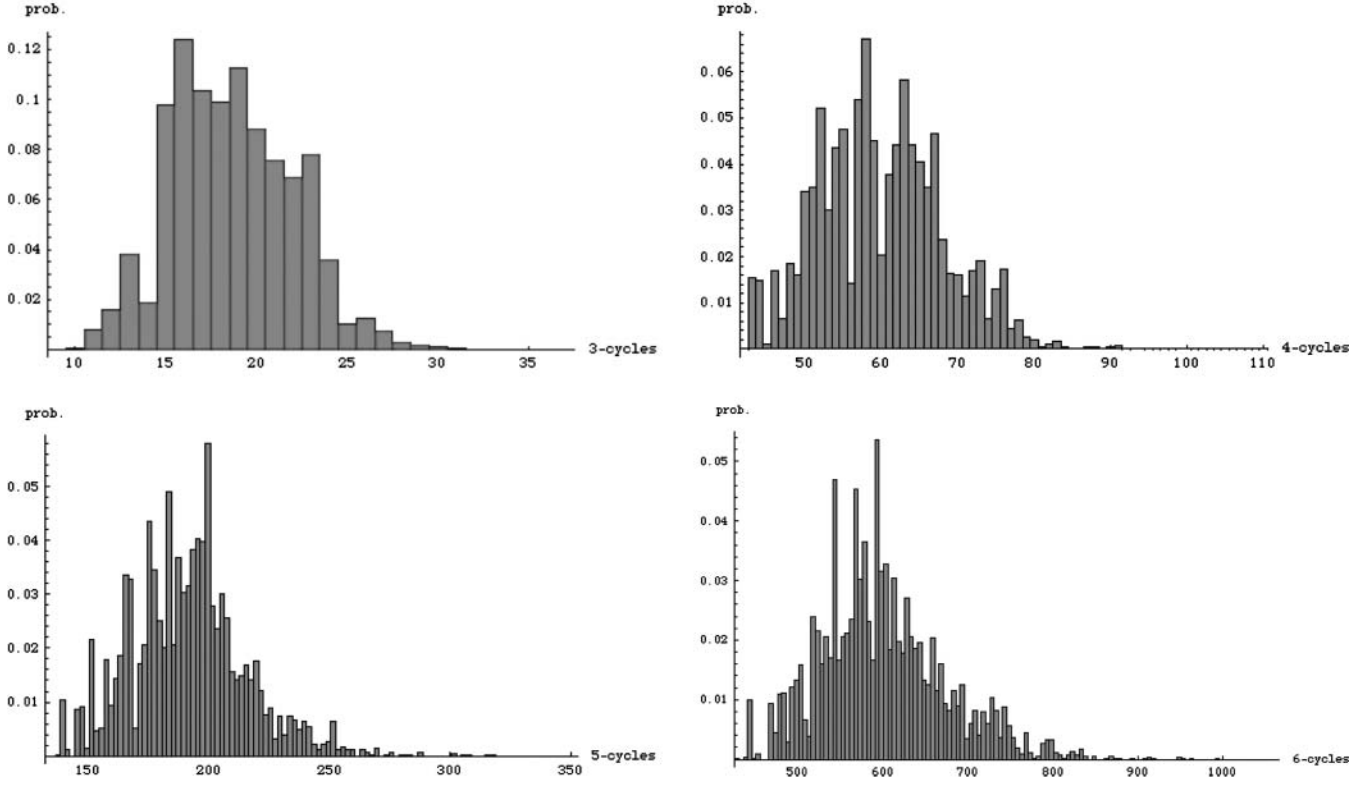

Figure 6. Histograms of k-cycle counts; the real food web has 18, 119, 153, 582, respectively.

[Wolfram Research 01] on a computer running Mac OS X. Code implementing the algorithm is available on the first author's web page or by e-mail request.

Acknowledgments. We thank Jose Blanchet, Sourav Chatterjee, Alex Gamburd, Susan Holmes, Brendan McKay, Amin Saberi, Richard Stanley, Patrick Wolfe, and Nick Wormald for their helpful suggestions and references.

References

[Admiraal and Handcock 08] R. Admiraal and M. S. Handcock. "Networksis: A Package to Simulate Bipartite Graphs with Fixed Marginals through Sequential Importance Sampling." *Journal of Statistical Software* 24:8 (2008), 1–21.

[Anderson et al. 99] C. Anderson, S. Wasserman, and B. Crouch. "A p^* Primer: Logit Models for Social Networks." *Social Networks* 21 (1999), 37–66.

[Arratia and Liggett 05] Richard Arratia and Thomas M. Liggett. "How Likely Is an I.I.D. Degree Sequence to Be Graphical?" *Ann. Appl. Probab.* 15:1B (2005), 652–670.

[Baird and Ulanowicz 89] Daniel Baird and Robert E. Ulanowicz. "The Seasonal Dynamics of the Chesapeake Bay Ecosystem." *Ecological Monographs* 59 (1989), 329–364.

[Barvinok and Hartigan 10] A. Barvinok and J. A. Hartigan. "The Number of Graphs and a Random Graph with a Given Degree Sequence." Preprint, 2010.

[Bassetti and Diaconis 05] Frederico Bassetti and Persi Diaconis. "Examples Comparing Importance Sampling and the Metropolis Algorithm." *Illinois Journal of Mathematics* 50, 2006.

[Bayati et al. 10] M. Bayati, J. H. Kim, and A. Saberi. "A Sequential Algorithm for Generating Random Graphs." *Algorithmica* 58 (2010), 860–910.

[Beichl and Cloteaux 08] I. Beichl and B. Cloteaux. "Measuring the Effectiveness of the s-Metric to Produce Better Network Models." In *Proceedings of the 40th Conference on Winter Simulation*, Winter Simulation Conference, pp. 1020–1028, 2008.

[Bender and Canfield 78] Edward A. Bender and E. Rodney Canfield. "The Asymptotic Number of Labeled Graphs with Given Degree Sequences." *J. Combinatorial Theory Ser. A*, 24:3 (1978), 296–307.

[Bezáková et al. 06] Ivona Bezáková, Alistair Sinclair, Daniel Štefankovič, and Eric Vigoda. "Analysis of Sequential Importance Sampling for Contingency Tables." Preprint, 2006.

[Blanchet 09] J. H. Blanchet. "Efficient Importance Sampling for Binary Contingency Tables." *Annals of Applied Probability* 19:3 (2009), 949–982.

[Blitzstein 06] Joseph K. Blitzstein. "Building Random Objects Sequentially: From Characterization to Algorithm." PhD thesis, Stanford University, 2006.

[Bollobás 80] Béla Bollobás. "A Probabilistic Proof of an Asymptotic Formula for the Number of Labelled Regular Graphs." *European J. Combin.* 1:4 (1980), 311–316.

[Chatterjee et al. 11] S. Chatterjee, P. Diaconis, and A. Sly. "Random Graphs with a Given Degree Sequence." To appear in *Annals of Applied Probability*, 2011.

[Chen 07] Yuguo Chen. "Conditional Inference on Tables with Structural Zeros." *Journal of Computational and Graphical Statistics* 16:2 (2007), 445–467.

[Chen et al. 05] Y. Chen, P. Diaconis, S. Holmes, and J. S. Liu. "Sequential Monte Carlo Methods for Statistical Analysis of Tables." *Journal of the American Statistical Association* 100 (2005), 109–120.

[Chen et al. 06] Y. Chen, I. H. Dinwoodie, and S. Sullivant. "Sequential Importance Sampling for Multiway Tables." *Annals of Statistics* 34:11 (2006), 523–545.

[Chung and Lu 2002a] Fan Chung and Linyuan Lu. "The Average Distances in Random Graphs with Given Expected Degrees." *Proc. Natl. Acad. Sci. USA* 99:25 (2002) (electronic), 15879–15882.

[Chung and Lu 2002b] Fan Chung and Linyuan Lu. "Connected Components in Random Graphs with Given Expected Degree Sequences." *Ann. Comb.* 6:2 (2002), 125–145.

[Cooper et al. 07] C. Cooper, M. Dyer, and C. Greenhill. "Sampling Regular Graphs and a Peer-to-Peer Network." *Combinatorics, Probability and Computing* 16 (2007), 557–594.

[Diaconis and Gangolli 95] Persi Diaconis and Anil Gangolli. "Rectangular Arrays with Fixed Margins." In *Discrete Probability and Algorithms (Minneapolis, MN, 1993)*, IMA Vol. Math. Appl. 72, pp. 15–41. New York: Springer, 1995.

[Diaconis and Sturmfels 98] Persi Diaconis and Bernd Sturmfels. "Algebraic Algorithms for Sampling from Conditional Distributions." *Ann. Statist.* 26:11 (1998), 363–397.

[Doucet et al. 01] Arnaud Doucet, Nando de Freitas, and Neil Gordon. "An Introduction to Sequential Monte Carlo Methods." In *Sequential Monte Carlo Methods in Practice*, Stat. Eng. Inf. Sci., pp. 3–14. New York: Springer, 2001.

[Ellson et al. 03] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull. "Graphviz and Dynagraph: Static and Dynamic Graph Drawing Tools." In *Graph Drawing Software*, edited by M. Junger and P. Mutzel, pp. 127–148. Berlin: Springer, 2003.

[Erdős and Gallai 60] P. Erdős and T. Gallai. "Graphen mit Punkten vorgeschriebenen Grades." *Mat. Lapok* 11 (1960), 264–274.

[Fishman 66] George S. Fishman. *Monte Carlo: Concepts, Algorithms, and Applications*, Springer Series in Operations Research. New York: Springer, 1996.

[Frank and Strauss 86] Ove Frank and David Strauss. "Markov Graphs." *J. Amer. Statist. Assoc.* 81:395 (1986), 832–842.

[Goulden and Jackson 04] Ian P. Goulden and David M. Jackson. *Combinatorial Enumeration*. Mineola, NY: Dover, 2004.

[Hakimi 62] S. L. Hakimi. "On Realizability of a Set of Integers as Degrees of the Vertices of a Linear Graph I." *J. Soc. Indust. Appl. Math.* 10 (1962), 496–506.

[Hammersley and Handscomb 65] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods*. London: Methuen & Co. Ltd., 1965.

[Havel 55] V. Havel. "A Remark on the Existence of Finite Graphs." *Časopis Pěst. Mat.* 80 (1955) 477–480.

[Holland and Leinhardt 81] Paul W. Holland and Samuel Leinhardt. "An Exponential Family of Probability Distributions for Directed Graphs." *J. Amer. Statist. Assoc.* 76:373 (1981), 33–65.

[Itzkovitz et al. 03] S. Itzkovitz, R. Milo, N. Kashtan, G. Ziv, and U. Alon. "Subgraphs in Random Networks." *Phys. Rev. E (3)* 68:2 (2003), 026127, 8.

[Jackson 08] M. O. Jackson. *Social and Economic Networks*. Princeton: Princeton University Press, 2008.

[Jerrum and Sinclair 90] Mark Jerrum and Alistair Sinclair. "Fast Uniform Generation of Regular Graphs." *Theoret. Comput. Sci.* 73:1 (1990), 91–100.

[Kim and Vu 04] J. H. Kim and V. H. Vu. "Sandwiching Random Graphs: Universality between Random Graph Models." *Adv. Math.* 188:2 (2004), 444–469.

[King 04] Oliver D. King. "Comment on 'Subgraphs in Random Networks.'" *Phys. Rev. E*, 70:5 (2004), 058101, 3.

[Kong et al. 94] A. Kong, J. S. Liu, and W. H. Wong. "Sequential Imputations and Bayesian Missing Data Problems." *Journal of the American Statistical Association* 89:425 (1994), 278–288.

[Lauritzen 88] Steffen L. Lauritzen. *Extremal Families and Systems of Sufficient Statistics*, Lecture Notes in Statistics 49. New York: Springer, 1988.

[Lehmann and Romano 05] E. L. Lehmann and Joseph P. Romano. *Testing Statistical Hypotheses*, Springer Texts in Statistics. New York: Springer, 2005.

[Liu 01] Jun S. Liu. *Monte Carlo Strategies in Scientific Computing*, Springer Series in Statistics. New York: Springer, 2001.

[Liu and Chen 98] Jun S. Liu and Rong Chen. "Sequential Monte Carlo Methods for Dynamic Systems." *J. Amer. Statist. Assoc.* 93:443 (1998), 1032–1044.

[Macdonald 95] I. G. Macdonald. *Symmetric Functions and Hall Polynomials*, Oxford Mathematical Monographs. New York: Oxford University Press, 1995.

[Mahadev and Peled 95] N. V. R. Mahadev and U. N. Peled. *Threshold Graphs and Related Topics*, Annals of Discrete Mathematics 56. Amsterdam: North-Holland, 1995.

[McKay and Wormald 90] Brendan D. McKay and Nicholas C. Wormald. "Asymptotic Enumeration by Degree Sequence of Graphs of High Degree." *European J. Combin.* 11:6 (1990), 565–580.

[McKay and Wormald 91] Brendan D. McKay and Nicholas C. Wormald. "Asymptotic Enumeration by Degree Sequence of Graphs with Degrees $o(n^{1/2})$." *Combinatorica* 11:4 (1991), 369–382.

[Milo et al. 02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. "Network Motifs: Simple Building Blocks of Complex Networks." *Science* 298 (2002), 824–827.

[Newman 01] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. "Random Graphs with Arbitrary Degree Distributions and Their Applications." *Physical Review E* 64 (2001), 026118.

[Newman 03] M. E. J. Newman. "The Structure and Function of Complex Networks." *SIAM Review* 45 (2003), 167–256.

[Olding and Wolfe 09] B. P. Olding and P. J. Wolfe. "Inference for Graphs and Networks: Extending Classical Tools to Modern Data." Preprint, 2009.

[Owen and Zhou 00] Art Owen and Yi Zhou. "Safe and Effective Importance Sampling." *Journal of the American Statistical Association* 95:449 (2000), 135–143.

[Park and Newman 04] Juyong Park and M. E. J. Newman. "Statistical Mechanics of Networks." *Phys. Rev. E (3)* 70:6 (2004), 066117, 13.

[R Core Team 05] R Development Core Team. *R: A Language and Environment for Statistical Computing*. Vienna: R Foundation for Statistical Computing, 2005.

[Robins et al. 07] G. Robins, T. Snijders, P. Wang, M. Handcock, and P. Pattison. "Recent Developments in Exponential Random Graph (P*) Models for Social Networks." *Social Networks* 29:2 (2007), 92–215.

[Rubinstein and Kroese 04] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method*, Information Science and Statistics. New York: Springer, 2004.

[Sloane 05] N. J. A. Sloane. "The On-Line Encyclopedia of Integer Sequences." Available online (http://www.research.att.com/~njas/sequences/), 2005.

[Snijders et al. 06] T. Snijders, P. Pattison, G. Robins, and M. Handcock. "New Specifications for Exponential Random Graph Models." *Sociological Methodology* 1 (2006), 99–154.

[Snijders 91] Tom A. B. Snijders. "Enumeration and Simulation Methods for 0–1 Matrices with Given Marginals." *Psychometrika* 56:3 (1991), 397–417.

[Snijders 02] T. A. B. Snijders. "Markov Chain Monte Carlo Estimation of Exponential Random Graph Models." *Journal of Social Structure* 3:2 (2002), 1–40.

[Steger and Wormald 99] A. Steger and N. C. Wormald. "Generating Random Regular Graphs Quickly." *Combin. Probab. Comput.* 8:4 (1999), 377–396.

[Strauss 86] David Strauss. "On a General Class of Models for Interaction." *SIAM Review* 28:4 (1986), 513–527.

[Tinhofer 79] G. Tinhofer. "On the Generation of Random Graphs with Given Properties and Known Distribution." *Applied Computer Science Berichte Praktische Informatik* 13 (1979), 265–296.

[Tinhofer 90] G. Tinhofer. "Generating Graphs Uniformly at Random." *Computing Supplementum* 7 (1990), 235–255.

[Ulanowicz 05] Robert E. Ulanowicz. "Ecosystem Network Analysis" Available online (http://www.cbl.umces.edu/~ulan/ntwk/network.html), 2005.

[Watts and Strogatz 98] Duncan J. Watts and Steven H. Strogatz. "Collective Dynamics of 'Small-World' Networks." *Nature* 393:6684 (1998), 440–442.

[Wolfram Research 01]Wolfram Research. "Mathematica," version 4.1, 2001.

[Wormald 99] Nicholas C. Wormald. "Models of Random Regular Graphs." In *Surveys in Combinatorics, 1999 (Canterbury)*, London Math. Soc. Lecture Note Ser. 267, pp. 239–298. Cambridge, UK: Cambridge Univ. Press, 1999.

Joseph Blitzstein, Department of Statistics, Harvard University, Cambridge, MA 02138 (blitzstein@stat.harvard.edu)

Persi Diaconis, Departments of Mathematics and Statistics, Stanford University, Stanford, CA 94305 (diaconis@math.stanford.edu)

Received November 9, 2009; accepted June 1, 2010.