Internet Mathematics, 12:205–219, 2016
ISSN: 1542-7951 print/1944-9488 online
DOI: 10.1080/15427951.2016.1164768

Taylor & Francis
Taylor & Francis Group

FAST SEQUENTIAL CREATION OF RANDOM REALIZATIONS OF DEGREE SEQUENCES

Brian Cloteaux
Applied and Computational Mathematics Division, National Institute of Standards and Technology, Gaithersburg, Maryland, USA

Abstract *We examine the problem of creating random realizations of very large degree sequences. Although fast in practice, the Markov chain Monte Carlo (MCMC) method for selecting a realization has limited usefulness for creating large graphs because of memory constraints. Instead, we focus on sequential importance sampling (SIS) schemes for random graph creation. A difficulty with SIS schemes is assuring that they terminate in a reasonable amount of time. We introduce a new sampling method by which we guarantee termination while achieving speed comparable to the MCMC method.*

1. INTRODUCTION

Creating random graphs with a given degree sequence is useful for tasks from counting graphs with a given degree sequence to creating models of networks. Unfortunately, the problem of creating a uniformly sampled random graph for a given degree sequence, using a reasonable amount of time and space, is an open and difficult problem.

To formalize the problem, we use some basic terminology. An *degree sequence* $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$ is a set of nonnegative integers such that $n - 1 \geq \alpha_1 \geq \alpha_2 \geq ... \geq \alpha_n \geq 0$. For any simple, undirected graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, with node degrees α, G is termed a *realization* of α. We use the standard notation of n and m to represent number of nodes and edges, respectively. If a degree sequence α has at least one realization, then the sequence is *graphic*.

Using this notation, the basic problem is to select a near-uniform random realization of a graphic degree sequence α. Additionally, we are concerned with issues such as speed and memory usage, which affect the creation of very large graphs. This article describes a new approach to this problem.

For the remainder of this article, we first review previous work to this problem and why there is a need for a new approach. Following that section, we give needed background results and definitions. The next four sections introduce the algorithm and provide analysis of its runtime and the quality of its results. Finally, we will offer some conclusions and suggest future directions of investigation.

Address correspondence to Brian Cloteaux, National Institute of Standards and Technology, 100 Bureau Drive M/S 8910, Gaithersburg, MD 20899-8910, USA. Email: brian.cloteaux@nist.gov

2. PREVIOUS WORK

The most common method for creating a random realization of a degree sequence is the Monte Carlo Markov chain (MCMC) approach. In this approach, an initial (nonrandom) realization is created and then a series of random edge switches are chosen, creating a walk among different realizations of the same degree distribution [11, 18]. This approach is popular because of its ease of implementation and its speed.

The major disadvantage of this approach is that it requires access to all edges in the graph in order to perform the random walk. This limits the size of the graph that can be generated using this method. If the size of the graph is too large to fit within the memory of the computer, there is a significant access penalty incurred for each random edge selection.

Additionally, understanding the mixing time of the Markov chain is an open problem; it is unknown whether the random walk on a general degree distribution is rapid mixing. Significant progress was made [6] by showing that, for regular sequences, the MCMC approach is rapid mixing, but the bound they give is

$$\tau(\epsilon) \leq \alpha_{\max}^{17} n^7 \log\left(\alpha n \epsilon^{-1}\right),$$

where $\alpha_{\max}$ is the maximum value in the degree sequence α. More recently, [10] showed that if, for the maximum degree in a sequence, $\alpha_{\max}$,

$$3 \leq \alpha_{\max} \leq \frac{\sqrt{\sum_{i=1}^{n} \alpha_i}}{4}, \tag{2.1}$$

then the resulting Markov chain is rapid mixing, but the resultant bound is an order of n larger than the bound for the regular case. Unfortunately, both bounds are much too large for practical use.

There is empirical evidence that the actual mixing times are much lower, perhaps as low as $O(m)$ time [8]. With the time needed to created a Havel–Hakimi instance in order to start the random walk, a typical MCMC implementation uses $O(m \log n)$ time and $O(m)$ space to generate a random realization.

Another approach to generating random realizations is by using a sequential importance sampling (SIS) method to randomly select edges until a realization is built. This allows selected edges to be saved at each step and, thus, requires only that the degree sequence, not the entire graph, fits into memory.

The difficulty with an SIS approach is that while we are selecting edges to add to our random realization, it is possible to become "stuck." In other words, we cannot arbitrarily choose edges and guarantee that a graph exists that has the given degree sequence and the chosen edges.

The first algorithm to overcome this problem of creating an SIS algorithm that could guarantee termination is from [4]. Unfortunately, the Blitzstein–Diaconis algorithm (BD) has two drawbacks. The first is that the method has not been shown to sample uniformly. The authors were able to show empirical evidence that their algorithm gave only a reasonable sampling of the realizations. The second and more serious setback of this approach is that the algorithm's runtime of $O(mn^2)$ makes it much slower than the MCMC approach. This runtime can be lowered to $O(mn)$ time, but this is still significantly slower than the MCMC algorithm [13].

A second SIS method was introduced, which we will denote as BKS [3]. The advantage of their method stems from the result that if $\alpha_{\max} \leq O(m^{\frac{1}{4}-\tau})$, then their algorithm is

expected to run in $O(\alpha_{\max} m)$ time and asymptotically approaches uniform sampling. This is a powerful result but it requires a very strong constraint on the degree sequence.

In addition, unlike the BD algorithm, this algorithm does not have a termination guarantee; it has only a high probability of succeeding for sequences that meet the maximum degree-size bound. This lack of a termination guarantee becomes problematic when using the BKS method for instantiating sequences that are outside of the prescribed limits. For instance, from the set of threshold degree sequences [12], as the length of the sequences grows, the probability that the BKS method is able to construct a realization of the sequences quickly approaches zero.

Finding fast SIS methods is essential to being able to create very large random graphs. In this article we introduce a new SIS algorithm that maintains the termination guarantee of the BD algorithm while having a runtime that is competitive with the MCMC and BKS methods.

3. DEFINITIONS AND BASIC RESULTS

We begin with some needed preliminary definitions and results. In order to compare sequences, we will use majorization, which is a partial order over the set of degree sequences. The degree sequence α *majorizes* (or *dominates*) the degree sequence β, denoted by $\alpha \succcurlyeq \beta$, if for all k from 1 to n,

$$\sum_{i=1}^{k} \alpha_i \geq \sum_{i=1}^{k} \beta_i, \tag{3.1}$$

and if the sums of the two sequences are equal, i.e.,

$$\sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} \beta_i. \tag{3.2}$$

The sequence α *strictly majorizes* β, denoted by $\alpha \succ \beta$, if $\alpha_i \neq \beta_i$ for at least one i.

If α is a degree sequence and $\alpha_i \geq \alpha_j + 2$, then the operation of subtracting 1 from α_i and adding 1 to α_j is called the *unit transformation* from i to j on α. The new sequence created by this operation is denoted by $\alpha[i \rightarrow j]$. To designate the application to a degree sequence α of a series of unit transformations, we use the shorthand notation $\alpha[i_1 \rightarrow j_1][i_2 \rightarrow j_2] = (\alpha[i_1 \rightarrow j_1])[i_2 \rightarrow j_2]$.

There is a close relationship between majorization and unit transformations, as shown by the following theorem.

Theorem 3.1 ([14]) *For any degree sequences α and β where $\alpha \succcurlyeq \beta$, β can be obtained from α by a finite sequence of unit transformations.*

The power of comparing degree sequences using majorization comes from the following theorem.

Theorem 3.2 ([17], Theorem 1) *If the degree sequence α is graphic and $\alpha \succcurlyeq \beta$, then β is graphic.*

We introduce an additional notation for showing modifications to a degree sequence. For a degree sequence α, index i, and a set of indices ω, where $i \notin \omega$ and $|\omega| = \alpha_i$, a *reduction* $R(\alpha; i, \omega)$ is the integer sequence where we delete the term α_i and subtract 1 from the terms in the index set ω and the resulting sequence is sorted into reverse order. The set of all reductions from an index i on the sequence α we will denote as $\mathcal{R}(\alpha, i)$.

In the set $\mathcal{R}(\alpha, i)$ there exists a reduction $R_{\max}(\alpha; i)$ that majorizes all the other reductions. This reduction is created by setting the index set ω to the α_i largest indices, where $i \notin \omega$. At the same time, there exists a reduction $R_{\min}(\alpha; i)$ that is majorized by all the other reductions and is created by setting the index set ω to the α_i smallest indices, where $i \notin \omega$. An example is for the sequence $\alpha = (5, 5, 5, 5, 4, 4, 1, 1, 1, 1)$, then $R_{\max}(\alpha; 1) = (5, 5, 5, 4, 3, 0, 0, 0, 0)$ and $R_{\min}(\alpha; 1) = (4, 4, 4, 3, 3, 1, 1, 1, 1)$. Performing the reduction $R_{\min}(\alpha; i)$ to a degree sequence is commonly called a "laying-off" procedure in the literature. We now restate a classic result in terms of reductions.

Theorem 3.3 ([11], Theorem 2.1) *A degree sequence α is graphic if and only if for any index i, $R_{\min}(\alpha; i)$ is graphic.*

This result is a simple consequence of the theorem of [17]. If α is graphic, then for a given index i, there must exist some reduction $R(\alpha; i, \omega)$ that is also graphic. This reduction comes from simply taking the adjacent vertices to v_i in the realization of α. Since $R(\alpha; i, \omega) \succcurlyeq R_{\min}(\alpha; i)$, then $R_{\min}(\alpha; i)$ must also be graphic.

4. AN SIS APPROACH USING REDUCTIONS

Previously used SIS algorithms for creating random realizations involve choosing individual edges at each step. This approach leads to either slow and complicated checks to ensure that the resulting edge in the graph is viable, as in case of the BD algorithm, or no guarantee of termination, as in the case of the BKS algorithm.

Rather than selecting individual edges, we instead choose all the potential edges to a given node at once, i.e., choosing a random reduction to the given node. Determining whether the selected edges are valid requires testing only if the resulting reduction is graphic; if the reduction is graphic, then there is a realization that contains the resulting edges in the reduction.

This leads to a direct SIS algorithm; for the sequence α and the index i, at each step, we choose a random reduction $R(\alpha; i, \omega)$. If the chosen reduction is nongraphic, we use it as an upper bound for selecting a new reduction, $R(\alpha; i, \omega')$, such that $R(\alpha; i, \omega') \prec R(\alpha; i, \omega)$. If this new sequence is still not graphic, it becomes the new upper bound and we iterate. From Theorem 3.3, this process will eventually find a graphic reduction, and thus, the algorithm itself is guaranteed to terminate with a random realization. This algorithm is shown in Figure 1.

The time needed for selecting a random reduction that respects these constraints needs some explanation. First, let us consider the case when there is no previous upper bound for the reduction $R(\alpha; i, \omega)$. We note that if a degree sequence is sorted and we guarantee that its sum is even, then we can test if the sequence is graphic in $O(\sqrt{m})$ time [19]. Using a data structure such as a binary indexed tree [7], we can both sample from a weighted distribution of the nodes and update the weighting after selection in $O(\log n)$ time per operation. Thus, the total time to select a reduction set ω for the reduction $R(\alpha; i, \omega)$ using a given weighting of the nodes is $O(\alpha_i \log n)$. If, for each iteration, we reduce from

Input: a graphic sequence α
Output: a graph G that is a random realization of α
$\alpha' \leftarrow \alpha$
$G \leftarrow \emptyset$
while α' *is not empty* **do**
  Choose an index i where $\alpha'_i > 0$ in α'
  Choose a random reduction set ω for i in α'
  while $R(\alpha'; i, \omega)$ *is not graphic* **do**
    $S \leftarrow R(\alpha'; i, \omega)$
    Randomly choose ω' using the Blitzstein-Diaconis weighting such that $R(\alpha'; i, \omega') \prec S$
    $\omega \leftarrow \omega'$
  end
  for $j \in \omega$ **do**
    add the edge $\{v_i, v_j\}$ to G
  end
  $\alpha' \leftarrow R(\alpha'; i, \omega)$
end
return G

Figure 1 Algorithm 1. SIS selection method based on random reductions.

an index i where $i \geq \alpha_i$ (such as choosing the index of the smallest value at each step), then $\alpha_i \leq \sqrt{m}$ and the total time needed to choose and test a random reduction becomes $O(\sqrt{m} \cdot \log n)$.

Now, consider that we have a previous upper bound $S = R(\alpha; i, \omega_S)$. In order to choose a new reduction $N = R(\alpha; i, \omega_N)$ where $R(\alpha; i, \omega_N) \prec S$, using the described method, first select a new reduction $R(\alpha; i, \omega_P)$. Then define ω_N as the α_i smallest indices in the set $\omega_S \cup \omega_P$. As we will show in Section 7, this selection process corresponds to the operation $R(\alpha; i, \omega_N) = R(\alpha; i, \omega_S) \wedge R(\alpha; i, \omega_P) \preccurlyeq R(\alpha; i, \omega_S)$. The only difficulty with this approach is when $R(\alpha; i, \omega_P) \succcurlyeq R(\alpha; i, \omega_S)$ and so $R(\alpha; i, \omega_N) = R(\alpha; i, \omega_S)$. When this occurs, we choose the largest index i where $i \notin \omega_S$ such that there exists an index $j \in \omega_S$ where $R(\alpha; i, \omega_S)[i \rightarrow j]$ is a valid reduction. By setting $R(\alpha; i, \omega_N) = R(\alpha; i, \omega_S)[i \rightarrow j] \prec R(\alpha; i, \omega_S)$, we guarantee that the algorithm will terminate with a set time bound.

Combining all these parts, the overall runtime of Algorithm 1 is $O(n \log n \cdot \sqrt{m} \cdot g(\alpha))$, where $g(\alpha)$ is the number of nongraphic reductions selected during the construction of the realization. We now examine the value $g(\alpha)$.

5. SELECTING GRAPHIC REDUCTIONS

The runtime of Algorithm 1 depends on the expected value of the number of nongraphic reductions selected, $g(\alpha)$, and this value $g(\alpha)$ strongly depends on the distribution of the degree sequence. To discuss this dependency on the sequence distribution, we will compare degree sequences by their *majorization gap* [2]. The majorization gap for a se-

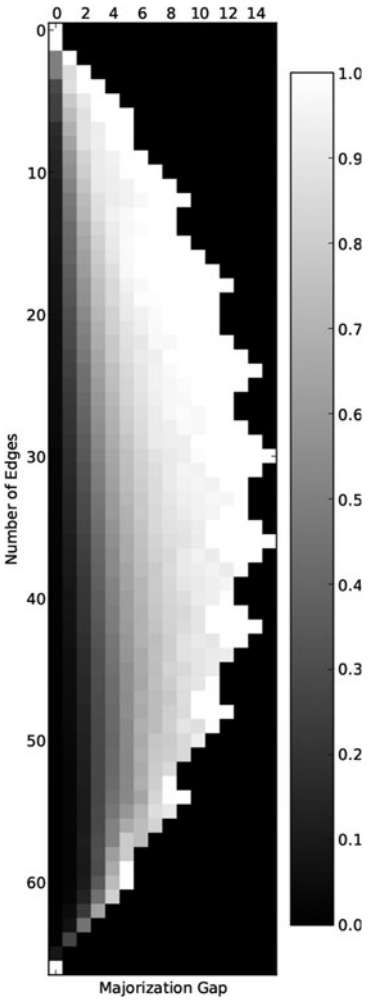

Figure 2 This figure displays the probabilities that a uniformly selected random reduction on the last index of a sequence will be graphic across all graphic degree sequences of a given length and majorization gap. This figure is a compilation of all graphic sequences of length 12 and less. The black outer border on the right-hand side of the figure represents combinations of majorization gap and number of edges where no sequence exists.

quence α is the minimum number of unit transformations from some threshold sequence β to α where $\beta \succcurlyeq \alpha$. This measures how close to being threshold a particular sequence is; the closer a sequence is to being threshold [12], then the smaller the sequence's majorization gap is and the fewer the number of graphic reductions there are in a reduction set for any given index. If the sequence α is a threshold sequence, i.e., has a majorization gap of 0, then there is exactly one reduction in the set $\mathcal{R}(\alpha, i)$ that is graphic. However, as the majorization gap becomes larger, the sequence is closer to being regular and the number of possible graphic reductions grows. For any sequence sum, there is a majorization gap value for which all degree sequences whose majorizations of all possible reductions are graphic.

Figure 2 shows the dependency of $g(\alpha)$ to its majorization gap. In this figure, we show the probability that a given uniformly sampled reduction is graphic from all degree

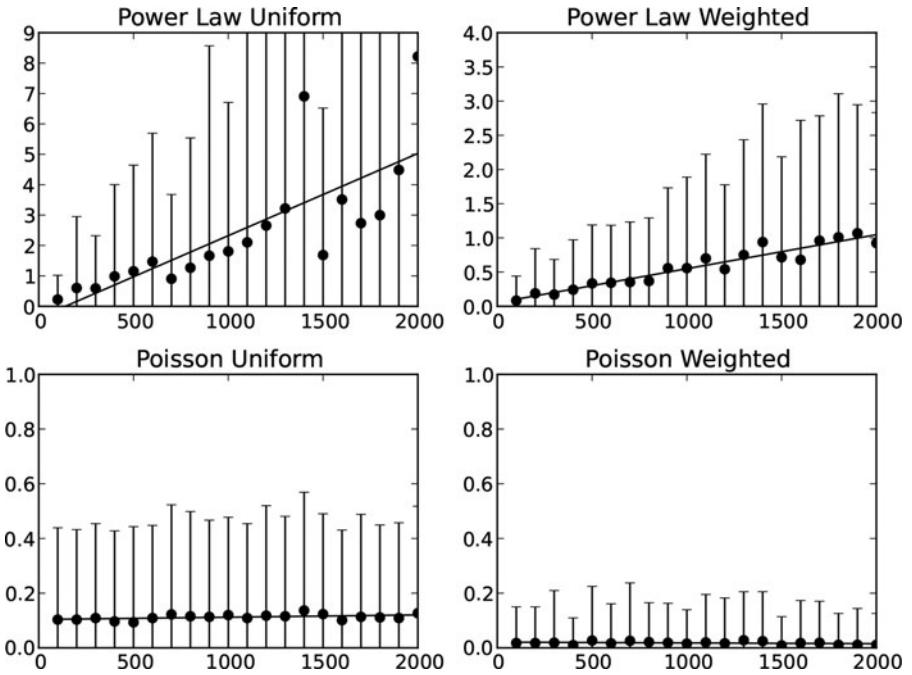

Figure 3 This figure shows the average number of nongraphic random reductions in which the x-axis of each plot is the sequence length. Each point is an average of 30 different degree distributions with the same sequence length and distribution parameters. Each graph shows the number of times a nongraphic reduction is selected while creating the graph along with a linear least-squares fit, and error bars for one standard deviation of the number of nongraphic reductions selected.

sequences with a fixed length and majorization gap. The first column, where the majorization gap is 0, represents all the threshold graphs for a given length. This figures that the closer a sequence's majorization gap is to zero, the higher the probability that a random reduction will not be graphic.

This dependency on the distribution is closely related to the probability distribution used for selecting the reductions. This weighting function for the selection probabilities is important for two reasons: the first is that having a uniform or near-uniform selection probability requires the correct edge-selection probabilities. Although it is an open problem to show that any weighting function produces uniform selection among the various realizations for a degree sequence, we use the weighting function described by Blitzstein and Diaconis for their algorithm. For the degree sequence α and index p, the weight given to an edge (p, q) to be selected for the realization G is proportional to the degree of q. Blitzstein and Diaconis empirically observed that this weighting gives a more uniform sampling of the realizations than simply using a uniform weighting of the nodes.

Because the BD weighting tends to create edges to vertices with larger degrees, the majorization gap of the resulting reduction typically is larger than a reduction using uniform selection. This tends to reduce the number of nongraphic reductions selected in the algorithm. In Figure 3, we see the results of distribution and the sampling method on the number of nongraphic reductions selected. For distributions that tend to have smaller majorization gaps (in this case, power-law distributions compared to Poisson distributions), they will be

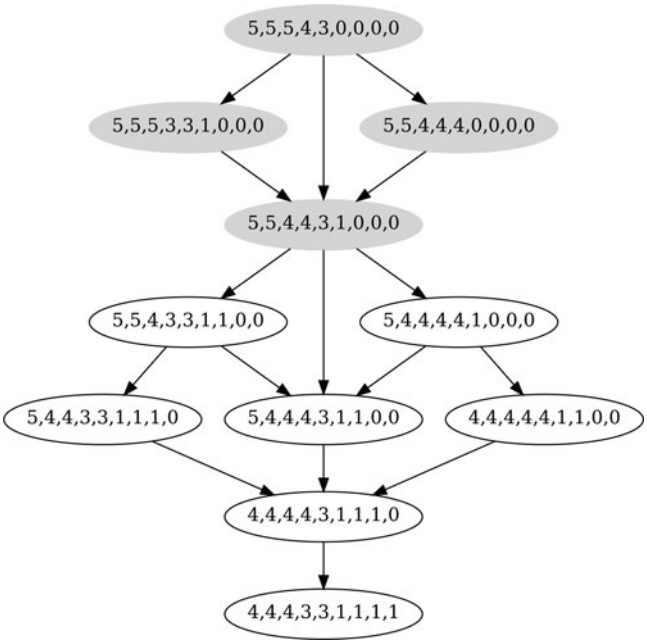

Figure 4 This figure shows a lattice of reductions for the sequence (5, 5, 5, 5, 4, 4, 1, 1, 1, 1) on its first index. The grey nodes signify nongraphic reductions, whereas the white nodes are graphic.

expected to have more nongraphic selections. Also, there are more nongraphic selections for uniform edge sampling versus the weighting scheme of Blitzstein and Diaconis. It should be noted that, for these test cases, there were a very small number of nongraphic reductions chosen. In fact, other than the case that used a power-law distribution along with uniform weighting combination, the algorithm averaged ≤ 1 nongraphic reduction selected out of the ≈ 2000 selections. For those cases, the algorithm essentially behaved as if it has a $O(n \log n \cdot \sqrt{m})$ runtime.

Because the maximum number of reductions the algorithm chooses before coming to the minimum reduction is polynomial in n [9], then from a uniform sampling from the reductions majorized by a given reduction, we would expect to find a graphic reduction in no more than $g(\alpha) = O(\log n)$ trial reductions. Thus, an expected time for this algorithm of $O(n \log^2 n \cdot \sqrt{m})$ within a logarithmic factor of the time for MCMC when the sequence is dense is $(m = O(n^2))$, and is at worst a $O(\sqrt{n} \log n)$ factor slower for sparse sequences $(m = O(n))$.

The worst-case bound for $g(\alpha)$ is much larger than its expected bound. From a result we will prove in Section 7, the number of nongraphic reductions that the algorithm can potentially choose before encountering a graphic reduction is $g(\alpha) = \Omega(n^2)$. We will now consider how to use the order structure of the reduction set to limit this worst-case behavior of the algorithm.

As we will show in Section 7, the partial-order $(\mathcal{R}(\alpha, i), \succcurlyeq)$ is a lattice. Theorem 3.3 shows that the bottom element $R_{\min}(\alpha, i)$ in $(\mathcal{R}(\alpha, i), \succcurlyeq)$ will always be graphic if the original sequence is graphic. Because all the graphic sequences are grouped at the bottom of the lattice, we reduce the possibility of selecting a nongraphic reduction by pruning some of the nongraphic reductions at the top of the lattice. If we descend the lattice starting

from the maximum element, we will eventually find a graphic sequence that we can use to provide a range for selecting reductions. But simply finding a maximal graphic reduction in the reduction lattice is not enough to create a selection range within which every realization has a nonzero probability of being selected. Consider the sequence with its corresponding reduction lattice in Figure 4. If we look at the set of graphic reductions, we notice that there are two maximal graphic reductions: (5, 5, 4, 3, 3, 1, 1, 0, 0) and (5, 4, 4, 4, 4, 1, 0, 0, 0). If we were to use one of those reductions as an upper bound for the set of reductions from which we are choosing, then the other reduction would have a zero probability of occurring in a realization.

In order to navigate the lattice to find an upper bound that does not preclude any graphic realizations, we use the following procedure. Start with the maximum reduction $r = R_{\max}(\alpha, d)$ and test if it is graphic. If it is not, then choose the smallest and largest indices in r, i, and j, respectively, where neither i nor j has been previously used in this search procedure, and create the new sequence $r[i \rightarrow j]$. It is straightforward to see that the sequence $r[i \rightarrow j]$ remains a reduction. We now retest whether this new reduction is graphic and repeat the procedure; if the reduction is not graphic, then we continue by choosing a new set of indices i, j that have not been used in this procedure. The maximum number of times this procedure will repeat is at most α_d, because after α_d iterations r becomes $R_{\min}(\alpha, d)$, which is graphic.

The point of this construction is that each time we choose the indices in this manner, the resulting reduction is majorized by all other valid reductions created by some unit transformation. Thus, for the nongraphic reduction r, where $r[i \rightarrow j]$ is graphic, any maximal graphic reduction p will be $r[i \rightarrow j] \preccurlyeq p \prec r$.

We now formalize this idea with a simple algorithm. As shown in Algorithm 2 in Figure 5, the basic idea is that before any random reduction is selected for an index d, we call the procedure to give bounds for the selection range of the reductions. For the reduction $R(\alpha, d)$, the maximum number of iterations in the algorithm is α_d. With an appropriate data structure, each unit transformation can be implemented in constant time, giving the algorithm a total runtime of $O(\alpha_d)$.

6. QUALITY OF RANDOM GRAPHS

Determining the quality, or how close to uniform, the random graphs produced by the reduction algorithm are remains an open question. Although we cannot prove that the sampling approaches uniform, we can empirically compare the graphs produced by the different algorithms. Figure 6 shows a comparison of the four algorithms: Algorithm 1, MCMC, BD, and BKS.

These algorithms were compared by creating a series of graph degree sequences drawn from Poisson and power-law distributions, where a single random sequence for both distributions was created for each length from 100 to 2,000 incremented by 100. The exact distributions are a Poisson distribution coming from a Erdős–Renyi random graph with an edge probability of $p = 0.1$ and a power-law distribution with an exponent of $k = 2.0$. For each sequence, 30 random graphs were created by each algorithm. For the MCMC algorithm, we took the running time as $30 \cdot m$ switches, as suggested in [16]. We compared three graph properties, again chosen from [16], to examine the similarity of the random graphs: the global clustering coefficient, the maximum eigenvalue, and the diameter.

Input: a graphic sequence α
Output: a graph G that is a random realization of α

$\alpha' \leftarrow \alpha$
$G \leftarrow \emptyset$
while α' *is not empty* **do**
  Choose an index i where $\alpha'_i > 0$ in α'
  $\beta_N, \beta_G =$ `Reduction-Bounds`(α', i)
  Choose a random reduction set ω for i in α' such that $R(\alpha'; \omega, i) \prec \beta_N$
  while $R(\alpha'; \omega, i) \not\succcurlyeq \beta_G$ **do**
    if $R(\alpha'; i, \omega)$ *is not graphic* **then**
      $S \leftarrow R(\alpha'; i, \omega)$
      Randomly choose ω' using the Blitzstein-Diaconis weighting such that $R(\alpha'; i, \omega') \prec S$
      $\omega \leftarrow \omega'$
    end
  end
  for $j \in \omega$ **do**
    add the edge $\{v_i, v_j\}$ to G
  end
  $\alpha' \leftarrow R(\alpha'; i, \omega)$
end
return G

`Reduction-Bounds` (α, d)
Input: α, d - degree sequence, reduction index
Output: β^N, β^G - bounds such that for every maximal graphic reduction p, $\beta^G \preccurlyeq p \prec \beta^N$ and $\beta^G = \beta^N[i \rightarrow j]$ for some indices i and j

$\beta^N \leftarrow R_{max}(\alpha, d)$
$\beta^G \leftarrow \beta^N$
$p \leftarrow 1$
while β^G *is not graphic* **do**
  $\beta^N \leftarrow \beta^G$
  decrement $\beta^G[p]$
  increment $\beta^G[n-p]$
  increment p
end
return β^N, β^G

Figure 5 Algorithm 2. SIS algorithm for creating a random model using the reduction bounds method. The reduction bounds routine returns two values, β_N which is a minimal nongraphic reduction such that there exists a graphic reduction that is a unit transformation of β_N and β_G, which is a graphic reduction where $\beta_G = \beta_N[i \rightarrow j]$ for some indices i and j.

We make two comments about our experimental setup. The first is that for the power-law distribution, the BKS algorithm could not converge to a random graph, thus, there are no results for BKS shown for those distributions. A second point is that a number of the random graphs produced for the power-law distributions by the other three methods were not connected and, thus, had an infinite diameter. Therefore, we do not show the diameter results for this case.

Examining the results, we note that all four algorithms produced virtually identical results for the Poisson distributions. At the scale shown in the figure, the average behavior between the four algorithms was indistinguishable.

For the power-law distributions, we note two points. First, the reduction and BD algorithms give essentially identical results. This can be expected, because the reduction algorithm is using the BD weighting for selecting its reductions. Thus, the quality of the results given by Algorithm 1 is essentially the same as that of the BD algorithm. A second point is that for the power-law case, we see the only noticeable difference between the results of the algorithms. There is a slight but discernible separation between the MCMC results and Algorithm 1 and BK results.

7. BOUNDING THE CHAIN LENGTHS

Over the set of partitions $\mathcal{L}_p$ for some positive integer p, majorization forms the lattice $(\mathcal{L}_p, \succcurlyeq)$ [5], where the meet and join operators are defined as follows:

$$\begin{aligned}(\alpha \wedge \beta)_k &= \min\left\{\sum_{i=1}^{k}\alpha_i, \sum_{i=1}^{k}\beta_i\right\} - \sum_{i=1}^{k-1}(\alpha \wedge \beta)_i \\ &= \min\left\{\sum_{i=1}^{k}\alpha_i, \sum_{i=1}^{k}\beta_i\right\} - \min\left\{\sum_{i=1}^{k-1}\alpha_i, \sum_{i=1}^{k-1}\beta_i\right\}, \qquad (7.1)\end{aligned}$$

$$\alpha \vee \beta = \bigwedge \{\phi : \phi \succcurlyeq \alpha, \phi \succcurlyeq \beta\}. \qquad (7.2)$$

If the integer p is even, then there are a small number of elements at the bottom of the partition lattice that are graphic, although the majority of the partitions in the lattice are nongraphic [15]. This integer partition lattice has provided a framework for considering realizability and uniqueness problems of degree sequences [1].

In this section, we use the lattice structure inherent in the set of reductions to establish bounds on the two algorithms. To show this, we will use the terminology and main result found in [9]. An H-step is a unit transform $[i \rightarrow i+1]$, whereas a V-step is a unit transform $[i \rightarrow j]$ where $\alpha_i = \alpha_j + 2$. An HV-chain between the sequences α and β is a series of transformations $\alpha = \alpha^{(0)} \succ \alpha^{(1)} \succ \alpha^{(2)} \succ \cdots \succ \alpha^{(i)} \succ \cdots \succ \alpha^{(L)} = \beta$, where every unit transformation from 1 to i is an H-step and every unit transformation from i to L is a V-step.

In an integer partition lattice, the length of the longest chain between two sequences, α and β where $\alpha \succcurlyeq \beta$, is defined as $h(\alpha, \beta)$. That HV-chains form the longest chains between sequences in these lattices was shown in [9].

Theorem 7.1. [9],Theorem 12 *Suppose that $\alpha \succcurlyeq \beta$. Then, all HV-chains from α to β have the same length and this length is $h(\alpha, \beta)$.*

The degree set $\mathcal{D}(\alpha)$ of a sequence α is the set of values in the sequence, i.e., $\mathcal{D}(\alpha) = \{d | d = \alpha_i \text{ for } 1 \le i \le n\}$. We denote the subsequence of α from the index i to the index j as $\alpha(i : j)$. We now relate the degree set between the indices in a unit transformation to the length of the longest chain between the two sequences.

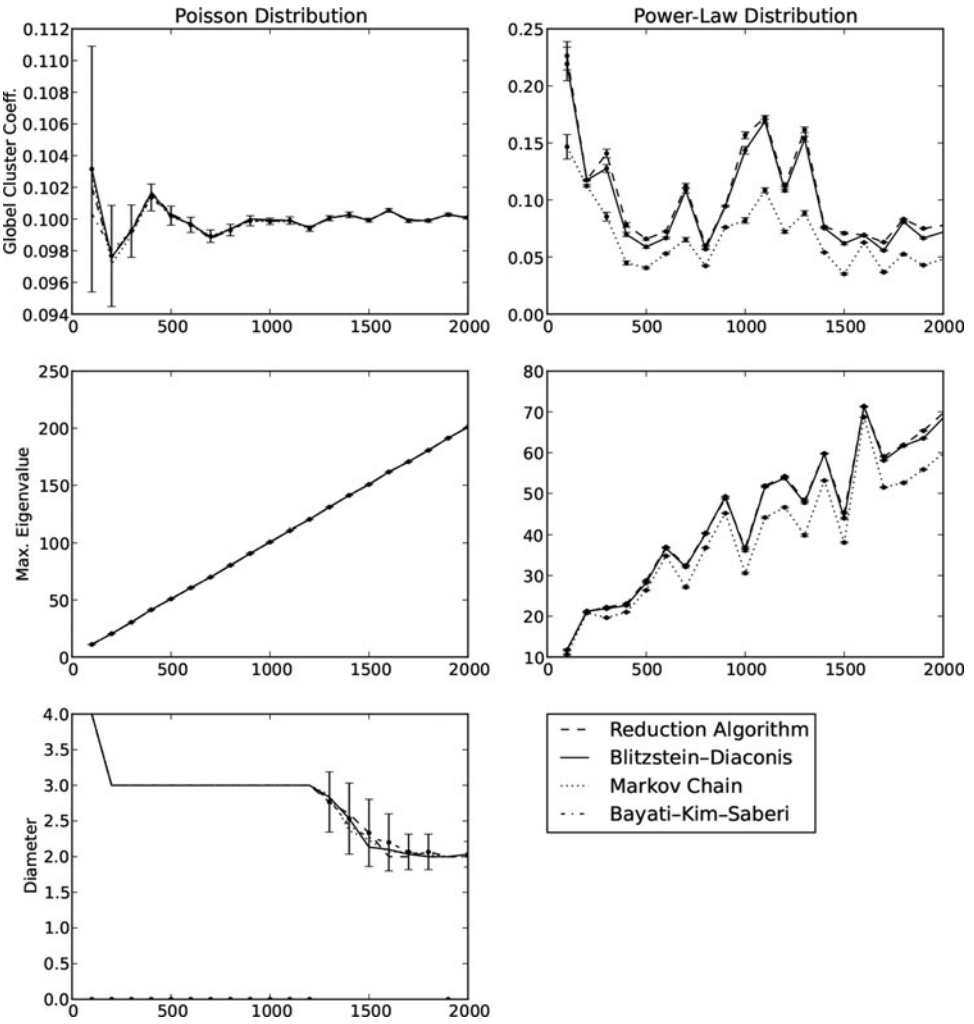

Figure 6 The x-axis shows the length for the random sequences. The Poisson degree distributions come from an Erdős–Renyi model where $p = 0.1$ and the power-law degree come from a power-law distribution with an exponent of 2.0. The points represent the mean of 30 random graphs generated by each algorithm, and the error bars represent one standard deviation. For the Poisson distributions, only the largest of error bars is shown because of the overlap.

Theorem 7.2. *For the positive integer sequence* α *and* $1 \leq i < j \leq n$, *then*

$$|\mathcal{D}(\alpha(i:j))| - 1 \leq h(\alpha, \alpha[i \rightarrow j]) \leq |\mathcal{D}(\alpha(i:j))|. \tag{7.3}$$

Proof. Theorem 7.1 shows that we need to construct an HV-chain only from α to $\alpha[i \rightarrow j]$ to establish the length of the maximum chain. To create an HV-chain, we will induct on the indices from i to j. To begin, we choose the smallest index $p > i$ such that $\alpha_i \geq \alpha_p + 2$. For the index p, there are three possibilities for maximum chain length $h(\alpha, \alpha[i \rightarrow p])$.

1. If $\alpha_i = \alpha_p + 2$, then $[i \rightarrow p]$ defines a HV-chain of length $2 \leq |\mathcal{D}(\alpha(i:p))|$.
2. If $\alpha_i > \alpha_p + 2$ and $\alpha_i = \alpha_{p-1}$, then $[p-1 \rightarrow p]$ defines an HV-chain of length $2 = |\mathcal{D}(\alpha(i:p))|$.
3. If $\alpha_i > \alpha_p + 2$ and $\alpha_i = \alpha_{p-1} + 1$, then $[p-1 \rightarrow p][i \rightarrow p-1]$ defines an HV-chain of length $3 = |\mathcal{D}(\alpha(i:p))|$.

We will now induct on the index number to show that this relationship continues to hold. Assume that for all indices $p \leq r \leq k$ there there is an HV-chain from α to $\alpha[i \rightarrow r]$ whose length is $|\mathcal{D}(\alpha(i:r))| - 1 \leq h(\alpha, \alpha[i \rightarrow r]) \leq |\mathcal{D}(\alpha(i:r))|$. If $\alpha_{k+1} = \alpha_k$, then $\alpha[i \rightarrow k] = \alpha[i \rightarrow k+1]$, and so they have the same HV-chain. Else, if $\alpha_k > \alpha_{k+1}$, then there are two cases. If $\alpha_k \geq \alpha_{k+1} + 2$, then $\alpha[i \rightarrow k+1] = \alpha[k \rightarrow k+1][i \rightarrow k]$. By prepending the H-step $[k \rightarrow k+1]$ to the front of the HV-chain for $[i \rightarrow k]$, we then have an HV-chain from α to $\alpha[i \rightarrow k+1]$ whose length is $\leq |\mathcal{D}(\alpha(1:k+1))|$. If $\alpha_k = \alpha_{k+1} + 1$, then there exists a smallest index s such that $\alpha_s = \alpha_k$. Now, $\alpha[i \rightarrow k+1] = \alpha[i \rightarrow s][s \rightarrow k+1]$. Again, by appending the V-step $[s \rightarrow k+1]$ to the end of the HV-chain for $\alpha[i \rightarrow s]$, we have a new HV-chain that satisfies the condition. □

This result bounds the number of possible nongraphic sequences that can be randomly selected before choosing a graphic sequence. The maximum number of nongraphic sequences in the SIS Algorithm 2 is no greater than the maximum chain length from α to $\alpha[i \rightarrow j]$, i.e., $h(\alpha, \alpha[i \rightarrow j]) \leq j - i + 1 \leq n$.

If we take the set of the possible reductions for some index i in the sequence α, they form a subset of the integer partitions. The majorization operation also forms a lattice over the set of reductions, $(\mathcal{R}(\alpha, i), \succcurlyeq)$, where the meet and join definitions are identical from the integer partition lattice. We see this by showing that the meet operator (and by extension, the join operator) are closed for the set of reductions.

Theorem 7.3. *For the sequence γ and the sequences $\alpha, \beta \in \mathcal{R}(\gamma, d)$, then $\alpha \wedge \beta \in \mathcal{R}(\gamma, d)$.*

Proof. From the definition of a reduction, $0 \leq \gamma_i - \alpha_i \leq 1$ and $0 \leq \gamma_i - \beta_i \leq 1$ for all $i \neq d$. It follows that for all i, that $|\alpha_i - \beta_i| \leq 1$. For the sequence $\alpha \wedge \beta$ to not be a reduction requires that there is an index p where $\gamma_p - (\alpha \wedge \beta)_p \geq 2$. From (7.1), if $\sum_{i=1}^{p} (\alpha \wedge \beta)_i = \sum_{i=1}^{p} \alpha_i$ and $\sum_{i=1}^{p-1} (\alpha \wedge \beta)_i = \sum_{i=1}^{p-1} \alpha_i$, then $(\alpha \wedge \beta)_p = \alpha_p$, and so $\gamma_p - (\alpha \wedge \beta)_p \leq 1$. There is an identical argument for when both sums are equal to the β sums. For a contradiction, we assume that $\sum_{i=1}^{p} \alpha_i > \sum_{i=1}^{p} \beta_i$ and $\sum_{i=1}^{p-1} \alpha_i < \sum_{i=1}^{p-1} \beta_i$. But these inequalities imply that $\alpha_p - \beta_p \geq 2$, which contradicts the fact that this difference can be no larger than 1. Thus, $\alpha \wedge \beta \in \mathcal{R}(\gamma, d)$. □

To establish a worst-case time bound for Algorithm 1, we show that there exist reduction lattices that contain quadratic-length chains of nongraphic reductions.

Theorem 7.4. *There exist reduction lattices of length n sequences that contain chains of nongraphic reductions of length $\Omega(n^2)$.*

Proof. Consider the degree sequence taken from the graph that is constructed by starting from the empty graph and then for m times adding an isolated node followed by a dominating

node. From the construction of this graph, it is straightforward to see that this sequence is both threshold ([12], Theorem 1.2.4) and that its degree set contains all the integers from 1 to $n-1$. Thus, this degree sequence with its length of $n = 2m$ is

$$\alpha = (n-1, n-2, ..., m+1, m, m, m-1, ..., 2, 1)\,. \tag{7.4}$$

Since α is threshold, then $R_{\min}(\alpha, m)$ must be the only graphic sequence in $\mathcal{R}(\alpha, m)$. We construct $R_{\min}(\alpha, m)$ from $R_{\max}(\alpha, m)$ with the following series of unit transformations:

$$R_{\min}(\alpha, m) = R_{\max}(\alpha, m)[m-1 \;\rightarrow\; n][m-2 \;\rightarrow\; n-1]...[1 \;\rightarrow\; m+1]. \tag{7.5}$$

From Theorem 7.1, for each of these unit transformations, there is an HV-chain of length of $\geq m$ in the integer partition lattice. The point of this construction is that for each sequence in the HV-chain of a unit transformation, it is also a valid reduction. Thus, between each unit transformation there is a sequence of m reductions, and for the m unit transformations there exists a chain of reductions whose length is at least $m^2 = \Theta(n^2)$ between $R_{\max}(\alpha, m)$ and $R_{\min}(\alpha, m)$. □

8. CONCLUSIONS

We have examined the problem of sequentially creating a random realization of a very large degree sequence and, in particular, in a reasonable amount of time. Current published algorithms all have failings that reduce their usefulness. The MCMC approach requires holding an entire graph in memory, the BKS algorithm does not guarantee termination and, for some degree sequences, has a high probability of not succeeding, and the BD algorithm is significantly slower than the other two approaches. The algorithm proposed in this article overcomes all of these difficulties.

This algorithm is competitive with the time requirement for the MCMC approach, but with a significant savings in memory usage. The quality of the selected realizations produced matches the output of the Blitzstein-Diaconis method. With this approach, it is feasible to create realizations containing millions of nodes.

Further research for problems in this area starts with the problem of determining whether a better sampling scheme than the BD weighting exists for the edge selection. Another important task is to prove the expected runtime of this algorithm, and in particular, the expectation of choosing a nongraphic reduction.

ACKNOWLEDGMENTS

The author would like to thank Zoe Park for her assistance in writing code. Also, the author would like to thank Peter Mell, Isabel Beichl, and especially the anonymous reviewer for their helpful comments, which substantially improved the presentation of this article.

REFERENCES

[1] M. Aigner and E. Triesch. "Realizability and Uniqueness in Graphs." *Discrete Mathematics* 136:1–3 (1994), 3–20. doi: 10.1016/0012-365X(94)00104-Q.

[2] S. R. Arikati and U. N. Peled. "Degree Sequences and Majorization." *Linear Algebra and its Applications* 199:Supplement 1 (1994), 179–211. doi: 10.1016/0024-3795(94)90349-2.

[3] M. Bayati, J. H. Kim, and A. Saberi. "A Sequential Algorithm for Generating Random Graphs." *Algorithmica* July (2009). doi: 10.1007/s00453-009-9340-1.

[4] J. Blitzstein and P. Diaconis. "A Sequential Importance Sampling Algorithm for Generating Random Graphs with Prescribed Degrees." *Internet Mathematics* 6:4 (2010), 489. doi: 10.1080/15427951.2010.557277.

[5] T. Brylawski. "The Lattice of Integer Partitions." *Discrete Mathematics* 6:3 (1973), 201–219. doi: 10.1016/0012-365X(73)90094-0.

[6] C. Cooper, M. Dyer, and C. Greenhill. "Sampling Regular Graphs and a Peer-to-Peer Network." *Combinatorics, Probability and Computing* 16:4 (2007), 557–593. doi: 10.1017/S0963548306007978.

[7] P. M. Fenwick. "A New Data Structure for Cumulative Frequency Tables." *Software: Practice and Experience* 24:3 (1994), 327–336. doi: 10.1002/spe.4380240306.

[8] C. Gkantsidis, M. Mihail, and E. W. Zegura. "The Markov Chain Simulation Method for Generating Connected Power Law Random Graphs." In *ALENEX*, pp. 16–25. SIAM, 2003.

[9] C. Greene and D. J. Kleitman. "Longest Chains in the Lattice of Integer Partitions Ordered by Majorization." *European Journal of Combinatorics* 7:1 (1986), 1–10. doi: 10.1016/S0195-6698(86)80013-0.

[10] C. Greenhill. "The Switch Markov Chain for Sampling Irregular Graphs: Extended Abstract." In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pp. 1564–1572. SIAM, 2015. Available online (`http://dl.acm.org/citation.cfm?id=2722129.2722232`).

[11] D. J. Kleitman and D. L. Wang. "Algorithms for Constructing Graphs and Digraphs with Given Valences and Factors." *Discrete Mathematics* 6:1 (1973), 79–88. doi: 10.1016/0012-365X(73)90037-X.

[12] N. V. R. Mahadev and U. N. Peled. *Threshold graphs and related topics*, Annals of Discrete Mathematics, 56. Amsterdam: North-Holland Publishing Co., 1995.

[13] E. Moseman. "Improving the computational efficiency of the Blitzstein-Diaconis algorithm for generating random graphs of prescribed degree." Technical Report, 2015, National Institute of Standards and Technology. doi: 10.6028/NIST.IR.8066.

[14] R. F. Muirhead. "Some Methods Applicable to Identities and Inequalities of Symmetric Algebraic Functions of *n* Letters." *Proceedings of the Edinburgh Mathematical Society*, 21: 144–157. London, UK: G. Bell and Sons, Ltd., 1903. doi: 10.1017/S001309150003460X.

[15] B. Pittel. "Confirming Two Conjectures about the Integer Partitions." *Journal of Combinatorial Theory, Series A* 88:1 (1999), 123–135. doi: 10.1006/jcta.1999.2986.

[16] J. Ray, A. Pinar, and C. Seshadhri. "Are We There Yet? When to Stop a Markov Chain while Generating Random Graphs." In *Proceedings of the 9th International Conference on Algorithms and Models for the Web Graph*, WAW'12, pp. 153–164. Berlin, Heidelberg: Springer-Verlag, 2012, doi: 10.1007/978-3-642-30541-2_12.

[17] E. Ruch and I. Gutman. "The Branching Extent of Graphs." *Journal of Combinatorics, Information, & System Sciences* 4:4 (1979), 285–295.

[18] R. Taylor. "Constrained Switchings in Graphs." In *Combinatorial Mathematics VIII*, vol. 884, edited by K. L. McAvaney, pp. 314–336. Berlin, Heidelberg: Springer, 1981. doi: 10.1007/BFb0091828.

[19] A. Tripathi and S. Vijay. "A Note on a Theorem of Erdős & Gallai." *Discrete Mathematics* 265:1–3 (2003), 417–420. doi: 10.1016/S0012-365X(02)00886-5.