SWE580- Fall 2021

Final Project Report
Alternative Approach on Identification of Influential Spreaders in Complex Networks

Osman Selçuk Sarıoğlu

2021-06-30

Abstract

The knowledge of the spreading pathways through the network of social interactions is crucial for developing efficient methods to either hinder spreading in the case of diseases, or accelerate spreading in the case of information dissemination [1]. To enhance performance on spreading and less costly way to transfer information, it's better to find nodes having most influence in the network. Maksim Kitsak showed that the most efficient spreaders are those located within the core of the network as identified by the k-shell decomposition analysis, on the contrary to the general expectation that most connected parties are supposed to have most influence.

In this project paper, it will be analyzed to find an alternative approach to find less costly and more effective influencers by utilizing the k-shell approach and number of connection together.

Keywords - efficient influential spreaders, k-shell decomposition, betweenness centrality, complex network

1 Introduction

For the networks having a broad number of links between nodes, it is expected that the most connected nodes are the key players. These nodes, called as hub, play important role in process of large scale spreading in network. This leads to an understanding of prioritizing hub with highest degree (k) in spreading process. Moreover, in social networks, it's believed that spreading is associated with the betweenness centrality (c_b) because of having more 'interpersonal influence' on members of social network [1].

In his study [1], Maksim Kitsak showed that the most efficient spreaders are those located within the core of the network as identified by the k-shell (k_s) decomposition analysis, on the contrary to the general expectation that most connected parties are supposed to have most influence. Kitsak applied the susceptible–infectious–recovered (SIR) and susceptible–infectious–susceptible (SIS) models to study the spreading process, and compared the results on average infection rate (R) through all network. During the study, it's applied various infection probabilities (β)

In this project paper, it's analyzed to find an alternative approach to find less costly and more effective influencers by utilizing the k-shell (k_s) and number of connection together (k). I applied SIR as spreading process with various probabilities of infection (β). I analyzed k_s and k to find an alternative parameter, which is derived from (k_s) and (k)values.

My initial intenstion was to test my data with same networks Kitsak used in his study [1]: the friendship network between the members of the LiveJournal.com community, the network of actors who have

costarred in movies labelled by imdb.com to compare my outputs with the available k_s results. However, after making several test runs, due to computational constraints, I had to generate my own network (G_{gen}). This network consists of 1000 nodes, and 38,168 edges, and it has 76.33 as average degree (k_M).

Kitsak showed in his study that highest k_s values have more effect on influence the network more than other parameters such has degree or betweenness. However, as expected, reaching these nodes in high k_s might be more costly (C_{k_s}) for any other nodes in outer shells.

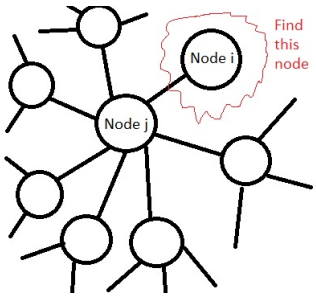


Figure 1: Objective is to find node i having highest n_{si}

Neighbor nodes having lower connection may influence these core nodes, less costly, and can have similar infection affect like high k_s ones. Therefore, I assume to find a relation between k and k_s to inspect the influencer low k-valued neighbors. After my study, I derived a new attribute for nodes in G_{gen}. This attribute is called n-shell (n_s) and it mentions the strength of motivation to spend extra cost for this node, illustrated in Figure 1, to achieve a similar output from a core node.

2 Method for Study

At this study, I used Python 3 for coding network analyzing and programming tools. I intensely used NetworkX library [2] for graph analysis. I coded several python files to run algorithms for generating graph, k-shell decomposition, n-shell decomposition, SIR spread, and plotting for results. Relevant python files are in attachment section of the report.

2.1 Generating Graph

As mentioned in the introduction section, due to computational constraints, I had to generate my own network (G_{gen}) instead of using networks mentioned at Kitsak's study. I used Python's NetworkX library [2], and applied an extended Barabási–Albert model graph. This is a built-in functionality of NetworkX. With this function, I generated random graphs, which are constructed using preferential attachment. This model allows new edges, rewired edges or new nodes. It constructs graph based on the probabilities p and q with $p+q<1$. The growing behavior of the graph is as follows:

- With p probability, new edges are added to the graph, starting from randomly chosen existing nodes and attached preferentially at the other end.
- With q probability, existing edges are rewired by randomly choosing an edge and rewiring one end to a preferentially chosen node.
- With $1-p-q$ probability, new nodes are added to the graph with edges attached preferentially.

By applying this algorithm into my generator code, I constructed 10 different graphs by using different values for p, q and number of edges to connect randomly at time. The python file to generate the graphs is given in Attachment 1.

I evaluated these graphs based on intensity of connections and number of shells to the core of the graph. Table 1 shows summary of information about graphs generated. From this list of graph, number of cores (k_s) and average degree (k_M) is the highest for G_{gen_6}. Therefore, I selected to run my analysis on G_{gen_6} for the further part of the study.

Graph	N	E	k_s	k_M
G_{gen_1}	1,000	9,921	20	19.84
G_{gen_2}	10,000	33,902	11	6.78
G_{gen_3}	20,000	109,386	14	10.93
G_{gen_4}	100	558	12	11.16
G_{gen_5}	1,000	7,570	19	15.14
G_{gen_6}	1,000	38,168	105	76.33
G_{gen_7}	500	6,080	32	24.32
G_{gen_8}	1,000	8,070	20	16.14
G_{gen_9}	1,000	16,224	47	32.44
$G_{gen_{10}}$	2,000	16,835	26	16.83

Table 1: Graphs generated with various combinations of N, p and q

2.2 K-shell Decomposition

In order to calculate k_s values of the nodes in G_{gen_6}, I wrote a code in Python by applying the algorithm for k-shell composition detailed in Algorithm 1 [3]. This code file is provided as Attachment 2.

Algorithm 1: Algorithm for defining k-shell (k_s) value of a node

Result: K-shell (k_s) for $G(V, E)$ undirected network
$C \leftarrow V$;
$s \leftarrow 0$;
$V_s = \emptyset$;
while $C \neq \emptyset$ **do**
  while *There are vertices of degree s in C* **do**
    Select i with degree s in C;
    $C \leftarrow C - \{i\}$;
    $V_s = V_s \cup \{i\}$;
    Remove all edges of i;
    Update the degrees in C;
  end
  $s \leftarrow s + 1$;
  $V_s = \emptyset$;
end
$s : V \mapsto \mathbb{N}$;
$i \mapsto k_{si}$;

After decomposing shells of the graph, nodes are divided into 105 different shells. Table 2 demonstrates the number nodes in different shells. Since I used my own generated graph for analysis, I couldn't compare results for Kitsak's results. This is why I run my own results for k_s against c_b and k in order to measure performance after spreading algorithm.

k_s Range	Number of N
01-10	186
10-20	161
20-30	110
30-80	334
80-100	65
101	1
102	4
103	1
104	1
105	137

Table 2: Number of nodes in different k_s ranges

2.3 Spreading Algorithm for Susceptible – Infectious – Recovered (SIR)

I applied the susceptible – infectious – recovered (SIR) algorithm to study the spreading process. This model can be describe as disease spreading, or information and rumor spreading in social processes. In this algorithm, we define a probability (p) that an infectious node will infect a susceptible neighbor. The value of the probability is important during the analysis. If you apply large values, there is a risk of spreading to large portion of population, so you cannot observe a significant difference regarding origin of the spread. Therefore, I simulated by study with small values as p:{ 0.005, 0.01, 0.015, 0.025, 0.05 }.

Since this model is stochastic model, I applied the algorithm for each case in 100 runs. I construct a python code for this iterative run. The python file for the algorithm is provide in Attachment 3.

The model I studied assumes that infected nodes are recovered after the cycle that it interacts within the neighbors once,

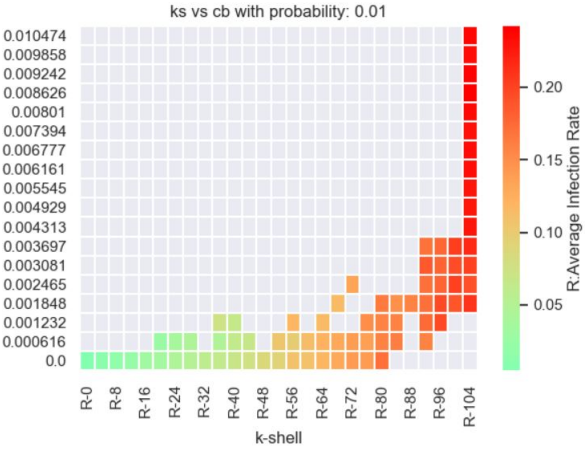
and spread is triggered from a single node (i). I didn't apply a probabilistic approach on recovery, so if a node i is infected, then it will be recovered eventually. Algorithm 2 illustrates the flow of the process.

Algorithm 2: Spreading algorithm of Susceptible – Infectious – Recovered (SIR)

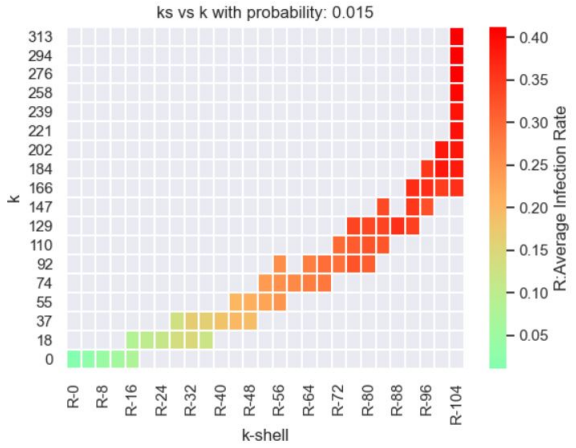
Result: SIR for $G(V, E)$ undirected network

Let $i \in V$;
I: Set of current spreaders;
R: Set of recovered;
$N(i)$: Set of neighbors of i;
β: Probability of spreading infection;
$I = \{i\}$, $R = \emptyset$;
while $I \neq \emptyset$ **do**
  Select $j \in V$;
  $I \leftarrow I - \{\mathrm{j}\}$;
  $R = R \cup \{\mathrm{j}\}$;
  $C = N(j) \setminus (I \cup R)$;
  while $C \neq \emptyset$ **do**
    Select $l \in C$;
    $C \leftarrow C - \{\mathrm{l}\}$;
    Infect l with probability of β;
    if l *is infected* **then**
      $I \leftarrow I \cup \{\mathrm{l}\}$;
    end
  end
end
Spread Ratio (r) is ratio of recovered nodes R againts all nodes in V ;

After I run the code for 100 iterations, I could simulated similar outputs that Kitsak presented his study [1] with my generated graph . Considering 5 different probabilities (p:{ 0.005, 0.01, 0.015, 0.025, 0.05 }), I received the similar result. There are significant improvement on infection rate when k_s increased, while it doesn't have significance impact on performance when you increase the c_b or k values in same k_s. Figure 2 has a sample illustration of results achieved. Figures for other p values are provided in Attachment 4.



(a) c_b against k_s at $p = 0.01$



(b) k against k_s at $p = 0.015$

Figure 2: Comparison of spreading performances

2.4 Algorithm for n-shell decomposition

Since the graph I studied provided similar results with Kitsak paper, then I focus on the question of finding efficient spreader interrelated with neighbors having highest k_s values. I used a attribute ***neighborhood-to-shell*** (n-shell) to differentiate nodes from each other. I represent this value as n_{si}, n-shell value of node i.

To compute the n-shell, there are three critical parameters: k-shell, degree of node, and number of nodes that has a common relationship. I initially set all n_s equal to k_s, so all the nodes should have at least a value equal to its k-shell value.

As illustrated at Figure 1, main objective is to find nodes which are located to strategically closer to core nodes. As explained by Kitsak, highest k_s values have better impact on spreading than highest degree ones. Therefore, neighbors of these highest k_s nodes are potential. Since I focus on cost effective spreaders, from these neighbors I spotlighted the ones having lowest k values. Here, there is another factor: how many nodes sharing the same highest k_s valued node. I formulated the relationship of these parameters as follows for node i which is the neighbor of node j:

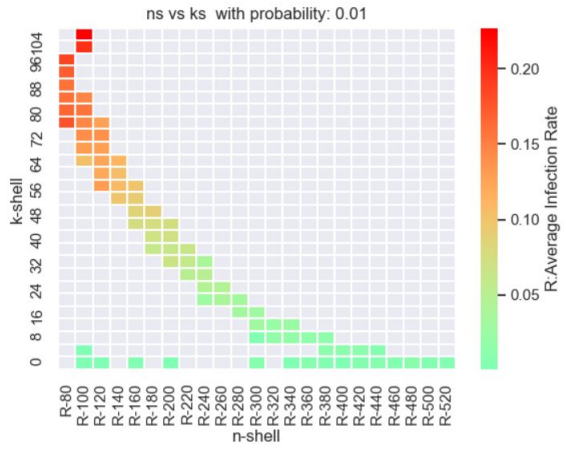
$$n_{si} = \ln(\frac{k_j}{k_i})k_{sj}$$

Algorithm 3: Algorithm for assigning n_s for each node i

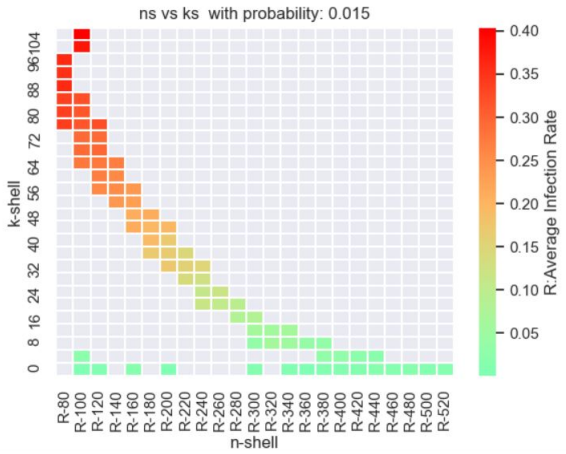
Result: n_s for $G(V, E)$
Let $i \in V$;
$N(i)$: Set of neighbors of i;
$n_{si} \leftarrow k_{si} : \forall i \in V$;
$C = V$;
while $C \neq \emptyset$ **do**
  Select $i \in C$;
  $C \leftarrow C - \{\text{i}\}$;
  $A = N(j)$;
  while $A \neq \emptyset$ **do**
    Select $j \in A$;
    $A \leftarrow A - \{\text{j}\}$;
    Calculate formula
     $n_{sj} = \ln(\frac{k_i}{k_j})k_{si}$;
    if *Calculation is greater than* n_{si} **then**
      $n_{sj} \leftarrow CalculatedValue$;
    end
  end
end

In order to compute this ratio and assign to each node, I coded python file using Algorithm 3. With this formula, being a neighbor to a node having high k_s, having a low degree (meaning your connection to this neighbor is import for that node), and interacting to a neighbor with having large connection becomes significant. In order to increase prioritization of k-shell and balance the affect of other two parameters, I utilized logarithmic effect on the ratio.

After running this code on the SIR results, I draw graphs showing relationship between k_s and n_s. Figure 3 illustrates this relationship for two different probabilities.



(a) n_s against k_s at $p = 0.01$



(b) n_s against k_s at $p = 0.015$

Figure 3: Comparison of k_s and n_s

As seen in the figures, for the nodes having n_s values greater than 300 has low values on both k_s. This means that they are connected to some higher k_s nodes. However, as seen at average infection rate, they don't have significant affect on spreading. At the next section, I will focus on question whether the spread rate is affected if there is a motivation factor for these high n_s node by assigning them different probability.

3 Motivation Factor for high n_s nodes

As discussed at the beginning of this paper, in equal spreading conditions (same $p = \beta$), higher k_s valued nodes have better performance on spreading if they are originator of the spread. However, it's also known that reaching a node with higher k_s also cost higher than other nodes. This case can be validated with real-life problems such as advertising a new product.

Therefore, I analyze whether will there be any significant change on spreading ratio, if I apply a "higher" probability of spread ratio for only the originator node, and same probability for the rest, if the spread is originated from a node having n_s values greater than 300.

In order to apply this approach, I have modified the SIR algorithm. With this change, algorithm gets two probability parameters: β_1 and β_2. β_1 represents probability of infection for only spread's starting node, whereas β_2 is the rate of infection for all other nodes. Algorithm 4 illustrates the updated SIR algorithm.

After modifying the code for simulation, I recalculated infection rates for spread originated from high n_s valued nodes. I make the iterations with the parameters belows:

- $\beta_1 = 0.25$ and $\beta_2 = 0.01$
- $\beta_1 = 0.5$ and $\beta_2 = 0.01$
- $\beta_1 = 0.25$ and $\beta_2 = 0.005$
- $\beta_1 = 0.5$ and $\beta_2 = 0.005$

I choose small β_2 values intentionally to observe whether there will be significant change on infection even though spread origins have low k_s values.

After running the simulation with updated SIR algorithm, results shown in Figure 4 and 5 are achieved.

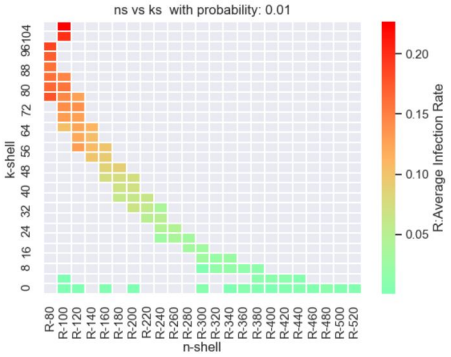
In Figure 4, there is a comparison between original k-shell comparison simulation with $\beta_2 = 0.01$ and modified SIR simulation with

Algorithm 4: Alternative Spreading algorithm of SIR with different spread probability of seed node

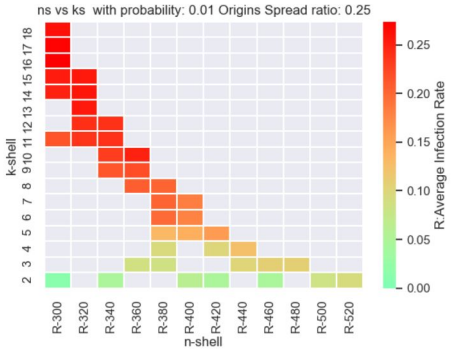
Result: SIR for $G(V, E)$ undirected network

Let $i \in V$;
I: Set of current spreaders;
R: Set of recovered;
$N(i)$: Set of neighbors of i;
β: Probability of spreading infection;
β_1: Probability of spreading infection from origin;
β_2: Probability of spreading infection for rest;
$I = \{i\}$, $R = \emptyset$;
while $I \neq \emptyset$ **do**
  Select $j \in V$;
  if j *is* i **then**
    $\beta \leftarrow \beta_1$;
  else
    $\beta \leftarrow \beta_2$;
  end
  $I \leftarrow I - \{\mathrm{j}\}$;
  $R = R \cup \{\mathrm{j}\}$;
  $C = N(j) \setminus (I \cup R)$;
  while $C \neq \emptyset$ **do**
    Select $l \in C$;
    $C \leftarrow C - \{\mathrm{l}\}$;
    Infect l with probability of β;
    if l *is infected* **then**
      $I \leftarrow I \cup \{\mathrm{l}\}$;
    end
  end
end
Spread Ratio (r) is ratio of recovered nodes R againts all nodes in V ;

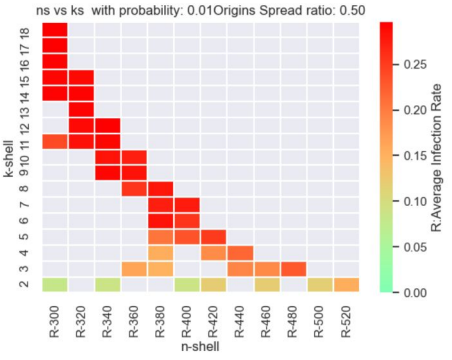
a motivation factors for the nodes having n_s values greater than 300. In the second cases, spread probability from these nodes are increased significantly with a cost of C_m. As seen in the figure 4, after increasing the probability of spread for origin only, higher n_s nodes has reached the infection rates of these nodes are significantly increased and reached to level similar to high k_s nodes. If the motivation factor is increased to higher level, the expectation to have higher infection rate with higher n_s increases.



(a) n_s against k_s at $p = 0.01$



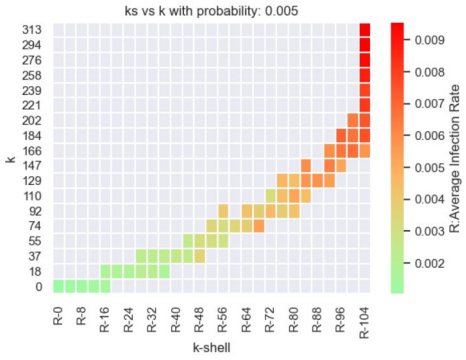
(b) n_s against k_s at $\beta_1 = 0.25$



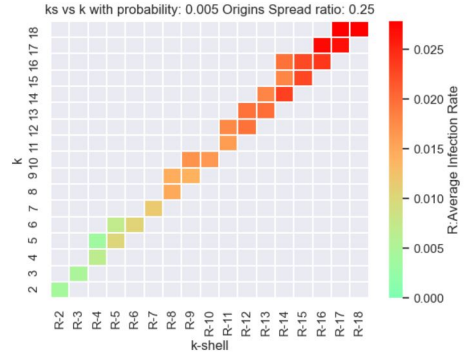
(c) n_s against k_s at $\beta_1 = 0.50$

Figure 4: Comparison of k_s and n_s

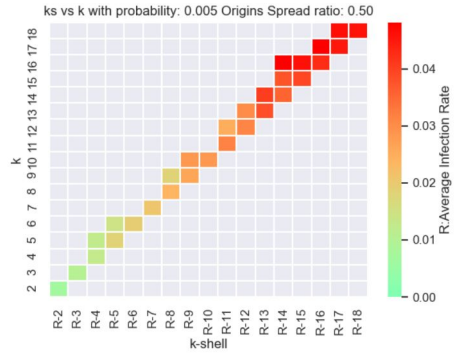
As seen in Figure 5, even at a small rate of infection such as $p = 0.005$, if you apply a motivation factor for origin of spreads having small k_s or k values, it's possible to exceed performance of high k_s valued nodes.



(a) k against k_s at $p = 0.005$



(b) k against k_s at $\beta_1 = 0.25$



(c) k against k_s at $\beta_1 = 0.50$

Figure 5: Comparison of k and k_s

Other figures related to this comparison for k , k_s and k and k_s is also provided at Attachment 4. Moreover, simulation results for each node and case is also provided in Attachment 5.

4 Conclusion

In this project paper, I analyzed a network to find an alternative approach for finding a less costly and more effective influencer. To achieve this I utilized the k-shell approach, degree of node and number of neighbor connections together, and I construct a formula to calculate a new attribute, $n - shell$.

$$n_{si} = \ln(\frac{k_j}{k_i})k_{sj}$$

Moreover, by modifying the SIR algorithm, I applied a new parameter origin infection probability, into my simulations. This helped me to analyze whether there are any positive correlation if a motivation factor is applied for infection probability of origin spreader. In my study, I observed that it's possible to reach similar or better spreading performances than high k_s nodes, if you increase spreading probability of high n_s nodes. However, motivation cost C_m should not exceed to cost of spreading from higher k_s nodes, C_{k_s}. To sum up, n-shell approach can be further studied considering cost relationship as well.

5 References

[1] Maksim Kitsak, Lazaros K. Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik,H. Eugene Stanley and Hernán A. Makse, *Identification of influential spreaders in complex networks*, Nature Physics, 2010 [2] Network X: Network Analysis in Python *https://networkx.org/* [3] SWE580 Complex Network Course notes *https://www.cmpe.boun.edu.tr/courses/cmpe556/*

6 Attachments

[1] Python code to generate graphs with extended Barabási–Albert model [2] Python code for k-shell decomposition [3] Python code for spreading algorithm of Susceptible – Infectious – Recovered (SIR) [4] Figures for infection rates at different probability combinations [5] Simulation Results