

# Influence Maximization Problem

Chu Tingxuan

May 2018

## 1 Introduction

In this project, I have done some researches on the Network Control Problem and the Influence Maximization problem. During this project, I have read some papers and relevant Chinese materials relating to the Influence Maximization problem. I use matlab for modeling and use python to implement the Influence Maximization algorithm. This report mainly analyzes and compares different social network models and algorithms to maximize social network influence.

## 2 Background

In recent years, with the development of the Internet, social network has developed rapidly and has received more and more attention. Many researches are devoted to the analysis of social networks. Social network is the carrier of the people through the network to connect, to form have the characteristics of a group, is composed of the relationship between the individual and individual of a complex network, this kind of complex social structure of information dissemination and diffusion plays an important role, when a person is to adopt a new idea or to accept a product, he would recommend to his friends or colleagues, some people may accept or follow his recommendation, and recommend to their friends or colleagues, this process is called propagation or diffusion, a person's behavior depends largely on the surrounding friends or colleagues. The research on influence propagation in social networks has practical significance. It has very important applications in marketing, advertising, public opinion early warning, and social stability.

## 3 Network Control

### 3.1 The social Network

The issue of maximizing the influence on social networks needs to use the corresponding influence propagation model. In research, social networks are generally abstracted into a graph  $G(V, E)$ , Each node  $v \in V$  represents an agent, and contains an instantaneous state value  $s(v, t)$  at each time step  $t$ . The set of all

node states cumulatively define the network state  $s(V, t) = s(v, t) \mid v \in V$ . The domain of node states is problem dependent. Each edge  $e \in E$  is a directed relation between two nodes, and is associated with a weight value  $w(e, t)$ . Each node on network  $G$  has two states at the initial moment. That is active and inactive, only the nodes that are active and have influence on the nodes they point to, An inactive node has no influence on the node to which it points. When a node is successfully affected by another node, it is said to be activated; when a node is activated, there are more and more neighbors, the probability that the node is activated. Increasingly larger, the activated node can affect the nodes it points to. Each node can only be converted from inactive state to active state, and cannot be reversed.

### 3.2 The diffusion Model

The Diffusion Model (DM) defines the behaviours of nodes as communicating agents in the social network. A DM can be defined as the pair of behaviours  $DM(v) = Sh(v), L(v, I)$ . The two parts of the DM correspond to two phases of communication in the social network. The process of state transition for a given node can now be described formally as the following:

$$s(v, t + 1) = L(v, I(v, t)) \quad (1)$$

$$I(v, t) = (w(e_{u,v}, t), Sh(u)) \mid u \in Ni(v, t) \quad (2)$$

Here  $Ni(v, t) = \{u \mid e_{u,v} \in E\}$  is the inbound neighbourhood of  $v$  at time  $t$ , and  $I(v, t)$  is the set of information tokens transmitted from all such neighbouring nodes. An information token is the output of a node's sharing strategy paired with the weight of the edge along which the token travelled. There are two kinds of DM: progressive and non-progressive. A progressive DM is one in which the state of a node can only be changed a single time. Non-progressive models, are those for which node states may change any number of times during simulation. Both of them will be used later. The model is shown in Figure 1.

### 3.3 The Control System

There are two parts to the control system corresponding to two dependent optimization problems. The first part of the control system deals with configuration. The second part deals with behaviour.

#### 3.3.1 Controller Configuration

A controller configuration defines the locations in the network to which the controller will apply its signals, and from which it observes the state of the networks. The set of all controlled nodes is directly set to the value of the control signal it receives. The control system may control the state of the social networks directly or indirectly. Direct control means that a controller will directly set the state of nodes at each time step. Indirect control means that a controller is setting the states of nodes outside of the network, which then communicate with nodes

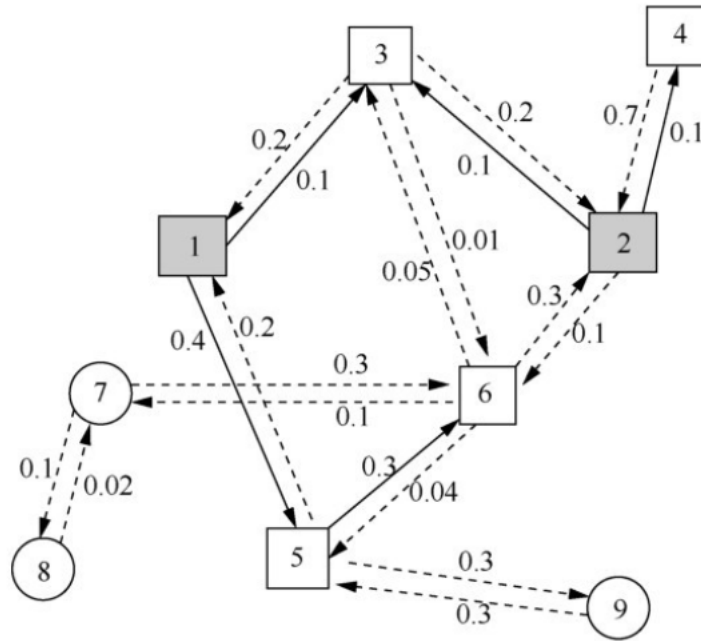


Figure 1: The Diffusion Model

in the network via the diffusion model. The types of controller configuration is shown in Figure 2.

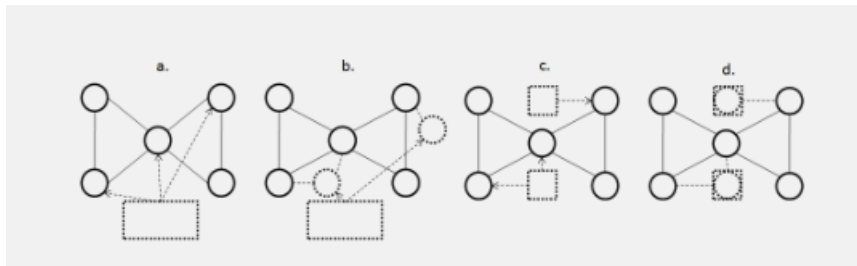


Figure 2: Types of controller configuration. a-Direct control. b - Indirect control. c-Distributed Direct control. d-Distributed Indirect control.

### 3.3.2 Control System Behaviour

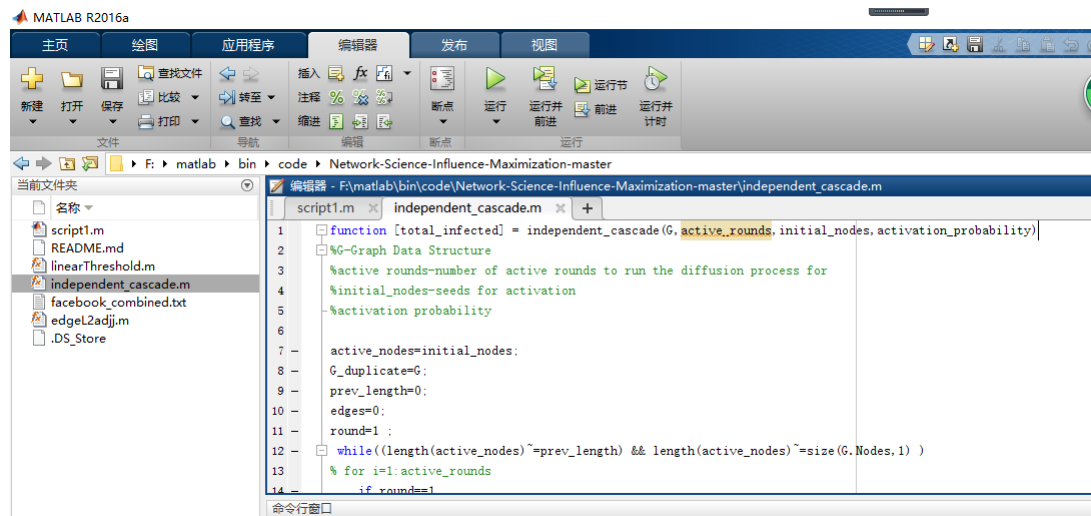
The objective for IMP assumes the optimal control behaviour, thus eliminating the need to optimize this NCP component.

## 4 The Influence Maximization Problem

The problem to be solved in maximizing the influence of social networks is how to select  $k$  seed nodes for information dissemination and diffusion so that the number of nodes that are eventually activated is the most. The propagation process in a social network is that when a node is activated, it will try to activate every unactivated neighbor node connected to it. When a neighbor node is activated, it will try to activate its own neighbor node. This process will continue until no new nodes are activated. At present, there are mainly two propagation models in the spread of social network influence the Linear Threshold(LT) and Independent Cascade(IC).I wrote IC model algorithm and LT model algorithm with MATLAB.The code is given in the appendix.

### 4.1 Independent Cascade

In the progressive IC, once a node is activated, it has a one-time chance of activating each neighbouring node on the subsequent time-step. It is important to note that in the process of the spread, whether a node  $v$  to succeed in  $t$  time to activate its neighbor nodes, in the later time,  $v$  itself, though still active, but it has no influence. At present, there are many researches on the influence maximization of independent cascaded model. The characteristic of IC model is that it only considers the influence of  $v$  and outgoing adjacent  $w$ . However, because it is a probability model, its activation process is uncertain, and the final result of activation of the same seed node for the same network may be quite different. Kempe and Kleinberg argue that the probability of activation success is not independent and will become smaller and smaller over time. I use matlab to model IC model,the algorithm is as Figure3 shows:

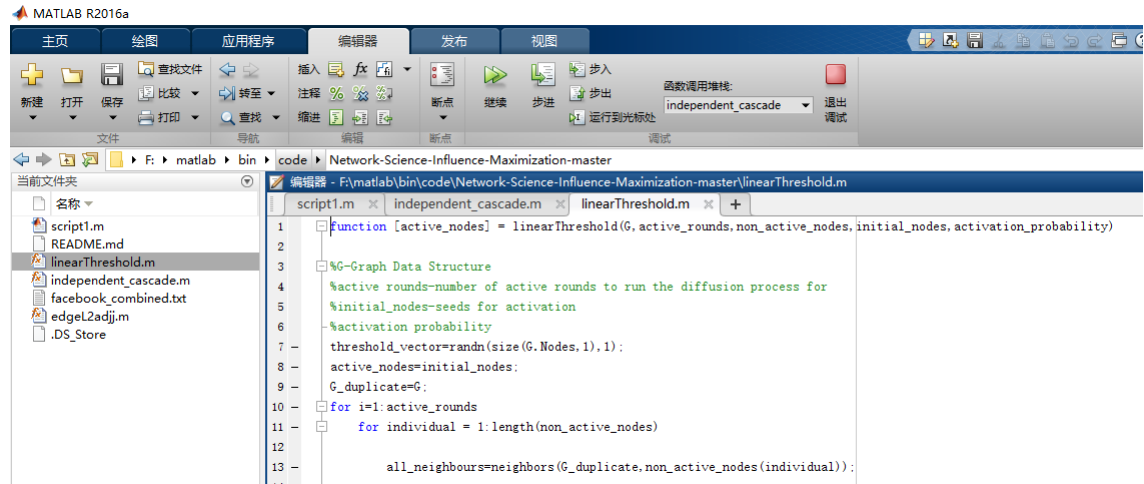


```
1 function [total_infected] = independent_cascade(G, active_rounds, initial_nodes, activation_probability)
2 %G-Graph Data Structure
3 %active_rounds-number of active rounds to run the diffusion process for
4 %initial_nodes-seeds for activation
5 %activation probability
6
7 active_nodes=initial_nodes;
8 G_duplicate=G;
9 prev_length=0;
10 edges=0;
11 round=1 ;
12 while((length(active_nodes)~=prev_length) && length(active_nodes)~=size(G.Nodes,1) )
13     % for i=1:active_rounds
14         if round==1
```

Figure 3: The Independent Cascaded Algorithm

## 4.2 Linear Threshold

In the LT, a node becomes activated if there is sufficient pressure from activated neighbouring nodes. In the LT model, the activation process is determined, and when we activate the same seed nodes on a graph, the final propagation range is exactly the same. When an active node  $w$  attempts to activate its inactive neighbor node  $v$  without success, the propagation range is exactly the same. When an active node  $w$  try to activate it inactive neighbor node  $v$  without success, influence on node  $v$  BWV node  $w$  will be "accumulation", rather than being abandoned, the accumulation of node  $v$  behind other neighbors to activate the contribution of the  $v$ , until the end of the node  $v$  is activated or propagation, this suggests that the activation of LT model engineering is a cooperative process of activation, the activation of each attempt will be accumulated, this is LT model the influence of the accumulation feature, it is different and the IC model. I use matlab to model LT model,the algorithm is as Figure4 shows:



```
1 function [active_nodes] = linearThreshold(G, active_rounds, non_active_nodes, initial_nodes, activation_probability)
2
3 %G-Graph Data Structure
4 %active_rounds-number of active rounds to run the diffusion process for
5 %initial_nodes-seeds for activation
6 %activation probability
7 threshold_vector=randn(size(G.Nodes,1),1);
8 active_nodes=initial_nodes;
9 G_duplicate=G;
10 for i=1:active_rounds
11     for individual = 1:length(non_active_nodes)
12
13         all_neighbours=neighbors(G_duplicate,non_active_nodes(individual));
```

Figure 4: The Linear Threshold Algorithm

## 4.3 the Influence Maximization Algorithm

There are various kinds of the Influence Maximization Algorithm.I have read some relevant materials.Such as greedy algorithm,node selection strategy based on degree,OASNET algorithm,CGA algorithm and Heuristic algorithm. I use Python to implement the Linear Threshold (linear threshold model) algorithm for maximizing social network impact and implement greedy algorithm. The specific experimental steps areuse parameter  $k$ - represents the number of child nodes to obtain,obtain all the nodes in the figure.Store all nodes in the list,the activation node list and activation list length for each node.Traverse all nodes,find the active node set corresponding to each node and the length of the active node

set. Get the number of nodes with the most active nodes, the list with the most active nodes and the child with the most active nodes. Use the loop to delete the set of seed nodes in all nodes. Return the child node set obtained by the greedy algorithm and the maximum node set activated by this child node set. Create a directed graph based on a data set. Invoke the greedy algorithm to obtain the subset of nodes and subset of nodes. The detailed code will be given in the appendix. The results about the Linear Threshold is shown in Figure 5. The result shows the layers (the child node set and the maximum activation node set of this child node), the lengths (the number of active nodes of child nodes) and the running time. The results about the greedy algorithm is shown in Figure 6. At first, I

```
In [10]: runfile('C:/Users/lenovo/Desktop/Linear_Threshold-master/
test_linear_threshold.py', wdir='C:/Users/lenovo/Desktop/Linear_Threshold-
master')
Reloaded modules: linear_threshold_clime, linear_threshold
[[25], [33, 3, 35, 6, 8, 75, 80, 50, 19, 55, 54, 23, 28, 29, 30]]
15
Running time: 0.006535026619530981 Seconds
```

Figure 5: The Linear Threshold Result

input the number of seed nodes. The result shows the seed nodes, layers (the child node set and the maximum activation node set of this child node), the lengths (maximum activation node set) and the running time.

```
In [11]: runfile('C:/Users/lenovo/Desktop/Linear_Threshold-master/
test_linear_threshold_clime.py', wdir='C:/Users/lenovo/Desktop/Linear_Threshold-
master')
Reloaded modules: linear_threshold

Please input the number of seeds: k=7
[25, 30, 1412, 3352, 5254, 5543, 7478]
[33, 3, 35, 6, 8, 75, 80, 50, 19, 55, 23, 29, 271, 152, 28, 286, 39, 300, 178,
54, 567, 182, 581, 584, 586, 590, 214, 8283, 348, 349, 604, 611, 108, 371]
34
Running time: 104.58678415999793 Seconds
```

Figure 6: The Linear Threshold Clime Result

## 5 Conclusion

During this program, I do some researches on the Control Problem and the Influence Maximization Problem. The report focused on the network propagation model and influence maximization algorithm. Related theoretical and experimental results are shown.

## 6 Experience

This is my first contact with social networking. When I started reading papers, I had a lot of questions. I communicated with the teacher Luoyi FU and solved a lot of problems. The teacher has been very patient in guiding me and telling me the most advanced papers. I also shared my experience with the teacher after reading. In class, there were teachers who talked about social networking related content, such as anonymous networks. I also had new experiences and understandings of social networking during the course of class. Searched a lot of related Chinese and English data on the Internet, and also searched many algorithms on GitHub for research. The process of writing code using matlab and Python also has a deeper understanding of these algorithms. Although there was no outstanding result in this project, it was a process from zero to one for me. I have learned a lot of new knowledge in this project. Thank the teacher Luoyi Fu for providing me with a lot of help and guidance!

## 7 Appendix

### 7.1 The diffusion Model in Matlab

```
function [active_nodes] = linearThreshold(G, active_rounds, non_active_nodes, initial_nodes, activation_probability)

%G-Graph Data Structure
%active rounds-number of active rounds to run the diffusion process for
%initial_nodes-seeds for activation
%activation probability
threshold_vector=randn(size(G.Nodes,1),1);
active_nodes=initial_nodes;
G_duplicate=G;
for i=1:active_rounds
    for individual = 1:length(non_active_nodes)

        all_neighbours=neighbors(G_duplicate,non_active_nodes(individual));

        infected_neighbours = intersect(all_neighbours,active_nodes);
        % all_neighbours = setxor(all_neighbours,Acommon);
        prob_infection=length(infected_neighbours)*activation_probability;

        if(prob_infection>threshold_vector(non_active_nodes(individual)))
            active_nodes=[active_nodes;all_neighbours(individual)];
            non_active_nodes(individual)=[];
        end

    end

end
end
```

Figure 7: The Linear Threshold Algorithm

### 7.2 The Influence Maximization Algorithm in Python

```

function [total_infected] = independent_cascade(G,active_rounds,initial_nodes,activation_probability)
%G-Graph Data Structure
%active rounds-number of active rounds to run the diffusion process for
%initial_nodes-seeds for activation
%activation probability

active_nodes=initial_nodes;
G_duplicate=G;
prev_length=0;
edges=0;
round=1 ;
while((length(active_nodes)~=prev_length) && length(active_nodes)~=size(G.Nodes,1) )
% for i=1:active_rounds
    if round==1
        active_nodes_last_round=active_nodes;
    else
        active_nodes_last_round=active_nodes_current_round;
    end
    active_nodes_current_round=[];
    prev_length=length(active_nodes)
    for individual = 1:length(active_nodes_last_round)

        potential_infected=neighbors(G_duplicate,active_nodes_last_round(individual));
        Acommon = intersect(potential_infected,active_nodes);
        potential_infected = setxor(potential_infected,Acommon);
        for victim = 1:length(potential_infected)
            temp=rand;
            if(activation_probability>temp)
                active_nodes_current_round=[active_nodes_current_round;potential_infected(victim)];
                edges=edges+1;
                G_duplicate=G_duplicate.rmedge(active_nodes_last_round(individual),potential_infected(victim));

            else
                edges=edges+1;
                G_duplicate=G_duplicate.rmedge(active_nodes_last_round(individual),potential_infected(victim));
            end
        end

    end
    active_nodes=[active_nodes;unique(active_nodes_current_round)];
    length(active_nodes);
    round=round+1;
end
total_infected=size(active_nodes,1); 8

```

Figure 8: The Independent Cascade Algorithm



```

#贪心算法
def _linear_clime(G,k):      #参数 k-表示要获取的子节点的个数
    allnodes = G.nodes()   #获取图中所有节点数
    seed_nodes = []        #该列表存储选取的子节点集
    for m in range(k):
        all_nodes = list(allnodes) #将所有的节点存储在 all_nodes 列表里
        layers_activite = []      # 存储每个节点的激活节点列表
        lengths = []              # 存储每个节点的激活列表长度
        datas = []
        for i in all_nodes:      #遍历所有的节点，分别求出每个节点对应的激活节点集以及激活节点集的长度
            data = []
            data.append(i)
            datas.append(i)
            data_test=seed_nodes+data
            layers = linear_threshold(G,data_test)
            data.pop()
            del layers[-1]
            length = 0
            layer_data = []
            for j in range(len(layers)):
                length = length + len(layers[j])
                layer_data = layer_data + layers[j]
            length_s = length - len(layers[0])
            for s in range(len(layers[0])):
                del layer_data[0]
            layers_activite.append(layer_data)
            lengths.append(length_s)
        # length_max = max(lengths) # 获取激活节点最多的节点数
        layers_max = layers_activite[lengths.index(max(lengths))] # 获取被激活节点数最多的列表
        seed_nodes.append(datas[lengths.index(max(lengths))]) # 获取被激活节点最多的子节点集
        for i in all_nodes:      #该循环是删除所有节点中 seed_nodes 节点集
            if i in seed_nodes:
                del all_nodes[all_nodes.index(i)]
        allnodes=all_nodes
    return seed_nodes,layers_max      #返回值是贪心算法求得的子节点集和该子节点集激活的最大节点集

#测试算法
if __name__ == '__main__':
    start=time.clock()
    datasets=[]
    f=open("Wiki-Vote.txt")
    data=f.read()
    rows=data.split("\n")
    for row in rows:
        split_row=row.split("\t")
        name=(int(split_row[0]),int(split_row[1]))
        datasets.append(name)
    G=networkx.DiGraph()
    G.add_edges_from(datasets) #根据数据集创建有向图

    n=input("Please input the number of seeds: k=")
    k=int(n)
    seed_nodes, layers_max=_linear_clime(G,k) #调用贪心算法获取节点子集和节点子集的最大激活节点集
    lenth=len(layers_max)
    end=time.clock()
    print(seed_nodes)
    print(layers_max)
    print(lenth)
    print("Running time: %s Seconds"%(end-start))

```

Figure 9: The Greedy Algorithm