Monte Carlo Maximum Likelihood for Exponential Random Graph Models: From Snowballs to Umbrella Densities

Monte Carlo Maximum Likelihood for Exponential Random Graph Models: From Snowballs to Umbrella Densities

Kevin Bartz, Joseph K. Blitzstein, and Jun S. Liu

Department of Statistics, Harvard University

September 2008

Abstract

Performing maximum-likelihood estimation for parameters in an exponential random graph model is challenging because of the unknown normalizing constant. Geyer and Thompson (1992) provide a Monte Carlo algorithm that uses samples from a distribution with known parameters to approximate the full likelihood, which is then maximized to estimate the MLE. We refine the approximation to use sample draws collected from differently parameterized distributions, increasing the effective sample size and improving the accuracy of the MLE estimate. Substantially lower estimation variance is demonstrated in simulated and actual network data. We also propose a new method for finding a starting point: scaling the MLE parameters of a small graph subsampled from the original graph. Through simulation with the triad model, this starting point produces convergence in many cases where the standard starting point (based on pseudolikelihood) fails to give convergence, though the reverse is also true.[1]

[1] Kevin Bartz is Ph.D. Student, Statistics (e-mail: bartz@fas.harvard.edu); Joseph K. Blitzstein is Assistant Professor, Statistics (e-mail: blitz@stat.harvard.edu); and Jun S. Liu is Professor, Statistics (e-mail: jliu@stat.harvard.edu). All are at Harvard University, Cambridge, MA 02138. Comments are welcome. An R package graphestimate, which reproduces the results of this paper, is available at www.kevinbartz.com/graph. The authors thank Xiao-Li Meng and the referees for helpful comments.

Keywords: likelihood approximation, discrete exponential family, network, pseudolikelihood, graph subsampling, likelihood approximation

1 INTRODUCTION

Exponential families of random graphs are among the most widely-used, flexible models for complex networks, particularly social networks. In these models, a set of (hopefully) scientifically interesting choice of graph statistics is made, and a discrete exponential family is formed with the graph statistics as sufficient statistics. So prevalent have these models become that there are two well-established names: *exponential random graph models* (*ERGM*), and the shorter term p^* (see Robins et al Robins et al. (2007) for a recent review).

Parameter estimation in ERGMs tends to be very challenging due to the normalizing constant typically being unknown, requiring a combinatorially massive effort to evaluate a sum over all graphs. An early approach by Strauss and Ikeda (1990) define a maximum pseudolikelihood estimator (MPLE), which is computationally fast but is rather unreliable (it is unsurprising that ignoring the dependencies in the model is perilous). For example, Corander et al. (1998) present a simulation study suggesting that the MPLE may be inconsistent. Thus, recent efforts have focused on using MCMC to approximate the likelihood function. In particular, Geyer and Thompson (1992) introduce "Monte Carlo maximum likelihood" (MCMCMLE), which neatly uses random draws from the distribution with a *chosen* β_0 to approximate the likelihood function. This method is implemented for ERGMs in the network analysis software suite statnet Handcock et al. (2003). If β_0 is not close to the MLE, it may require an infeasibly large MCMC sample size to obtain a reasonable approximation, so an iterative method is used: choose β_1 to maximize the approximate likelihood obtained from the β_0-graphs, then choose β_2 based on β_1-graphs, etc., until convergence. While clearly an improvement over pseudolikelihood, the method is computationally expensive, often requiring a huge number of draws and iterations. Moreover, the method is highly sensitive to the initial parameter values, as demonstrated in Handcock (2003) and Snijders (2002).

In this paper, we propose a method for reducing the computational burden of MCMCMLE by using the generated graphs more efficiently, avoiding the waste of discarding all the previously generated graphs from the earlier iterations. For example, the thousands of graphs generated from the β_0 distribution still contain valuable information that can be leveraged in later stages of the estimation. This is achieved with the help of an idea from

Kong et al. (2003), which provides a semi-parametric method for simultaneously estimating multiple normalizing constants, given samples from the corresponding distributions. This enables using *all* the previous iterations' simulation draws to approximate the likelihood, rather than just the samples at the current parameter value. Our simulations show large efficiency gains from this method, and suggest that for practical examples the effective sample size is nearly equal to the total number of simulations. The simulations are based on several standard network data sets and reasonably simple ERGMs where the estimation is already challenging, such as the *triad model*, a canonical specification of Frank and Strauss (1986) with features for the number of edges, triangles, and two-stars in the graph.

Additionally, we propose an alternative way to pick the starting point β_0, rather than the traditional choice of the MPLE. Using the MPLE for the initial parameter value often results in nonconvergence. Instead, we *subsample* from the graph to obtain a smaller, more tractable graph, find the MLE for the subgraph, and then fit a scaling transformation to obtain an estimate for the full graph. While this method does not dominate the MPLE, we find that it leads to convergence for many graphs for which the MPLE fails as a starting point. Thus, both the subsample-based on the MPLE-based initial values are useful tools to have on hand.

The paper is organized as follows. Section 2 provides background on the model, the problem considered, and previous approaches to finding the MLE. Section 3 presents two novel improvements to the efficiency of the methods: one based on importance sampling, and a further refinement based on Kong et al. (2003). Section 4 presents our subsample-based procedure for choosing starting points for an MLE-finding algorithm, and Section 7 discusses the results and possible future directions. An R package `graphestimate` implementing the methods proposed is available at `www.kevinbartz.com/graph`.

2 BACKGROUND

A *graph* consists of a set of *nodes* together with a set of *edges*, each of which corresponds to a tie between two nodes. For notational simplicity, we focus here on undirected graphs, in which ties are nondirectional. Let $x(G)$ denote a feature of the graph G (a graph statistic).

Typically features are geometrically natural quantities such as the number of edges or the number of triangles in the graph. Given a set of features $\{x_i\}_{i=1,\dots,q}$, the exponential random graph model (ERGM, also often called p^*) assumes that G arises from a discrete exponential family

$$p_{\boldsymbol{\beta}}(G) \quad = \quad \frac{\exp(\boldsymbol{\beta}'\boldsymbol{x}(G))}{c(\boldsymbol{\beta})} \triangleq \frac{q_{\boldsymbol{\beta}}(G)}{c(\boldsymbol{\beta})}, \tag{1}$$

where $c(\boldsymbol{\beta})$ is the normalizing constant, whose analytic form is generally unknown due to the combinatorial complexity of summing over all possible graphs. Note that there is an implicit conditioning on the number of nodes in the graph, which is fixed for all graphs generated by $p_{\boldsymbol{\beta}}$.

Our goal is to find the MLE of $\boldsymbol{\beta}$. However, since $c(\boldsymbol{\beta})$ is unknown, the MLE generally cannot be found analytically, so we seek a reliable approximation of the MLE.

2.1 Pseudolikelihood

The maximum pseudo-likelihood estimator (MPLE) of Strauss and Ikeda (1990) provides fast and convenient parameter estimates, but without guarantees of consistency or reliable guidance as to when the method will work well. The pseudolikelihood takes on an easy analytical form by ignoring dependency, thereby giving a likelihood for which standard logistic regression algorithms can be employed. Given a graph G, let G_{ij} be an indicator variable representing the state of the edge linking nodes i and j. Let $G'(i,j)$ be the same graph as G, but with the i,j edge toggled, and let $G_{-i,-j}$ be the set of all other edges. The pseudo-likelihood is

$$\begin{aligned}\prod_{i,j} P(G_{ij} = g_{ij} | G_{-i,-j}) \quad &= \quad \prod_{i,j} \frac{p_{\boldsymbol{\beta}}(G)}{p_{\boldsymbol{\beta}}(G) + p_{\boldsymbol{\beta}}(G'(i,j))} \\ &= \quad \prod_{i,j} \operatorname{logit}^{-1}\left(\boldsymbol{\beta}'[\boldsymbol{x}(G) - \boldsymbol{x}(G'(i,j))]\right)\end{aligned} \tag{2}$$

The change statistics $\boldsymbol{x}(G) - \boldsymbol{x}(G')$ are generally easy to compute, depending on the features used. For example, for the edge feature, $x(G) - x(G')$ increments by one if an edge is added and decrements if an edge is deleted. Table 1 gives the change statistics for the triad model.

The form (2) is identical to the likelihood of a logistic regression, where the true edge states are the dependent variable and the change statistics $\boldsymbol{x}(G) - \boldsymbol{x}(G')$ are the independent

Table 1: Change Statistics $x(G) - x(G'(i,j))$ for the Triad Model

	Edge G_{ij} Perturbed	
Feature	**Added**	**Deleted**
Edges	$+1$	-1
Triangles	$\sum_{i'} G_{ji'} G_{i'i}$	$-\sum_{i'} G_{ji'} G_{i'i}$
Two-stars	$\sum_{i'} G_{ii'} + G_{ji'}$	$-\sum_{i'} G_{ii'} + G_{ji'}$

variables. The maximizing $\widehat{\boldsymbol{\beta}}_{\text{MPLE}}$ are the usual logistic regression estimates. Note though that the standard errors estimated by logistic regression are not reliable here, and the MPLE's mathematical properties are poorly understood. Thus, the MPLE is most useful as a rough estimate of the MLE for use as a starting point for other, iterative methods.

2.2 Markov Chain Monte Carlo Maximum Likelihood

Geyer and Thompson (1992) approximate the likelihood using m samples $\{G_i\} \sim p_{\boldsymbol{\beta}_0}$ for known $\boldsymbol{\beta}_0$, which can be realized through an MCMC procedure as described in Section 2.3. The approximated likelihood is maximized to approximate the MLE.

To obtain the approximated likelihood, write the log likelihood ratio of $\boldsymbol{\beta}$ to $\boldsymbol{\beta}_0$ as

$$\begin{aligned} \Lambda(\boldsymbol{\beta}; G, \boldsymbol{\beta}_0) &= l(\boldsymbol{\beta}; G) - l(\boldsymbol{\beta}_0; G) \\ &= \log \frac{q_{\boldsymbol{\beta}}(G)}{q_{\boldsymbol{\beta}_0}(G)} - \log \frac{c(\boldsymbol{\beta})}{c(\boldsymbol{\beta}_0)} \\ &\triangleq A(\boldsymbol{\beta}; G, \boldsymbol{\beta}_0) - B(\boldsymbol{\beta}; \boldsymbol{\beta}_0) \end{aligned} \tag{3}$$

Since $q_{\boldsymbol{\beta}}(G)$ is simply the exponential component from (1), A has a known, simple form:

$$A(\boldsymbol{\beta}; G, \boldsymbol{\beta}_0) = (\boldsymbol{\beta} - \boldsymbol{\beta}_0)' \boldsymbol{x}(G)$$

The ratio B of normalizing constants is unknown, but can be approximated using the sampled graphs and importance sampling (see, for example, Liu (2002)):

$$\begin{aligned}
G_i &\sim p_{\boldsymbol{\beta}_0} \\
\hat{B}(\boldsymbol{\beta};\boldsymbol{\beta}_0,\{G_i\}) &= \log\left\{\frac{1}{m}\sum_{i=1}^{m}\frac{\hat{c}(\boldsymbol{\beta})}{\hat{c}(\boldsymbol{\beta}_0)}\right\} \\
&= \log\left\{\frac{1}{m}\sum_{i=1}^{m}\frac{q_{\boldsymbol{\beta}}(G_i)}{q_{\boldsymbol{\beta}_0}(G_i)}\right\} \\
&= \log\left\{\frac{1}{m}\sum_{i=1}^{m}\exp((\boldsymbol{\beta}-\boldsymbol{\beta}_0)'\boldsymbol{x}(G_i))\right\} \qquad (4)
\end{aligned}$$

Note the dependencies: A depends on the parameters $\boldsymbol{\beta}$ and $\boldsymbol{\beta}_0$ and the data G; B depends on only the parameters, while $\hat{B}$ uses the parameters and the samples but not the data.

Plugging in $\hat{B}$ for B yields the approximated log likelihood ratio,

$$\hat{\Lambda}(\boldsymbol{\beta};G,\boldsymbol{\beta}_0,\{G_i\}) = A(\boldsymbol{\beta};G,\boldsymbol{\beta}_0) - \hat{B}(\boldsymbol{\beta};\boldsymbol{\beta}_0,\{G_i\})$$

Note that as a function of $\boldsymbol{\beta}$, Λ is the log likelihood $l(\boldsymbol{\beta};G)$ plus a constant, so maximizing Λ yields the MLE. Likewise, maximizing $\hat{\Lambda}$ provides the MCMCMLE, an estimate of the MLE.

Of course, the MCMCMLE changes based on the drawn $\{G_i\}$. The error depends greatly on the accuracy of the approximation $\hat{\Lambda}$, which drops the further $\boldsymbol{\beta}$ is from $\boldsymbol{\beta}_0$. Geyer and Thompson (1992) recommend an iterative approach, sampling new data at the current MCMCMLE and re-estimating. The maximum found is unique since the likelihood, arising from an exponential family, is globally concave.

1. Initialize with a point $\boldsymbol{\beta}^{(0)}$, usually taken to be the MPLE. Then for each t,
   (a) Sample m realizations $\{G_i^{(t)}\} \sim p_{\boldsymbol{\beta}^{(t)}}$.
   (b) Set $\boldsymbol{\beta}^{(t+1)} = \arg\max_{\boldsymbol{\beta}} \hat{\Lambda}(\boldsymbol{\beta};G,\boldsymbol{\beta}^{(t)},\{G_i\})$.
2. Stop after T iterations, or once $\max_{\boldsymbol{\beta}} \hat{\Lambda}(\boldsymbol{\beta};G,\boldsymbol{\beta}^{(t)},\{G_i\}) < \varepsilon$ for some fixed ε. $\boldsymbol{\beta}^{(T)}$ is the MCMCMLE.

2.3 Sampling Graphs

To sample graphs with parameters $\boldsymbol{\beta}$ we employ the Metropolis algorithm. We mutate graph G_i to G_i' through by toggling a single possible edge from its current state Y_{ij} to $1-Y_{ij}$. We select the edge uniformly at random, so the proposal distribution is symmetric. The ratio of probabilities, governing acceptance of G_i' as G_{i+1}, is then

$$\frac{p(G_i')}{p(G_i)} \quad = \quad \exp(\boldsymbol{\beta}'(\boldsymbol{x}(G_i') - \boldsymbol{x}(G_i))).$$

As in the pseudo-likelihood case, the change statistics $\boldsymbol{x}(G_i') - \boldsymbol{x}(G_i)$ are easily computed using Table 1. Note that it is particularly important for ERGMs to allow sufficient burn-in because each proposal modifies only a single edge. All of our simulations use $2\binom{n}{2}$ burn-in steps, twice as many as needed to allow an empty graph to change to a full graph.

There are more possibilities for MCMC-based sampling, such as Gibbs sampling or Metropolis-Hastings with exotic proposal functions, but these do not lead to significant gains in speed or accuracy for ERGMs, as demonstrated by Snijders (2002).

3 IMPROVING EFFICIENCY WITH MIXTURES

Geyer's MCMCMLE has the weakness that the sampled $\{G_i^{(t)}\}$ are thrown away after each iteration in favor of a new set at the updated parameter value. This seems inefficient, given the computational expense of generating the sampled graphs through MCMC. In this section, we introduce two improvements to the MCMCMLE procedure, using *all* the draws $\{G_i^{(s)}, s \leq t\}$ from the previous iterations. Both generalize the importance sampling relationship (4) to provide an improved estimate $\hat{B}$. The first is a weighted average of t independent estimates of $\hat{B}$, and the second is an adaptation of a semiparametric method from Kong et al. (2003).

The other estimation steps in our two procedures are identical to Geyer's; the only difference is the improved $\hat{B}$ used to approximate the likelihood.

3.1 Weighted-average Importance Sampling

Here we apply the same relationship (4) to each of t sets of draws $\{G_i^{(s)}\}$, yielding t approximations $\hat{\Lambda}^{(s)}(\boldsymbol{\beta})$. Intuitively, the most reliable approximation is the one with $\boldsymbol{\beta}^{(s)}$ closest to the likelihood argument $\boldsymbol{\beta}$. In this section we form a weighted average estimator $\hat{\Lambda} = \sum w_s \hat{\Lambda}^{(s)}$ whose weights are proportional to inverse variance.

Formally, at stage t, we form the approximated likelihood (3) by writing the ratio of normalizing constants as

$$\begin{aligned} B(\boldsymbol{\beta};\boldsymbol{\beta}^{(t)}) &= \log \frac{c(\boldsymbol{\beta})/c(\boldsymbol{\beta}^{(s)})}{c(\boldsymbol{\beta}^{(t)})/c(\boldsymbol{\beta}^{(s)})} \\ &= B(\boldsymbol{\beta};\boldsymbol{\beta}^{(s)}) - B(\boldsymbol{\beta}^{(t)};\boldsymbol{\beta}^{(s)}),\ s \in \{1,\ldots,t-1\} \end{aligned}$$

Noting that this holds for each $s \in \{1,\ldots,t-1\}$, we form an estimator for $B(\boldsymbol{\beta};\boldsymbol{\beta}^{(t)})$ as the weighted average of estimators over s; namely,

$$\begin{aligned} \hat{B}_w(\boldsymbol{\beta}) &\triangleq \sum_{s=1}^{t} w_s \left(\hat{B}_s(\boldsymbol{\beta}) - \hat{B}_s(\boldsymbol{\beta}^{(t)})\right) \\ \hat{B}_s(\boldsymbol{b}) &\triangleq \hat{B}(\boldsymbol{b};\boldsymbol{\beta}^{(s)},\{G_i^{(s)}\}) \end{aligned}$$

for weights with $\sum w_s = 1$. Note that the second summand, which does not involve $\boldsymbol{\beta}$, does not affect optimization. In turn, the estimated log likelihood ratio is

$$\hat{\Lambda}(\boldsymbol{\beta}) = A(\boldsymbol{\beta}) - \hat{B}_w(\boldsymbol{\beta}),$$

which is a weighted average of log likelihood approximations.

A natural weighting is $w_s \propto \hat{V}_s^{-1}$, with $V_s = \text{Var}[\hat{B}(\boldsymbol{\beta};\boldsymbol{\beta}^{(s)})]$. The Delta Method approximates V_s from expression (4) as

$$\hat{V}_s \approx \frac{\frac{1}{m}\widehat{\text{Var}}[\exp((\boldsymbol{\beta}-\boldsymbol{\beta}^{(s)})'\boldsymbol{x}(G_i^{(s)}))]}{\left(\frac{1}{m}\sum_{i=1}^{m}\exp((\boldsymbol{\beta}-\boldsymbol{\beta}^{(s)})'\boldsymbol{x}(G_i^{(s)}))\right)^2}$$

The numerator can be estimated as the sample variance of the expression over the draws $\{G_i^{(s)}\}$. Note that as $\boldsymbol{\beta}$ approaches $\boldsymbol{\beta}^{(s)}$, the numerator approaches zero while the denominator approaches one. Hence $\hat{V}_s \to 0$, which has the desired effect of increasing w_s when $\boldsymbol{\beta}$ is close to $\boldsymbol{\beta}^{(s)}$. Our estimator leans on a particular $\boldsymbol{\beta}^{(s)}$'s draws as $\boldsymbol{\beta}$ approaches $\boldsymbol{\beta}^{(i)}$.

Here $\hat{B}_w$ is simply the best linear unbiased estimate formed from the t estimators of B, one of which is $\hat{B}$. Hence $V[\hat{B}_w] \leq V[\hat{B}]$; specifically,

$$\begin{aligned} \text{Var}[\hat{B}] &\approx \hat{V}_t \\ \text{Var}[\hat{B}_w] &\approx \left(\sum_{s=1}^{t} \hat{V}_s^{-1}\right)^{-1} \end{aligned}$$

3.2 Mixture Umbrella Distribution

The $\hat{B}$ in (4) uses $p_{\boldsymbol{\beta}}$ as the umbrella density since $\{G_i^{(t)}\} \sim p_{\boldsymbol{\beta}^{(t)}}$. In this section we improve the estimator by redefining the umbrella density as a mixture density with equal weight on each $p_{\boldsymbol{\beta}^{(s)}}, s \leq t$ as motivated by Kong et al. (2003).

At iteration t, re-index all samples to date as the collapsed set $\{\Gamma_j\} = \{G_i^{(s)}\}_{s \leq t}$, $j \in \{1, \ldots, mt\}$. The Γ_j can be viewed as draws from a mixture density with

$$p_\Gamma(G) \;=\; \frac{1}{t}\sum_{s=1}^{t} p_{\boldsymbol{\beta}^{(s)}}(G) = \frac{1}{t}\sum_{s=1}^{t} \frac{q_{\boldsymbol{\beta}^{(s)}}(G)}{c(\boldsymbol{\beta}^{(s)})} \tag{5}$$

Then the ratio of normalizing constants can be written

$$\begin{aligned} B(\boldsymbol{\beta};\boldsymbol{\beta}^{(t)}) = \frac{c(\boldsymbol{\beta})}{c(\boldsymbol{\beta}^{(t)})} &= \frac{1}{c(\boldsymbol{\beta}^{(t)})}\int q_{\boldsymbol{\beta}}(H)dH \\ &= \frac{1}{c(\boldsymbol{\beta}^{(t)})}\int \frac{q_{\boldsymbol{\beta}}(H)(p_\Gamma(H)dH)}{p_\Gamma(H)} \end{aligned}$$

A natural estimator of this integral is the weighted average over all graph draws available:

$$\hat{B}_m(\boldsymbol{\beta};\boldsymbol{\beta}^{(t)}, \{\Gamma_j\}) \;=\; \frac{1}{\hat{c}(\boldsymbol{\beta}^{(t)})mt}\sum_{j=1}^{mt} \frac{q_{\boldsymbol{\beta}}(\Gamma_j)}{\hat{p}_\Gamma(\Gamma_j)} \tag{6}$$

Note the $\hat{p}_\Gamma$ and $\hat{c}$ in the denominator; the normalizing constants are unknown and must be estimated. Kong et al. (2003) provide a semiparametric estimate of all normalizing constants as:

$$\hat{c}(\boldsymbol{\beta}^{(s)}) \;=\; \sum_{j=1}^{mt} \frac{q_{\boldsymbol{\beta}^{(s)}}(\Gamma_j)}{m\sum_{i=1}^{t}\hat{c}(\boldsymbol{\beta}^{(i)})^{-1} q_{\boldsymbol{\beta}^{(i)}}(\Gamma_j)} \tag{7}$$

Note that in terms of p_Γ these equations simplify to

$$\hat{c}(\boldsymbol{\beta}^{(i)}) \;=\; \frac{1}{mt}\sum_{j=1}^{mt} \frac{q_{\boldsymbol{\beta}^{(i)}}(\Gamma_j)}{\hat{p}_\Gamma(\Gamma_j)},$$

which takes the same form as (6). This is not surprising in light of the discussion in Kong et al. (2003), which points out that system (7) of equations can be generated by starting wth a mixture density of the generating distributions.

For comparison, consider the algorithm of Geyer and Thompson (1992) described in Section 2.2, which can be viewed as an application of importance sampling to the statistic

$q_{\boldsymbol{\beta}}(G_i)$. Importance sampling performs best when the umbrella distribution covers the most important regions of the statistic, with the proposal distribution giving a good approximation to the target distribution. Here, the statistic is the likelihood function across all $\boldsymbol{\beta}$. When the argument $\boldsymbol{\beta}$ is far from the $\boldsymbol{\beta}^{(t)}$ used to generate the graphs, $q_{\boldsymbol{\beta}}(G)$ is several orders of magnitude smaller than $q_{\boldsymbol{\beta}^{(t)}}(G)$. This makes the approximated likelihood highly variable since the umbrella distribution is a poor cover of the sampling distribution.

The problem is that the likelihood calculations are based on β varying, essentially presenting a moving target. To accommodate this, the umbrella distribution should produce G_i from a wide range of $p_{\boldsymbol{\beta}}$. A natural choice is a mixture distribution including draws from all $\{p_{\boldsymbol{\beta}^{(s)}}, s \leq t\}$ from which we have sampled graphs. This provides support for the umbrella distribution over the swaths of the sampling distribution encompassing all previously drawn graphs. Note that although the Γ_j are not actually draws from such a mixture, they can be viewed as thus for the purpose of importance sampling.

Because of the complexity of estimating $c(\boldsymbol{\beta}^{(s)})$, $s \leq t$ simultaneously, it is not easy to write an expression for the variance of this estimator. Instead, we compute variance via simulation in later sections and compare this to the other estimators.

Despite the extra complexity, this method is only modestly more computationally expensive than that of Geyer and Thompson (1992). This is because runtime under both is dominated by MCMC sampling, not the likelihood approximation. In our simulation study, we compare the two methods' runtime after a fixed number of iterations and consider whether the expense is warranted by an improvement in accuracy.

4 STARTING POINT

A weakness of all methods mentioned above is that they depend on the likelihood approximation being accurate in the region of the MLE. If the starting point $\boldsymbol{\beta}^{(0)}$ is too far away, a large number of samples is needed for a suitable approximation. Without enough samples, the algorithm can bounce around for a long time without converging. In practice nonconvergence is quite common in ERGMs since the conventional starting point, the MPLE, is often far from the MLE. Simulation results in Section 5 suggest that starting at the MPLE and running 10 iterations, MCMCMLE converges for less than 90% of 100-node graphs under the triad model.

In this section we aim to find a better starting point than the MPLE. Our approach is based on subsampling from the observed graph G:

1. Subsample a smaller graph g from G by taking a subset of its nodes and edges. We present three methods of accomplishing this.
2. Estimate the MLE $\widehat{\boldsymbol{\beta}}_{\text{SMLE}}$ of g under the same model desired for the MLE of G. Call this the SMLE.
3. Transform $\widehat{\boldsymbol{\beta}}_{\text{SMLE}}$ into an estimate of $\widehat{\boldsymbol{\beta}}_{\text{MLE}}$ by applying a scaling function S. The scaling function can be learned by simulating from the desired model.

4.1 Subsampling Methods

We consider three methods of sampling a subgraph from the observed graph G. For the sake of comparison, we define all three to subsample down to a fixed number n_g of nodes.

- **Node sampling**. Sample n_g nodes from G, including edges therein.
- **Edge sampling**. Sample edges from G one at a time until the subgraph has n_g nodes.
- **Snowball sampling**, introduced in Goodman (1961) as a technique for sampling from hard-to-reach populations. Sample a node from G. Include both the node itself, all its neighbors, and all edges therein. Continue sampling one node at a time and including all its neighbors until n_g nodes are obtained in the subgraph. To ensure exactly n_g nodes, it may be necessary at the final stage to sample from among the neighbors if including all would carry the total over n_g. (Note that there are several variants, such as including second-degree neighbors)

The former two may seem tempting at first due to a statistician's desire for simple random sampling, but closer thought (and simulations) show that snowball sampling is by far the best of the three for this purpose. Intuitively, this is because the node sampling and edge sampling *destroy the structure and connectivity of the graph*, while snowball sampling at least provides a more complete picture of the graph within a local neighborhood. Of course, there are many other possible subsampling schemes, and finding the optimal one for the above subsample down/scale up procedure remains an open problem.

4.2 Estimation and Transformation

After subsampling, we estimate the MLE $\widehat{\boldsymbol{\beta}}_{\text{SMLE}}$ under the subsampled graph g under the same model. Our goal is to use this as a starting point for the MCMCMLE procedure for G. But subsampling impacts the MLE in an unknown way; empirically, MLEs of subsampled graphs tend to be greater in magnitude than the MLE of the original graph.

To accommodate this we learn a scaling function S mapping $\widehat{\boldsymbol{\beta}}_{\text{SMLE}}$ to the $\widehat{\boldsymbol{\beta}}_{\text{MLE}}$. Pairs of known SMLEs and MLEs are needed in order to learn their relationship. We apply a simulation-based procedure to obtain the pairs.

1. Simulate graphs for which the MLE of the desired model parameters is known or can be approximated with low error.

2. Subsample from each such graph; find the SMLE.

3. Using all such pairs, fit a function S mapping the SMLE to the MLE. In our application of this method, we employ GLS to find a linear S.

4.3 Practical Use

The primary use case for the scaling function is to provide a sane alternative starting point for the MPLE that provides improved convergence behavior. However, compared to the MPLE, learning a scaling function is expensive: it requires a multiple of the runtime required to the ERGM to the original data. This is typically much longer than the MPLE, which requires only a few seconds to find even for a large network. As such, when runtime is an issue, we recommend the scaling function only as a fallback when the MPLE does not converge.

Note that the expense is mitigated for applications in which a researcher wishes to fit the same ERGM to each of several network data sets of the same number of nodes. There, one scaling function can be learned for the desired ERGM specification and node size, and then applied to generate starting points for each of the networks. Note that users who wish to fit one of the three models we consider (ET, ETK, ETKS) to a 100-node network can use our learned scaling functions directly. The accompanying R package `graphestimate` implements our scaling function as an optional fallback behavior for these three feature sets.

5 SIMULATION STUDY

We next test our methods using a simulation study. We compare three MCMCMLE methods:

- *Geyer*: Simple method of Geyer and Thompson (1992) from Section 2.2, discarding draws after every iteration and using $\hat{B}$ to approximate the likelihood
- *Imp.WA*: Our weighted-average importance sampling method from Section 3.1, making use of all iterations' draws in $\hat{B}_w$
- *Kong*: Our mixture umbrella method from Section 3.2, making use of all iterations' draws in $\hat{B}_m$

We compare two starting-point methods:

- The MPLE from Section 2.1
- Our subsample-based estimate from Section 4

To compare the MCMCMLE methods, we first simulate several thousand graphs from three different ERGM specifications using random parameter values. We then execute each MCMCMLE method several times using the MPLE as a starting point and compute the variance and MSE of the each method's parameter estimates.

To compare the starting point methods, we then repeat the Geyer and Thompson (1992) MCMCMLE methodology using our subsample-based starting point, comparing convergence results with those of the MPLE. We judge convergence by whether the likelihood ratio increments by less than 0.01 in the final iteration; while this is arbitrary, it provides a fixed yardstick for comparison.

5.1 Data

Our simulation study considers graphs with $n = 100$ nodes. We use the following procedure to generate a collection of graphs for evaluation.

1. Sample a true parameter p-vector $\boldsymbol{\beta} \sim N_p(-\mathbf{1}, I_p)$.

2. Sample a graph from $p_{\boldsymbol{\beta}}$ using a Metropolis scheme.

We generate graphs for three different parameterizations:

- **ETK**, $(\beta_{\text{edges}}, \beta_{\text{triangles}}, \beta_{\text{two-stars}})$: the triad model; 10,000 graphs.

- **ET**, with $(\beta_{\text{edges}}, \beta_{\text{triangles}})$: 1,000 graphs.

- **ETKS**, $(\beta_{\text{edges}}, \beta_{\text{triangles}}, \beta_{\text{two-stars}}, \beta_{\text{squares}})$: 1,000 graphs. β_{sqaures} corresponds to the number of squares formed by edges in the graph, a feature proposed by Snijders et al. (2004).

Just over half of the graphs drawn for the triad model are degenerate — that is, they have the minimum or maximum number of a given feature possible. For instance, both full and empty graphs are degenerate under the triad model, as are graphs with the maximum number of triangles possible with the same number of edges. In these cases, the MLE does not exist, and so degenerate graphs are excluded from the estimation study in Section 5.2.

Statistics on degenerate graphs are given in Table 2. Of the 10,000 sampled graphs for the triad model, 4,057 are nondegenerate. Degeneracy is common for significant values of the two-star parameter because there are far more two-stars possible in a 100-node graph than edges and triangles. The proportion of degeneracy is much lower for the ET model, which does not have a two-star parameter. Additionally, about 10% of nondegenerate graphs lead to nonconvergence of all three MCMCMLE methods; these are excluded from the MLE estimation comparison but are examined in detail in Section 5.3.

5.2 Estimation Results

For each sampled graph, we repeat each MCMCMLE technique $r = 10$ times, each for five iterations using an MCMC sample size of m =1,001 graphs at each iteration. From the r

Table 2: Convergence and Degeneracy Totals by Model

	ETK	**ET**	**ETKS**
Total graphs sampled	10,000	1,000	1,000
Nondegenerate graphs	4,057	914	299
At least one method converges	3,707	651	282
`Kong` converges	3,569	651	281
`Imp.WA` converges	3,517	651	280
`Geyer` converges	3,387	650	268

estimates at each iteration, we then compute two measures of accuracy:

- Sample variance of the estimates.
- Mean-squared error of the estimates, defining the truth as the MLE found using a large MCMC sample size of m_{truth} =100,001. Note that although the exact parameters used to simulate the graphs are available, we do not use them as the truth because the direct goal of the MCMCMLE methods is to estimate the MLE. A comparison with the actual simulation parameters would conflate the MLE's MSE with that of the MCMCMLE methodology. Hence we treat a well-approximated MLE as the ground truth.

Our primary metric for comparison is the median of these quantities across the sampled graphs. To avoid conflating the MLE and starting point comparisons, we use the MPLE as the starting point for results in this subsection.

The results for each parameter in the triad model are given in Figure 1. Each panel shows the variance and MSE of parameter estimates in the number of iterations. For simplicity, the results are reported as a fraction of the variance seen in the first iteration (Note that all three methods produce identical estimates after the first iteration). For later iterations, our methods exhibit decaying variance in the number of iterations because they make use of increasingly many draws, while Geyer's plateaus because it uses only the fresh m at each iteration.

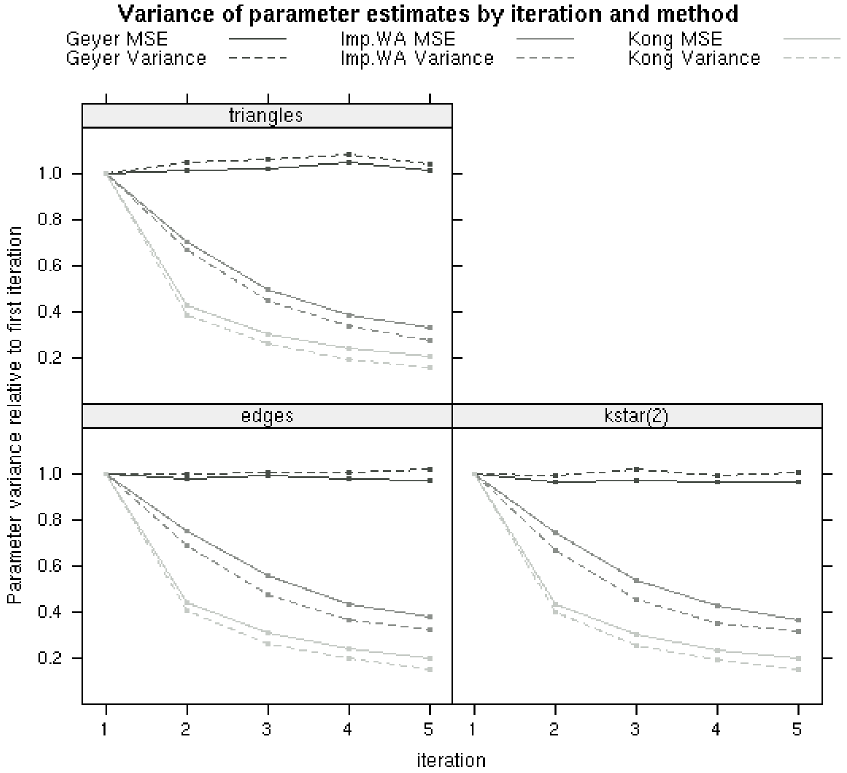

Figure 1: Median sample variance ratios and MSE ratios of parameter estimates plotted over optimization iterations for each method. The ratios are given as a fraction of variance and MSE after the first iteration, at which all methods are the same.

The Kong-based mixture method performs best by an outstanding margin. Its MLEs are nearly five times as efficient as Geyer's by the fifth iteration and about twice as efficient as the weighted-averaging method. The trend holds for all parameters. It also holds for both variance and MSE, suggesting that it is not a result of bias. Similar trends also hold for the ET and ETKS models, whose results are shown in Figures 2 and 3.

Computationally, the Kong-based method more than takes about 50% more time than the Geyer-based method after five iterations. In our simulations using a 64-bit 2GHz Intel Core 2 Duo, the Kong-based mixture method took an average 2.1 s for five iterations, versus 1.4 s for Geyer's. We observed the same pattern on a 64-bit dual-core AMD Turion, though both took about 60% longer. Consequently, the five-fold variance improvement under Kong's method translates to a still-sizable three-fold gain in computational efficiency (defined as the ratio of runtime to inverse estimation variance).

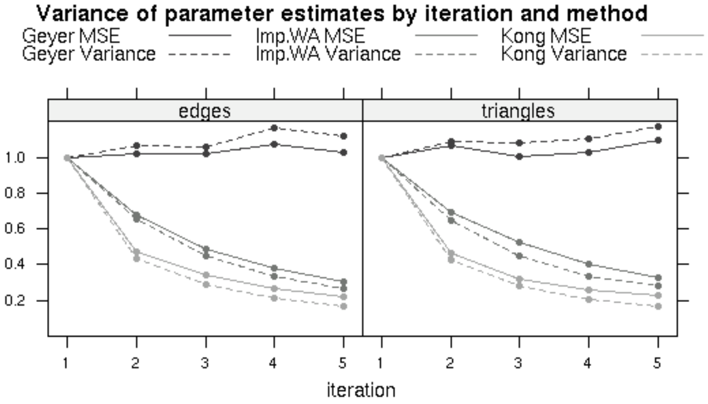

Figure 2: Variance profile of parameter estimates for the two-parameter ET model.

Table 3 shows the elementwise average of the sample correlation matrices for the Geyer-based method's fifth-iteration MLE estimates. All three methods produced similar estimation correlation structures. Of note is the high negative correlation between the edge and two-stars parameters. This is because both capture the same higher-level effect of the density of edges in the graph. These are the hardest two to estimate.

Table 3: Average Correlation Matrix of Geyer-based MCMLE

	$\hat{\beta}_{\text{edges}}$	$\hat{\beta}_{\text{triangles}}$	$\hat{\beta}_{\text{two-stars}}$
$\hat{\beta}_{\text{edges}}$	1.00	0.08	-0.96
$\hat{\beta}_{\text{triangles}}$	0.08	1.00	-0.19
$\hat{\beta}_{\text{two-stars}}$	-0.96	-0.19	1.00

5.3 Starting Point Results

For about 10% of the simulated graphs, none of the three methods converge. One reason for this is that the starting point is too far from MLE, leading to a poor likelihood approximation. Here we compare convergence under the two types of starting points: the conventional MPLE starting point; and our subsample-based starting point.

Convergence statistics for the MPLE starting point were given in Table 2, as recorded in the experiment of Section 5.2. Of nondegenerate graphs, about 90% lead to convergence in the triad model. The convergence rate is similar across all three MLE estimation methods.

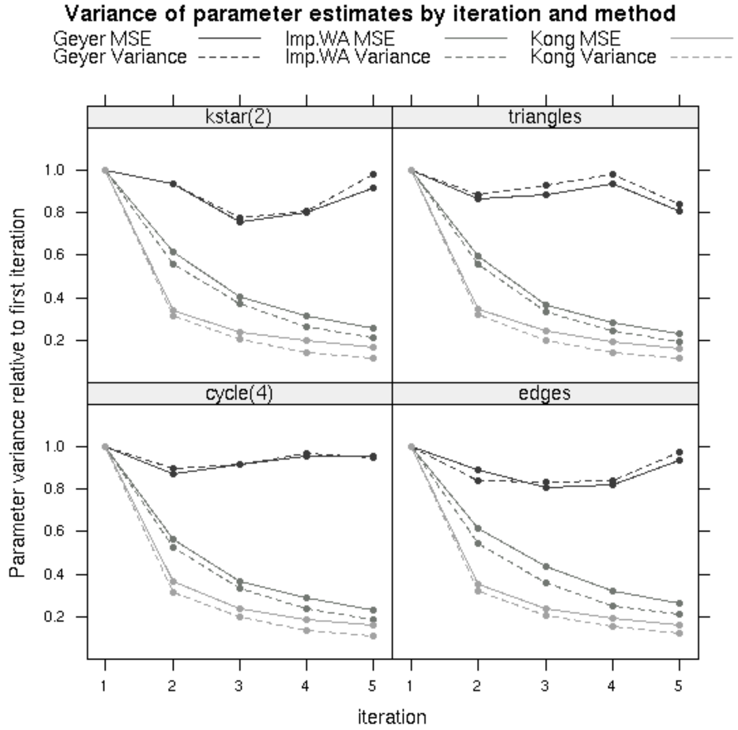

Figure 3: Variance profile of parameter estimates for the four-parameter ETKS model.

Our results in this section are broken into two parts: first, in Section 5.3.1, the scaling function learned to relate the SMLE to the MLE; next, in Section 5.3.2, the convergence results under the subsampling-based starting point. For this comparison we focus on convergence of Geyer's MCMCMLE for the 4,057 nondegenerate triad-model graphs. We also use snowball rather than node or edge sampling, because the latter two commonly produce degenerate subgraphs. We use subgraphs of $n_g = 50$ nodes out of $n = 100$ in the full graph.

5.3.1 Scaling Function

Before we can find subsample-based starting points, we first must perform a separate simulation experiment to learn a scaling function that relates the SMLE to the estimated MLE. This experiment requires a set of graphs whose MLEs are estimated with high confidence; to avoid in-sample error, we select a new set of graphs using a methodology identical to Section 5.1. This produces d =3,765 graphs. Next we snowball-sample 50 nodes from each such graph and find the MCMLE of the subgraph, yielding the SMLE. This procedure takes a total of about 3 hours on our 2GHz Core 2 Duo laptop.

Figure 4 shows paired scatterplots comparing $\hat{\beta}_{\text{MLE}}$ to $\hat{\beta}_{\text{SMLE}}$. Estimates of both $\beta_{\text{triangles}}$ and $\beta_{\text{two-stars}}$ under subsampling correlate near linearly to their whole-graph analogs. The cross-plots of different parameters mirror the correlation matrix in Table 3.

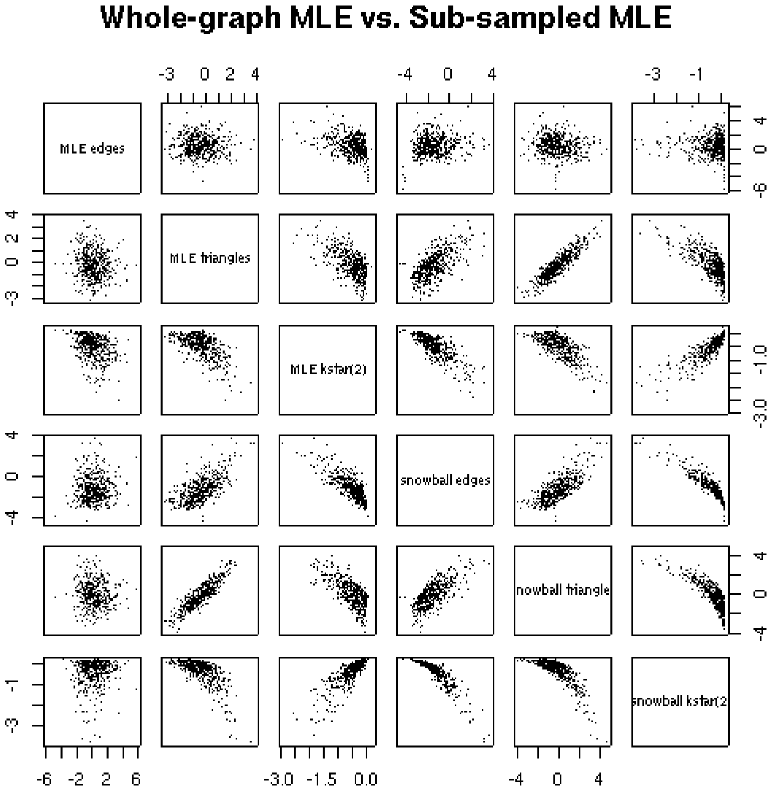

Figure 4: Scatterplots comparing MLE parameters of the entire graph to MLE parameters of sub-sampled graphs.

With most correlations linear, a linear scaling function seems reasonable. We fit $\widehat{\boldsymbol{\theta}}$ in a feasible GLS (FGLS) regression

$$\begin{aligned} \widehat{\boldsymbol{\beta}}_{\text{MLE}} &= \boldsymbol{\theta}'\widehat{\boldsymbol{\beta}}_{\text{SMLE}} + \boldsymbol{\varepsilon}, \\ \text{Cov}(\boldsymbol{\varepsilon}) &= \begin{pmatrix} \widehat{\text{Cov}}(\widehat{\boldsymbol{\beta}}^{(1)}_{\text{MLE}}) & & 0 \\ & \ddots & \\ 0 & & \widehat{\text{Cov}}(\widehat{\boldsymbol{\beta}}^{(d)}_{\text{MLE}}) \end{pmatrix} \end{aligned}$$

The covariance for each set of $\hat{\beta}_{\text{MLE}}$ is taken as the sample covariance matrix of the MLE estimates over that graph's $r = 10$ repetitions. Table 4 shows our fitted scaling function along with a marginal R^2 for each parameter. Having coefficients less than one, the triangle and edge parameter models tend to be lower for the whole graph than for the subsample. The comparatively poor R^2 for the $\hat{\beta}_{\text{MLE,edges}}$ model confirms what is visible in the first row of Figure 4, that the edge parameter has the poorest fit.

Table 4: FGLS-fitted Snowball Sampling Scaling Function

	$\widehat{\boldsymbol{\theta}}_{\text{edges}}$	$\widehat{\boldsymbol{\theta}}_{\text{triangles}}$	$\widehat{\boldsymbol{\theta}}_{\text{two-stars}}$
$\boldsymbol{\beta}_0$ (intercept)	1.59 (0.17)	-0.27 (0.07)	-0.37 (0.03)
$\hat{\boldsymbol{\beta}}_{\text{SMLE}}$, edges	0.67 (0.07)	-0.05 (0.03)	-0.04 (0.01)
$\hat{\boldsymbol{\beta}}_{\text{SMLE}}$, triangles	-0.15 (0.05)	0.81 (0.02)	-0.03 (0.01)
$\hat{\boldsymbol{\beta}}_{\text{SMLE}}$, two-stars	0.94 (0.17)	0.10 (0.06)	0.48 (0.03)
Marginal R^2	**0.07**	**0.81**	**0.74**

5.3.2 Convergence

With the scaling function learned, we next return to the graphs from the MLE experiment of Section 5.2; we re-run Geyer starting at $\hat{S}(\widehat{\boldsymbol{\beta}}_{\text{SMLE}})$ instead of $\widehat{\boldsymbol{\beta}}_{\text{MPLE}}$. Table 5 shows a contingency table comparing the two starting points' convergence results. We can see that the scaling function is not universally better, but does result in convergence in 133 of the 670 graphs for which the MPLE fails. This suggests that the subsampling technique may be most useful as a fallback when the MPLE fails. Failure can be detected using degeneracy detection methods such as those given in Handcock (2003) (e.g., extreme values of the likelihood ratio, degenerate sufficient statistics in the sampled graphs).

6 APPLICATION

Finally, we return to our MCMCMLE methods, testing them on triad models for two well-known network data sets on which convergence could be reached.

- `florentine`, a nondirectional set of 20 marriage ties among 16 Renaissance Florentine

Table 5: Convergence (C) and Nonconvergence (NC) Contingency Table

		$\hat{S}(\widehat{\boldsymbol{\beta}}_{\text{SMLE}})$		
		C	NC	
$\hat{\boldsymbol{\beta}}_{\text{MPLE}}$	C	3,101	286	**3,387**
	NC	133	537	**670**
		3,234	**823**	**4,057**

families from Padgett (1994) and part of the statnet R package of Handcock et al. (2003).

- molecule, a nondirectional data set with 20 nodes and 28 edges meant to resemble a synthetic molecule, used within statnet.

To test our methods, we estimate $r = 100$ times for each data set using all three methods for five iterations with m =10,000 samples at each iteration as recommended by Handcock et al. (2003).

Figure 5 plots the estimates of the MLE under each MCMCMLE method, and standard errors are given in Table 6. In all cases, the Kong mixture-based estimates are clearly less variable than the others, while all have the same mean up to two significant digits. This confirms the simulation results of Section 5.2. The standard errors show that mixture-based estimates have about a five-fold efficiency gain over Geyer-based estimates, which is comparable to the simulation results for the triad model. Weighted-averaging again falls in between.

7 CONCLUSION

A mixture umbrella distribution, with normalizing constants estimated as in Kong et al. (2003), greatly refines approximated likelihoods. Using this method to incorporate draws from previous iterations yields a markedly more accurate MCMCMLE for a wide range of ERGMs. The reduction in variance is roughly linear in the number of iterations. By five iterations of our refined procedure, the MLE estimate is about 1/5 as variable as under the

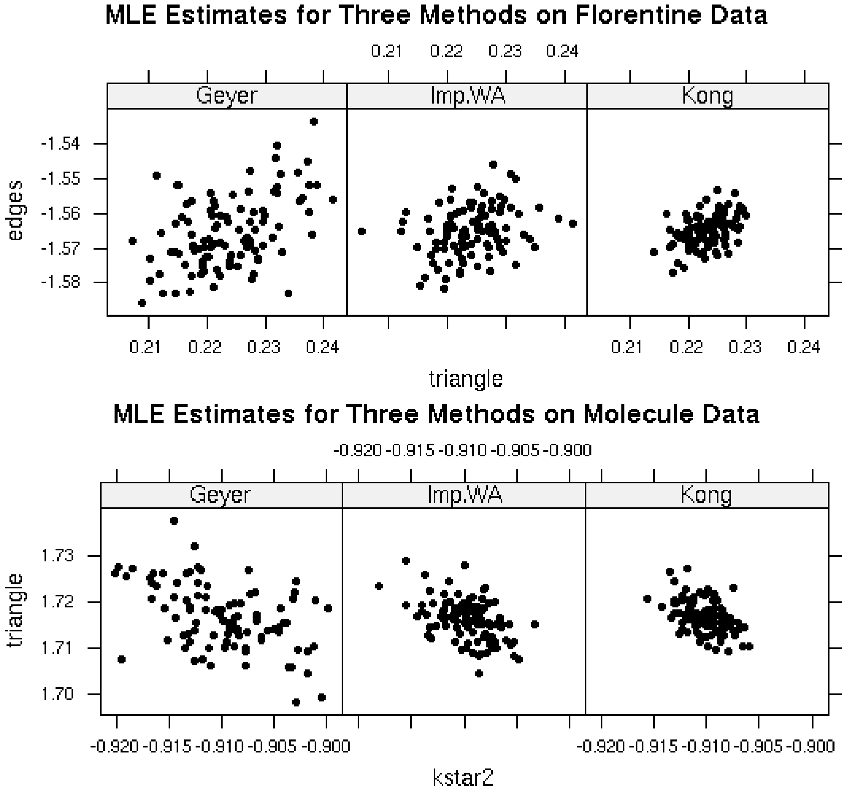

Figure 5: Scatterplots of MCMCMLEs for two network data sets.

approach proposed by Geyer and Thompson (1992).

Subsampling with snowball sampling as a means of finding starting points for MCMCMLE is a promising alternative to the MPLE; it is common for one to yield convergence but not the other. Further development of improved subsample estimates is warranted, either in terms of the scaling function used or the subsampling scheme). Meanwhile, it is helpful to have both the MPLE and the subsampling starting point as standard choices of initial value, to improve the chance of obtaining convergence.

References

J. Corander, K. Dahmstrom, and P. Dahmstrom. Maximum likelihood estimation for markov graphs. *Research Report, Department of Statistics, University of Stockholm*, 8, 1998.

Ove Frank and David Strauss. Markov graphs. *Journal of the American Statistical Associ-*

Table 6: Sample MLE Mean and Variance

		florentine			molecule		
		E	T	2S	E	T	2S
Mean	Geyer	-1.57	.22	-.029	1.90	1.72	-.91
	Imp.WA	-1.57	.22	-.029	1.90	1.72	-.91
	Kong	-1.57	.22	-.029	1.90	1.72	-.91
Var.	G/K	5.2	5.3	5.4	6.1	3.6	6.1
	IWA/K	2.4	3.0	2.3	1.9	1.5	1.8

ation, 81(395):832–842, sep 1986. ISSN 0162-1459.

Charles J. Geyer and Elizabeth A. Thompson. Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 54 (3):657–699, 1992. ISSN 0035-9246.

Leo A. Goodman. Snowball sampling. *The Annals of Mathematical Statistics*, 32(1):148–170, mar 1961. ISSN 0003-4851.

M. Handcock. Statistical models for social networks: Inference and degeneracy. *Dynamic Social Network Modeling and Analysis*, page 22940, 2003.

Mark S. Handcock, David R. Hunter, Carter T. Butts, Steven M. Goodreau, and Martina Morris. statnet: An r package for the statistical modeling of social networks, 2003. URL `http://www.csde.washington.edu/statnet`.

A. Kong, P. McCullagh, X.L. Meng, D. Nicolae, and Z. Tan. A theory of statistical models for monte carlo integrations (with discussion). *The Journal of Royal Statistical Society*, 2003.

Jun S. Liu. *Monte Carlo Strategies in Scientific Computing.* Springer, 2nd edition, 2002. ISBN 0387952306.

John F. Padgett. Marriage and elite structure in renaissance florence. *Paper delivered to the Social Science History Association*, pages 1282–1500, 1994.

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): 192–215, 2007.

T. Snijders, P. Pattison, and M. Handcock. New specifications for exponential random graph models. *Working Paper no. 42, Center for Statistics and Social Sciences, University of Washington*, 2004.

Tom A.B. Snijders. Markov chain monte carlo estimation of exponential random graph models. *Research Report, Department of Statistics and Measurement Theory, University of Groningen*, 2002.

David Strauss and Michael Ikeda. Pseudolikelihood estimation for social networks. *Journal of the American Statistical Association*, 85(409):204–212, mar 1990. ISSN 0162-1459.

W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.