Discrete Applied Mathematics Final Part II

Jason Chung and Jasdeep Singh

December 15, 2021

1 Introduction

In Part I of this Project, we compared face-to-face contact data sets with the co-presence data sets. Some important distinctions to make are that face-to-face networks are precise yet difficult to obtain, while co-presence networks are less precise yet easily available. Our comparison of the network statistics was insightful, but did not necessarily yield useful information about the possibility of using one data set as a replacement of the other.

In order to determine if two types of network and their related data are interchangeable, we will consider a simple model of an infectious disease spreading on the networks that is transmitted when two individuals are in contact (either face-to-face or co-presence). Historic examples of such epidemics include the Great Plague of London (17th century), the 1918 influenza pandemic, the Severe Acute Respiratory Syndrome (SARS) outbreak of 2003, the 2009 influenza A (H1N1) epidemic, the 2019 SARS-CoV-2 (COVID-19) pandemic, etc. Similar mathematical models can also be used to model gossips on social networks, or the spread of computer viruses.

2 Methodology

2.1 The Susceptible-Infected-Recovered (SIR) Model on a Network

The Kermack-McKendrick epidemic model, also known as the susceptible-infected-recovered (SIR) model, is a model in which each individual is a node and the total number of nodes n is divided into three subsets:

$$n = S(t) + I(t) + R(t)$$

where

1. $S(t)$ is the number of susceptible nodes who have not been infected, but have no immunity and can still be infected if exposed to an infected neighboring node
2. $I(t)$ is the number of infected nodes who can transmit the infection to neighboring nodes that are susceptible, and $R(t)$ is the number of recovered nodes who were previously infected
3. but have recovered and gained immunity against the infection (they can no longer transmit the disease either).

The random variables $S(t), I(t)$ and $R(t)$ change according to the following conditions:

- If a node v is susceptible at time t and it is the neighbor of an infected node u, then v can become infected at time $t + \Delta t$ with a probability proportional to

  $$\beta A_{uv} \Delta t$$

  where A_{uv} is the entry in the adjacency matrix of the contact network, which accounts for the amount of contacts that happened between u and v.

- If v is infected at time t, then v can recover at time $t + \Delta t$ with a probability proportional to

$$\mu \Delta t$$

- If v is recovered (immune) at time t, then its status no longer changes.

The parameter β (measured in inverse of time units) controls the infection rate: if v is the neighbor of an infected node, then the average time interval for v to become infected is $1/\beta$.

Similarly, the parameter μ (also measured in inverse of time units) is the recovery rate since it takes a time interval of $1/\mu$ for node v to recover once it has been infected, or equivalently the mean infectious period is $1/\mu$

To better understand the roles of β and μ, let us consider the situation where all the nodes are initially healthy but susceptible.
Once a single node is infected, it will it will infect about $\beta\overline{\mathrm{d}}\Delta t$ nodes over the time step Δt, where $\overline{\mathrm{d}}$ is the average degree of the graph G. If we integrate this number of infections over the mean infection period, we get about $\beta\overline{\mathrm{d}}/\mu$ nodes infected by node v.

Each infected node will infect about $\beta\overline{\mathrm{d}}/\mu$ nodes during the mean infectious period. The number

$$p_0 = \frac{\beta}{\mu}\overline{\mathrm{d}}$$

where $\frac{\beta}{\mu}\overline{\mathrm{d}}$ is the average degree, otherwise known as the basic reproduction number. p_0 determines if an epidemic will occur: if it is greater than 1, then the infection will die out, whereas if the value is less than 1 the number of infected nodes will increase exponentially. In this model, an epidemic will come to an end when enough nodes are recovered and thus protected from the virus.

2.2 Numerical simulation of the SIR model on a network

Consider the problem of simulating an epidemic on a graph G = (V,E) with adjacency matrix A. To carry out these simulations, we need the following variables:

- β: infection rate
- μ: recovery rate
- Δt: time step
- T: time interval for the simulation
- A: adjacency matrix

All epidemics are initiated with a source node chosen randomly amongst the set of nodes. The simulation stops when t is infinite because that is when the set of infectious nodes is empty and all nodes are either recovered (immune) or were never susceptible (infected).
The impact of the epidemic is quantified using the fraction of the number of vertices n_r that recovered (that were previously infected),

$$n_r = \frac{R(t_\infty)}{n}$$

For each simulation, when $n_r > 0.2$, 20 percent of the populated was infected and the epidemic was considered a large outbreak. In all simulations we use $\beta = 4 * 10^{-4}$, and we change the value of μ

Below is the pseudo-code for the simulation of the SIR infection on the graph G = (V,E). The source of infection is a random node v_0

```
1:procedure SIR(A,$\beta$,$\mu$,$\Delta t$,$v_0$,V)

// Initialize the variables
2:  nTimeSteps <— T/$\Delta t$
3:  q := $\mu\Delta t$
4:  susceptibleNodes <— V
5:  infectiousNodes <— {$v_0$}
6:  recoveredNodes <— $\emptyset$

// main loop
7:  for t := 1,nTimeSteps do
8:      for all v elements in infectiousNodes do

// neighbors of v become infected with probability p
9:          for all u elements in neighbors(v) do
10:             if u elements in susceptibleNodes then
11:                 p := $\beta A_{uv}\Delta t$
12:                 infectiousNodes <— $v_0$ with probability p
13:             end if
14:         end for

// v becomes recovered with probability q
15:         recoveredNodes <— $v_0$ with probability q
16:     end for
17: end for
18:end procedure
```

The Matlab code we used for the simulations was the above pseudocode converted to Matlab Code, and is included in section 7.1: SIR.m.

For each of the six face-to-face datasets, we performed 100 simulations of the SIR epidemic, using the following parameters:

- the unique vertex that is the source of the infection is chosen uniformly at random
- $\beta = 4 * 10^{-4}$
- $\mu = X/k$, where k is a number from 1 to 5 and X is 10 if the network is a face-to-face network and 50 if the network is a co-presence network
- $\Delta t = 5 * 10^{-3} * 1/\beta$

Plots showing the number of nodes that are susceptible, infected, and recovered throughout SIR simulations of each data set using a k value of 1 and 5 are included in section 8: SIR Models.

3 Question Three

For this question, we needed to plot as a function of p_0 the distributions of recovered nodes. Our code to run the simulations, calculate the percentage of recovered nodes, and display the distribution is found in section 7.2: SIRquestion3.m. The distributions that were plotted as a result are included in section 9: Recovered Node Distribution Plots.

4 Question Four

For this question, we needed to plot the fraction of epidemics where the final fraction of recovered nodes is greater than 20 percent as a function of p_0. Our code to run the simulations, take out epidemics where the final fraction was less than 20 percent, and display the fraction of epidemics is found in section 7.3: SIRquestion4.m. The resulting plot is below:

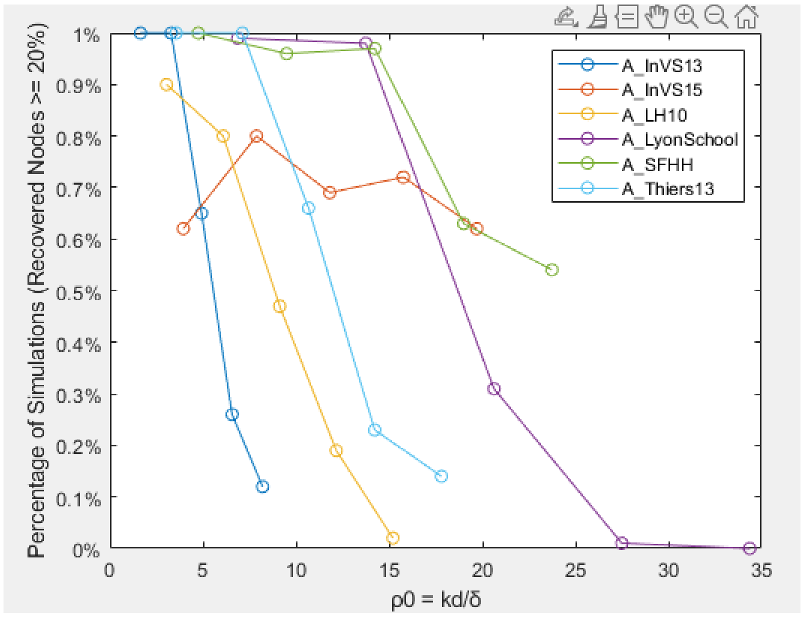


Figure 1: Fraction of Epidemics where the Final Fraction of Recovered Nodes, n_r , is Greater than 20 Percent as a Function of p_0

5 Question Five

For this question, we needed to plot the average number of recovered nodes for those epidemics where the final fraction of recovered nodes is greater than 20 percent as a function of p_0. Our code to run the simulations, take out epidemics where the final fraction was less than 20 percent, and display the fraction of epidemics is found in section 7.4: SIRquestion5.m. The resulting plot is below:

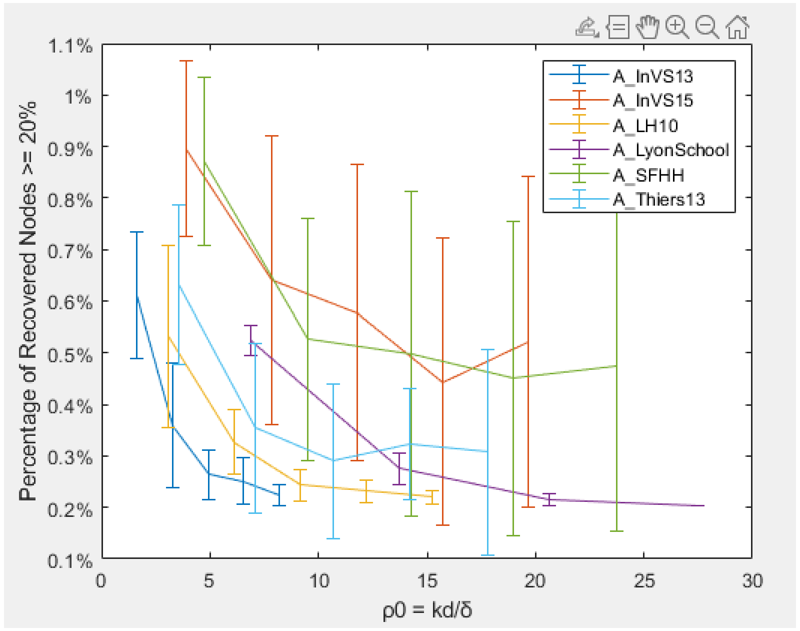


Figure 2: The Average Number of Recovered Nodes for Epidemics where the Final Fraction of Recovered Nodes is Greater than 20 Percent as a Function of p_0

6 Conclusion

Since the mean infection period is proportional to k, smaller values of k mean epidemics die out quickly, so that the entire population is not infected. In this situation, the spread of the virus, or simulation, ends because the infected population no longer exists.

Conversely, a larger value of k (k = 5) means that infections spread throughout the entire population, or all of the nodes. The simulations stops because there are no more susceptible nodes left, or people that can be infected left. The infected nodes will eventually recover but only after the simulation is stopped.

For each of the questions the simulation is stopped when either there are no more infected nodes or when there are no more susceptible nodes. We notice in each of the graph for question 3, as k increases, the graph starts shifting to the left. For each location, the histogram for question three for k = 1 is heavily concentrated to the right which means there is a high percentage of simulations that had a high percentage of recovered nodes. But as k increases we notice the histogram shift to the left. This means that there is a large percentage of simulations that have a low percentage of recovered nodes.

The same goes for question 4. We notice that as p_0 increases or (as k increases because k is proportional to p_0), figure 1 shows that starts to decay exponentially. It suddenly drops. This same effect goes to question 5. As p_0 increases or (as k increases because k is proportional to p_0), figure 2 shows the mean of each percentage of recovered nodes decrease.

Also we notice that locations with higher average degree distributions have lower percentage of recovered nodes. The virus/disease spreads faster when there the network is well connected when compared to networks that aren't as connected. Networks that aren't as connected wont have many infected nodes which means there will be more recover

7 Code Appendix

7.1 SIR.m

```
function SIR(filename, k, del)

    b = 4 * 10^-4;
    u = del * b/k;
    dt = 5 * 10^-3 * 1/b;

    q = u * dt;

    load("-mat", filename, 'U');

    susceptibleNodes = ones(1, length(U));
    infectiousNodes = zeros(1, length(U));
    recoveredNodes = zeros(1, length(U));

    susceptibleNodesGraph = [sum(susceptibleNodes, "all")];
    infectiousNodesGraph = [sum(infectiousNodes, "all")];
    recoveredNodesGraph = [sum(recoveredNodes, "all")];

    infectiousNodes(1, 1) = 1;
    susceptibleNodes(1, 1) = 0;
    susceptibleNodesGraph(end + 1) = sum(susceptibleNodes, "all");
    infectiousNodesGraph(end + 1) = sum(infectiousNodes, "all");
    recoveredNodesGraph(end + 1) = sum(recoveredNodes, "all");

    while sum(susceptibleNodes) ~= 0 && sum(infectiousNodes) ~= 0
        for i = find(infectiousNodes)
```

```
            for s = find(susceptibleNodes)
                if U(i, s) == 1 && s ~= i && susceptibleNodes(1, s) == 1
                    infected = b*dt;
                    probinfected = rand/100;
                    if infected > probinfected
                        susceptibleNodes(1, s) = 0;
                        infectiousNodes(1, s) = 1;
                    end
                end
            end

            recovered = q;
            probrecovered = rand/10;
            if recovered > probrecovered
                infectiousNodes(1, i) = 0;
                recoveredNodes(1, i) = 1;
            end
            susceptibleNodesGraph(end + 1) = sum(susceptibleNodes, "all")
                ;
            infectiousNodesGraph(end + 1) = sum(infectiousNodes, "all");
            recoveredNodesGraph(end + 1) = sum(recoveredNodes, "all");
            disp("susceptible: " + sum(susceptibleNodes, "all") + "
                infected: " + sum(infectiousNodes, "all") + " recovered: "
                 + sum(recoveredNodes, "all"))
        end
    end

    disp(sum(recoveredNodes, "all")/length(U));
    plot(recoveredNodesGraph);
    hold on;
    plot(infectiousNodesGraph);
    plot(susceptibleNodesGraph);
    ylabel("Number of Nodes Affected");
    xlabel("Time");
    legend('Recovered Nodes', 'Infected Nodes', 'Susceptible Nodes');
    hold off;
end
```

7.2 SIRquestion3.m

```
function SIRquestion3(filename, k)
recoveredpercentage = zeros(1, 100);
    for numbersim = 1:100
        del = 10;
        b = 4 * 10^-4;
        u = del * b/k;
        dt = 5 * 10^-3 * 1/b;

        q = u * dt;

        load("-mat", filename, 'U');

        susceptibleNodes = ones(1, length(U));
        infectiousNodes = zeros(1, length(U));
        recoveredNodes = zeros(1, length(U));
```

```matlab

        susceptibleNodesGraph = [sum(susceptibleNodes, "all")];
        infectiousNodesGraph = [sum(infectiousNodes, "all")];
        recoveredNodesGraph = [sum(recoveredNodes, "all")];

        infectiousNodes(1, 1) = 1;
        susceptibleNodes(1, 1) = 0;
        susceptibleNodesGraph(end + 1) = sum(susceptibleNodes, "all");
        infectiousNodesGraph(end + 1) = sum(infectiousNodes, "all");
        recoveredNodesGraph(end + 1) = sum(recoveredNodes, "all");

        while sum(susceptibleNodes) ~= 0 && sum(infectiousNodes) ~= 0
            for i = find(infectiousNodes)
                for s = find(susceptibleNodes)
                    if U(i, s) == 1 && s ~= i && susceptibleNodes(1, s) == 1
                        infected = b*dt;
                        probinfected = rand/100;
                        if infected > probinfected
                            susceptibleNodes(1, s) = 0;
                            infectiousNodes(1, s) = 1;
                        end
                    end
                end

                recovered = q;
                probrecovered = rand/10;
                if recovered > probrecovered
                    infectiousNodes(1, i) = 0;
                    recoveredNodes(1, i) = 1;
                end
                susceptibleNodesGraph(end + 1) = sum(susceptibleNodes, "all");
                infectiousNodesGraph(end + 1) = sum(infectiousNodes, "all");
                recoveredNodesGraph(end + 1) = sum(recoveredNodes, "all");
                disp("susceptible: " + sum(susceptibleNodes, "all") + " infected: " + sum(infectiousNodes, "all") + " recovered: " + sum(recoveredNodes, "all"))
            end
        end

        disp(sum(recoveredNodes, "all")/length(U));
        recoveredpercentage(1, numbersim) = sum(recoveredNodes, "all")/length(U);
    end
    disp(recoveredpercentage);
    histogram(recoveredpercentage, 'BinWidth', 0.01);
    ytickformat('percentage');
    ylabel("Percentage of Simulations");
    xlabel("Percentage of Recovered Nodes");
    hold off;
end
```

7.3 SIRquestion4.m

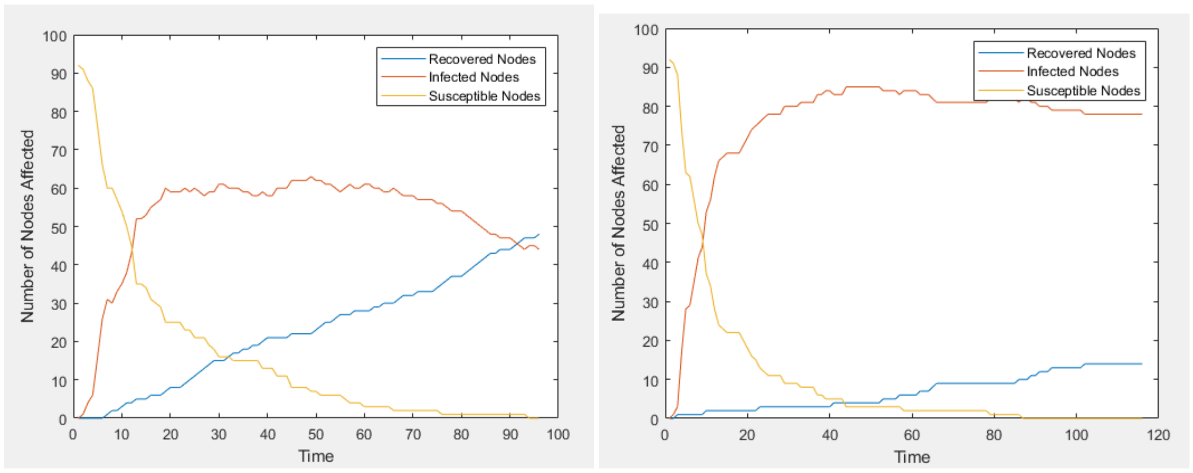
```matlab
function SIRquestion4()
    filenames = ["A_InVS13.mat", "A_InVS15.mat", "A_LH10.mat", "A_LyonSchool.mat", "A_SFHH.mat", "A_Thiers13.mat"];

    for filenameindex = 1:6
        x = zeros(1, 5);
        y = zeros(1, 5);
        for k = 1:5
            recoveredpercentage = zeros(1, 100);
            for numbersim = 1:100
                del = 10;
                b = 4 * 10^-4;
                u = del * b/k;
                dt = 5 * 10^-3 * 1/b;

                q = u * dt;

                load("-mat", filenames(filenameindex), 'U');

                susceptibleNodes = ones(1, length(U));
                infectiousNodes = zeros(1, length(U));
                recoveredNodes = zeros(1, length(U));

                susceptibleNodesGraph = [sum(susceptibleNodes, "all")];
                infectiousNodesGraph = [sum(infectiousNodes, "all")];
                recoveredNodesGraph = [sum(recoveredNodes, "all")];

                infectiousNodes(1, 1) = 1;
                susceptibleNodes(1, 1) = 0;
                susceptibleNodesGraph(end + 1) = sum(susceptibleNodes, "all");
                infectiousNodesGraph(end + 1) = sum(infectiousNodes, "all");
                recoveredNodesGraph(end + 1) = sum(recoveredNodes, "all");

                while sum(susceptibleNodes) ~= 0 && sum(infectiousNodes) ~= 0
                    for i = find(infectiousNodes)
                        for s = find(susceptibleNodes)
                            if U(i, s) == 1 && s ~= i && susceptibleNodes(1, s) == 1
                                infected = b*dt;
                                probinfected = rand/100;
                                if infected > probinfected
                                    susceptibleNodes(1, s) = 0;
                                    infectiousNodes(1, s) = 1;
                                end
                            end
                        end

                        recovered = q;
                        probrecovered = rand/10;
                        if recovered > probrecovered
                            infectiousNodes(1, i) = 0;
                            recoveredNodes(1, i) = 1;
                        end
                        susceptibleNodesGraph(end + 1) = sum(susceptibleNodes, "all");
                        infectiousNodesGraph(end + 1) = sum(infectiousNodes, "all");
                        recoveredNodesGraph(end + 1) = sum(recoveredNodes, "all");
                        disp("susceptible: " + sum(susceptibleNodes, "all") + " infected: " + sum(infectiousNodes, "all") + " recovered: " + sum(recoveredNodes, "all"))
                    end
                end

                disp(sum(recoveredNodes, "all")/length(U));
                if sum(recoveredNodes, "all")/length(U) >= 0.2
                    recoveredpercentage(1, numbersim) = sum(recoveredNodes, "all")/length(U);
                end
            end
            recoveredpercentage = nonzeros(recoveredpercentage');
            x(1, k) = k*mean(sum(U, 1))/del;
            y(1, k) = length(recoveredpercentage)/100;
        end
        plot(x, y, '-o'); hold on;
    end
    ytickformat('percentage');
    ylabel("Percentage of Simulations (Recovered Nodes >= 20%)");
    xlabel(" 0  = kd/  ");
    legend('A\_InVS13', 'A\_InVS15', 'A\_LH10', 'A\_LyonSchool', 'A\_SFHH', 'A\_Thiers13');
    hold off;
end
```

7.4 SIRquestion5.m

```
function SIRquestion5()
    filenames = ["A_InVS13.mat", "A_InVS15.mat", "A_LH10.mat", "A_LyonSchool.mat", "A_SFHH.mat", "A_Thiers13.mat"];

    for filenameindex = 1:6
        x = zeros(1, 5);
        y = zeros(1, 5);
        stddev = zeros(1, 5);
        for k = 1:5
            recoveredpercentage = zeros(1, 100);
            for numbersim = 1:100
                del = 10;
                b = 4 * 10^-4;
                u = del * b/k;
                dt = 5 * 10^-3 * 1/b;

                q = u * dt;

```

```
load("-mat", filenames(filenameindex), 'U');

susceptibleNodes = ones(1, length(U));
infectiousNodes = zeros(1, length(U));
recoveredNodes = zeros(1, length(U));

susceptibleNodesGraph = [sum(susceptibleNodes, "all")];
infectiousNodesGraph = [sum(infectiousNodes, "all")];
recoveredNodesGraph = [sum(recoveredNodes, "all")];

infectiousNodes(1, 1) = 1;
susceptibleNodes(1, 1) = 0;
susceptibleNodesGraph(end + 1) = sum(susceptibleNodes, "all");
infectiousNodesGraph(end + 1) = sum(infectiousNodes, "all");
recoveredNodesGraph(end + 1) = sum(recoveredNodes, "all");

while sum(susceptibleNodes) ~= 0 && sum(infectiousNodes) ~= 0
    for i = find(infectiousNodes)
        for s = find(susceptibleNodes)
            if U(i, s) == 1 && s ~= i && susceptibleNodes(1, s) == 1
                infected = b*dt;
                probinfected = rand/100;
                if infected > probinfected
                    susceptibleNodes(1, s) = 0;
                    infectiousNodes(1, s) = 1;
                end
            end
        end

        recovered = q;
        probrecovered = rand/10;
        if recovered > probrecovered
            infectiousNodes(1, i) = 0;
            recoveredNodes(1, i) = 1;
        end
        susceptibleNodesGraph(end + 1) = sum(susceptibleNodes, "all");
        infectiousNodesGraph(end + 1) = sum(infectiousNodes, "all");
        recoveredNodesGraph(end + 1) = sum(recoveredNodes, "all");
        disp("susceptible: " + sum(susceptibleNodes, "all") + " infected: " + sum(infectiousNodes, "all") + " recovered: " + sum(recoveredNodes, "all"))
    end
end

disp(sum(recoveredNodes, "all")/length(U));
if sum(recoveredNodes, "all")/length(U) >= 0.2
    recoveredpercentage(1, numbersim) = sum(
                        recoveredNodes, "all")/length(U);
                end
            end
            recoveredpercentage = nonzeros(recoveredpercentage');
            x(1, k) = k*mean(sum(U, 1))/del;
            y(1, k) = mean(recoveredpercentage);
            stddev(1, k) = std(recoveredpercentage);
        end
        errorbar(x, y, stddev); hold on;
    end
    ytickformat('percentage');
    ylabel("Percentage of Recovered Nodes >= 20%");
    xlabel(" 0  = kd/  ");
    legend('A\_InVS13', 'A\_InVS15', 'A\_LH10', 'A\_LyonSchool', 'A\_SFHH
        ', 'A\_Thiers13');
    hold off;
end
```

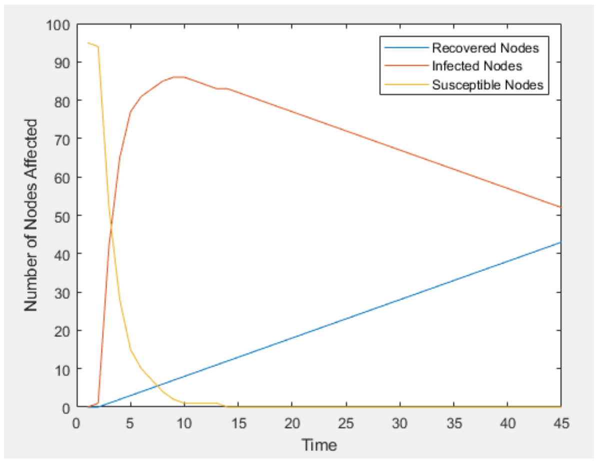
8 SIR Models



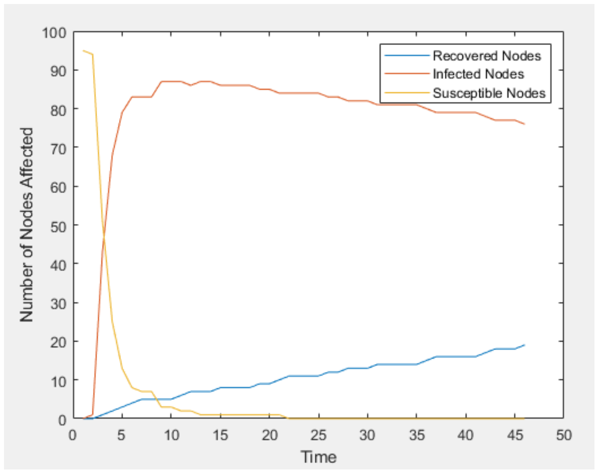
(a) k = 1

(b) k = 5

Figure 3: lnVS13 Face-to-face SIR Model

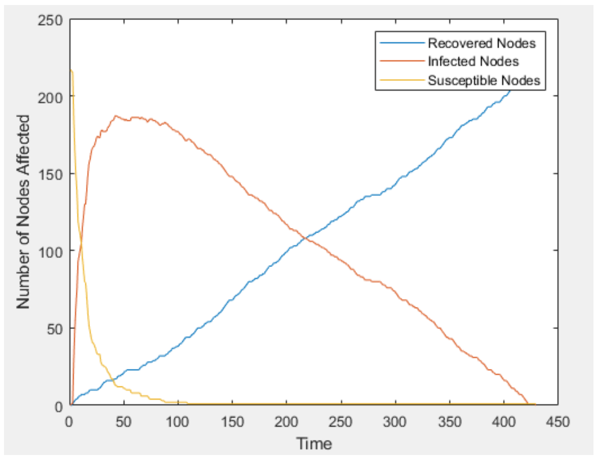


(a) k = 1

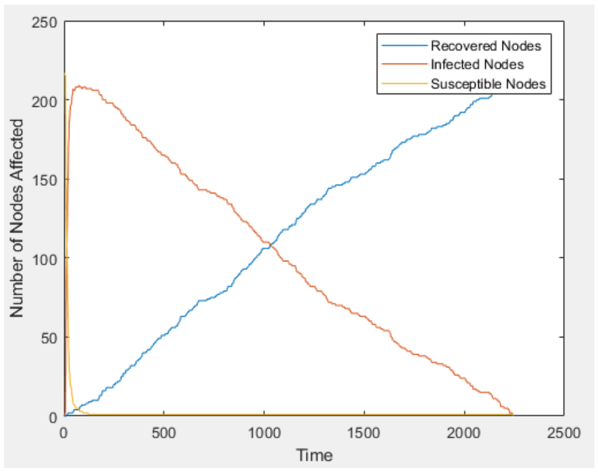


(b) k = 5

Figure 4: lnVS13 Co-presence SIR Model

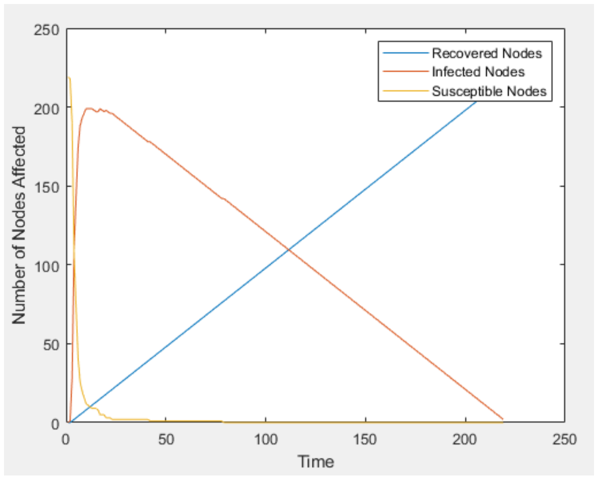


(a) k = 1

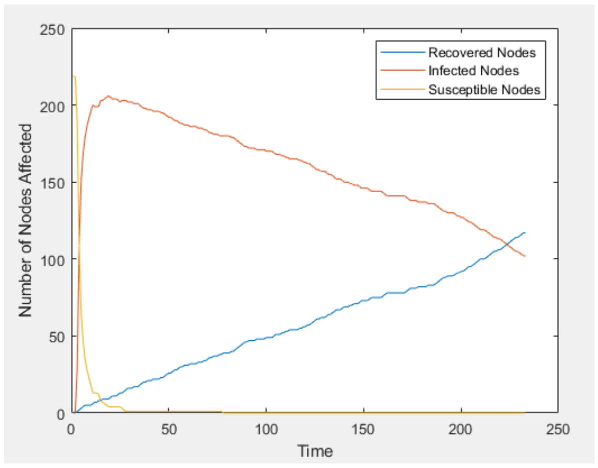


(b) k = 5

Figure 5: InVS15 Face-to-face SIR Model

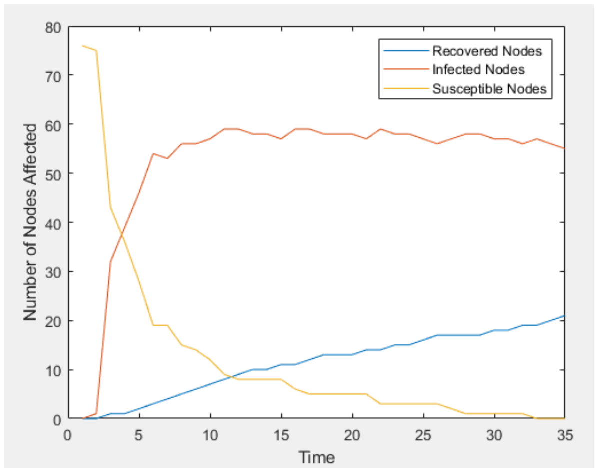


(a) k = 1

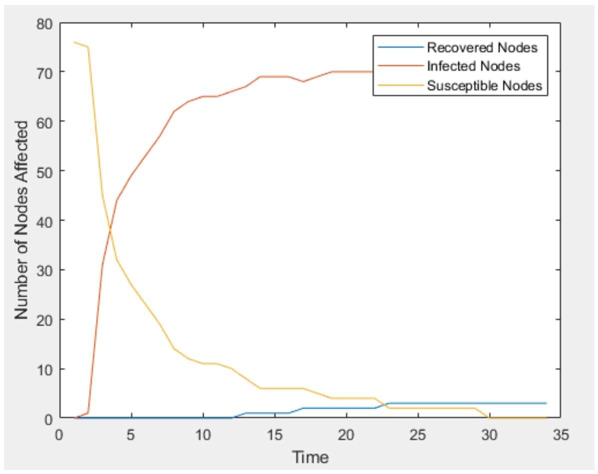


(b) k = 5

Figure 6: InVS15 Co-presence SIR Model

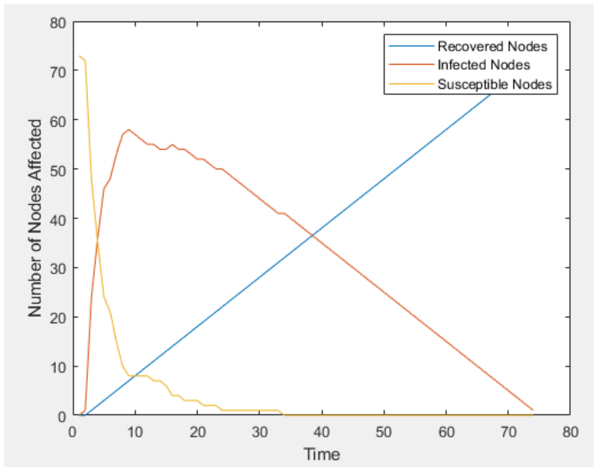


(a) k = 1

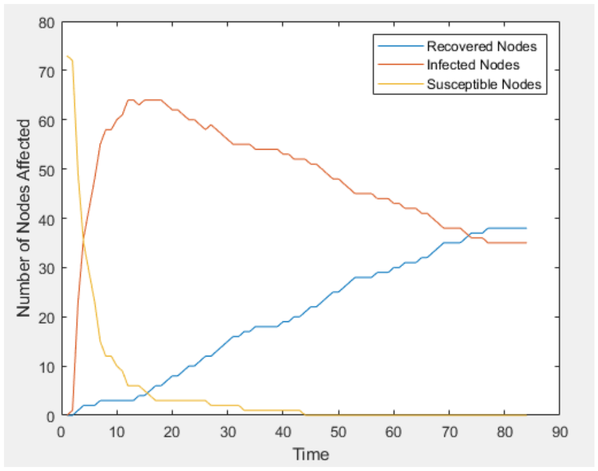


(b) k = 5

Figure 7: LH10 Face-to-face SIR Model

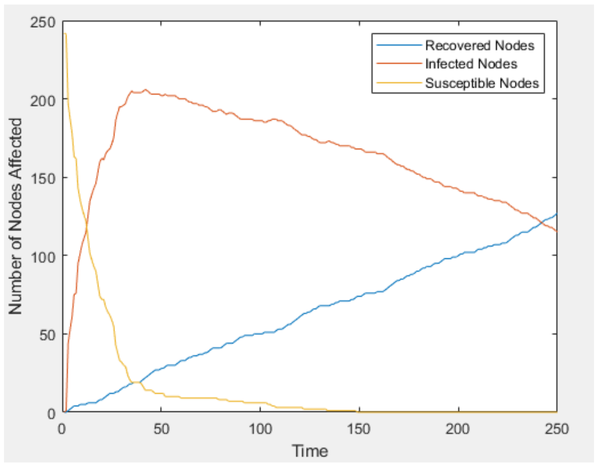


(a) k = 1

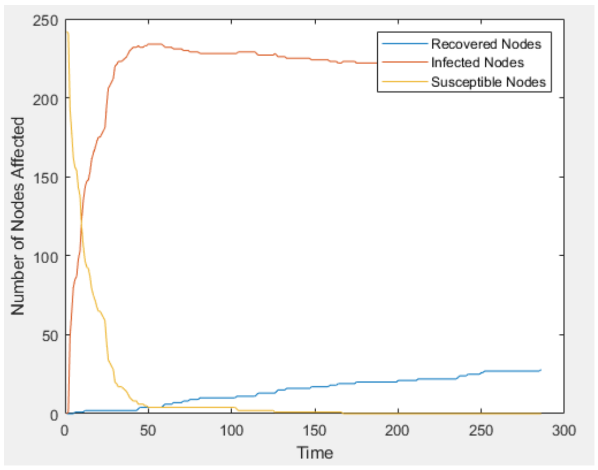


(b) k = 5

Figure 8: LH10 Co-presence SIR Model

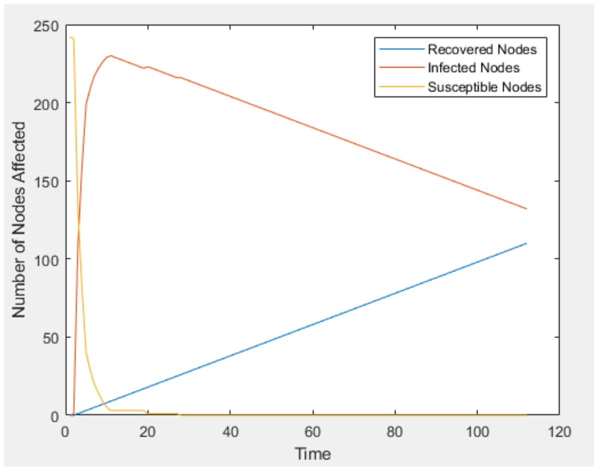


(a) k = 1

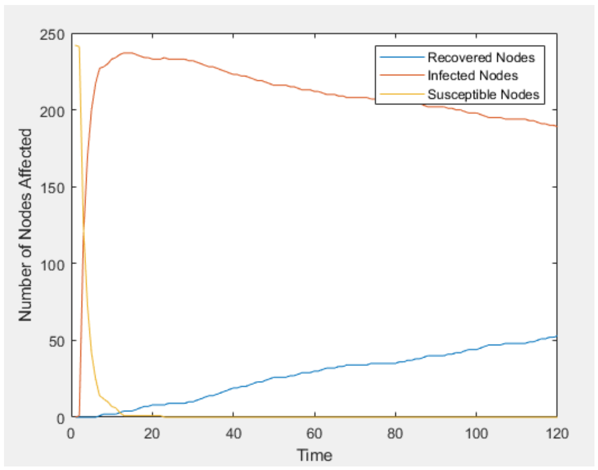


(b) k = 5

Figure 9: LyonSchool Face-to-face SIR Model

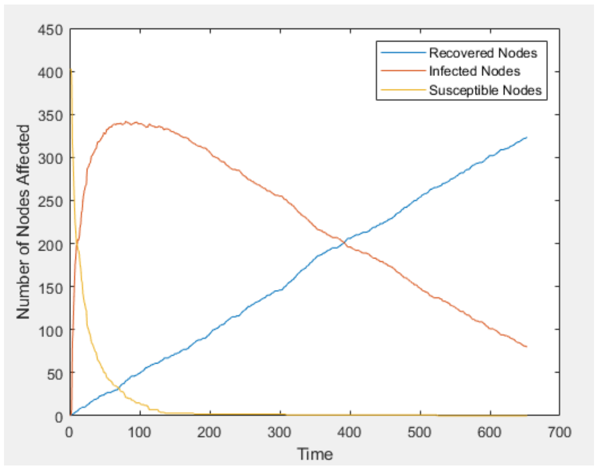


(a) k = 1

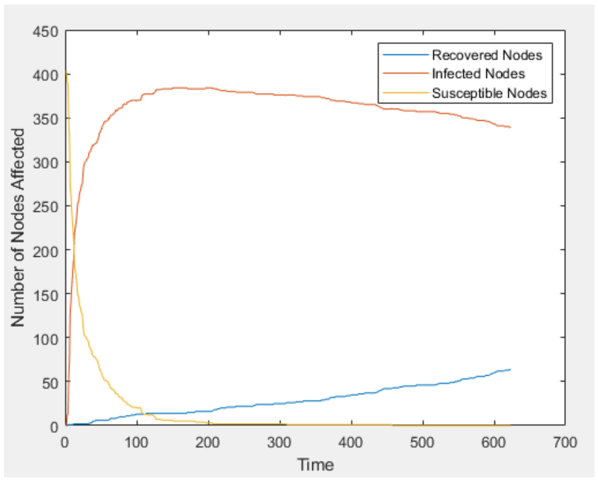


(b) k = 5

Figure 10: LyonSchool Co-presence SIR Model

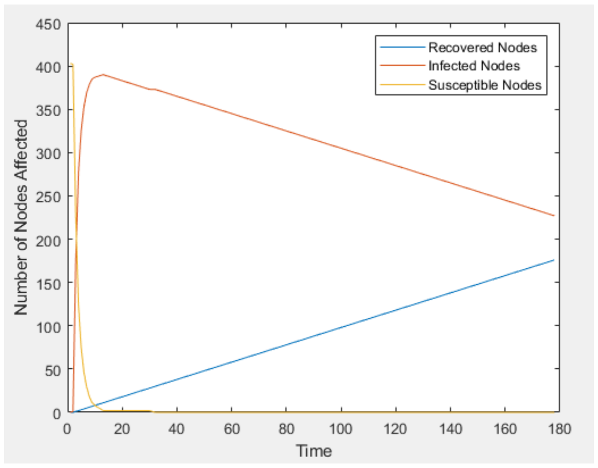


(a) k = 1

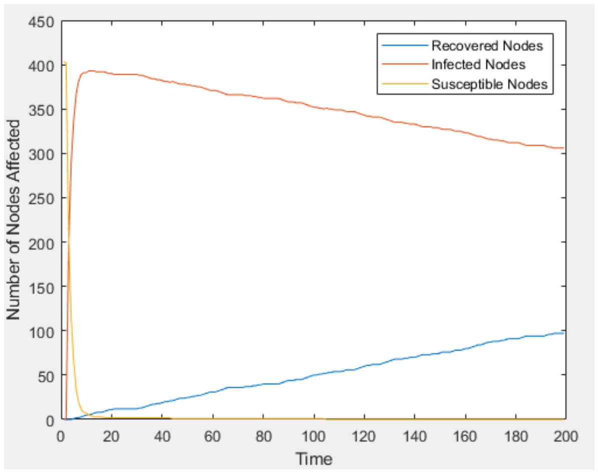


(b) k = 5

Figure 11: SFHH Face-to-face SIR Model

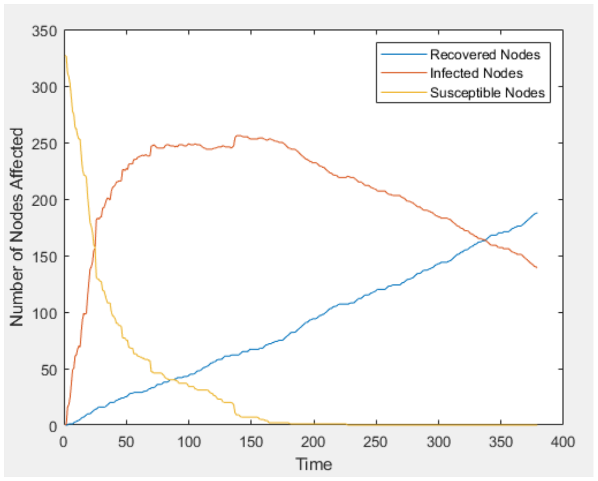


(a) k = 1

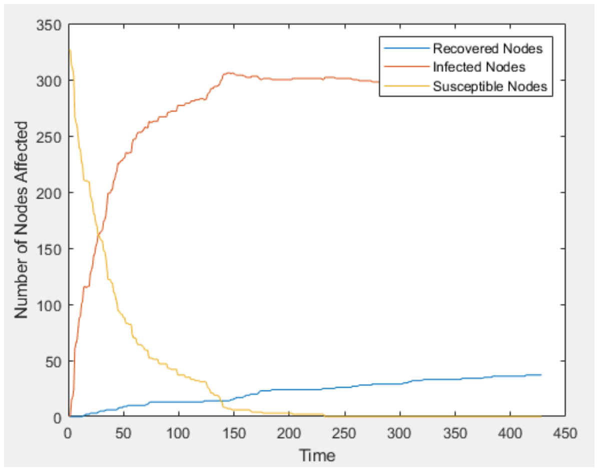


(b) k = 5

Figure 12: SFHH Co-presence SIR Model

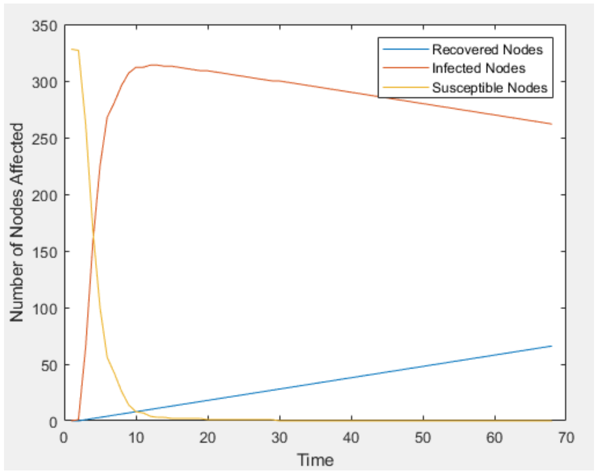


(a) k = 1

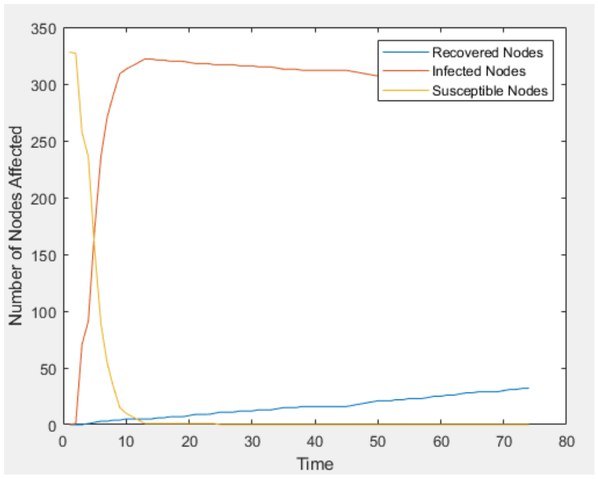


(b) k = 5

Figure 13: Thiers13 Face-to-face SIR Model



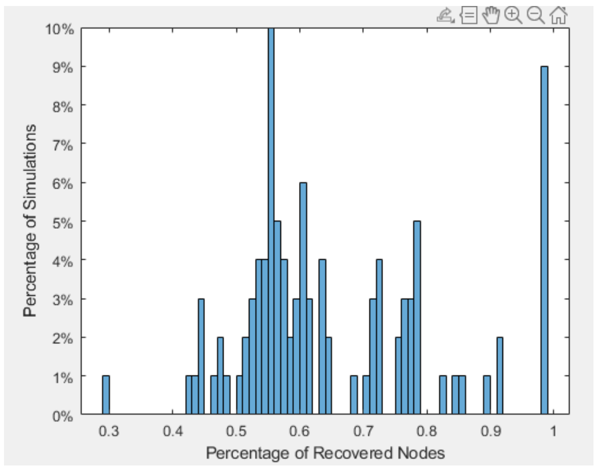
(a) k = 1



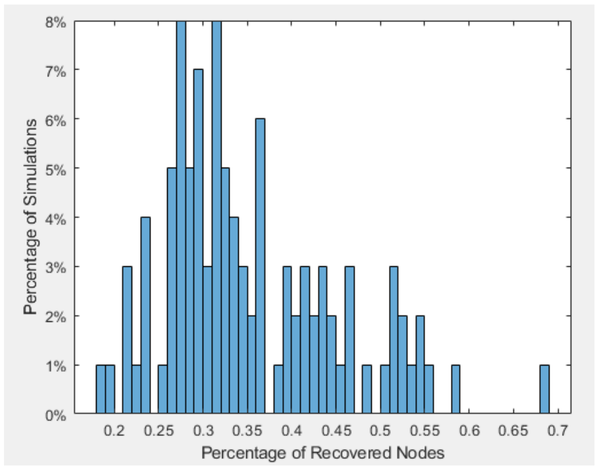
(b) k = 5

Figure 14: Thiers13 Co-presence SIR Model

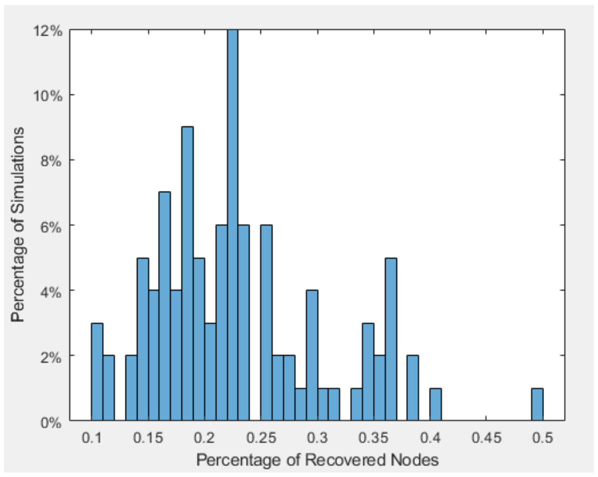
9 Recovered Node Distribution Plots



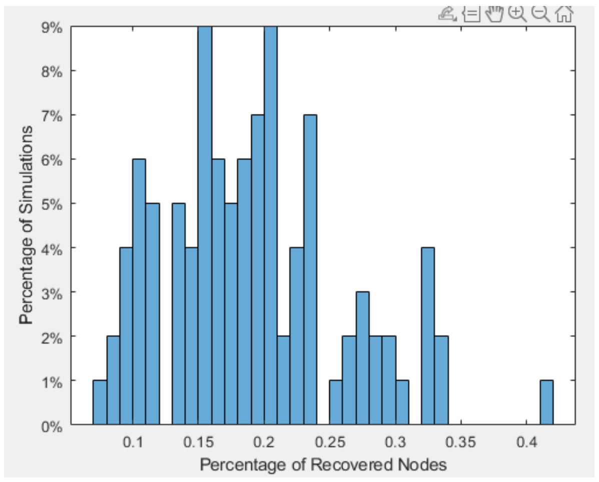
(a) k = 1



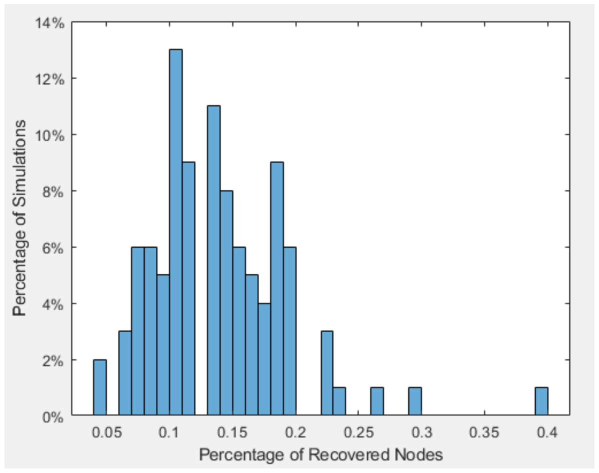
(b) k = 2



(c) k = 3

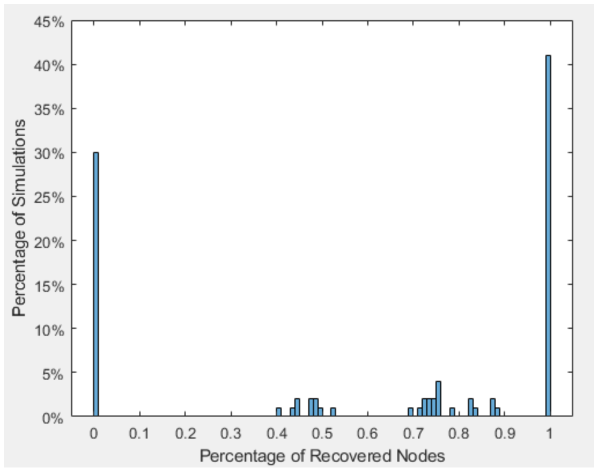


(d) k = 4

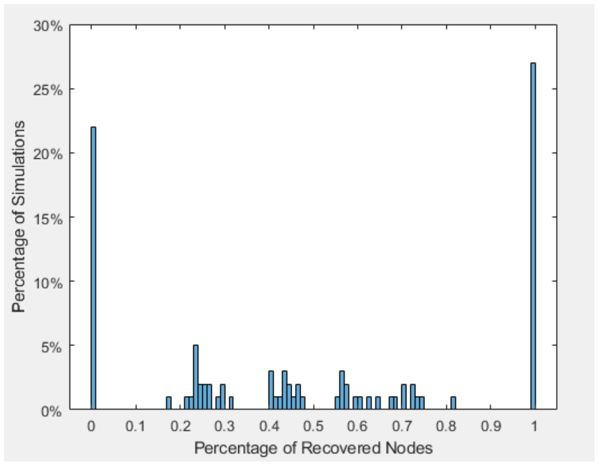


(e) k = 5

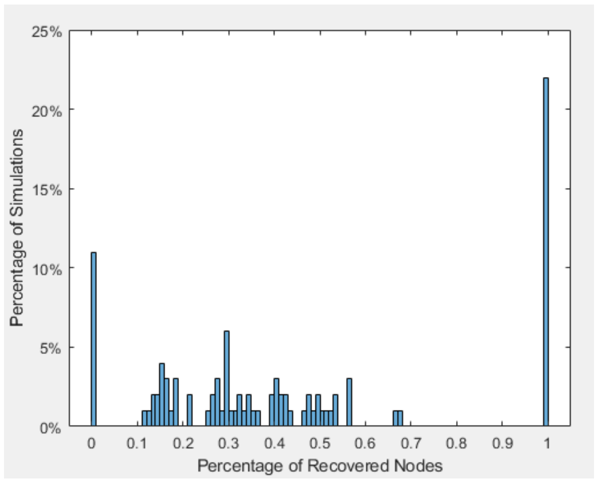
Figure 15: InVS13 Recovered Node Distribution



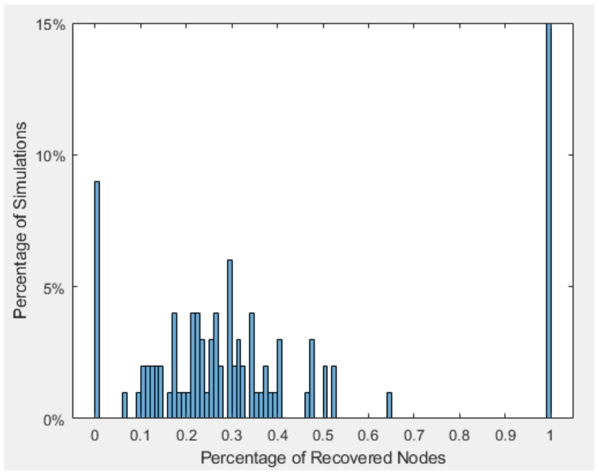
(a) k = 1



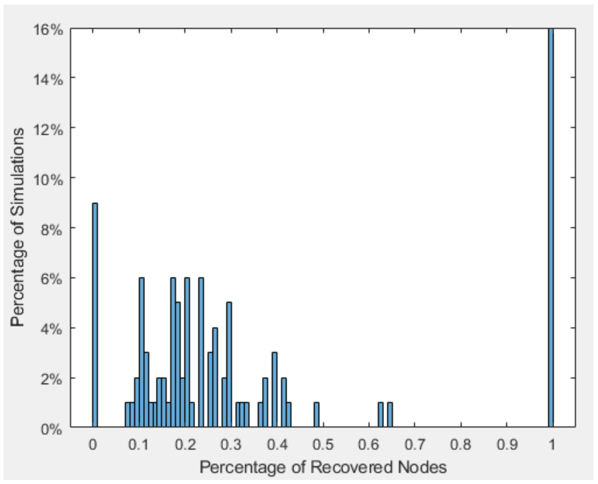
(b) k = 2



(c) k = 3

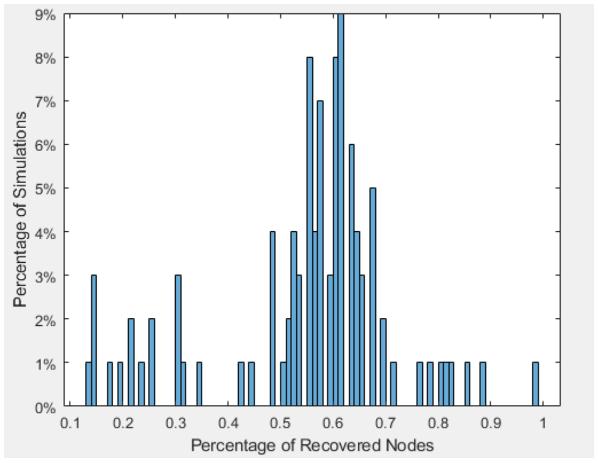


(d) k = 4

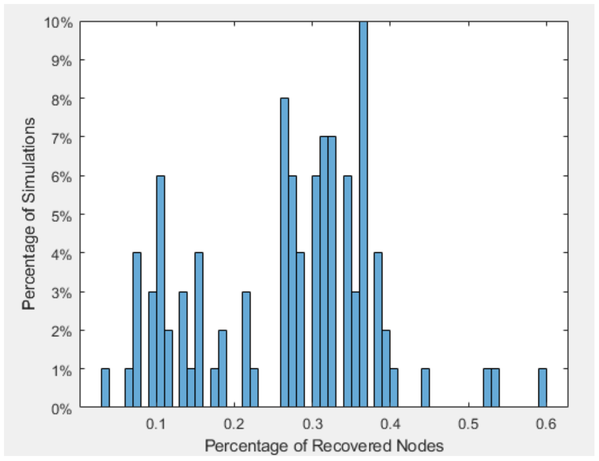


(e) k = 5

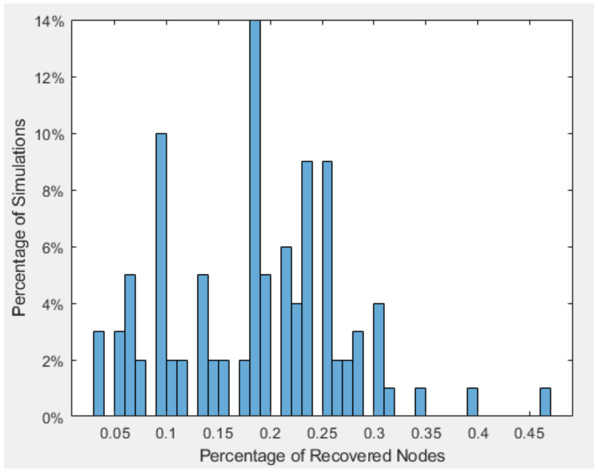
Figure 16: InVS15 Recovered Node Distribution



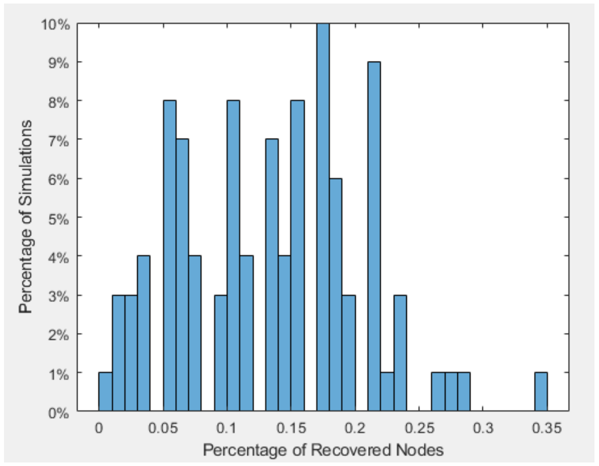
(a) k = 1



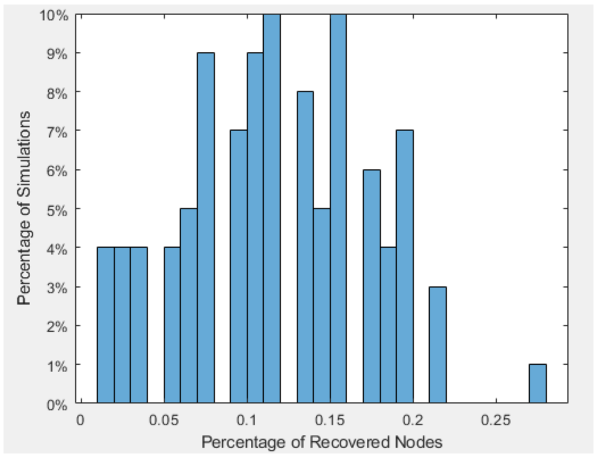
(b) k = 2



(c) k = 3

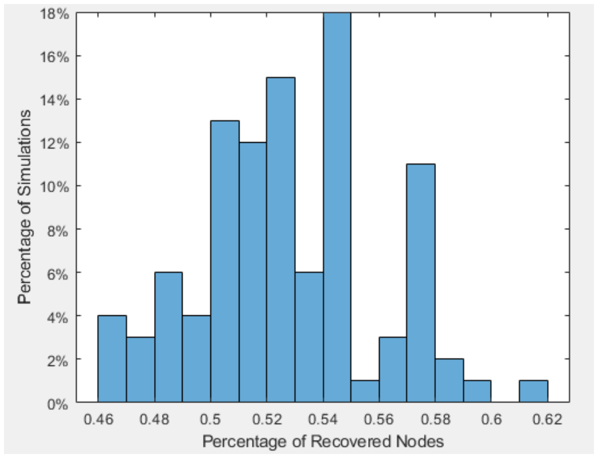


(d) k = 4

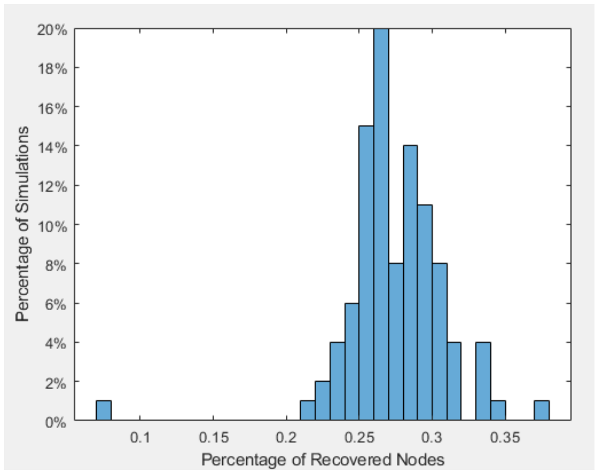


(e) k = 5

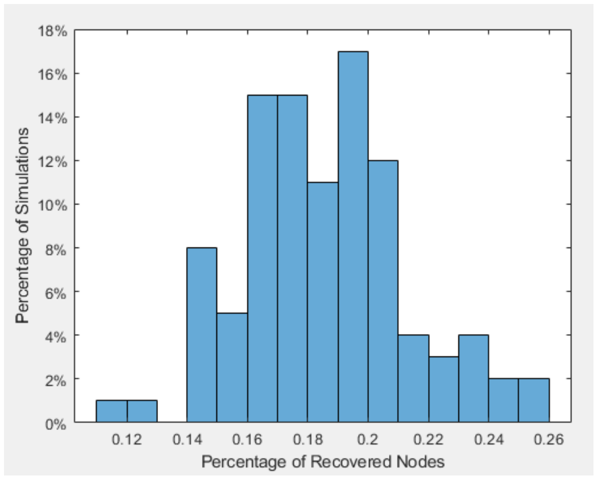
Figure 17: LH10 Recovered Node Distribution



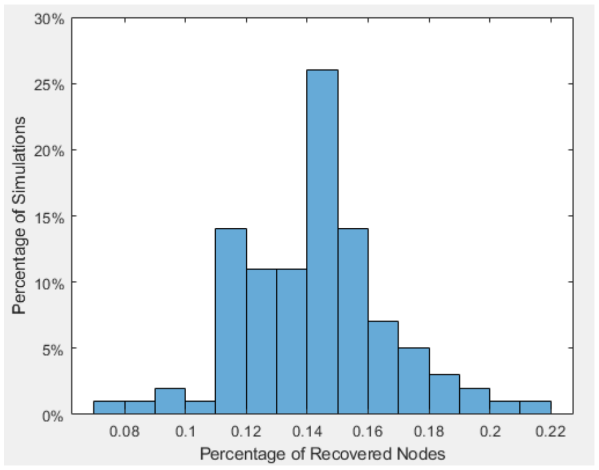
(a) k = 1



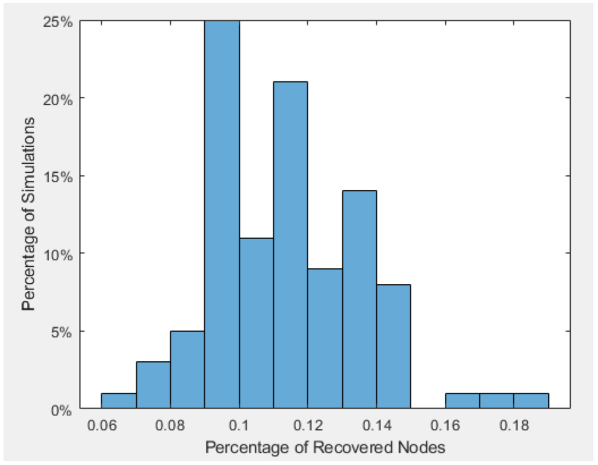
(b) k = 2



(c) k = 3

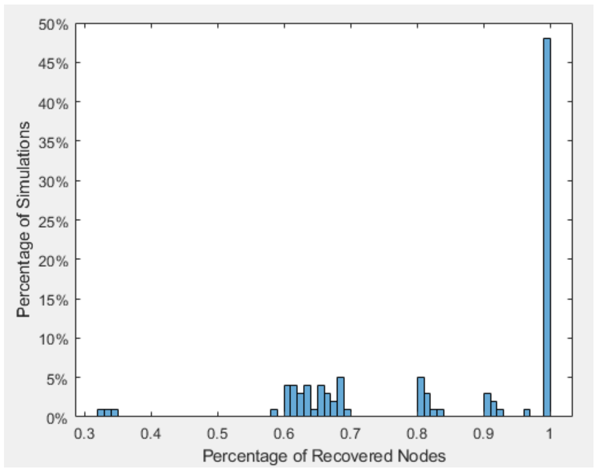


(d) k = 4

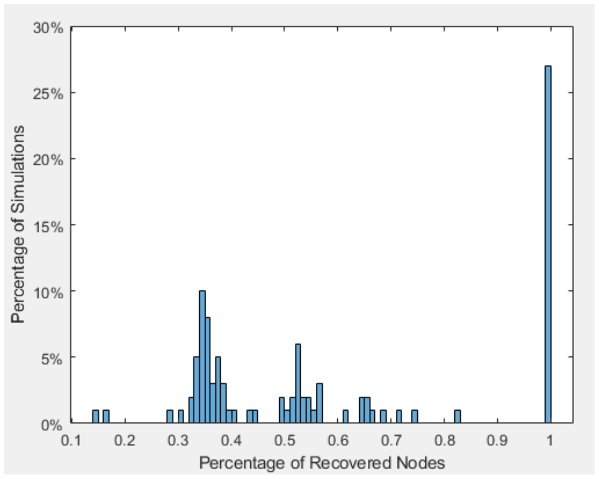


(e) k = 5

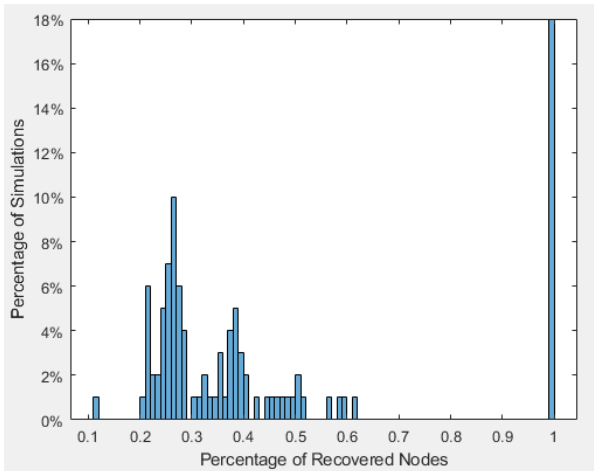
Figure 18: LyonSchool Recovered Node Distribution



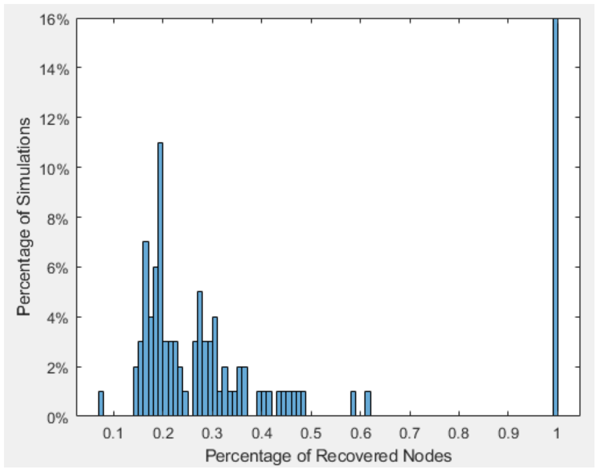
(a) k = 1



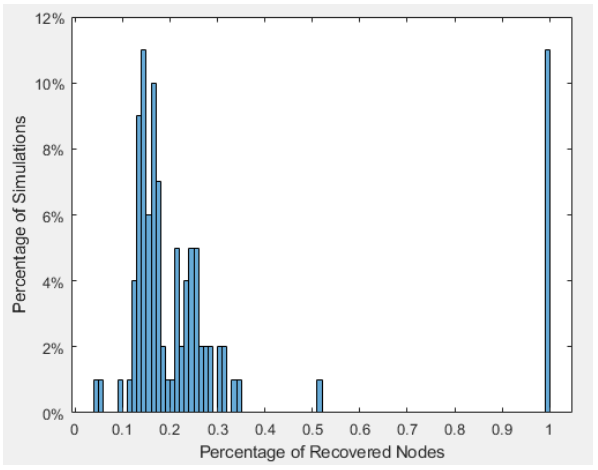
(b) k = 2



(c) k = 3

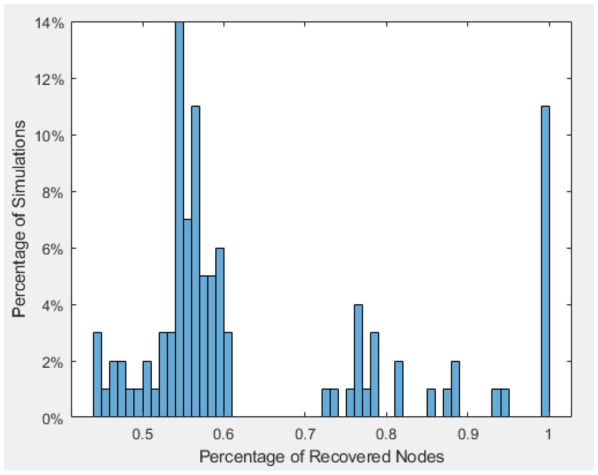


(d) k = 4

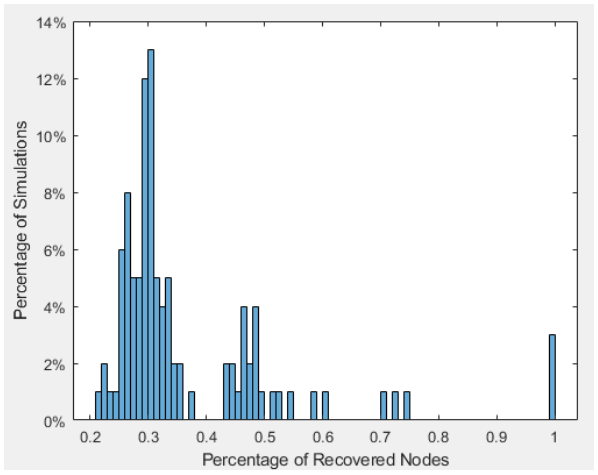


(e) k = 5

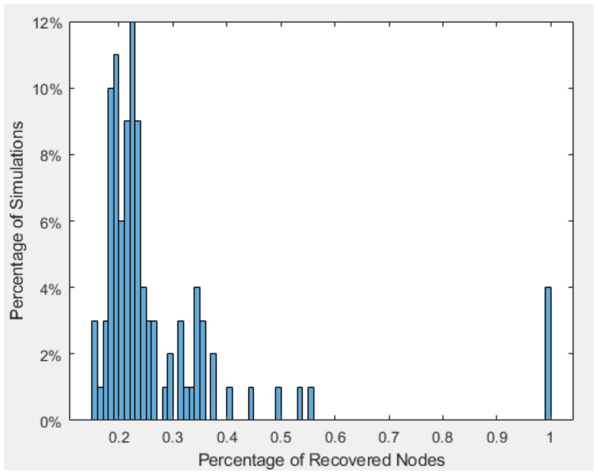
Figure 19: SFHH Recovered Node Distribution



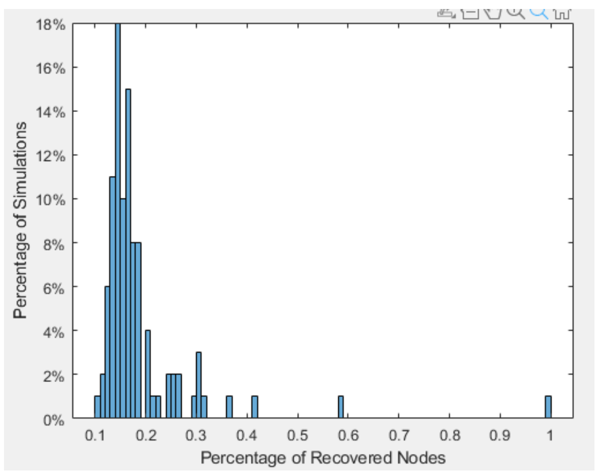
(a) k = 1



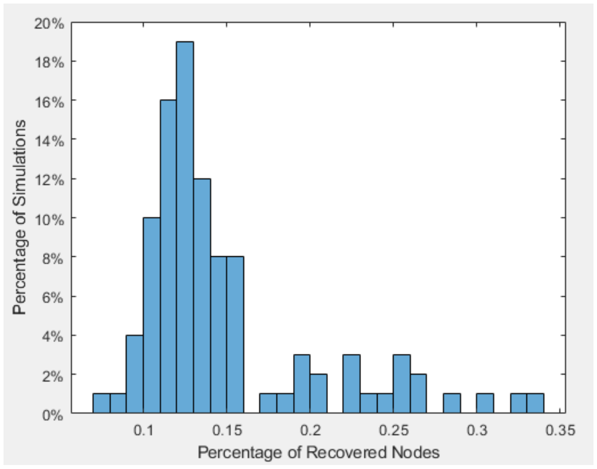
(b) k = 2



(c) k = 3



(d) k = 4



(e) k = 5

Figure 20: Thiers13 Recovered Node Distribution