Journal of Machine Learning Research 19 (2018) 1-13 Submitted 3/17; Revised 12/17; Published 11/18

A Note on Quickly Sampling a Sparse Matrix with Low Rank Expectation

Karl Rohe KARLROHE@STAT.WISC.EDU
Statistics Department
University of Wisconsin-Madison
Madison, WI 53706, USA

Jun Tao ITAOJUN@PKU.EDU.CN
School of Mathematical Sciences
Peking University
Beijing, 100871, China

Xintian Han HANXINTIAN@PKU.EDU.CN
Yuanpei College
Peking University
Beijing, 100871, China

Norbert Binkiewicz NORBERTBIN@GMAIL.COM
Statistics Department
University of Wisconsin-Madison
Madison, WI 53706, USA

Editor: Inderjit Dhillon

Abstract

Given matrices $X, Y \in \mathbb{R}^{n\times K}$ and $S \in \mathbb{R}^{K\times K}$ with positive elements, this paper proposes an algorithm `fastRG` to sample a sparse matrix A with low rank expectation $E(A) = XSY^T$ and independent Poisson elements. This allows for quickly sampling from a broad class of stochastic blockmodel graphs (degree-corrected, mixed membership, overlapping) all of which are specific parameterizations of the generalized random product graph model defined in Section 2.2. The basic idea of `fastRG` is to first sample the number of edges m and then sample each edge. The key insight is that because of the the low rank expectation, it is easy to sample individual edges. The naive "element-wise" algorithm requires $O(n^2)$ operations to generate the $n \times n$ adjacency matrix A. In sparse graphs, where $m = O(n)$, ignoring log terms, `fastRG` runs in time $O(n)$. An implementation in `R` is available on github. A computational experiment in Section 2.4 simulates graphs up to $n = 10,000,000$ nodes with $m = 100,000,000$ edges. For example, on a graph with $n = 500,000$ and $m = 5,000,000$, `fastRG` runs in less than one second on a 3.5 GHz Intel i5.

Keywords: Random Dot Product Graph, Edge exchangeable, Simulation

1. Introduction

The random dot product graph (RDPG) model serves as a test bed for various types of clustering and statistical inference algorithms. This model generalizes the Stochastic Blockmodel (SBM), the Overlapping SBM, the Mixed Membership SBM, and the Degree-Corrected SBM (Holland et al., 1983; Latouche and Ambroise, 2011; Airoldi et al., 2008; Karrer and Newman, 2011). Under the

RDPG, each node i has a (latent) node feature $x_i \in \mathbb{R}^K$ and the probability that node i and j share an edge is parameterized by $\langle x_i, x_j \rangle$ (Young and Scheinerman, 2007).

While many network analysis algorithms only require $O(m)$ operations, where m is the number of edges in the graph, sampling RDPGs with the naive "element-wise" algorithm takes $O(n^2)$ operations, where n is the number of nodes. In particular, sparse eigensolvers compute the leading k eigenvectors of such graphs in $O(km)$ operations (e.g. in ARPACK Lehoucq et al. 1995). As such, sampling an RDPG is a computational bottleneck in simulations to examine many network analysis algorithms.

The organization of this paper is as follows. Section 2 gives `fastRG`. Section 2.1 relates `fastRG` to xlr, a new class of edge-exchangeable random graphs with low-rank expectation. Section 2.2 presents a generalization of the random dot product graph. Theorem 4 shows that `fastRG` samples Poisson-edge graphs from this model. Then, Theorem 7 in Section 2.3 shows how `fastRG` can be used to approximate a certain class of Bernoulli-edge graphs. Section 2.4 describes our implementation of `fastRG` (available at `https://github.com/karlrohe/fastRG`) and assesses the empirical run time of the algorithm.

1.1. Notation

Let $G = (V, E)$ be a graph with the node set $V = \{1, \ldots, n\}$ and the edge set E contains edge (i, j) if node i is connected to node j. In a directed graph, each edge is an ordered pair of nodes while in an undirected graph, each edge is an unordered pair of nodes. A multi-edge graph allows for repeated edges. In any graph, a self-loop is an edge that connects a node to itself. Let the adjacency matrix $A \in \mathbb{R}^{n \times n}$ contain the number of edges from i to j in element A_{ij}. For column vectors $x \in \mathbb{R}^a$, $z \in \mathbb{R}^b$ and $S \in \mathbb{R}^{a \times b}$, define $\langle x, z \rangle_S = x^T S z$; this function is not necessarily a proper inner product because it does not require that S is non-negative definite. We use standard big-O and little-o notations, i.e. for sequence x_n, y_n; $x_n = o(y_n)$ when y_n is nonzero implies $\lim_{n\to\infty} x_n/y_n = 0$; $x_n = O(y_n)$ implies there exists a positive real number M and an integer N such that $|x_n| \leq M|y_n|$, $\forall n \geq N$.

2. `fastRG`

`fastRG` is motivated by the wide variety of low rank graph models that specify the expectation of the adjacency matrix as $E(A) = XSX^T$ for some matrix (or vector) X and some matrix (or value) S.

Types of low rank models	$X \in$	In each row...
SBM	$\{0,1\}^{n\times K}$	a single one
Degree-Corrected SBM	$\mathbb{R}^{n\times K}$	a single non-zero positive entry
Mixed Membership SBM	$\mathbb{R}^{n\times K}$	non-negative and sum to one
Overlapping SBM	$\{0,1\}^{n\times K}$	a mix of 1s and 0s
Erdős-Rényi	$\{1\}^n$	a single one
Chung-Lu	$\mathbb{R}^n$	a single value

Table 1: Restrictions on the matrix X create different types of well known low rank models. There are further differences between these models that are not emphasized by this table.

Given $X \in \mathbb{R}^{n \times K_x}$, $Y \in \mathbb{R}^{d \times K_y}$ and $S \in \mathbb{R}^{K_x \times K_y}$, `fastRG` samples a random graph. Define A as the $n \times d$ matrix where A_{ij} counts the number of times edge (I,J) was sampled by `fastRG`. In expectation A is XSY^T. Importantly, `fastRG` requires that the elements of X,Y, and S are non-negative. This condition holds for all of the low rank models in the above table. Each of those models set $Y = X$ and enforce different restrictions on the matrix X.

As stated below, `fastRG` samples a (i) directed graph with (ii) multiple edges and (iii) self-loops. After sampling, these properties can be modified to create a simple graph (undirected, no repeated edges, and no self-loops); see Remarks 5 and 6 in Section 2.2 and Theorem 7 in Section 2.3.

Algorithm 1 `fastRG`(X,S,Y)

Require: $X \in \mathbb{R}^{n \times K_x}$, $S \in \mathbb{R}^{K_x \times K_y}$, and $Y \in \mathbb{R}^{d \times K_y}$ with all matrices containing non-negative entries.
  Compute diagonal matrix $C_X \in \mathbb{R}^{K_x \times K_x}$ with $C_X = diag(\sum_i X_{i1}\ ,\ \ldots\ ,\ \sum_i X_{iK_x})$.
  Compute diagonal matrix $C_Y \in \mathbb{R}^{K_y \times K_y}$ with $C_Y = diag(\sum_i Y_{i1}\ ,\ \ldots\ ,\ \sum_i Y_{iK_y})$.
  Define $\tilde{X} = XC_X^{-1}$, $\tilde{S} = C_X S C_Y$, and $\tilde{Y} = YC_Y^{-1}$.
  Sample the number of edges $m \sim Poisson(\sum_{u,v} \tilde{S}_{uv})$.
  for $\ell = 1 : m$ **do**
    Sample $U \in \{1,\ \ldots,\ K_x\}, V \in \{1,\ \ldots,\ K_y\}$ with $\mathbb{P}(U = u, V = v) \propto \tilde{S}_{uv}$.
    Sample $I \in \{1,\ \ldots,\ n\}$ with $\mathbb{P}(I = i) = \tilde{X}_{iU}$.
    Sample $J \in \{1,\ \ldots,\ d\}$ with $\mathbb{P}(J = j) = \tilde{Y}_{jV}$.
    Add edge (I,J) to the graph, allowing for multiple edges (I,J).
  end for

An implementation in `R` is available at `https://github.com/karlrohe/fastRG`. As discussed in Section 2.4, in order to make the algorithm more efficient, the implementation is slightly different from the statement of the algorithm above.

There are two model classes that can help to interpret the graphs generated from `fastRG` and those model classes are explored in the next two subsections. Throughout all of the discussion, the key fact that is exploited by `fastRG` is given in the next Theorem.

Theorem 1 *Suppose that $X \in \mathbb{R}^{n \times K_x}$, $Y \in \mathbb{R}^{d \times K_y}$ and $S \in \mathbb{R}^{K_x \times K_y}$ all contain non-negative entries. Define $x_i \in \mathbb{R}^{K_x}$ as the ith row of X. Define $y_j \in \mathbb{R}^{K_y}$ as the jth row of Y. Let (I,J) be a single edge sampled inside the for loop in `fastRG`(X,S,Y), then*

$$\mathbb{P}\left((I,J) = (i,j)\right) \propto \langle x_i, y_j \rangle_S.$$

Proof

$$\begin{aligned}
\mathbb{P}\big((I,J) = (i,j)\big) &= \sum_{u,v} \mathbb{P}((I,J) = (i,j) | (U,V) = (u,v))\ \mathbb{P}((U,V) = (u,v))) \\
&= \frac{\sum_{u,v} \tilde{X}_{iu} \tilde{Y}_{jv} \tilde{S}_{uv}}{\sum_{u,v} \tilde{S}_{u,v}} = \frac{\sum_{u,v} X_{iu} Y_{jv} S_{uv}}{\sum_{u,v} \tilde{S}_{u,v}} = \frac{x_i^T S y_j}{\sum_{a,b} x_a^T S y_b}
\end{aligned}$$

■

2.1. `fastRG` samples from xlr: a class of edge-exchangeable random graphs

There has been recent interest in edge exchangeable graph models with blockmodel structure (e.g. Crane and Dempsey 2016; Cai et al. 2016; Herlau et al. 2016; Todeschini and Caron 2016). To characterize a broad class of such models, we propose xlr. For notational simplicity, the rest of the paper will suppose that $Y = X \in \mathbb{R}^{n\times K}$ and `fastRG`(X,S) = `fastRG`(X,S,X).

Definition 2 *[xlr] An xlr graph on n nodes and K dimensions is generated as follows,*

1. *Sample $(X,S) \in \mathbb{R}^{n\times K} \times \mathbb{R}^{K\times K}$ from some distribution and define x_i as the ith row of X.*
2. *Initialize the graph to be empty.*
3. *Add independent edges $e_1, e_2, \ldots$ to the graph, where*

$$\mathbb{P}(e_\ell = (i,j)) = \frac{\langle x_i, x_j\rangle_S}{\sum_{a,b}\langle x_a, x_b\rangle_S}.$$

From Theorem 1, `fastRG` samples the edges in xlr.

An xlr graph is both (i) edge-exchangeable as defined by Crane and Dempsey (2016) and (ii) conditional on X and S, its expected adjacency matrix is low rank. By sampling X to satisfy one set of restrictions specified in Table 1, xlr provides a way to sample edge exchangeable blockmodels. xlr stands for *edge-exchangeable and low rank* because it characterizes all edge-exchangeable and low rank random graph models on a finite number of nodes. In particular, by Theorem 4.2 in Crane and Dempsey (2016) if a random undirected graph with an infinite number of edges is edge exchangeable, then the edges are drawn iid from some randomly chosen distribution on edges f. Moreover, let B be the adjacency matrix of a single edge drawn from f. Under the assumption that $E(B|f)$ is rank K, there exist matrices $X \in \mathbb{R}^{n\times K}$ and $S \in \mathbb{R}^{K\times K}$ that are a function of f and give the eigendecomposition $E(B|f) = XSX^T$. This implies that $\mathbb{P}(e_1 = (i,j)|f) \propto \langle x_i, x_j\rangle_S$, where x_i is the ith row of X.

2.2. `fastRG` samples from a generalization of the RDPG

Under the RDPG as described in Young and Scheinerman (2007), the expectation of the adjacency matrix is XX^T for some matrix $X \in \mathbb{R}^{n\times K}$. This implies that the expected adjacency matrix is always non-negative definite (i.e. its eigenvalues are non-negative). However, some parameterizations of the SBM (and other blockmodels) lead to an expected adjacency matrix with negative eigenvalues (i.e. it is not non-negative definite); for example, if the off-diagonal elements of S are larger than the diagonal elements, then XSX^T could have negative eigenvalues. Moreover, even if the elements of X and S are positive, as is the case for the low rank models in Table 1 and as is required for `fastRG`, it is still possible for XSX^T to have negative eigenvalues. By modifying the RDPG to incorporate a matrix S, the model class below incorporates all types of blockmodels.

Definition 3 *[Generalized Random Product Graph (gRPG) model] For n nodes in K dimensions, the gRPG is parameterized by $X \in \mathbb{R}^{n\times K}$ and $S \in \mathbb{R}^{K\times K}$, where each node i is assigned the ith row of X, $x_i = (X_{i1}, \ldots, X_{iK})^T \in \mathbb{R}^K$. For $i, j \in V$, define*

$$\lambda_{ij} = \langle x_i, x_j\rangle_S = \sum_k^K \sum_l^K X_{ik} S_{kl} X_{jl}.$$

Under the gRPG, the adjacency matrix $A \in \mathbb{R}^{n\times n}$ contains independent elements and the distribution of A_{ij} (i.e. the number of edge from i to j) is fully parameterized by $f(\lambda_{ij})$, where f is some mean function.

Below, we will use the fact that the gRPG only requires that the λ_{ij}s specify the distribution of A_{ij}, allowing for A_{ij} to be non-binary (as in multi-graphs and weighted graphs) or to have edge probabilities which are a function of λ_{ij}.

Theorem 4 *For $X \in \mathbb{R}^{n\times K}$ and $S \in \mathbb{R}^{K\times K}$, each with non-negative elements, if $\tilde{A}$ is the adjacency matrix of a graph sampled with `fastRG`(X,S), then $\tilde{A}$ is a Poisson gRPG with $\tilde{A}_{ij} \sim Poisson(\langle x_i, x_j\rangle_S)$.*

The proof is contained in the appendix.

Remark 5 (Simulating an undirected graph) *As defined, both the gRPG model and `fastRG` generate directed graphs. An "undirected gRPG" should add a constraint to Definition 3 that $A_{ij} = A_{ji}$ for all i, j. To sample such a graph with `fastRG`, input $S/2$ instead of S, then after sampling a directed graph with `fastRG`, symmetrize each edge by removing its direction (this doubles the probability of an edge, hence the need to input $S/2$). Theorem 4 can be easily extended to show this is an undirected gRPG.*

Remark 6 (Simulating a graph without self-loops) *As defined, both the gRPG model and `fastRG` generate graphs with self-loops. A "gRPG without self-loops" should add a constraint to Definition 3 that $A_{ii} = 0$ for all i. A graph from `fastRG` can be converted to a gRPG without self-loops by simply (1) sampling $m \sim Poisson(\sum_{u,v}\tilde{S}_{uv} - \sum_i \langle x_i, x_i\rangle_S)$ and (2) resampling any edge that is a self-loop. The proof of Theorem 4 can be extended to show that this is equivalent.*

2.3. Approximate Bernoulli-edges

To create a simple graph with `fastRG` (i.e. no multiple edges, no self-loops, and undirected), first sample a graph with `fastRG`. Then, perform the modifications described in Remarks 5 and 6. Then, keep an edge between i and j if there is at least one edge in the multiple edge graph; define the threshold function, $t(A_{ij}) = \mathbb{1}(A_{ij} > 0)$, where $t(A)$ applies element-wise.

If $\tilde{A}$ is a Poisson gRPG, then $t(\tilde{A})$ is a Bernoulli gRPG with mean function $f(\lambda_{ij}) = 1 - exp(-\lambda_{ij})$. Let B be distributed as Bernoulli gRPG(X,S) with identity mean function,

$$B_{ij} \sim Bernoulli(\lambda_{ij}).$$

Theorem 7 shows that in the sparse setting, there is a coupling between $t(\tilde{A})$ and B such that $t(\tilde{A})$ is approximately equal to B. The theorem is asymptotic in n; a superscript of n is suppressed on $\tilde{A}, B$ and λ.

Theorem 7 *Let $\tilde{A}$ be a Poisson gRPG and let B be a Bernoulli gRPG using the same set of λ_{ij}s, with $\tilde{A}_{ij} \sim Poisson(\lambda_{ij})$ and $B_{ij} \sim Bernoulli(\lambda_{ij})$. Let $t(\cdot)$ be the thresholding function for $\tilde{A}$.*

Let α_n be a sequence. If $\lambda_{ij} = O(\alpha_n/n)$ for all i, j and there exists some constant $c > 0$ and $N > 0$ such that $\sum_{ij}\lambda_{ij} > c\alpha_n n$ for all $n > N$, then there exists a coupling between $t(\tilde{A})$ and B such that

$$\frac{E\|t(\tilde{A}) - B\|_F^2}{E\|B\|_F^2} = O(\alpha_n/n).$$

For example, in the sparse graph setting where $\lambda_{ij} = O(1/n)$ and $\sum_{ij} \lambda_{ij} = O(n)$, $\alpha_n = 1$. Under this setting and the coupling defined in the proof, all of the $O(n)$ edges in $t(\tilde{A})$ are contained in B and B has an extra $O(1)$ more edges than $t(\tilde{A})$.

The condition $\alpha_n = o(n)$ implies that all edge probabilities decay. If one is interested in models where some λ_{ij}'s are constant (e.g. certain models with heavy tailed degree distributions), then there are three possible paths forward.

1. Segment the pairs i, j into two sets (large and small λ_{ij}'s) and use two different sampling techniques on each set.

2. Use `fastRG` as a proposal distribution for rejection sampling (if for some $\varepsilon > 0$, $\lambda_{ij} < 1 - \varepsilon$ for all i, j, then rejection sampling would still be $O(m)$ operations).

3. Use `fastRG` and expect edge attenuation (no greater than 37%) for high probability edges.

Regarding the third point, consider the coupling in the proof of Theorem 7 without any condition on λ_{ij}. The coupling ensures that every edge in $t(\tilde{A})$ is also contained in B. Conversely, conditioned on edge i, j appearing in B, then the probability that this edge is included in $t(\tilde{A})$ is a greater than $1 - exp(-\lambda_{ij}) \geq 1 - exp(-1) > .63$.

2.4. Implementation of `fastRG`

Code at `https://github.com/karlrohe/fastRG` gives an implementation of `fastRG` in `R`. It also provides wrappers that simulate the SBM, Degree-Corrected SBM, Overlapping SBM, and Mixed Membership SBM. The code for these models first generates the appropriate X and then calls `fastRG`. In order to help control the edge density of the graph, `fastRG` and its wrappers can be given an additional argument avgDeg. If avgDeg is given, then the matrix S is scaled so that `fastRG` simulates a graph with expected average degree equal to avgDeg. Without this, parameterizations can easily produce very dense graphs.

To accelerate the running time of `fastRG`, the implementation is slightly different than the statement of the algorithm above. The difference can be thought of as sampling all of the (U,V) pairs before sampling any of the Is or Js. In particular, the implementation samples $\varpi \in R^{K \times K}$ as multinomial$(m, \tilde{S}/\sum_{u,v} \tilde{S}_{uv})$. Then, for each $u \in \{1, \dots K\}$, it samples $\sum_v \varpi_{u,v}$-many Is from the distribution $\tilde{X}_{\cdot u}$. Similarly, for each $v \in \{1, \dots K\}$, it samples $\sum_u \varpi_{u,v}$-many Js from the distribution $\tilde{X}_{\cdot v}$. Finally, the indexes are appropriately arranged so that there are $\varpi_{u,v}$-many edges (I,J) where $I \sim X_{\cdot u}$ and $J \sim X_{\cdot v}$. Recall that the statement of `fastRG` above allows for X and Y, where those matrices can have different numbers of rows and/or columns; the implementation also allows for this.

Under the SBM, it is possible to use `fastRG` to sample from the Bernoulli gRPG with the *identity* mean function instead of the mean function $1 - \exp(-\langle x_i, x_j \rangle_S)$ that is created by the thresholding function t from Section 2.3. The wrapper for the SBM does this by first transforming each element of S as $-\ln(1 - S_{ij})$ and then calling `fastRG`. The others models are not amenable to this trick; by default, they sample from the Poisson gRPG with identity mean function.

3. Experiments

3.1. Running time of `fastRG` on large and sparse graphs

To examine the running time of `fastRG`, we simulated a range of different values of n and $E(m)$, where $E(m)$ is the expected number of edges. In all simulations $X = Y$ and $K = 5$. The elements of X are independent $Poisson(1)$ random variables and the elements of S are independent $Uniform[0,1]$ random variables. To specify $E(m)$, the parameter avgDeg is set to $E(m)/n$. The values of n range from $10,000$ to $10,000,000$ and the values of $E(m)$ range from $100,000$ to $100,000,000$. The graph was taken to be directed, with self-loops and multiple edges. Moreover, the reported times are only to generate the edge list of the random graph; the edge list is not converted into a sparse adjacency matrix, which in some experiments would have more than doubled the running time. Each pair of n and $E(m)$ is simulated one time; deviations around the trend lines indicate the variability in run time.

In Figure 1, the vertical axes present the running time in `R` on a Retina 5K iMac, 27-inch, Late 2014 with 3.5 GHz Intel i5 and 8GB of 1600 MHz DDR3 memory. In the left panel of Figure 1, each line corresponds to a single value of n and $E(m)$ increases along the horizontal axis. In the right panel of Figure 1, each line corresponds to a single value of $E(m)$ and n increases along the horizontal axis. All axes are on the log_{10} scale. The solid black line has a slope of 1. Because the data aligns with this black line, this suggests that `fastRG` runs in linear time.

The computational bottleneck is sampling the Is and Js. The implementation uses Walker's Alias Method (Walker, 1977) (via `sample` in `R`). To take m samples from a distribution over n elements, Walker's Alias Method requires $O(m + \ln(n)n)$ operations (Vose, 1991). However, the log dependence is not clearly evident in the right plot of Figure 1; perhaps it would be visible for larger values of n.

3.2. Comparison to a previous technique

Previously, Hagberg and Lemons (2015) studied a fast technique to generate sparse random kernel graphs. Under the random kernel graph model, nodes i and j connect with probability $\kappa(i/n, j/n)$, where the function κ is non-negative, bounded, symmetric, measurable, and almost surely continuous. This model class includes the SBM and the Degree-Corrected SBM. It is difficult to see how a more general low rank model could be parameterized as a random kernel graph with an almost surely continuous κ. For example, we suspect that Mixed Membership SBMs could not be parameterized as such.

The algorithm proposed in Hagberg and Lemons (2015) is fast when it is fast to compute (i) the integral $F(y,a,b) = \int_a^b \kappa(x,y)dx$ and (ii) its roots, that is for any y,a,r, solve for $F(y,a,b) = r$. Their software, which we will refer to as $fast$-κ is in python and generates a NetworkX graph.

For a simple benchmark to compare the running times of `fastRG` and $fast$-κ, Figure 2 repeats the run time experiment that was performed in Hagberg and Lemons (2015). This simulation is for an Erdős-Rényi graph with expected degree 10, for n ranging between $5,000$ and $5M$. Speed comparisons are troubled by the fact that our code returns an edge list in `R` and $fast$-κ returns a NetworkX graph in python. Converting from an edge list to other data types takes longer than sampling the edge list with `fastRG`. For example, converting the edge list to a sparse matrix (a type that is convenient for spectral estimators) takes about as long as sampling the edge list with `fastRG`. Converting the edge list to an igraph takes about 10x longer than sampling the edge list

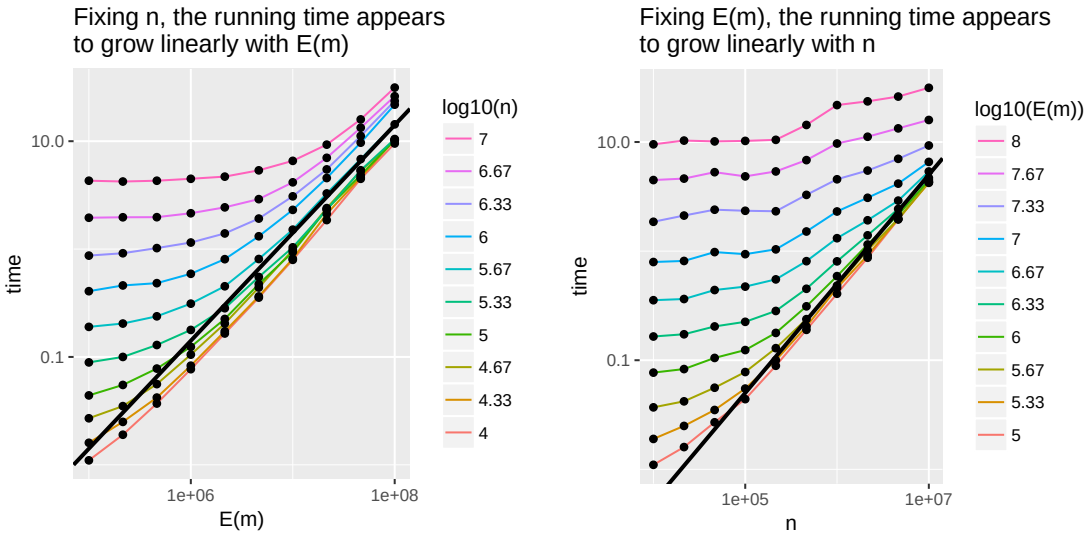

Figure 1: Both plots present the same experimental data. In the left plot, each line corresponds to a different value of n and they are presented as a function of $E(m)$. In the right plot, each line corresponds to a different value of $E(m)$ and they are presented as a function of n. On the right side of both plots, the lines start to align with the solid black line, suggesting a linear dependence on $E(m)$ and n.

with `fastRG`. There are three lines in Figure 2 for `fastRG`, one line for each data type (edge list, sparse adjacency matrix, and igraph). The speed comparison in Figure 2 corresponds to the average running time over 10 simulations performed on a 2015 MacBook Pro, 2.8 GHz Intel Core i7, with 16 GB 1600 MHz DDR3 running Python 3.5.2 and `R` 3.3.2. The slope of the blue line corresponds to the running time $O(n \log n)$. While none of these packages have been optimized for speed, they are all sufficiently fast for a wide range of purposes.

3.3. Simulating small and dense graphs with `fastRG`

This section investigates the graph density at which `fastRG` becomes slower than simulating each element A_{ij} as a Bernoulli random variable. In Figure 3, the reported time to compute this naive (element-wise) algorithm includes both (i) the time it takes to compute the probabilities `eA = X%*%B%*%t(X)` and (ii) sample the edges `z = rbinom(length(eA), 1, eA)`. The time for `fastRG` is for a directed graph, represented as a sparse matrix. The time to compute `fastRG` also includes the time it takes to construct X and B.

Figure 3 compares these two approaches on a set of Stochastic Blockmodels for values $K \in \{2,5,10\}$, $n \in \{500,1000,5000,10000\}$, and graph density $\rho = n^{-2}E(m)$ varying between .02 and .35. In all simulations, the S matrix is proportional to $I_K + J_K \in \mathbb{R}^{K \times K}$, where I_K is the identity matrix and J_K is the matrix of ones. The scale of S is adjusted to ensure the correct density ρ. For each model, both `fastRG` and the naive simulation are used to simulate two graphs. The line is a quadratic fit with ordinary least squares. In each panel, the vertical line is at $\rho = .25$, which is

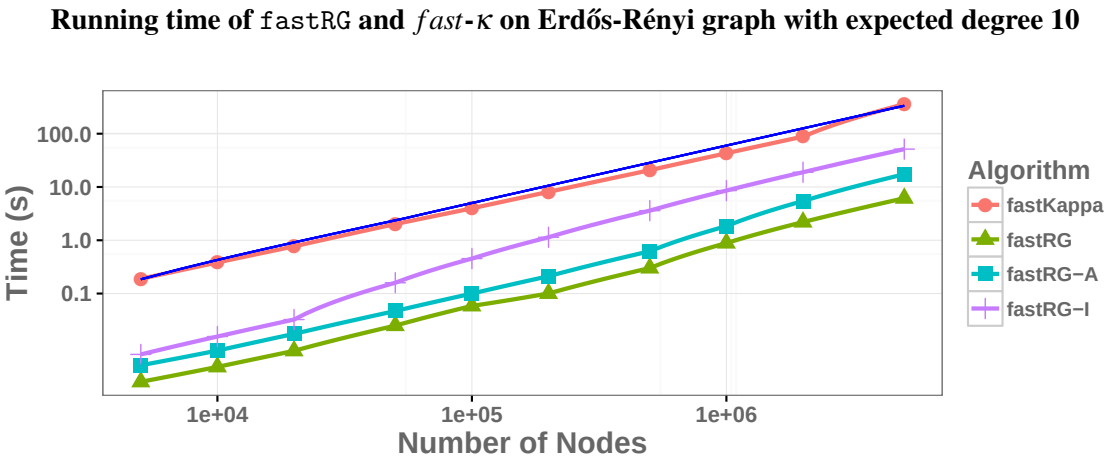

Figure 2: As the number of nodes increases (horizontal axis), all of the running times increase in parallel to the solid blue line which gives the rate $O(n \log n)$. The bottom three lines all correspond to `fastRG`, outputting three different graph types (edge list, sparse adjacency matrix, and igraph). For example, in roughly 8 seconds, *fast*-κ generates a graph with 20k nodes and `fastRG` generates an igraph with 1M nodes. To generate the random edge list on 1M nodes with `fastRG` takes less than 1 second

approximately the crossover point in all simulations. For scale, $\rho = .25$ corresponds to expected degrees 125, 250, 1250, and 2,500 respectively for n values $500, 1000, 5000$, and 10000.

Acknowledgments

This research was supported by NSF grant DMS-1612456 and ARO grant W911NF-15-1-0423.

Appendix A. Proofs

For an integer d, define $1_d \in \mathbb{R}^d$ as a vector of ones. The proof of Theorem 4 requires the following lemma, which says that a vector (or matrix) of independent Poisson entries becomes multinomial when you condition on the sum of the vector (or matrix).

Lemma 8 *Let $A \in \mathbb{R}^{n \times n}$ be the random matrix whose i,jth element $A_{ij} \overset{i.d.}{\sim} Pois(\lambda_{ij})$ $i, j = 1, \ldots, n$. Then conditioned on $\sum_{i,j} A_{ij} = 1_n^T A 1_n = m$,*

$$(A_{11}, A_{12}, \ldots, A_{nn}) \sim Multinomial(m, \lambda / \sum_{ij} \lambda_{ij})$$

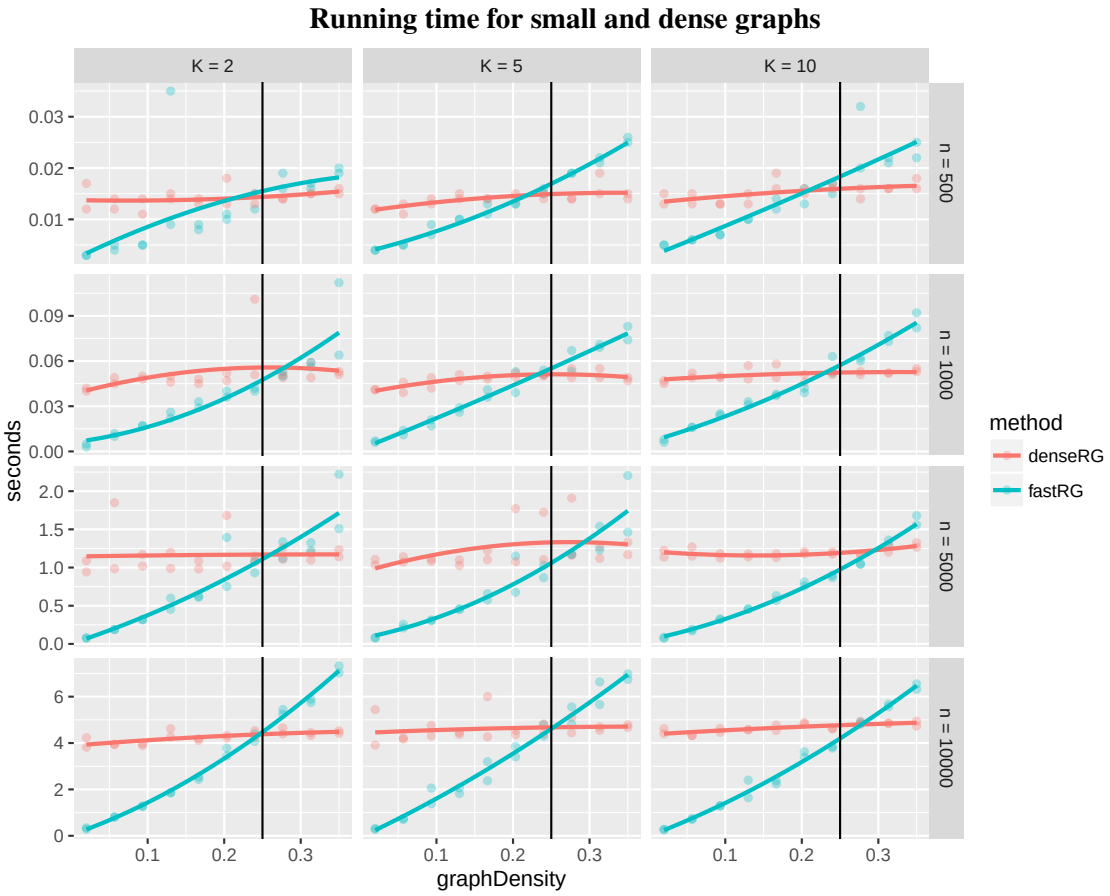

Figure 3: This figure compares the run time of `fastRG` to the run time of simulating each A_{ij} as a Bernoulli random variable (`denseRG` in the legend). Note that the number of edges grows quadratically with the edge density. So, as the density of the graph increases (horizontal axis), the running time of `fastRG` grows quadratically, whereas the running time of the naive algorithm does not depend on the density of the graph. In this simulation, the crossover is around $\rho = .25$ (black line). This figure shows that even for relatively small n and in the dense regime, `fastRG` has potential to be faster than the naive approach.

where $\lambda = (\lambda_{11},\ \lambda_{12},\ \ldots,\ \lambda_{nn})$. *That is, let* $a \in \mathbb{R}^{n\times n}$ *be a fixed matrix of integers with* $1_n^T a 1_n = m$, *then*

$$\begin{aligned}\mathbb{P}(A = a | 1_n^T A 1_n = m) &= \mathbb{P}(A_{11} = a_{11},\ A_{12} = a_{12},\ \ldots,\ A_{nn} = a_{nn} | 1_n^T A 1_n = m) \\ &= \frac{m!}{\prod_{i,j} a_{ij}!} \prod_{i,j} \left(\frac{\lambda_{ij}}{\lambda_{11} + \lambda_{12} + \cdots + \lambda_{nn}} \right)^{a_{ij}}.\end{aligned}$$

For completeness, a proof of this classical result is given at the end of the paper. The next proof is a proof of Theorem 4.

Proof Let A come from the Poisson gRPG with X and S and identity mean function. Let $\tilde{A}$ be a sample from `fastRG`. For any fixed adjacency matrix a, we will show that $\mathbb{P}(A = a) = \mathbb{P}(\tilde{A} = a)$.

Define $m = 1_n^T a 1_n$ and decompose the probabilities,

$$\mathbb{P}(A = a) = \mathbb{P}(1_n^T A 1_n = m)\mathbb{P}(A = a | 1_n^T A 1_n = m) \quad (1)$$
$$\mathbb{P}(\tilde{A} = a) = \mathbb{P}(1_n^T \tilde{A} 1_n = m)\mathbb{P}(\tilde{A} = a | 1_n^T \tilde{A} 1_n = m). \quad (2)$$

The proof will be divided into two parts. The first part shows that $\mathbb{P}(1_n^T A 1_n = m) = \mathbb{P}(1_n^T \tilde{A} 1_n = m)$ and the second part will show that $\mathbb{P}(A = a | 1_n^T A 1_n = m) = \mathbb{P}(\tilde{A} = a | 1_n^T \tilde{A} 1_n = m)$.

Part 1: The sum of independent Poisson variables is still Poisson,

$$\sum_{ij} A_{ij} \sim Poisson(\sum_{ij} \lambda_{ij}).$$

So, we must only show that $1_n^T A 1_n$ and $1_n^T \tilde{A} 1_n$ have the same Poisson parameter:

$$\sum_{ij} \lambda_{ij} = 1_n^T XSX 1_n = 1_n^T XCC^{-1} SC^{-1} CX 1_n = 1_n^T \tilde{X} \tilde{S} \tilde{X} 1_n = 1_n^T \tilde{X} \tilde{S} \tilde{X} 1_n = 1_K^T \tilde{S} 1_K = \sum_{u,v} \tilde{S}_{uv}.$$

Part 2: After conditioning on $1_n^T A 1_n = m$, Lemma 8 shows that A has the multinomial distribution. In `fastRG`, we first sample $1_n^T \tilde{A} 1_n$ and then add edges with the multinomial distribution. So, we must only show that the multinomial edge probabilities are equal for A and $\tilde{A}$. From Lemma 8, the multinomial edge probabilities for A are $\lambda_{ij} / \sum_{a,b} \lambda_{ab}$. To compute the multinomial edge probabilities for $\tilde{A}$, recall that (I,J) is a single edge added to the graph in `fastRG`. By Theorem 1,

$$\mathbb{P}(\tilde{A}_{ij} = 1 | 1_n^T \tilde{A} 1_n = 1) = \mathbb{P}\big((I,J) = (i,j)\big) = \frac{\langle x_i, x_j \rangle_S}{\sum_{a,b} \langle x_a, x_b \rangle_S} = \frac{\lambda_{ij}}{\sum_{a,b} \lambda_{ab}}$$

This concludes the proof. ■

Proof [Proof of Theorem 7] Let $U_{ij} \overset{i.i.d}{\sim} Uniform(0,1)$. Define $\mathscr{A}$ and $\mathscr{B}$:

$$\mathscr{A}_{ij} = \mathbf{1}(U_{ij} > 1 - e^{-\lambda_{ij}}),$$
$$\mathscr{B}_{ij} = \mathbf{1}(U_{ij} > \lambda_{ij}).$$

Note that $\mathscr{A}$ and $\mathscr{B}$ are equal in distribution to $t(\tilde{A})$ and B respectively. By Taylor expansion,

$$E\|\mathscr{A}-\mathscr{B}\|_F^2 = \sum_{i,j}(\lambda_{ij}-(1-e^{-\lambda_{ij}})) = \sum_{i,j}\sum_{k=2}^{\infty}(-\lambda_{ij})^k/k! = \sum_{i,j}O(\lambda_{ij}^2) = \sum_{i,j}O((\alpha_n/n)^2) = O(\alpha_n^2).$$

Then, $E\|B\|_F^2 = \sum_{ij}\lambda_{ij} > c\alpha_n n$. So, defining $t(\tilde{A})$ and B with the above coupling yields the result. ■

Proof [proof of Lemma 1]

$$\begin{aligned}\mathbb{P}(A=a|1_n^TA1_n=m) &= \frac{\mathbb{P}(A=a)}{\mathbb{P}(1_n^TA1_n=m)} = \frac{\prod_{i,j}\frac{\lambda_{ij}^{a_{ij}}}{a_{ij}!}e^{-\lambda_{ij}}}{\frac{(\lambda_{11}+\lambda_{12}+\cdots+\lambda_{nn})^m}{m!}e^{-(\lambda_{11}+\lambda_{12}+\cdots+\lambda_{nn})}} \\ &= \frac{m!}{\prod_{i,j}a_{ij}!}\prod_{i,j}\left(\frac{\lambda_{ij}}{\lambda_{11}+\lambda_{12}+\cdots+\lambda_{nn}}\right)^{a_{ij}}\end{aligned}$$

■

References

E Airoldi, E Blei, S Fienberg, and E Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(5):1981–2014, 2008.

D Cai, T Campbell, and T Broderick. Edge-exchangeable graphs and sparsity. In *Advances in Neural Information Processing Systems*, pages 4242–4250, 2016.

H Crane and W Dempsey. Edge exchangeable models for network data. *arXiv preprint arXiv:1603.04571*, 2016.

A Hagberg and N Lemons. Fast generation of sparse random kernel graphs. *PloS one*, 10(9): e0135177, 2015.

T Herlau, M Schmidt, and M Mø rup. Completely random measures for modelling block-structured sparse networks. In *Advances in Neural Information Processing Systems 29*. 2016.

P Holland, K Laskey, and S Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2): 109–137, 1983.

B Karrer and M Newman. Stochastic blockmodels and community structure in networks. *Phys. Rev. E*, 83:016107, Jan 2011.

P Latouche and C Ambroise. Overlapping stochastic block models with application to the french political blogosphere. *Annals of Applied Statistics*, 5(1):309–336, 2011.

R Lehoucq, D Sorensen, and P Vu. Arpack: An implementation of the implicitly re-started arnoldi iteration that computes some of the eigenvalues and eigenvectors of a large sparse matrix. *Available from netlib@ ornl. gov under the directory scalapack*, 1995.

A Todeschini and F Caron. Exchangeable Random Measures for Sparse and Modular Graphs with Overlapping Communities. *arXiv preprint arXiv:1602.02114*, February 2016.

M Vose. A linear algorithm for generating random numbers with a given distribution. *IEEE Transactions on software engineering*, 17(9):972–975, 1991.

A Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, 3(3):253–256, 1977.

S Young and E Scheinerman. *Random Dot Product Graph Models for Social Networks*, pages 138–149. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.