Sequential Importance Sampling for Bipartite Graphs With Applications to Likelihood-Based Inference [1]

Ryan Admiraal
University of Washington, Seattle

Mark S. Handcock
University of Washington, Seattle

Technical Report No. 502
Department of Statistics
University of Washington

August 2006

[1]Ryan Admiraal is a graduate student, Department of Statistics, University of Washington, Box 354322, Seattle WA 98195-4332. E-mail: ryan@stat.washington.edu; Web: http://www.stat.washington.edu/ryan. Mark S. Handcock is Professor of Statistics and Sociology, Department of Statistics, University of Washington, Box 354322, Seattle WA 98195-4322. E-mail: handcock@stat.washington.edu; Web: http://www.stat.washington.edu/handcock.

Abstract

The ability to simulate graphs with given properties is important for the analysis of social networks. Sequential importance sampling has been shown to be particularly effective in estimating the number of graphs adhering to fixed marginals and in estimating the null distribution of test statistics. This paper builds on the work of Chen et al. (2005), providing an intuitive explanation of the sequential importance sampling algorithm as well as several examples to illustrate how the algorithm can be implemented for bipartite graphs. We examine the performance of sequential importance sampling for likelihood-based inference in comparison with Markov chain Monte Carlo, and find little empirical evidence to suggest that sequential importance sampling outperforms Markov chain Monte Carlo, even for sparse graphs or graphs with skewed marginals.

KEY WORDS: Sequential importance sampling; bipartite graph; Markov chain Monte Carlo; likelihood inference; graph counting;

1 Introduction

A bipartite graph is a graph for nodes of two distinct types with the relation defined to be between nodes of different types. The set of nodes can be represented by two subsets $\mathcal{R}$ and $\mathcal{C}$ for which nodes in $\mathcal{R}$ only have ties to nodes in $\mathcal{C}$, and nodes in $\mathcal{C}$ only have ties to nodes in $\mathcal{R}$. One of the more common types of bipartite graphs is an affiliation network for which the two types can be called "actor" and "event" and the relation indicates the affiliation of the actor with the given event. As an example of an affiliation network, we will later examine the occurrence of finch species on different islands. Bipartite graphs can be represented by a matrix $A = a_{ij}$, where $i \in \mathcal{R}$, $j \in \mathcal{C}$, and $a_{ij} = 1$ if there is a relational tie from i to j and 0 otherwise. In the case of an affiliation matrix, $a_{ij} = 1$ would correspond to actor i being affiliated with event j.

In the analysis of bipartite graphs, most research has focused on analytic methods of analyzing properties of the graph or calculating graph statistics. Where analytic methods are impractical, research has generally turned to approximation methods. The approximation method most commonly utilized is Markov chain Monte Carlo (MCMC), which often provides a means to representatively sample the graph space. The standard MCMC algorithm is that developed by Snijders (1991), and extended by Rao et al. (1996). While useful in solving many difficult problems, MCMC has its limitations, and alternative Monte Carlo methods are of interest. Recent research has brought to light the effectiveness of sequential importance sampling (SIS) in solving certain problems for which analytic methods and current MCMC algorithms do not provide good solutions or any solution at all.

SIS has proved particularly useful in counting bipartite graphs with given marginals. For many graphs, exact enumeration of all graphs adhering to marginal constraints simply is not practical. To illustrate the issues, consider the bipartite graph in Table 1, which consists of only 13 rows, 17 columns, and 122 ties. For this graph, a graph of moderate size, the total number of unique graphs matching the row and column sums is more than 6.7×10^{16}. While several elaborate algorithms can perform such tasks of exact graph counting, the amount of time required to directly compute the number of graphs meeting the marginal constraints generally makes such a computation impractical. Attempts to approximate the number of graphs meeting marginal constraints by means of MCMC are more practical in terms of computing time, but, for graphs similar in size to Table 1, they still require an exorbitant run time for a low degree of accuracy. SIS, on the other hand, requires relatively few sampled graphs and minimal computing time to produce a highly accurate estimate

of the number of graphs meeting the marginal constraints. It does this by sampling columns of the graph sequentially, ensuring that new graphs meet the column and row sum constraints of the observed graph. In each column, sampling of rows to receive 1's is done according to specified inclusion probabilities to ensure that the new graph comes from a distribution that is relative uniform (and known).

	Island															
Finch	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
Large ground finch	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
Medium ground finch	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	0	0
Small ground finch	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0
Sharp-beaked ground finch	0	0	1	1	1	0	0	1	0	1	0	1	1	0	1	1	1
Cactus ground finch	1	1	1	0	1	1	1	1	1	1	0	1	0	1	1	0	0
Large cactus ground finch	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
Large tree finch	0	0	1	1	1	1	1	1	1	0	0	1	0	1	1	0	0
Medium tree finch	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Small tree finch	0	0	1	1	1	1	1	1	1	1	0	1	0	0	1	0	0
Vegetarian finch	0	0	1	1	1	1	1	1	1	1	0	1	0	1	1	0	0
Woodpecker finch	0	0	1	1	1	0	1	1	0	1	0	0	0	0	0	0	0
Mangrove finch	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Warbler finch	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 1: Darwin's finch data

Another application for which SIS has proved useful is in approximating the null distribution of a test statistic. Such capabilities are important in significance testing. For the null distribution of most test statistics for a graph of moderate size, there are no known analytic approaches that can accurately approximate the distribution. This is primarily because such algorithms must be modified for the test statistic under consideration, leading to even more complex algorithms than those used in the graph counting problem. MCMC also appears to be an inferior approach, as Chen et al. (2005) (CDHL) provide numerous examples in which MCMC algorithms are consistently less efficient than SIS. CDHL suggests that this is to be expected, as the problems of counting graphs and approximating the null distribution of a test statistic are both easily solved when sampling graphs uniformly or nearly uniformly, and this is exactly what SIS attempts to do. However, the validity of CDHL's assessment is an empirical question and will depend on the specifics of the problem addressed.

With the apparent advantages of SIS over analytic methods and MCMC when considering the problems of approximating the size of a graph space or the null distribution of a test statistic, it seems plausible that SIS may also have a distinct advantage in handling difficult likelihood inference problems. To the best of our knowledge, this application of SIS has not been explored. In cases where direct computation of the likelihood is impractical, research has focused on alternative estimators or approximations. One such alternative is pseudo-likelihood estimation,

which constructs a surrogate for the likelihood based on the product of the full conditional distribution for each edge. Generally, this can be computed exactly. However, Robins et al. (2006) argue that this method is intrinsically highly dependent on the observed graph and, consequently, may result in substantial bias in the parameter estimates for certain graphs. In addition, the naïve standard errors are often too small. van Duijn et al. (2006) show that the maximum pseudo-likelihood estimate (MPLE) performs substantially worse than the MLE in terms of efficiency and bias. They show that the estimates of the standard errors of the MPLE can be both higher and lower than their actual values depending on the observed graph. Because of these potential problems, we generally avoid pseudo-likelihood and consider other approximations, such as MCMC. Unfortunately, in the case of MCMC there can be strong dependence between graphs that are sampled successively, meaning that a larger number of graphs must be sampled to obtain a desired effective sample size. More problematic for MCMC is the possibility that certain regions of the graph space will not be sampled or that the chain will remain in certain regions of the space for extended periods of time, leading to incorrect inference (Gelman, 1996).

Although generally slower than MCMC in sampling valid graphs, SIS has the advantage of sampling new graphs independently and with the goal of an approximately uniform probability, meaning that a much smaller sample of graphs is required to obtain a specified effective sample size. Because it samples new graphs independently, it is able to quickly sample from different regions of the space. This means that it should be able to better avoid the problems that MCMC encounters with failing to sample certain regions or occasionally remaining in certain regions of the graph space for a large number of iterations. This leads us to believe that SIS may be comparable to MCMC in maximum likelihood estimation and may actually produce more accurate parameter estimates in certain instances.

In this paper we give a detailed exposition of the computational aspects of SIS, following the structure and notation of Chen et al. (2005). After describing the components of the algorithm in Section 2, we provide a simple example of how to implement SIS in Section 3, guiding the reader through each of the steps. In Section 4 we show how SIS can be used in maximum likelihood estimation. In Section 5 we discuss the results of our algorithm as they pertain to graph counting, approximating the null distribution of a test statistic, and likelihood inference. Finally, in Section 6 we discuss the results in relation to other methods, paying particular attention to the likelihood inference results.

2 Sequential Importance Sampling for Bipartite Graphs

Let $\boldsymbol{r} = (r_1, r_2, ..., r_m)$ denote the row sums and $\boldsymbol{c} = (c_1, c_2, ..., c_n)$ the column sums of an $m \times n$ bipartite graph. Then we will denote the space of all graphs with column sums $\boldsymbol{c}$ by $\mathcal{A}_{\boldsymbol{c}}$, and we will denote the space of all graphs with row sums $\boldsymbol{r}$ and column sums $\boldsymbol{c}$ by $\mathcal{A}_{\boldsymbol{rc}}$. For most significance tests and for likelihood inference, we must generate new graphs that have the same marginals as the observed graph. Consequently, our focus will be on generating graphs in $\mathcal{A}_{\boldsymbol{rc}}$.

For the graph in Table 2, $\mathcal{A}_{\boldsymbol{c}}$ is the space of all graphs with column sums $(3, 3, 3, 1)$, and $\mathcal{A}_{\boldsymbol{rc}}$ is the space of all graphs with row sums $(3, 3, 2, 2)$ and column sums $(3, 3, 3, 1)$. Suppose we want to generate new graphs with the same row and column sums as this graph. If we simply ensure that we sample according to the column sum constraints and the algorithm samples $(0\ 1\ 1\ 1)^T$ for the first column, sampling $(0\ 1\ 1\ 1)^T$ for the second column could produce a valid graph for $\mathcal{A}_{\boldsymbol{c}}$ but not for $\mathcal{A}_{\boldsymbol{rc}}$. This should be clear in that only two columns remain to be sampled, and yet we still need to sample three 1's for the first row in order to obtain a valid graph in $\mathcal{A}_{\boldsymbol{rc}}$. Since our goal is to sample new graphs in $\mathcal{A}_{\boldsymbol{rc}}$, we will need to address this problem.

1	1	1	0	3
1	1	0	1	3
1	0	1	0	2
0	1	1	0	2
3	3	3	1	

Table 2: First Example

2.1 Conjugate Sequences

One possible remedy for this problem is to generate graphs in $\mathcal{A}_{\boldsymbol{c}}$ and simply discard those graphs that are not in $\mathcal{A}_{\boldsymbol{rc}}$. This is highly inefficient, however, so we consider a different approach and implement a forward-looking step to ensure that all sampled graphs are valid. To do this, we define the conjugate sequence of column sums $c_1, c_2, ..., c_n$ by $C_i^{(0)} = \#\{c_j : c_j \geq i\}$, $i = 1, 2, ..., m$, $j = 1, 2, ..., n$. Thus, for any given column sum c_j, we will increment $C_i^{(0)}$ by 1 for each j satisfying $1 \leq C_i^{(0)} \leq c_j$. This means that $C_1^{(0)}$ counts the number of non-zero column sums, $C_2^{(0)}$ counts the number of column sums that are at least two, $C_3^{(0)}$ counts the number of column sums that are at least three, and so on.

In general, we will let $\boldsymbol{C}^{(j)} = \left(C_1^{(j)}, ..., C_m^{(j)}\right)$ denote the conjugate sequence of $c_{j+1}, ..., c_n$. This represents the conjugate sequence after the first j columns have been sampled. For example, for the graph in Table 2, $\boldsymbol{C}^{(0)} = (4, 3, 3, 0)$, $\boldsymbol{C}^{(1)} = (3, 2, 2, 0)$, $\boldsymbol{C}^{(2)} = (2, 1, 1, 0)$, $\boldsymbol{C}^{(3)} = (1, 0, 0, 0)$, and $\boldsymbol{C}^{(4)} = (0, 0, 0, 0)$. By construction, $\boldsymbol{C}^{(0)}$, $\boldsymbol{C}^{(1)}$, ..., $\boldsymbol{C}^{(n)}$, are independent of the sampling mechanism for the columns, so all conjugate sequences can be computed prior to sampling any of the n columns. Also by construction, $C_i^{(0)} \geq C_i^{(1)} \geq \cdots \geq C_i^{(n)}$ for all i. The algorithm we implemented for computing the conjugate sequences can be found in Algorithm 1 of the appendix.

2.2 Knots and Corresponding Restrictions

For a given conjugate sequence, we can determine the maximum number of ones that can be sampled in a specified set of columns for a given number of rows by computing partial sums from the conjugate sequence. In particular, $C_1^{(j)}$ gives the maximum number of 1's that can be sampled in columns $j+1$ to n for any one row, $\sum_{i=1}^{2} C_i^{(j)}$ gives the maximum number of 1's that can be sampled in columns $j+1$ to n for any two rows, and $\sum_{i=1}^{t} C_i^{(j)}$ gives the maximum number of 1's that can be sampled in columns $j+1$ to n for any t rows. Thus, for the graph in Table 2, $\boldsymbol{C}^{(1)} = (3, 2, 2, 0)$ tells us that, in the last three columns, we can sample no more than three 1's for any given row, no more than five 1's for any two rows, no more than seven 1's for any three rows, and no more than seven 1's for any four rows.

Likewise, the row sums tell us how many 1's must be sampled for a specific set of rows. Since rows with larger row sums will have greater restrictions in the sampling process, we rearrange the rows so that row sums are ordered from largest to smallest. Then $\sum_{i=1}^{t} r_i$ tells us the total number of 1's that must be sampled for the first t rows. Note that $\sum_{i=1}^{m} C_i^{(0)} = \sum_{j=1}^{n} c_j = \sum_{i=1}^{m} r_i$. Then, since $C_i^{(0)} \geq C_i^{(1)}$ for all i, we have $\sum_{i=1}^{m} r_i \geq \sum_{i=1}^{m} C_i^{(1)}$. In particular, if $\sum_{i=1}^{t} r_i > \sum_{i=1}^{t} C_i^{(1)}$, then at least $\sum_{i=1}^{t} r_i - C_i^{(1)}$ ones must be sampled in rows 1 to t of the first column. Failure to do so will result in a graph that fails to meet our marginal constraints, as the first t rows of the graph will require more ones than afforded by the remaining column sums of size t or less.

2.2.1 Motivation

To illustrate this, suppose $C_1^{(1)} = 5$, $C_2^{(1)} = 4$, and $C_3^{(1)} = 2$; and suppose $r_1 = 4$, $r_2 = 4$, and $r_3 = 4$. Then $\sum_{i=1}^{3} r_i > \sum_{i=1}^{3} C_i^{(1)}$, and $\sum_{i=1}^{3} r_i - C_i^{(1)} = 1$. Let x denote instances of a 1 being sampled for the first row, y denote instances of a 1 being sampled for the second row, and z denote instances of a 1 being sampled for

the third row. If we fail to sample a 1 in the first column for any of these three rows, satisfying the row sum requirements for the first row requires that we sample 1's for four of the five remaining columns with non-zero column sums (i.e. $c_2, c_3, ..., c_m$), leaving unsampled only one of the columns with a non-zero column sum.

x	x		x	x

Satisfying the row sum requirements for the second row requires that we sample four 1's from either the column with a non-zero column sum that has yet to be sampled or the columns with column sums of at least two which have already been sampled once. This results in all columns with non-zero column sums being sampled at least once and all but one of the columns with column sums of at least two being sampled twice.

x	x	y	x	x
y	y			y

Finally, satisfying the row sum requirements for the third row requires that we sample four 1's for either the column with a column sum of at least two which has been sampled only once or the columns with column sums of at least three which have already been sampled twice. Here, we encounter a problem, as there are not enough column sums of at least three to accommodate the number of 1's required by this third row sum. Of course, we cannot sample a column twice for the same row either, so it becomes clear that the only solution is to sample a 1 for at least one of the first three rows in the first column.

x	x	y	x	x	
y	y	z		y	
	z			z	z

2.2.2 Determination of Knots and Corresponding Restrictions

To determine how many 1's must be sampled in certain rows for a given column, we record the row number t whenever $\sum_{i=1}^{t} r_i > \sum_{i=1}^{t} C_i^{(1)}$, and we also record the

corresponding difference $\sum_{i=1}^{t} r_i - C_i^{(1)}$. Let $k_1, k_2, ...$ (which we will refer to as *knots*) take on the values of t for instances where $\sum_{i=1}^{t} r_i > \sum_{i=1}^{t} C_i^{(1)}$, and let $v_1, v_2, ...$ take on the corresponding differences $\sum_{i=1}^{t} r_i - C_i^{(1)}$. Thus, v_i tells us how many ones must be sampled by row k_i. If $v_j \leq v_i$ for some $j > i$, we will remove k_j and v_j, as the restrictions placed on our sampling through k_i and v_i ensure that the restrictions placed on our sampling through k_j and v_j are met. Likewise, if $v_j - v_i \geq k_j - k_i$ for any $j > i$, then we will remove k_i and v_i, as the restrictions placed on our sampling through k_j and v_j ensure that the restrictions placed on our sampling through k_i and v_i are met.

For the first column of the graph given in Table 2, we initially record the knots and corresponding restrictions

$$\begin{aligned} k_1 &= 2, \quad v_1 = 1 \\ k_2 &= 3, \quad v_2 = 1 \\ k_3 &= 4, \quad v_3 = 3. \end{aligned}$$

However, k_1 and v_1 ensure that the restrictions stipulated by k_2 and v_2 are met, so we remove k_2 and v_2 and reorder all subsequent knots to obtain

$$\begin{aligned} k_1 &= 2, \quad v_1 = 1 \\ k_2 &= 4, \quad v_2 = 3. \end{aligned}$$

The algorithm we implemented for computing the knots and corresponding values for a given column can be found in Algorithm 3 of the appendix.

2.3 Sampling a Column

After we have recorded the knots $\boldsymbol{k} = (k_1, k_2, ...)$ and corresponding restrictions $\boldsymbol{v} = (v_1, v_2, ...)$ for a column, we can begin sampling for that column. Before providing a general rule for sampling a column, we will first demonstrate how sampling is executed for a simple example.

2.3.1 Example

To sample the first column for the graph in Table 2, we record the knots and corresponding restrictions for the first column. These are given by

$$\begin{aligned} k_1 &= 2, \quad v_1 = 1 \\ k_2 &= 4, \quad v_2 = 3. \end{aligned}$$

With the knots and corresponding restrictions recorded, our first step is to sample 1's for rows 1 to $k_1 = 2$. Because $v_1 = 1$, we know that we must sample at least one

of these rows to receive a 1. At the same time, we cannot sample more 1's than what the column sum allows, so we may not sample more than four rows to receive a 1 from the first two rows. Clearly, it is not possible to sample more 1's than number of rows, so we may not sample more than two of the first two rows to receive a 1. Thus, we know that we must sample at least one but no more than two of the first two rows to receive a 1, and we randomly select one of the two possibilities. Suppose we randomly select one, so only one of the first two rows will receive a 1. We then randomly choose one of the two rows based on some probability proportional to the row sum. We will assume that the first row was chosen to receive the 1, so our current sampling scheme for the first column is

1				3
0				3
				2
				2
3	3	3	1	

The second knot k_2 and its corresponding restriction v_2 tell us that we must sample at least three rows from the first four rows to receive a 1. We have already sampled one row to receive a 1 from the first two rows, so we must sample two rows from the third and fourth rows to receive a 1. Once again, we may not sample more 1's than what the number of rows permits, so we may not sample more than two rows to receive a 1. Consequently, we must sample both the third and fourth rows to receive 1's, so our sampling scheme for the first column will be

1				3
0				3
1				2
1				2
3	3	3	1	

2.3.2 General Column Sampling Procedure

In general, then, to sample column j we first determine d_1, the total number of 1's to be sampled for the first k_1 rows. As stated before, v_1 denotes the number of ones that must be sampled by row k_1, so this is our lower bound for d_1. At the same time, we cannot sample more 1's than rows, nor can we sample more 1's than what the column sum allows, so we are bounded above by the minimum of k_1 and c_j. Thus, we sample a value d_1 uniformly from $\{v_1, ..., \min\{k_1, c_j\}\}$. Once we have sampled a value for d_1, we sample the rows that are to receive a 1. The procedure for sampling

the d_1 rows to receive a 1 is described later. Next, we uniformly sample a value d_2 from $\{\max\{v_2 - d_1, 0\}, \min\{k_2 - k_1, c_j - d_1\}\}$ to determine the number of 1's to be sampled for rows $k_1 + 1$ to k_2, and the row sampling procedure is repeated. This continues until we have either sampled c_j rows to receive ones or have reached our last knot.

Each time we sample a value d_i for a knot k_i, we compute inclusion probabilities for rows $k_{i-1} + 1$ to k_i, and then we randomly sample one of these rows to receive a 1 according to the inclusion probabilities. The inclusion probabilities are then updated for the unsampled rows, and we again randomly select one of these rows to receive a 1 according to the new inclusion probabilities. The procedure is repeated until we have sampled d_i rows. The algorithm we implemented for sampling rows to receive a 1 for a given column can be found in Algorithm 4 of the appendix.

2.4 Updating

After we have sampled the first column, we decrement by one the row sums for each row that was sampled and record these new row sums $\boldsymbol{r}^{(1)}$. Then we consider the conjugate sequence $\boldsymbol{C}^{(2)}$ and the new row sums and repeat the procedure of ordering row sums in decreasing order, determining knots, recording restrictions corresponding to each knot, and sampling according to these restrictions. In essence, we are sampling the first column for a new $m \times n - 1$ graph with row sums equal to $\boldsymbol{r}^{(1)}$.

2.5 Graph Probability

As sampling is occurring, we update the probability of the new graph. Since sampling is not uniform, it is vital that we know the probability of generating the new graph that we observe. Columns are sampled sequentially and conditional on previous columns sampled, so the graph probability is given by the product of the conditional probabilities of the columns. For each column, this conditional probability is simply the product of the uniform probabilities used to choose d_i for each knot k_i and the probabilities of the rows that are sampled for each of those knots. Let $\mathcal{S}$ represent the rows $k_{i-1} + 1$ to k_i, the rows from which we will be sampling d_i times, and A_l $(l = 0, ..., d_i)$ the rows in $\mathcal{S}$ that have been sampled after d_i draws. Then $\prod_{l=1}^{d_i} P\left(s_l, A_{l-1}^c\right)$ gives the probability of sampling row s_1, then row s_2, and so on. According to the distribution that we will use in computing inclusion probabilities, however, $\prod_{l=1}^{d_i} P\left(s_l, A_{l-1}^c\right)$ does not depend on the ordering of s_1 to s_{d_i}. Consequently, by computing the probability of the permutation that we observe,

we can easily compute the probability of the combination of 1's and 0's that are sampled for a particular column, as this will simply be $d_i! \prod_{l=1}^{d_i} P\left(s_l, A_{l-1}^c\right)$.

2.6 Essential Algorithmic Considerations

Ideally, we would like to sample graphs uniformly. For SIS, this is rarely possible, but we can often sample graphs from a distribution that is nearly uniform. As mentioned previously, sampling graphs from a relative uniform distribution is vital in enabling us to accurately estimate the number of graphs meeting marginal constraints and to approximate the null distribution of a test statistic. Additionally, the effective sample size increases as the sampling distribution approaches a uniform distribution. To ensure that graphs are sampled according to a relative uniform distribution, columns should almost always be arranged in decreasing order by column sum, and sampling of rows to receive a 1 should be done according to the conditional Poisson distribution.

2.6.1 Effective Sample Size and Squared Coefficient of Variation

The effective sample size gives a calculation of the equivalent uniform probability sample for the sample under consideration. Thus, if we sample N graphs uniformly, our effective sample size is simply N. Kong et al. (1994) show that, in the case of SIS, if we sample N graphs, the effective sample size is $\frac{N}{1+cv^2}$, where cv^2 is the square of the coefficient of variation of the standardized graph weights. If the graph probabilities are $p_1, p_2, ..., p_N$, then the standardized graph probabilities are given by $\frac{\frac{1}{p_1}N}{\sum_{i=1}^N \frac{1}{p_i}}, \frac{\frac{1}{p_2}N}{\sum_{i=1}^N \frac{1}{p_i}}, ..., \frac{\frac{1}{p_N}N}{\sum_{i=1}^N \frac{1}{p_i}}$, which have mean $\mu = 1$. Recall that the coefficient of variation is given by $cv = \frac{\sigma}{\mu}$, so, in the case of SIS, $cv^2 = \sigma^2$. This is approximated by the sample variance

$$\begin{aligned} s^2\left(\frac{\frac{1}{p_i}N}{\sum_{i=1}^N \frac{1}{p_i}}\right) &= \left(\frac{N}{\sum_{i=1}^N \frac{1}{p_i}}\right)^2 s^2\left(\frac{1}{p_i}\right) \\ &= \frac{\frac{1}{N-2}\sum_{i=1}^N \left[\frac{1}{p_i} - \frac{1}{N}\sum_{j=1}^N \frac{1}{p_j}\right]^2}{\left[\frac{1}{N}\sum_{j=1}^N \frac{1}{p_j}\right]^2}. \end{aligned} \tag{1}$$

In essence, cv^2 provides a measure of the distance between a uniform distribution and the SIS distribution, and $1 + cv^2$ measures the efficiency of the SIS distribution, relative to a uniform sampling distribution. To maximize the effective sample size, it is clear that we must minimize cv^2. Chen et al. (2005) argue that, in the case of zero-one tables, cv^2 is almost always minimized by rearranging columns so that the

column sums are in decreasing order and by sampling rows to receive a 1 according to a conditional Poisson distribution.

2.6.2 Conditional Poisson Distribution

The conditional Poisson distribution arises from the conditional distribution of a Poisson-binomial distribution. Borrowing from the notation of Chen et al. (2005), the Poisson-binomial distribution is given by a random variable $S_{\boldsymbol{Z}} = Z_1 + \cdots + Z_l$, where $\boldsymbol{Z} = (Z_1, ..., Z_l)$ denote Bernoulli trials with corresponding probability of success $\boldsymbol{p} = (p_1, ..., p_l)$. If we condition on $S_{\boldsymbol{Z}}$, the resulting distribution is the conditional Poisson distribution. Chen et al. (1994) argue that sampling rows according to such a distribution is much more efficient than sampling rows uniformly. With the conditional Poisson distribution, rows are sampled without replacement with probabilities that are proportional to $\boldsymbol{r}/n$. Let $\mathcal{S}$ again represent the rows from which we will be sampling d_i times and A_l $(l = 0, ..., d_i)$ the rows in $\mathcal{S}$ that have been sampled after l draws. Then row t will be sampled on draw l with probability

$$P\left(t, A_{l-1}^c\right) = \frac{w_t \Psi\left(d_i - l, A_{l-1}^c \backslash t\right)}{\left(d_i - l + 1\right) \Psi\left(d_i - l + 1, A_{l-1}^c\right)}, \tag{2}$$

where $w_t = \frac{r_t}{n - r_t}$ and Ψ is given by the recursive formula

$$\begin{aligned} \Psi\left(z, A\right) &= \sum_{B \subset A, |B| = z} \left(\prod_{i \in B} w_i \right) \\ &= \Psi\left(z, A \backslash \{z\}\right) + w_z \Psi\left(z - 1, A \backslash \{z\}\right) \end{aligned} \tag{3}$$

This distribution has the nice property that $\prod_{l=1}^{d_i} P\left(s_l, A_{l-1}^c\right)$ does not depend on the ordering of s_1 to s_{d_i}, greatly simplifying the computation of the graph probability. The algorithm for the recursive probability computation can be found in Algorithm 7 of the appendix.

3 A Simple Illustration of Sequential Importance Sampling

To illustrate the steps of sequential importance sampling for bipartite graphs, consider the graph presented in Table 3. We will illustrate the sampling procedure for the first column.

1	1	0	0	2
0	0	1	1	2
1	1	1	0	3
2	2	2	1	

Table 3: Second example

- **Reorder columns**

  The column sums $\boldsymbol{c} = (2, 2, 2, 1)$ are already in decreasing order, so there is no need to reorder the column sums.

- **Compute conjugate sequences**

  From the column sums, we calculate conjugate sequences $\boldsymbol{C}^{(1)} = (3, 2, 0)$, $\boldsymbol{C}^{(2)} = (2, 1, 0)$, $\boldsymbol{C}^{(3)} = (1, 0, 0)$, and $\boldsymbol{C}^{(4)} = (0, 0, 0)$

- **Determine knots and corresponding restrictions**

  Before we determine the knots, we arrange the rows so that the row sums are in decreasing order. Thus, the third row will now become the first row, and the other rows will move down one to produce the following graph:

1	1	1	0	3
1	1	0	0	2
0	0	1	1	2
2	2	2	1	

  We record the ordered row sums $\boldsymbol{r}^{ord} = (3, 2, 2)$ and find that the smallest value for t that produces $\sum_{i=1}^{t} r_i^{ord} > \sum_{i=1}^{t} C_i^{(1)}$ is $t = 3$. Thus, $k_1 = 3$, and $v_1 = \sum_{i=1}^{3} r_i^{ord} - C_i^{(1)} = 2$. Since this exhausts our rows, there are no other knots to record.

- **Sample rows to receive a 1**

  First, we determine the value of d_1, the number of 1's to sample from the first three rows, by uniformly sampling from $\{v_1, \min\{k_1, c_1\}\}=\{2, \min\{3, 2\}\}=\{2\}$. Thus, $d_1 = 2$ with probability 1. For each of the first three rows, we compute the probability of being sampled to receive a 1 on the first draw. To do this, we use the weights $\boldsymbol{w} = \frac{\boldsymbol{r}}{n-\boldsymbol{r}} = \{\frac{3}{4-3}, \frac{2}{4-2}, \frac{2}{4-2}\} = \{3, 1, 1\}$ and let $A_0^c = \{1, 2, 3\}$ represent rows one to three. Then the probability that we sample the first row to receive a 1 on our first draw is

$$P\left(1, A_0^c\right) = \frac{w_1 \Psi\left(1, A_0^c \backslash 1\right)}{2 \times \Psi\left(2, A_0^c\right)}.$$

Now $\Psi\left(0, A_0^c\right) = 1$, $\Psi\left(1, A_0^c\right) = \sum_{i \in A_0^c} w_i$ and $\Psi\left(2, A_0^c\right) = \sum_{i \neq j \in A_0^c} w_i w_j$, so

$$P\left(1, A_0^c\right) = \frac{3 \times (1+1)}{2 \times (3 \times 1 + 3 \times 1 + 1 \times 1)} = \frac{3}{7}.$$

Similar computations produce $P\left(2, A_0^c\right) = P\left(3, A_0^c\right) = \frac{2}{7}$. Thus, the first row is selected on the first draw with probability $\frac{3}{7}$, while the second and third row are each selected on the first draw with probability $\frac{2}{7}$. If the second row is selected first, then $A_1 = \{2\}$ and we recalculate inclusion probabilities on the second draw for the first and third columns, obtaining $P\left(1, A_1^c\right) = \frac{3 \times 1}{1 \times (3+1)} = \frac{3}{4}$ and $P\left(3, A_1^c\right) = \frac{1}{4}$. Suppose we select the first row on the second draw. Then our sample for the first column is $(1\ 1\ 0)^T$. This is not quite right, however, as we initially rearranged our rows to ensure that row sums were in decreasing order, so we must reorder our sample accordingly. Thus, our sample for the first column is in fact $(1\ 0\ 1)^T$.

- **Update row sums and repeat**
  Once we have sampled the first column, we need to decrement the row sums for rows that were sampled. This means that our new row sums are $\boldsymbol{r} = (1, 2, 2)$. Now we repeat the procedure, using the updated row sums and $\boldsymbol{C}^{(2)}$ on the 3×3 graph that has yet to be sampled. This process is repeated until the entire graph has been sampled.

- **Compute graph probabilities**
  The probability for the graph we are sampling is updated as sampling occurs. For the first column, we sampled $d = 2$ with probability 1, and we sampled the first and third (reordered) rows with probability $2! P\left(1, A_0^c\right) P\left(3, A_1^c\right) = 2! \times \frac{2}{7} \times \frac{3}{4} = \frac{3}{7}$, so the probability for our first column is $1 \times \frac{3}{7}$. Once we have sampled all columns, we simply multiply the probabilities corresponding to each column to obtain our graph probability. In practice, we consider log-probabilities and sum the log-probabilities to ensure that we do not encounter computational underflow problems in our calculations.

4 Estimation of the Likelihood Based on Sampling

A simple statistical model for a network is to posit that it is equally likely to be any member of a class of networks. Often the class of networks formed by all combinations of arcs is chosen. In our context the class could be all bipartite networks

with the same marginal totals as the observed network. More sophisticated models allow the probabilities of class members to differ and model these probabilities in a parsimonious manner. This allows researchers to statistically compare the observed network to the patterns that might have been observed if the network had been drawn with equal probability from the class.

Statistical models based on exponential families have a long history in social network analysis (Holland and Leinhardt, 1981; Frank and Strauss, 1986). These models allow complex social structure to be represented in an interpretable and parsimonious manner. For example, in the case of Darwin's finch data, we may suspect that competition and cooperation among finch species may be important in explaining the observed distribution of finches on the islands, so we would include a statistic in our model that would measure frequency of coexistence for different finch species.

Below we consider exponential random graph (ERG) models, although the framework is broad enough to encompass other model classes. An ERG model for bipartite graphs is an exponential family for which the sufficient statistics are a set of user-defined functions $Z(a)$ of the affiliation matrix a. The statistics $Z(a)$ are chosen to capture the hypothesized social structure of the bipartite network (Frank and Strauss, 1986; Morris, 2003). Models take the form:

$$P_\theta(A = a) = \frac{\exp\left(\theta \cdot Z\left(a\right)\right)}{\sum_{b \in \mathcal{A}} \exp\left(\theta \cdot Z\left(b\right)\right)}, \tag{4}$$

where $\mathcal{A}$ is our graph space, θ is our parameter vector and $Z(a)$ is the vector of sufficient statistics. In this form, it is easy to see that $\sum_{b \in \mathcal{A}} \exp\left(\theta \cdot Z\left(b\right)\right)$ normalizes our probabilities to ensure a valid distribution.

Inference for the model parameter θ can be based on the (logarithm of the) likelihood function corresponding to the model (4):

$$l(\theta; a_{obs}) \equiv \log\left\{P_\theta\left(A = a_{obs}\right)\right\} = -\log\left\{\sum_{a \in \mathcal{A}} \exp\left(\theta \cdot [Z\left(a\right) - Z\left(a_{obs}\right)]\right)\right\}. \tag{5}$$

The instances where this can be computed easily are uncommon, so an alternative approach is required. Here we apply the general approach proposed by Geyer and Thompson (1992). Suppose we have an independent sample from $\mathcal{A}$ denoted by $\{a_1, \ldots, a_M\}$, where a_k is selected from the set $\mathcal{A}$ with probability $p_k = \exp(q_k)$, and $Z_k = Z\left(a_k\right) - Z\left(a_{obs}\right)$ and $Z_{ik} = Z_i\left(a_k\right) - Z_i\left(a_{obs}\right)$.

We can estimate the log-likelihood via

$$\hat{l}(\theta) \;=\; -\log\left\{\sum_{k=1}^{M}\frac{1/p_k}{\sum_{j=1}^{M}1/p_j}\exp\left(\theta\cdot Z_k\right)\right\}. \tag{6}$$

Note that the graphs are weighted according to the inverse probability of being sampled, the importance or sample weight for the graph. Now suppose we sample under the model with parameter θ_0. Then for MCMC, the p_k are of the form $p_k = P_{\theta_0}\left(A = a_k\right)$ so

$$\begin{aligned}
\hat{l}(\theta) &= -\log\left\{\sum_{k=1}^{M}\frac{1/p_k}{\sum_{j=1}^{M}1/p_j}\exp\left(\theta\cdot Z_k\right)\right\} \\
&= -\log\left\{\sum_{k=1}^{M}\frac{\exp\left(-\theta_0\cdot Z\left(a_k\right)\right)\sum_{a\in\mathcal{A}}\exp\left(\theta_0\cdot Z\left(a\right)\right)}{\sum_{j=1}^{M}\exp\left(-\theta_0\cdot Z\left(a_j\right)\right)\sum_{a\in\mathcal{A}}\exp\left(\theta_0\cdot Z\left(a\right)\right)}\exp\left(\theta\cdot\left[Z\left(a_k\right)-Z\left(a_{obs}\right)\right]\right)\right\} \\
&= -\log\left\{\frac{\exp\left(-\theta\cdot Z\left(a_{obs}\right)\right)}{\sum_{j=1}^{M}\exp\left(-\theta_0\cdot Z\left(a_j\right)\right)}\sum_{k=1}^{M}\exp\left(\left(\theta-\theta_0\right)\cdot Z\left(a_k\right)\right)\right\}. \qquad (7)
\end{aligned}$$

This is simply a uniform weighting for each of the M sampled graphs.

Referring back to (6), if we ignore the constant shift in the log-likelihood, $\log(\sum_{j=1}^{M}1/p_j)$, we obtain

$$\hat{l}(\theta) = -\log\left\{\sum_{k=1}^{M}\exp\left(\theta\cdot Z_k - q_k\right)\right\}. \tag{8}$$

As equation (8) makes clear, the set $\{Z_k, p_k\}_{k=1}^{M}$ implicitly forms a discrete exponential family over the sample space $\{Z_k\}_{k=1}^{M}$ with probabilities $\{p_k\}_{k=1}^{M}$. Hence (8) is concave and will have a unique maxima if, and only if, the convex hull of $\{Z_k\}_{k=1}^{M}$ has the zero vector in its interior (Handcock, 2003). Let $\tilde{\theta}$ be the value of θ that maximizes (8). Under these conditions, (8) is smooth as a function of θ and $\tilde{\theta}$ can be found by standard Newton-type algorithms (Handcock, 2003). Specifically, if we define

$$w_k^* = \frac{M\exp\left(\theta\cdot Z_k - q_k\right)}{\sum_{j=1}^{M}\exp\left(\theta\cdot Z_j - q_j\right)},$$

we obtain the partial derivatives

$$\frac{\partial\hat{l}(\theta)}{\partial\theta_i} \;=\; -\exp\left(\hat{l}(\theta)\right)\sum_{k=1}^{M}Z_{ik}\exp\left(\theta\cdot Z_k - q_k\right)$$

$$
\begin{aligned}
&= -\frac{\sum_{k=1}^{M} Z_{ik} \exp\left(\theta \cdot Z_k - q_k\right)}{\sum_{k=1}^{M} \exp\left(\theta \cdot Z_k - q_k\right)} \\
&= -\frac{1}{M} \sum_{k=1}^{M} w_k^* Z_{ik} \qquad (9) \\
\frac{\partial^2 \hat{l}(\theta)}{\partial \theta_i \theta_j} &= \frac{\partial \hat{l}(\theta)}{\partial \theta_j} \frac{\partial \hat{l}(\theta)}{\partial \theta_i} - \frac{1}{M} \sum_{k=1}^{M} w_k^* Z_{ik} Z_{jk}, \\
&= \frac{1}{M^2} \left(\sum_{k=1}^{M} w_k^* Z_{ik} \right) \left(\sum_{k=1}^{M} w_k^* Z_{jk} \right) - \frac{1}{M} \sum_{k=1}^{M} w_k^* Z_{ik} Z_{jk}, \\
&= \frac{1}{M} \left[\frac{1}{M} \left(\sum_{k=1}^{M} w_k^* Z_{ik} \right) \left(\sum_{k=1}^{M} w_k^* Z_{jk} \right) - \left(\sum_{k=1}^{M} w_k^* Z_{ik} Z_{jk} \right) \right] \qquad (10)
\end{aligned}
$$

From the form of the log-likelihood and partials, it should be clear that, for each graph $a_k \in \mathcal{A}$ that we sample, we need only record the graph probability p_k and sufficient statistics $Z(a_k)$. We then use these probabilities, sufficient statistics and formula to compute $\tilde{\theta}$ using a Newton-Raphson algorithm.

Let $\hat{\theta}$ be the value of θ that maximizes the log-likelihood presented in (5). To compute the Monte Carlo standard error, Geyer (1994) shows that

$$
\sqrt{M} \frac{\partial \hat{l}(\hat{\theta})}{\partial \theta} \xrightarrow{\mathcal{L}} N(0, \Omega)
$$

and

$$
\sqrt{M} \left(\tilde{\theta} - \hat{\theta} \right) \xrightarrow{\mathcal{L}} N\left(0, \mathcal{I}^{-1} \Omega \mathcal{I}^{-1}\right).
$$

Here, Ω is approximated by

$$
\mathbb{V}\left(\sqrt{M} \frac{\partial \hat{l}(\tilde{\theta})}{\partial \theta} \right) = M \sum_{k=1}^{M} \frac{1}{M^2} \mathbb{V}\left(w_k^* Z_k\right) = \mathbb{V}\left(w_1^* Z_1\right), \qquad (11)
$$

where $w_1^* Z_1, \ldots, w_M^* Z_M$ are i.i.d. Hunter and Handcock (2006) show that the Fisher information matrix, $\mathcal{I}$, can be approximated by

$$
\hat{\mathcal{I}}(\tilde{\theta}) = \frac{1}{M} \left[\left(\sum_{k=1}^{M} w_k^* Z_k Z_k \right) - \frac{1}{M} \left(\sum_{k=1}^{M} w_k^* Z_k \right) \left(\sum_{k=1}^{M} w_k^* Z_k \right) \right] \qquad (12)
$$

Thus,

$$\mathbb{V}\left(\tilde{\theta}-\hat{\theta}\right) \approx \frac{1}{M}\left[\hat{\mathcal{I}}(\tilde{\theta})\right]^{-1}\mathbb{V}\left(w_1^* Z_1\right)\left[\hat{\mathcal{I}}(\tilde{\theta})\right]^{-1}, \tag{13}$$

and we obtain the Monte Carlo standard error through

$$\frac{1}{\sqrt{M}}\left(\operatorname{diag}\left\{\left[\hat{\mathcal{I}}(\tilde{\theta})\right]^{-1}\mathbb{V}\left(w_1^* Z_1\right)\left[\hat{\mathcal{I}}(\tilde{\theta})\right]^{-1}\right\}\right)^{\frac{1}{2}}. \tag{14}$$

5 Implementation

We have implemented the SIS algorithm of Section 2 and the likelihood estimators of Section 4. The core routines are coded in the **C** programming language because of its speed and efficiency. The support routines are coded in the **R** language (R Development Core Team, 2006) due to its flexibility and power. The MCMC algorithm of Snijders (1991) and Rao et al. (1996) has also been implemented. Finally, the code has been incorporated in to the **R** package **statnet** to provide access to other network analysis tools (e.g., plotting, summarization, goodness-of-fit, and simulation)(Handcock et al., 2003). In addition this presents a consistent user-interface for modeling and simulation of bipartite graphs using SIS. Both **R** and the **statnet** package are publicly available (See websites in the references for details). Code written in the **R** language and using **statnet** will be made available along with the data so that the analysis used in this paper can be reconstructed by the reader.

6 Applications and Comparison

In this section we apply the SIS sampling algorithm and likelihood framework to a number of common application problems. It is also compared to the competing MCMC algorithm.

6.1 Estimating Graph Space Size for Fixed Marginals

To establish the validity of the SIS algorithm and to apply it to a graph counting problem, we used Darwin's finch data, found in Table 1. Charles Darwin compiled this data on thirteen finch species on a visit to the Galapagos Islands. For each finch type, he recorded on which of seventeen islands that finch could be found. Sanderson (2000) argues that, in examining island biogeography, it is important to condition on the number of islands and species in order to sample from the appropriate null space, so graphs sampled from the null distribution of the observed graph should

have the same marginals. Chen et al. (2005) report the number of graphs matching the marginal constraints of Darwin's finch data to be 67,149,106,137,567,626. A sequential importance sample of size 10,000 estimated the total number of such graphs to be 6.722×10^{16} with a standard error of 7.2×10^{14}, estimates which closely match the results from CDHL's SIS algorithm. A separate sample of 1,000 graphs produced the histogram of inverse graph probabilities shown in Figure 1, which closely resembles CDHL's histogram of importance weights. Note the skewness of the weights. If SIS produced a simple random sample from the space of graphs, the weights would be equal (and equal to about 0.671×10^{17}). There are a substantial proportion of graphs with weights more than three times this level.

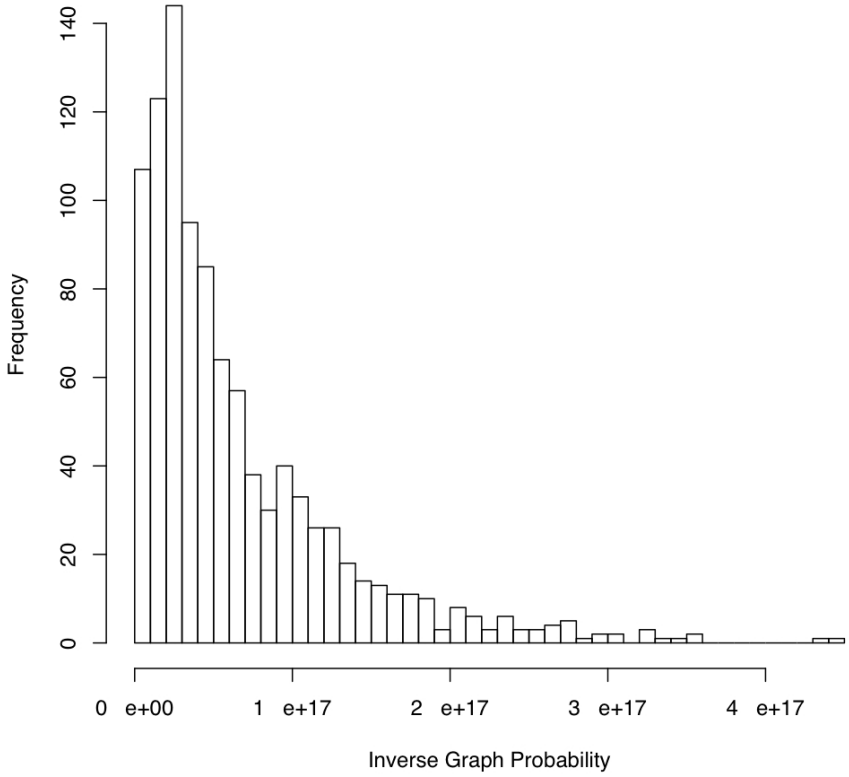

Figure 1: Histogram of Finch Data Importance Weights (1,000 Weights)

6.2 Approximating the Null Distribution of a Statistic

In considering approximations to the null distribution of a test statistic, CDHL again consider Darwin's finch data. In particular, they address the question of whether the observed grouping of finch species on islands happened by random chance or if it was the result of a struggle in which only species which depended on different food sources could coexist on an island. To test this hypothesis, CDHL consider the test statistic

$$\bar{S}^2 = \frac{1}{m(m-1)} \sum_{i \neq j} s_{ij}^2,$$

where m is the number of finch species, $\boldsymbol{S} = (s_{ij}) = \boldsymbol{A}\boldsymbol{A}^T$, and $\boldsymbol{A} = (a_{ij})$ is the bipartite graph in Table 1. This test statistic gives one of many possible measures of finch species coexistence. Here, s_{ij} is simply the number of islands on which finch species i and j coexist. At first glance, it may seem natural to consider the first moment, $\bar{S}$, but Roberts and Stone (1990) show that this will be the same for all graphs that meet the marginal constraints of the observed graph. Consequently, they suggest considering the second moment as a measure of coexistence. If there were no competition among the finch species, we would expect finches to share nearly equal numbers of islands with each different type of finch. If competition exists, however, we would expect that a finch will share a larger number of islands with non-competitive finch species and a smaller number of islands with competitive finch species. Since $\bar{S}^2$ is minimized for equal s_{ij} and maximized for values of s_{ij} that are furthest from $\bar{S}$, a large value of $\bar{S}^2$ would be consistent with competition among certain finch species and cooperation among others.

For the finch data, $\bar{S}^2$ has a value of 53.115. We would expect that, if the observed pattern happened by random chance, its value of $\bar{S}^2$ would not be an extreme value when compared with the values of $\bar{S}^2$ produced by randomly sampled graphs. Using the method of Section 4 and using SIS to sample $100,000$ independent graphs, we obtained the distribution of test statistics seen in Figure 2. Only one graph produced a test statistic larger than 53.115, providing strong evidence that it was unlikely that the grouping of finch species on islands was due to random chance and providing greater validity to the claim that it is the result of competition and cooperation among the species.

While $\bar{S}^2$ may prove useful in determining competition and cooperation among finch species, simply comparing how many finch pairs share a specified number of islands for both the observed graph and simulated graphs may better demonstrate the competition among certain species and cooperation among other species. Figure 3 attempts to do this by plotting the mean number of finch pairs sharing x islands, $x = 0, 1, ..., 17$, from 10,000 simulated graphs. These means are represented by a dot, and the trend is represented by a dashed line. Vertical lines for each possible value of x show the range of values represented in the simulated graphs for each of these values of x. For the sake of comparison, the number of finch pairs sharing x islands from the observed graph are also represented in Figure 3 by an $\times$, and the trend is represented by a solid line. The trend for the simulated graphs is almost the exact

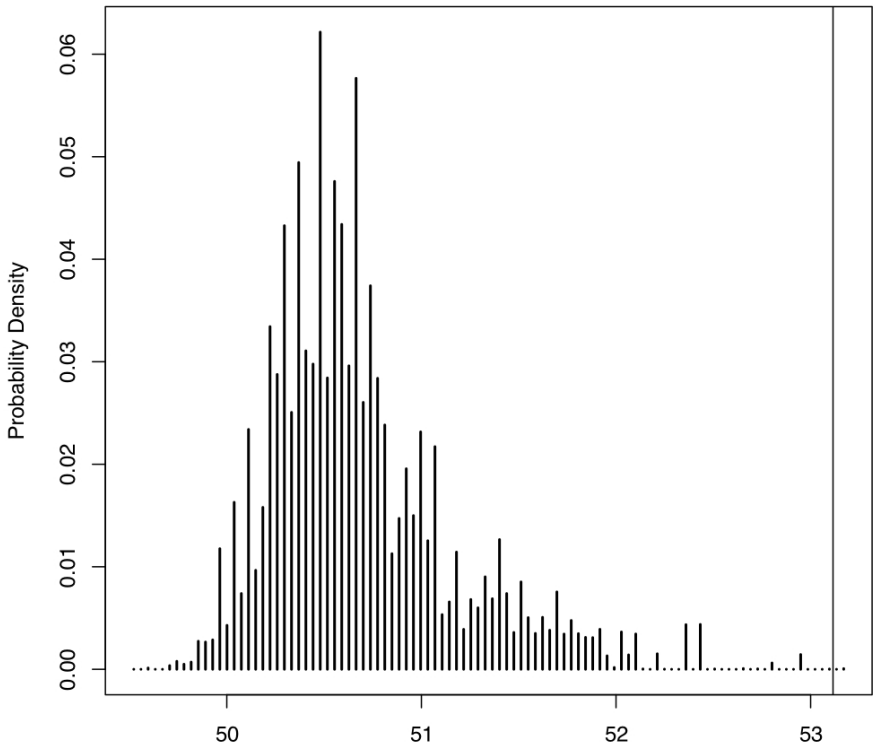

Figure 2: Null distribution of the test statistic $\bar{S}^2$

opposite of that for the observed graph, as valleys in the simulated graph primarily correspond with peaks in the observed graph, and vice versa. In addition, we would expect that, if competition were prevalent, the observed graph would produce larger numbers of pairs of finches sharing few islands and larger numbers of pairs of finches sharing many islands. This is what we observe, calling into question the hypothesis that what Darwin observed was the result of random chance.

6.3 Likelihood Inference for a Model of Competition Among Darwin's Finches

For Darwin's finch data, we could consider a number of statistical models to explain the observed graph. In any relevant model, it seems important to consider measures of competition and cooperation among finch species, as evidenced by Figures 2 and 3. Here, we considered an ERG model with a measure of competition that simply counts the number of pairings of finch species for which the two species share no islands in common. This measure, given by the sufficient statistic $G = \#\{(i,j) : \sum_k a_{ik}a_{jk} = 0\}$ for a_{ij} as defined previously, is similar to that proposed by Roberts and Stone in that it measures coexistence of finch species on islands, but it combines all instances where there is coexistence into one case, providing a simpler measure. If the coefficient θ for this term is not significantly different from 0, we will

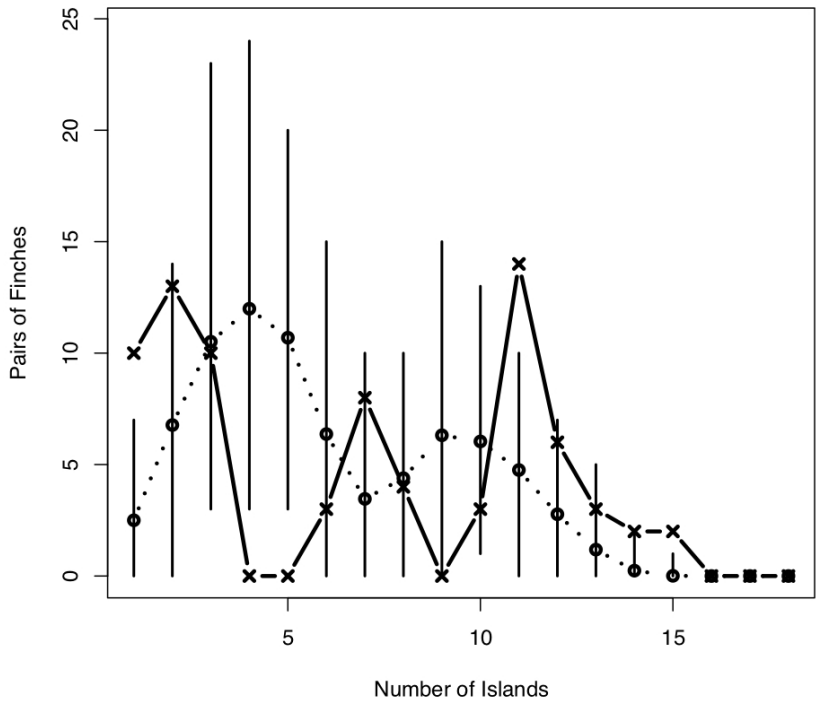

Figure 3: Number of pairs of finches sharing x islands, $x = 0, 1, ..., 17$

have evidence contradicting the claim that the observed sequence of finch species on islands is due to competition and cooperation among species.

For this measure, we obtained the histogram of sufficient statistics seen in Figure 4 for a sample of 10,000 graphs generated by SIS. As in the case of $\bar{S}^2$, the observed value of $G = 10$ is an extreme value, consistent with competition among finch species. Using the graphs generated by SIS, we estimated θ to be 0.835 with a standard error of 0.401, p-value for the Wald test of 0.039, and MCSE of 0.040. We can conduct a significance test of the null hypothesis that $\theta = 0$ using exact testing (Besag, 2000). The exact p-value of 6.25×10^{-4} suggests that the estimate is significantly different from 0, a result in line with competition existing among certain species.

These results are quite different from those produced by the MCMC algorithm. Using the **statnet** package to simulate graphs by MCMC, we sampled 10,000 graphs, using a burn-in of 10,000 and retaining only every hundredth graph sampled. The generating value for the MCMC algorithm, θ_0, was chosen to be the MLE. This choice optimally reduces the MCSE. The choice of $\theta_0 = 0$ corresponds to equally likely sampling of graphs that satisfy the marginal constraints. This choice will give a MCSE close to that of the SIS algorithm with a similar number of draws. It will be slightly worse if the interval for retaining graphs is small enough to induce significant positive correlation between the statistics. The choice of θ_0 equal to the MPLE (here, 0.361) will typically lead to an improvement over $\theta_0 = 0$ as it will be

closer to the optimal value. The MCSE for this choice is 0.0037, close to the optimal value.

The results for θ_0 set to the MLE will be closer to the actual performance. This is because a small initial run can be used (starting from the MPLE) to get an improved estimate of θ. Then a longer run can be used based on this improved estimate as the generating value. Thus typical implementations of the algorithm can use a simple iteration to improve estimation (a procedure that is not open to the SIS algorithm).

We present the results from the chain starting with a generating value given by the MLE, although the results from all three are nearly identical. The correlation between successive samples was 0.052. From these 10,000 graphs, and the Newton-Raphson estimate of Section 4, we estimated θ to be 0.609 with a standard error of 0.340 and p-value for the Wald test of 0.0744. Hence for a significance level of $\alpha = 0.05$, the Wald test would lead to completely different conclusions for SIS and MCMC. In the latter case, we would fail to conclude that there was evidence of competition and cooperation among the finch species.

To ascertain if these discrepancies between SIS and MCMC were the result of erroneous MCMC estimates, we considered a fourth Markov chain generated by $\theta_0 = 0.835$, the estimate for θ produced by SIS. If this was the MLE, then we would expect MCMC to produce an estimate of θ close to this. In fact, though, MCMC estimated θ to be 0.610 with a standard error of 0.344, results that seem to verify the results of our previous MCMC runs and discredit the SIS results. Increasing the SIS sample to 100,000 graphs, our estimates actually became worse, as θ was estimated to be 0.847 with a standard error of 0.387. Further increasing the sample size to 1,000,000 again failed to show any significant improvement in the estimates as θ was estimated to be 0.8261 with a standard error of 0.3659.

In addition to differences in the parameter estimate and standard error, another substantial difference between the two Monte Carlo methods is the Monte Carlo standard error (MCSE). For the sample of 10,000 graphs generated by SIS, the MCSE is estimated to be 0.040, about 10 times larger than the MCSE of 0.004 produced by MCMC. Hence the the effective sample size of the $10,000$ graphs generated by SIS is only slightly larger than 1,000 generated from the MCMC.

The MCMC intentionally generates graphs with sufficient statistics similar to the observed graph, thereby producing a lower MCSE. This can be seen in Figure 5, which shows a greater propensity for MCMC to produce sufficient statistics close to the observed value. SIS, on the other hand, samples from all graphs meeting the marginal constraints of the observed graph without intentionally giving preference to graphs with sufficient statistics similar to the observed graph. As a result, we

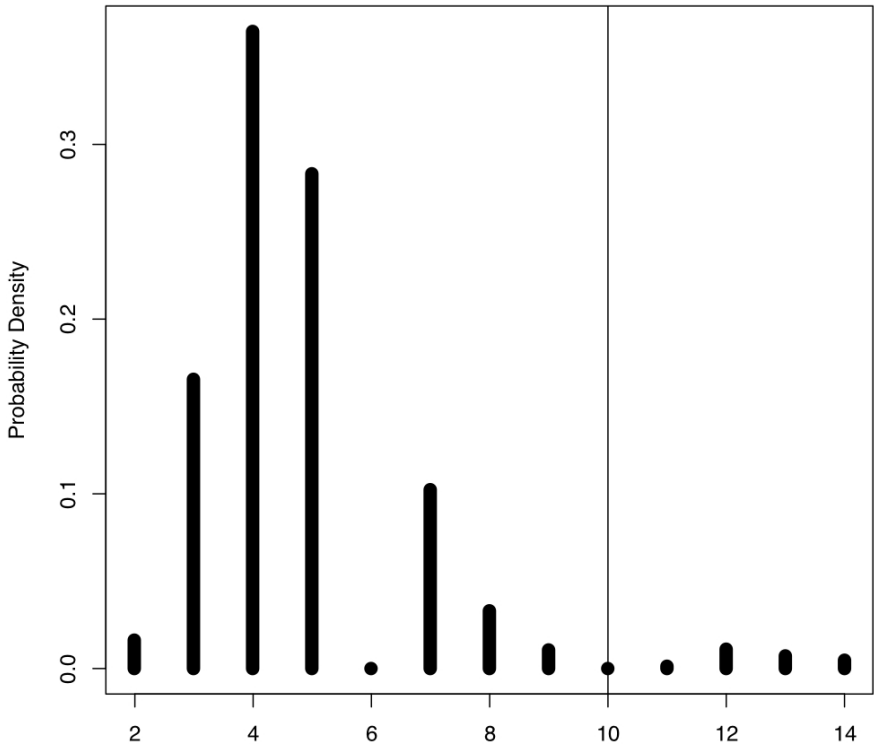

Figure 4: Null distribution of the test statistic G, as sampled by SIS

would expect a higher MCSE for SIS, as clearly evidenced when comparing Figure 4 and Figure 5.

We also remark that the increase in SIS sample size by a factor of ten decreased the estimate of the MCSE by a factor of 2.30 (versus the expected 3.16) based on (11). One possible explanation for this could be skewness in the distribution of $\boldsymbol{w}^*\boldsymbol{Z_k}$ or a small number of extreme values of $\boldsymbol{w}^*\boldsymbol{Z_k}$. In Figure 6, we see a histogram of $\log(\boldsymbol{w}^*\boldsymbol{Z_k})$, which is rather symmetric, suggesting skewness in the distribution of $\boldsymbol{w}^*\boldsymbol{Z_k}$. Comparing the sample variance of $\boldsymbol{w}^*\boldsymbol{Z_k}$ with the variance assuming a log-normal distribution, we obtain nearly identical estimates, so skewness does not appear to be a fundamental problem. Overall this comparison also illustrates a little appreciated aspect of SIS: despite being independent, the variation in the sample weights of the SIS reduces the effective sample size below that of a simple random sample.

Where we felt SIS would prove better than MCMC in likelihood inference was sparse graphs and graphs with skewed marginals. We created two sparse graphs:

- a graph with 1,000 rows, 5,000 columns, and 500 ties. For the rows we had 608 row sums of 0, 300 row sums of 1, 79 row sums of 2, 10 row sums of 3, and 3 row sums of 4; and for the columns we had 4,521 column sums of 0, 458 column sums of 1, and 21 column sums of 2;

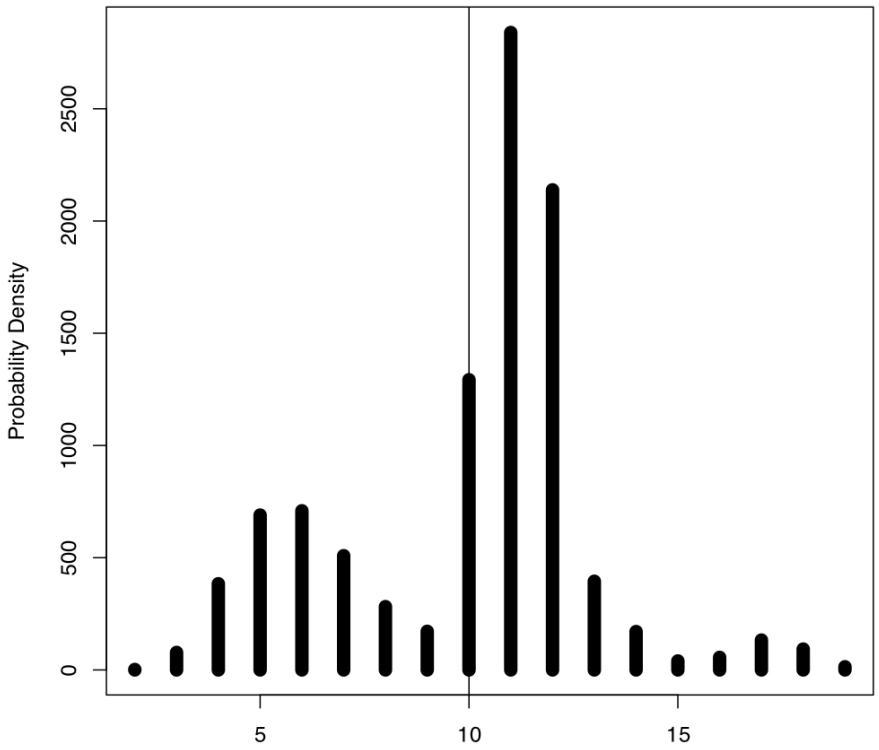

Figure 5: Distribution of the test statistic G produced by MCMC

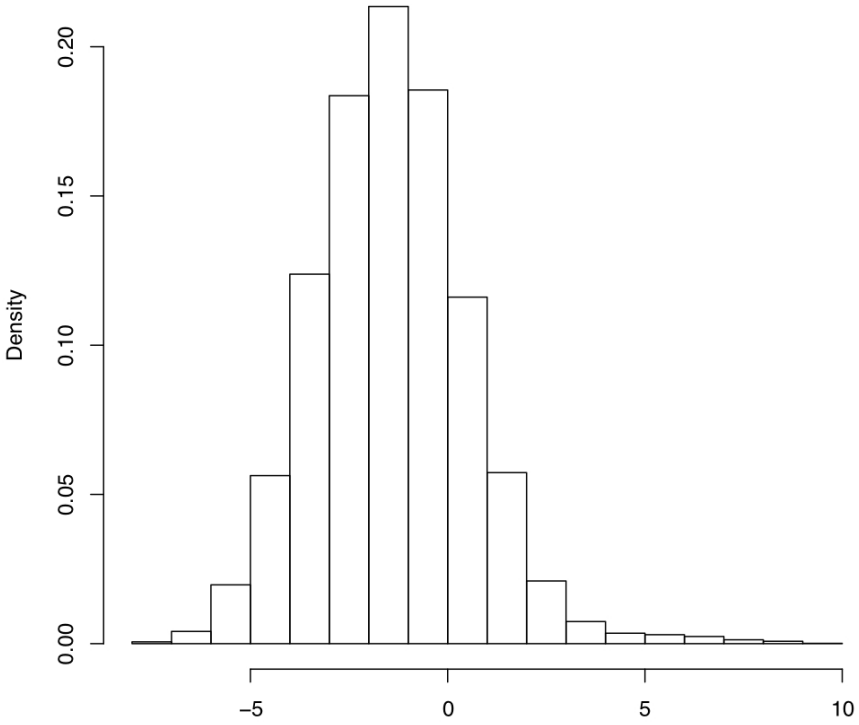

Figure 6: Distribution of $\log(\boldsymbol{w}^* \boldsymbol{Z}_{\boldsymbol{k}})$

- a graph with 50 rows, 100 columns, and 200 ties. All row sums were 4, and all column sums were 2.

For the first graph, we considered an ERG model with a term that measured instances where finches shared exactly one island. The sufficient statistic corresponding to this term appeared to be bi-modal with one of the modes being rare. In this case, MCMC was able to quickly sample from the entire distribution and produce consistent parameter estimates. SIS, on the other hand, had difficulty in sampling from regions of the graph space that corresponded to the rare mode, leading to either highly variable parameter estimates or an inability to estimate the parameter as a result of only the observed statistic being sampled.

For the second graph, we considered an ERG model which contained a term that measured instances where finches shared no islands. A MCMC sample of 10,000 graphs for which we had a burn-in of 10,000 graphs and retained only every 100th graph produced a parameter estimate of 1.385 with a standard error of 0.393 and MCSE of 0.006. A SIS sample of 10,000 graphs estimated the parameter to be 1.232 with a standard error of 0.388 and MCSE of 0.112. Increasing the SIS sample to 100,000, we estimated the parameter to be 1.797 with a standard error of 0.672 and MCSE of 0.089. Again, MCMC proved to be consistent in estimating the parameter, as the three different generating values resulted in the same estimates. SIS, on the other hand, proved wildly inconsistent, as an increase in the SIS sample size led to far worse estimates. Results were similar when examining graphs with skewed marginals.

7 Discussion

Unlike in the cases of the graph-counting problem and approximating the null distribution of a test statistic, empirical evidence fails to show that SIS has any distinct advantages over MCMC in likelihood inference problems. For Darwin's finch data, a graph with skewed marginals, and sparse graphs of large dimensions, MCMC proved to be much more efficient, producing consistent parameter estimates in a short period of time when compared with SIS. For instance, the SIS algorithm took nearly the same amount of time to simulate 10,000 graphs and compute parameter estimates as it took MCMC to do the same. However, considering that the effective sample size of the 10,000 graphs simulated by SIS was roughly equivalent to 1,000 graphs simulated by MCMC, MCMC is approximately 10 times faster in producing similar precision.

A more pressing matter is the inability of SIS to produce consistent parameter

estimates. This may be attributable to the observed statistic being near the edge of the sample space of the statistic. Since the probability of producing statistics exceeding the observed statistic is quite low, we might expect the SIS estimator to be poor. In addition, the estimate of the MCSE given by (14) will also be poor in these extreme cases.

Considering the struggles with obtaining consistent parameter estimates using SIS, it may be useful to explore modifications to the current SIS algorithm that will more frequently produce graphs with sufficient statistics similar to the observed graph, much in the way that MCMC does. Such modifications would undoubtedly affect the near-uniformity of the SIS sampling scheme, but it might help decrease the MCSE as well as increase the number of instances in which SIS could produce a valid parameter estimate, as there would be fewer instances where the observed sufficient statistic is either smaller than or larger than all simulated sufficient statistics. More importantly, it may better prevent the drastic fluctuation in parameter estimates, as most easily seen for sparse graphs.

In the future, it may be worthwhile to see if extensions of SIS to social networks and other graphs with structural zeros produce similar results when addressing graph counting problems, tests on null distributions of test statistics, and likelihood inference problems. Such extensions of SIS have only recently been developed, and the level of complexity of such algorithms is increased greatly because of the nature of the structural zero constraints. Considering other constraints may also be of interest in certain situations. For example, in the ongoing example of Darwin's finch data, Roberts and Stone (1990) suggest considering not only the marginal constraints but also an additional constraint which stipulates that species which do not occur on islands containing more than g species in the observed graph will not occur on islands containing more than g species in any of the simulated graphs. Consequently, it may be of use to explore how easily other constraints can be implemented.

References

Besag, J. (2000). Markov chain monte carlo for statistical inference. Working paper, Center for Statistics and the Social Sciences, University of Washington.

Chen, X. H., A. P. Dempster, and J. S. Liu (1994). Weighted finite population sampling to maximize entropy. *Biometrika 81*, 457–469.

Chen, Y., P. Diaconis, S. P. Holmes, and J. Liu (2005). Sequential monte carlo

methods for statistical analysis of tables. *Journal of the American Statistical Association 100*, 109–120.

Frank, O. and D. Strauss (1986). Markov graphs. *Journal of the American Statistical Association 81*(395), 832–842.

Gelman, A. (1996). Inference and monitoring convergence. In W. R. Gilks, S. Richardson, and D. J. Spiegelhalter (Eds.), *Markov Chain Monte Carlo in Practice*, pp. 131–144. New York: Chapman and Hall.

Geyer, C. J. (1994). On the convergence of monte carlo maximum likelihood calculations. *Journal of the Royal Statistical Society, Series B 56*, 261–274.

Geyer, C. J. and E. A. Thompson (1992). Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society, Series B 54*, 657–699.

Handcock, M. S. (2003). Assessing degeneracy in statistical models of social networks. Working paper, Center for Statistics and the Social Sciences, University of Washington.

Handcock, M. S., D. R. Hunter, C. T. Butts, S. M. Goodreau, and M. Morris (2003). *statnet: An R package for the Statistical Modeling of Social Networks*. `http://csde.washington.edu/statnet`.

Holland, P. W. and S. Leinhardt (1981). An exponential family of probability distributions for directed graphs. with comments by Ronald L. Breiger, Stephen E. Fienberg, Stanley S. Wasserman, Ove Frank and Shelby J. Haberman and a reply by the authors. *Journal of the American Statistical Association 76*(373), 33–65.

Hunter, D. R. and M. S. Handcock (2006). Inference in curved exponential family models for networks. *Journal of Computational and Graphical Statistics 15*, to appear.

Kong, A., J. Liu, and W. Wong (1994). Sequential imputations and bayesian missing data problems. *Journal of the American Statistical Association 89*, 278–288.

Morris, M. (2003). Local rules and global properties: Modeling the emergence of network structure. In R. Breiger, K. Carley, and P. Pattison (Eds.), *Dynamic Social Network Modeling and Analysis*, pp. 174–186. Committee on Human Factors, Board on Behavioral, Cognitive, and Sensory Sciences. National Academy Press: Washington, DC.

R Development Core Team (2006). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0.

Rao, A., R. Jana, and S. Bandyopadhyay (1996). A markov chain monte carlo method for generating random $(0, 1)$ matrices with given marginals. *Sankhya, Series A 58*, 225–242.

Roberts, A. and L. Stone (1990). Island-sharing by archipelago species. *Oecologia 83*, 560–567.

Robins, G., P. Pattison, Y. Kalish, and D. Lusher (2006). A workshop on exponential random graph (p^*) models for social networks. Submited to *Social Networks*, University of Melbourne.

Sanderson, J. (2000). Testing ecological patterns. *American Scientist 88*, 332–339.

Snijders, T. (1991). Enumeration and simulation methods for 0-1 matrices with given marginals. *Psychometrika 56*, 397–417.

van Duijn, M., K. Gile, and M. S. Handcock (2006). Comparison of maximum pseudo likelihood and maximum likelihood estimation of exponential random graph models. Manuscript, Center for Statistics and the Social Sciences, University of Washington.

APPENDIX

Algorithm 1 COMPUTE-CONJUGATE-SEQUENCES

The following is executed once.

1. Initialize conjugate sequences vector $c \leftarrow 0$
2. **for** $i \leftarrow 1$ to $ncol$
3. **for** $j \leftarrow i + 1$ to $ncol$
4. **for** $k \leftarrow 1$ to $colsum[j]$
5. **do** $c\,[nrow \times (i-1) + k] \leftarrow c\,[nrow \times (i-1) + k] + 1$

Algorithm 2 Sequential Importance Sampling

The following is executed for every iteration of SIS.

1. SORT-ROWS {Order rows in decreasing order by row sum}
2. COMPUTE-KNOTS {Determine k_i and v_i from row sums and conjugate sequence}
3. SIS-SAMPLE {Compute valid sample and record probability}
4. UPDATE-ROWS {Update row sums and reverse ordering from SORT-ROWS}

Algorithm 3 COMPUTE-KNOTS

The following is executed for each column of a bipartite graph.

1. Initialize row sums partial $rpart \leftarrow 0$ and conjugate partial $cpart \leftarrow 0$
2. **for** $i \leftarrow 1$ to $nrow$
3. **do** $rpart \leftarrow rpart + rowsum[i]$,
4. $cpart \leftarrow cpart + c[i]$
5. **if** $rpart > cpart$
6. **then** $knot[i] \leftarrow i$
7. $value[i] \leftarrow rpart - cpart$
8. **else** remove $knot[i]$, $value[i]$
9. **for** $i \leftarrow 1$ to $nrow$
10. **for** $j \leftarrow i + 1$ to $nrow$
11. **if** $value[j] \leq value[i]$
12. **then** remove $knot[j]$, $value[j]$
13. **if** $value[j] - value[i] \geq knot[j] - knot[i]$
14. **then** remove $knot[j]$, $value[j]$

Algorithm 4 SIS-Sample

1. Initialize total number sampled $total \leftarrow 0$
2. Generate random uniform(0,1) number *rand*
3. $nsamp \leftarrow value[0] + \left\lfloor \frac{rand}{\text{MIN}(knot[0], colsum) - value[0] + 1} \right\rfloor$
4. $total \leftarrow nsamp$
5. **for** $i \leftarrow 1$ to $knot[0]$
6. **do** $weights[i] \leftarrow \frac{rowsum[i]}{ncol - rowsum[i]}$
7. **for** $i \leftarrow knot[0]$ to $nrow$
8. **do** $weights[i] \leftarrow 0$
9. Sample-Without-Replacement(*weights, nsamp*)
10. **for** $i \leftarrow 1$ to Length(*knot*)
11. **do** Generate new *rand*
12. $nsamp \leftarrow value[i] - total + \left\lfloor \frac{rand}{\text{MIN}(knot[i] - knot[i-1], colsum - total) - \text{MAX}(value[i] - total, 0) + 1} \right\rfloor$
13. $total \leftarrow total + nsamp$
14. **for** $i \leftarrow 1$ to $knot[i-1]$
15. **do** $weights[i] \leftarrow 0$
16. **for** $i \leftarrow knot[i-1]$ to $knot[i]$
17. **do** $weights\,[i - knot[i-1] + 1] \leftarrow \frac{rowsum[i]}{ncol - rowsum[i]}$
18. **for** $i \leftarrow knot[i]$ to $nrow$
19. **do** $weights[i] \leftarrow 0$
20. Sample-Without-Replacement(*weights, nsamp*)

Algorithm 5 Sample-Without-Replacement

1. Initialize total number sampled $total \leftarrow 0$
2. Initialize inclusion probability vector $prob \leftarrow 0$
3. Initialize permutation vector $perm \leftarrow \{1, ..., \text{Length}(weights)\}$
4. **for** $i \leftarrow 1$ to $nsamp$
5. **for** $j \leftarrow 1$ to $\text{Length}(weights) - total$
6. **do** $probsum \leftarrow 0$
7. $prob[j] \leftarrow$ Compute-Inclusion-Probability(*weights, j, nsamp-i+1*)
8. Generate random uniform(0,1) number *rand*
9. **for** $l \leftarrow 1$ to $\text{Length}(weights) - total$
10. **do** $probsum \leftarrow probsum + prob[l]$
11. **if** $probsum > rand$
12. **then** break
13. Save $perm[l]$
14. Remove $weights[l]$, $perm[l]$
15. $total \leftarrow total + 1$

Algorithm 6 COMPUTE-INCLUSION-PROBABILITY

1. Initialize $l \leftarrow 0$
2. **for** $i \leftarrow 1$ to LENGTH($weights$)
3. **if** $i \neq j$
4. **then** $rweights[l] \leftarrow weights[i]$
5. $l \leftarrow l + 1$
6. $numerator \leftarrow$ RECURSIVE-PROBABILITY($rweights$, LENGTH($rweights$), $nsamp-1$)
7. $denominator \leftarrow$ RECURSIVE-PROBABILITY($weights$, LENGTH($weights$), $nsamp$)
8. $prob \leftarrow numerator/denominator$

Algorithm 7 RECURSIVE-PROBABILITY

1. Initialize vector $current \leftarrow 0$, $previous \leftarrow 0$
2. $current[1] \leftarrow 1$
3. **for** $i \leftarrow 1$ to LENGTH($rweights$)
4. **for** $j \leftarrow 1$ to $nsamp + 1$
5. **do** $previous[j] \leftarrow current[j]$
6. $minim \leftarrow$ MIN($i + 1, nsamp$)
7. **for** $j \leftarrow 2$ to $minim$
8. **do** $current[j] \leftarrow rweights[i] \times previous[j-1] + previous[j]$
9. **if** $i < nsamp$
10. **then** $current[minim] \leftarrow rweights[i] \times previous[minim-1]$
11. **else** $current[minim] \leftarrow rweights[i] \times previous[minim-1] + previous[minim]$