Behavior of Large Random Graph

Zheguang Zhao
Supervised by Prof. Paul Dupius

April 2017

1 Introduction

We consider the graph of N nodes, and each node with degree k, i.e. k half edges. We study the properties of the random graph where each node in the graph is equally likely to be paired with any other node. In particular, we are interested in the size of the largest connected component, and whether it is smaller than a fraction $\beta \in (0, 1)$ of the total number N of nodes.

To study this the largest connected component in this random graph, we use a Markov process to simulate the stochastic formulation of components in the graph. We develop both a Crude Markov Chain (CMC) algorithm and an importance sampling algorithm based on Gibbs conditioning principle and zero variance. We report the simulation result on a configuration of he number of repetitions $n = 10^6$, the node degree $k = 3$, the number of nodes $N = 100$, and the estimate of the probability that the largest component degree $d \leq \beta N$ with $\beta = \frac{7}{8}$.

2 CMC

We categorize the nodes of the random graph into three types based on the status of its half edges. A half edge is *assigned* if it is paired with another half edge. Otherwise it is *unassigned* and can be either *sleeping* or *awake*. A sleeping half edge can be waken so that it becomes available for any assignment. At the beginning all half edges are sleeping.

A node is *dead* if all of its half edges are already assigned. Otherwise the node is *alive* and has two cases. If all of its half edges are unassigned, then it is *sleeping*. Otherwise if some of its half edges are assigned, then it is *awake*. A sleeping node can be waken up by setting all of its half edges to be awake. Initially all nodes are sleeping.

Note that we do not simulate the entire graph, but instead we only keep track of two quantities as the states in the Markov process that evolve and lead us to a sample of the size of the largest connected component. Let $X_1(t)$ be the number of awake half edges, i.e. unassigned half edges left on the awake nodes, and $X_2(t)$ be the number of sleeping nodes, where $t = 0, 1, \dots$ is the time step of the Markov process. The state space of the Markov process is

$$\mathbf{X}(t) = \{(X_1(t), X_2(t)) = (x_1, x_2) : x_1 \geq 0, x_2 \geq 0, t = 0, 1, \dots\}$$

The initial state is

$$\mathbf{X}(0) = (X_1(0), X_2(0)) = (0, N)$$

In general, the number of half edges available for assignment is the sum of the number of unassigned half edges on both awake nodes and sleeping nodes,

$$X_1(t) + kX_2(t)$$

If $X_1(t) = 0$ and $X_2(t) = x_2 > 0$, we choose a sleeping node uniformly at random and wake it up by setting all of its half edges to be awake. The corresponding transition of states is

$$\mathbf{X}(i) \to \mathbf{X}(j) = (0, x_2) \to (k, x_2 - 1)$$

and the transition probability is

$$p_{ij} = 1$$

If $X_1(t) = x_1 > 0$ and $X_1(t) + kX_2(t) > 1$, we first pick an half edge from x_1 awake half edges according to uniform distribution. To pair this half edge, we pick another half edge from the remaining $X_1(t) + kX_2(t) - 1$ unassigned half edges again uniformly at random. Because the second step involves assigning a half edge that is either awake or sleeping, so we consider two cases.

In the first case, both half edges to pair are from the awake nodes. The corresponding state transition is

$$\mathbf{X}(i) \to \mathbf{X}(j) = (x_1, x_2) \to (x_1 - 2, x_2)$$

with transition probability

$$p_{ij} = \frac{x_1 - 1}{x_1 + kx_2 - 1} \tag{1}$$

By contrast, in the second case, the first half edge is from the awake nodes but the second half edge is from the sleeping nodes. To choose the second half edge, we randomly pick a sleeping half edge, and then as a side effect wake up the corresponding node by setting all of its half edges to be awake too. The corresponding state transition is

$$\mathbf{X}(i) \to \mathbf{X}(j) = (x_1, x_2) \to (x_1 - 2 + k, x_2 - 1)$$

with transition probability

$$p_{ij} = \frac{kx_2}{x_1 + kx_2 - 1} \tag{2}$$

Note that the transition probabilities (1) and (2) sum to 1.

When both $X_1(t) = 0$ and $X_2(t) = 0$, i.e. $X_1(t) + kX_2(t) = 0$, all half edges are assigned and the Markov process stops (i.e. with self transition probability $p_{ii} = 1$ at state $(x_1, x_2) = (0, 0)$).

When $X_1(t) = 0$, there is no more awake half edges, meaning each node is either dead or sleeping. All the dead nodes (if any) form a connected component because none of them is connected to any sleeping nodes (if any). So between two times τ_1, τ_2 where $X_1(\tau_1) = X_1(\tau_2) = 0$, we form a connected component of size

$$X_2(\tau_1) - X_2(\tau_2)$$

To estimate the probability of having a largest connected component with degree no greater than a fraction of the total number nodes, we use the Crude Monte Carlo sampling in Algorithm 1.

Algorithm 1 CMC

```
 1: function CMC($N, k, \beta, n$)
 2: input: Number of nodes $N$, node degree $k$, largest component degree ratio
    $\beta$, number of repetitions $n$.
 3: output: Estimate of $l = P(d \leq \beta N)$ where $d$ is the degree of the largest
    component.
 4:     for $j = 1, \cdots, n$ do
 5:         $(x_1, x_2) = (0, N)$.
 6:         for $i = 0, 1, \cdots, kN/2$ time steps do
 7:             if $i = 0$ then
 8:                 $(x_1, x_2) = (k, x_2 - 1)$
 9:             else if $x_1 = 0$ then
10:                 break
11:             else
12:                 $u \sim \mathrm{U}(0, 1)$
13:                 $(x_1, x_2) \leftarrow \begin{cases} (x_1 - 2, x_2) & u \leq \frac{x_1 - 1}{x_1 + kx_2 - 1} \\ (x_1 - 2 + k, x_2 - 1) & \textit{else} \end{cases}$
14:         $I_j = \begin{cases} 1 & N - x_2 \leq \beta * N \\ 0 & \text{else} \end{cases}$
15:     return $\hat{l} = \frac{1}{n} \sum_j^n I_j$
```

3 Importance sampling

However, the event of a largest component with degree no greater than βN can be rare if $\beta < 1$, because the Law of Large Numbers for stochastic processes states that the average largest component degree converges to N. So to improve the sampling efficiency, we use Gibbs conditioning principle to develop the importance sampling scheme.

By the Law of Large Numbers for stochastic processes, we have

$$\frac{1}{N} \times (Nt) \to \boldsymbol{\phi}(t) = (\phi_1(t), \phi_2(t)) \text{ as } N \to \infty$$

where $\boldsymbol{\phi}(t)$ is

$$\begin{aligned}\phi_1(t) &= 3 - 2t - 3(1 - 2t/3)^{3/2} \\ \phi_2(t) &= (1 - 2t/3)^{3/2}\end{aligned}$$

for $t \in [0, 3/2]$.

If we condition on returning to $X_1(t) = 0$ before $\frac{3N}{2}\beta$, then we have by conditional Law of Large Numbers

$$\frac{1}{N} X(Nt) \approx \bar{\boldsymbol{\phi}}(t) = (\bar{\phi}_1, \bar{\phi}_2)$$

where $\bar{\boldsymbol{\phi}}(t)$ is

$$\bar{\phi}_1(t) = 3(1 - \bar{\phi}_2(t)) - 2t \tag{3}$$

$$\bar{\phi}_2(t) = 1 - \beta(1 - (1 - 2t/3)^{3/2}) \tag{4}$$

for $t \in [0, 3/2]$. Note that $\bar{\boldsymbol{\phi}}(t) = \boldsymbol{\phi}(t)$ when $\beta = 1$.

We follow the trajectory $\bar{\boldsymbol{\phi}}$. At discrete time $i \in [0, 3N/2]$, we associate $t = i/N$ such that $t \in [0, 3/2]$ as in (3). Then the transition probability in the importance sampling is

$$\begin{aligned}q_1(i) &= P(X_i = (x_1, x_2) \to (x_1 - 2, x_2)) \\ q_2(i) &= P(X_i = (x_1, x_2) \to (x_1 - 2 + k, x_2 - 1))\end{aligned}$$

so that

$$\begin{aligned}E[\mathbf{X}(i+1) - \mathbf{X}(i)] &= (-2, 0)q_1(i) + (-2 + k, -1)q_2(i) \\ &= \left(\frac{d}{dt}\bar{\phi}_1(t), \frac{d}{dt}\bar{\phi}_2(t)\right)\end{aligned}$$

We can solve for the transition probability $q_1(i)$ and $q_2(i)$ in terms of the derivatives of $\bar{\phi}_1$ and $\bar{\phi}_2$

$$\begin{aligned}q_1(i) &= \left(\frac{d}{dt}\bar{\phi}_1(t) + \frac{d}{dt}\bar{\phi}_2(t)\right)/(-2) \\ q_2(i) &= -\frac{d}{dt}\bar{\phi}_2(t)\end{aligned}$$

where

$$\begin{aligned}\frac{d}{dt}\bar{\phi}_1(t) &= 3\beta(1 - 2t/3) - 2 \\ \frac{d}{dt}\bar{\phi}_2(t) &= -\beta(1 - 2t/3)\end{aligned}$$

The likelihood ratio of the importance sampling is

$$W(X) = \prod_i w(i)$$

where

$$w(i) = \begin{cases} p_1(i)/q_1(i) & (x_1, x_2) \to (x_1 - 2, x_2) \\ p_2(i)/q_2(i) & (x_1, x_2) \to (x_1 - 2 + k, x_2 - 1) \end{cases}$$

Note that by using the conditioning, the trajectory may not return to $\mathbf{X}_1(i) = 0$ for $i \in \frac{3N}{2}\beta$. In this case we do not count the sample. The detail algorithm is described in Algorithm 2.

Algorithm 2 Importance Sampler

```
1: function IMPORTANCE($N, k, \beta, n$)
2: input: Number of nodes $N$, node degree $k$, largest component degree ratio $\beta$, number of repetitions $n$.
3: output: Estimate of $l = P(d \le \beta N)$ where $d$ is the degree of the largest component.
4:     for $j = 1, \cdots, n$ do
5:         $(x_1, x_2) = (0, N)$.
6:         for $i = 0, 1, \cdots, kN/2$ time steps do
7:             if $i = 0$ then
8:                 $(x_1, x_2) = (k, x_2 - 1)$
9:             else if $x_1 = 0$ then
10:                break
11:            else
12:                $(p_1, p_2) = \left(\frac{x_1 - 1}{x_1 + kx_2 - 1}, 1 - p_1\right)$
13:                $(\bar{\phi}_1, \bar{\phi}_2)(t) = \left(3(1 - \bar{\phi}_2(t) - 2t, 1 - \beta(1 - (1 - 2t/3)^{3/2}\right)$
14:                $(q_1, q_2) = \left(\frac{D\bar{\phi}_1(t) + D\bar{\phi}_2(t)}{-2}, -D\bar{\phi}_2(t)\right)$
15:                $u \sim \mathrm{U}(0, 1)$
16:                $(x_1, x_2) \leftarrow \begin{cases} (x_1 - 2, x_2) & u \le q_1 \\ (x_1 - 2 + k, x_2 - 1) & else \end{cases}$
17:                $w_j \leftarrow \begin{cases} w_j p_1/q_1 & u \le q_1 \\ w_j p_2/q_2 & else \end{cases}$
18:        $I_j = \begin{cases} 1 & x_1 = 0 \text{ and } N - x_2 \le \beta * N \\ 0 & \text{else} \end{cases}$
19:    return $\hat{l} = \frac{1}{n}\sum_j^n I_j w_j$
```

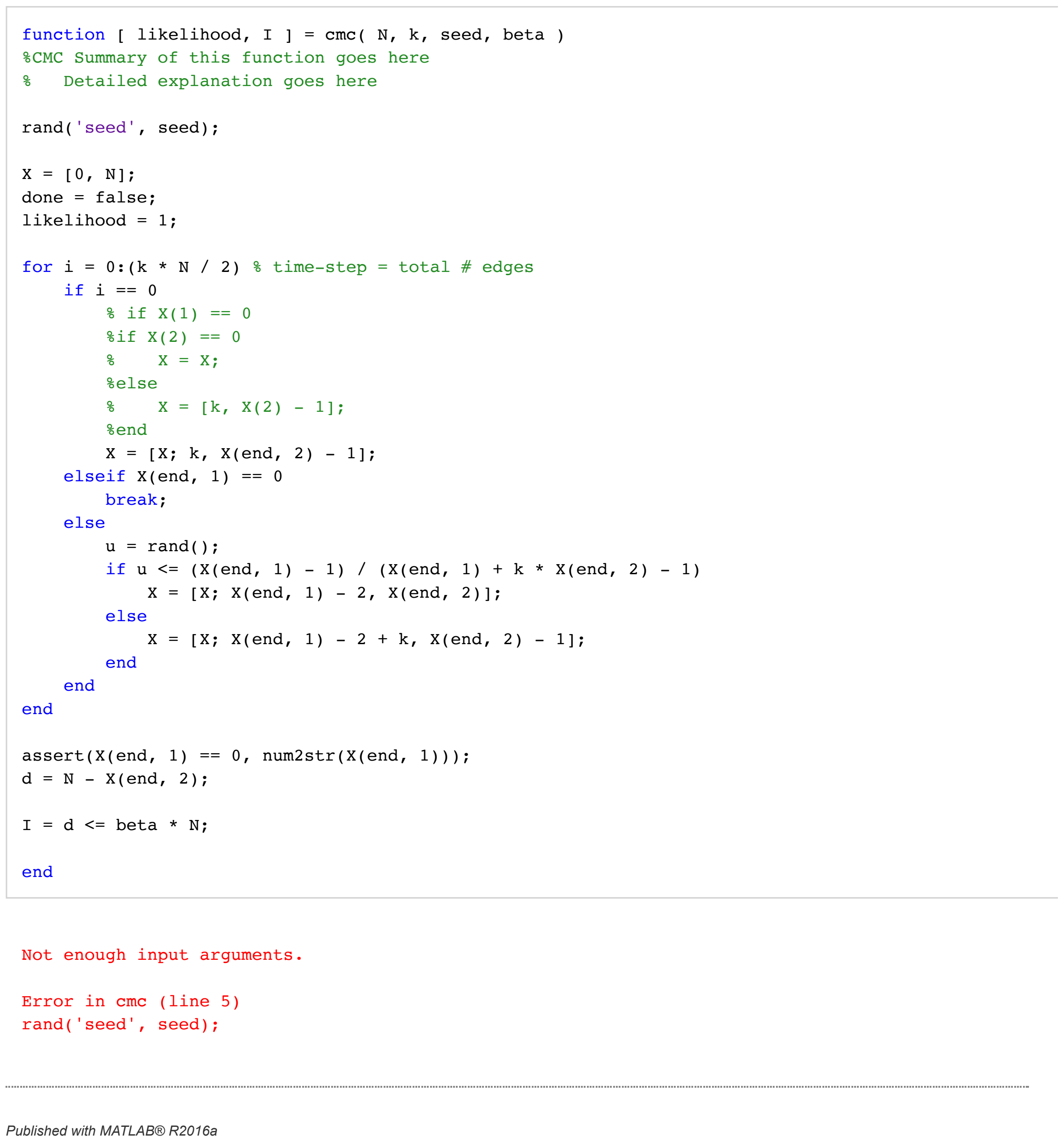
	Estimate	Standard error	Relative error
CMC	6.5000×10^{-5}	8.0620×10^{-6}	0.1240
Importance	5.9256×10^{-5}	3.1189×10^{-7}	0.0053

Table 1: Summary of results. The number of reptetions is $n = 10^6$, the node degree $k = 3$, the number of nodes $N = 100$, and the estimate of the probability that the largest component degree $d \leq \beta N$ with $\beta = \frac{7}{8}$.

4 Result

We run the CMC and importance sampling algorithm for the number of reptetions $n = 10^6$, the node degree $k = 3$, the number of nodes $N = 100$, and the estimate of the probability that the largest component degree $d \leq \beta N$ with $\beta = \frac{7}{8}$. The result is listed in Table 4. The Importance Sampling algorithm has about an order of magnitude smaller standard error.

5 Algorithms

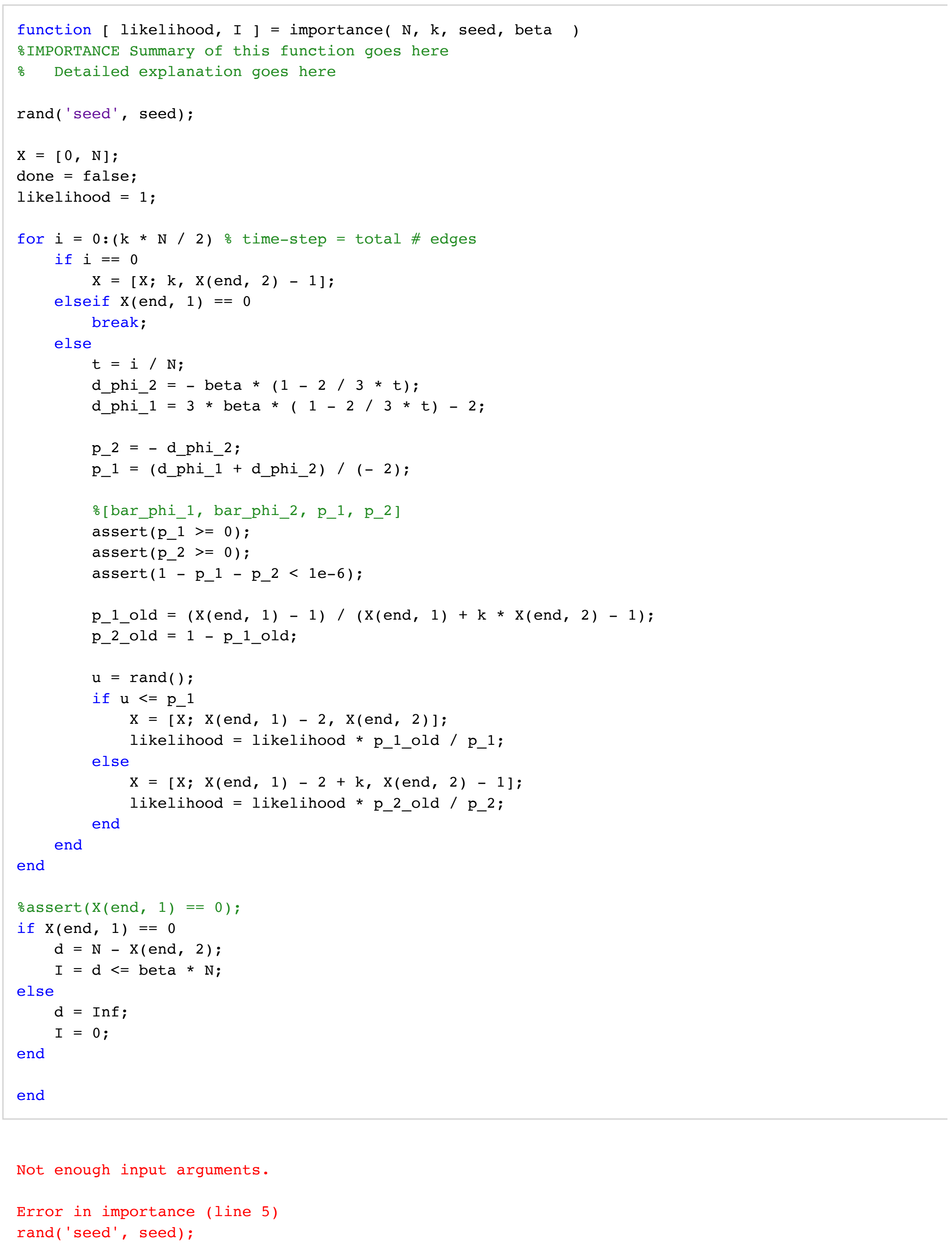


```matlab
function [ likelihood, I ] = cmc( N, k, seed, beta )
%CMC Summary of this function goes here
%   Detailed explanation goes here

rand('seed', seed);

X = [0, N];
done = false;
likelihood = 1;

for i = 0:(k * N / 2) % time-step = total # edges
    if i == 0
        % if X(1) == 0
        %if X(2) == 0
        %    X = X;
        %else
        %    X = [k, X(2) - 1];
        %end
        X = [X; k, X(end, 2) - 1];
    elseif X(end, 1) == 0
        break;
    else
        u = rand();
        if u <= (X(end, 1) - 1) / (X(end, 1) + k * X(end, 2) - 1)
            X = [X; X(end, 1) - 2, X(end, 2)];
        else
            X = [X; X(end, 1) - 2 + k, X(end, 2) - 1];
        end
    end
end

assert(X(end, 1) == 0, num2str(X(end, 1)));
d = N - X(end, 2);

I = d <= beta * N;

end
```

```
Not enough input arguments.

Error in cmc (line 5)
rand('seed', seed);
```

Published with MATLAB® R2016a

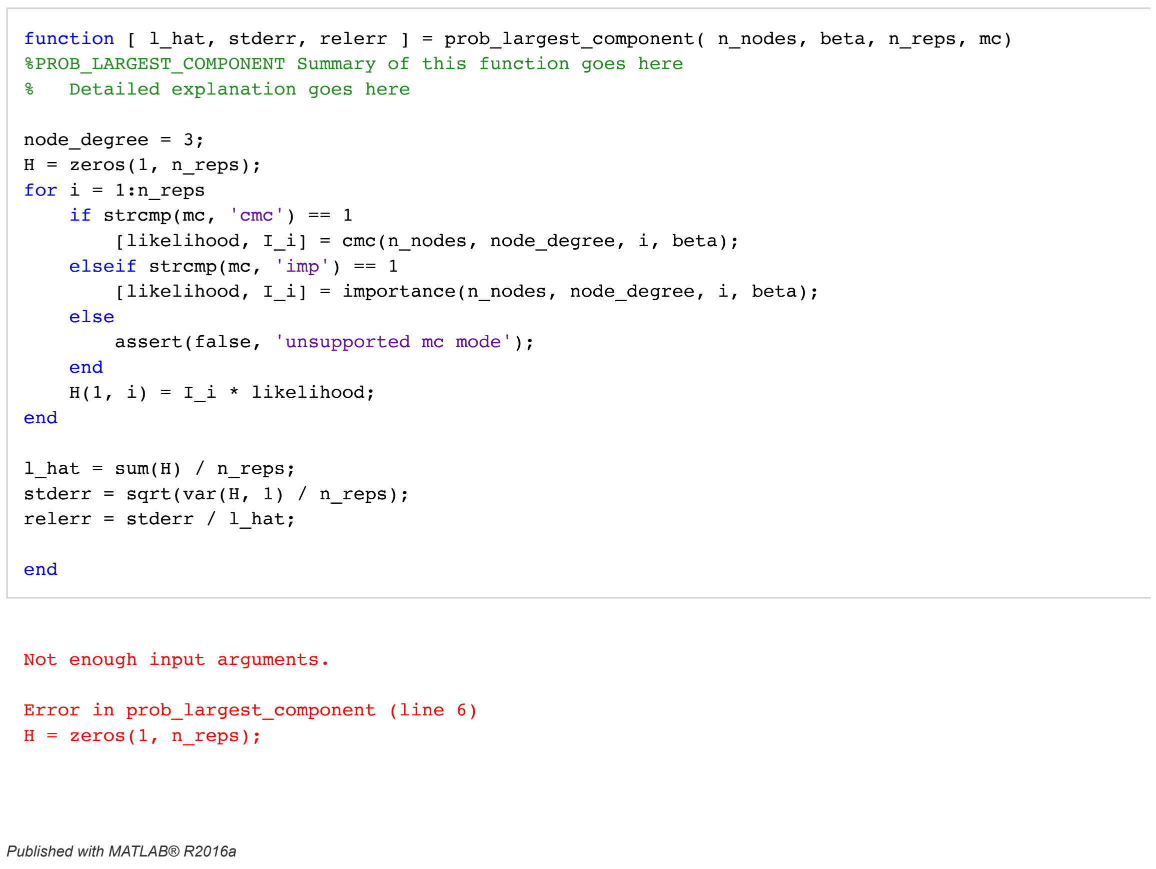
Figure 1: CMC



```
function [ likelihood, I ] = importance( N, k, seed, beta  )
%IMPORTANCE Summary of this function goes here
%   Detailed explanation goes here

rand('seed', seed);

X = [0, N];
done = false;
likelihood = 1;

for i = 0:(k * N / 2) % time-step = total # edges
    if i == 0
        X = [X; k, X(end, 2) - 1];
    elseif X(end, 1) == 0
        break;
    else
        t = i / N;
        d_phi_2 = - beta * (1 - 2 / 3 * t);
        d_phi_1 = 3 * beta * ( 1 - 2 / 3 * t) - 2;

        p_2 = - d_phi_2;
        p_1 = (d_phi_1 + d_phi_2) / (- 2);

        %[bar_phi_1, bar_phi_2, p_1, p_2]
        assert(p_1 >= 0);
        assert(p_2 >= 0);
        assert(1 - p_1 - p_2 < 1e-6);

        p_1_old = (X(end, 1) - 1) / (X(end, 1) + k * X(end, 2) - 1);
        p_2_old = 1 - p_1_old;

        u = rand();
        if u <= p_1
            X = [X; X(end, 1) - 2, X(end, 2)];
            likelihood = likelihood * p_1_old / p_1;
        else
            X = [X; X(end, 1) - 2 + k, X(end, 2) - 1];
            likelihood = likelihood * p_2_old / p_2;
        end
    end
end

%assert(X(end, 1) == 0);
if X(end, 1) == 0
    d = N - X(end, 2);
    I = d <= beta * N;
else
    d = Inf;
    I = 0;
end

end
```

```
Not enough input arguments.

Error in importance (line 5)
rand('seed', seed);
```

Figure 2: Importance sampling



```
function [ l_hat, stderr, relerr ] = prob_largest_component( n_nodes, beta, n_reps, mc)
%PROB_LARGEST_COMPONENT Summary of this function goes here
%   Detailed explanation goes here

node_degree = 3;
H = zeros(1, n_reps);
for i = 1:n_reps
    if strcmp(mc, 'cmc') == 1
        [likelihood, I_i] = cmc(n_nodes, node_degree, i, beta);
    elseif strcmp(mc, 'imp') == 1
        [likelihood, I_i] = importance(n_nodes, node_degree, i, beta);
    else
        assert(false, 'unsupported mc mode');
    end
    H(1, i) = I_i * likelihood;
end

l_hat = sum(H) / n_reps;
stderr = sqrt(var(H, 1) / n_reps);
relerr = stderr / l_hat;

end
```

```
Not enough input arguments.

Error in prob_largest_component (line 6)
H = zeros(1, n_reps);
```

Published with MATLAB® R2016a

Figure 3: Probability of the degree of largest connected component being less than or equal to β.