Report for Influence Maximization Problem

Name: Zhang YIfan

SID: 11711335

1. Preliminaries

1.1 Problem Description

One of the fundamental problems in a social network is influence maximization. Consider a social network as a graph G(V, E) consisting of individuals (node-set V) and relationships (edge set E); essentially influence maximization problem comes down to the problem of finding important nodes or structures in graphs.

To address the influence maximization problem, first, it is needed to understand the influence diffusion process in social networks. In other words, how does the influence propagate over time through a social network? Assume time is partitioned into discrete time slots, and then influence diffusion can be modelled as the process by which activations occur from neighbour to neighbour. In each time slot, all previously activated nodes remain active, and others either remain inactive or be activated by their neighbours according to the activation constraints. The whole process runs in a finite number of time slots and stops at a time slot when no more activation occurs. Let S denote the set of initially activated nodes; we mean by f(S) eventually the number of activations, and the target is to maximize f(S) with a limited budget.

There are two diffusion models: IC and LT. For the IC model, each node has a threshold, and it is randomized and reset at each iteration. And each edge is represented by three values: begin node, end node, activation probability. During each propagation, the begin node will try to activate the end node with the assigned probability. In the implementation, we realize this process by generating a random float between 0 and 1, if it's smaller than or equal to the probability, then it's activated successfully. Otherwise, it is not. For the LT model, it's not about the possibility. Each node keeps a record of its current value, and we achieve this by adding all the values that all the activated nodes which point to it contribute. If the current value exceeds its threshold, then it's activated.

The following discusses the INPUT and OUTPUT format of ISE and IMP:

1.1.1 ISE

INPUT: A graph G(V, E) where V is the set of individuals and E is the set of edges (relationships), a set S indicates all the seeds(already activated node in the graph), an activation model m, and a limited time budget t.

OUTPUT: A number answer indicates the most precise number your algorithm calculates that the number of nodes that are activative after propagation.

1.1.2 IMP

INPUT: A graph G(V, E) where V is the set of individuals and E is the set of relationships, an activation model m, a number k indicating the size of desired seed set, and a limited time budget t.

OUTPUT: A set S of nodes within V where S□ such that the final activations f (S) is maximized and |S| <= K .

1.2 Problem Applications

The maximization of influence has full application within social networks. We can consider a social network as a graph of relationships and interactions within a group of individuals — plays a fundamental role as a medium for the spread of information, ideas, and influence among its members. If we can convince a small number of individuals in a social network to adopt a new product or innovation, and the target is to trigger a maximum further adoptions, then which set of individuals should we convince? To understand how such "word-of-mouth” effects are hold, to what extent some people are affected by influential social accounts, friends, colleges or other people, we need to understand the dynamics of adoptions and also apply influencial maximization problems to such practical scenario.

Suppose that we have data on a social network, with estimates for the extent to which individuals influence one another, and we would like to market a new product that we hope will be adopted by a significant fraction of the network. The premise of viral marketing is that by initially targeting a few “influential” members of the system, for example, giving them free samples of the product — we can trigger a cascade of influence by which friends will recommend the product to other friends, and many individuals will ultimately try it. But how should we choose the few key individuals to use for seeding this process? This question is considered in a probabilistic model of interaction; heuristics were given for choosing customers with a significant overall effect on the network, and methods were also developed to infer the influence data necessary for posing these types of problems.

2. Methodology

2.1 Notation

2.1.1 Notations

f(S): eventually the number of activations of seedset S

|S|: the size of seed list

t(v): the threshold value of node v

w(i, j): the weight of edge. In IC model it means probability.

2.1.2 Names

nodeCount: the number of nodes

edgeCount: the number of edges

ActivitySet: the list of nodes which are activated

Total_sum: the sum of answers from each pool

Total_count: the number of different pools

2.2 Data Structure

Class node: an object named node; represents each node in the graph; it has attributes:

1. an integer: num (marked from 1 to nodeCount)
2. a list: neighbor, contains all the neighbors of this node
3. a list: value, contains all the values of influence to its neighbor
4. an integer: threshold, indicates its threshold to be activated(randomed at each iteration)
5. a boolean: active, indicates whether it's activated or not
6. an integer: influence, indicates the number of nodes this node can activate in total in IMP

List:

nodeList: contains all the nodes in this graph

Seed_list: contains all the seeds chosen at this iteration

ActivitySet: contains all the nodes that are activated for now

newActivitySet: contains all the nodes that are activated during this iteration

2.3 Model Design

2.3.1 ISE

First reading the requirements of ISE problem, I first concluded the INPUT and OUTPUT format of ISE:

INPUT: A graph G(V, E) where V is the set of individuals and E is the set of edges (relationships), a set S indicates all the seeds(already activated node in the graph), an activation model m, and a limited time budget t.

OUTPUT: A number answer indicates the most precise number your algorithm calculates that the number of nodes being activated after propagation.

Then I analyzed the main steps to solve this problem:

The main steps:

1. Use sys and getopt to read in commands, graph and seeds.
2. Initialized nodeList and set the attributes.
3. Start eight concurrent pools, which communicate through queue.
4. Recognize the required model, start calculation accordingly. Main algorithm used in this step were given by the lecture powerpoint. And in this report the details and pseudocoede is given in the next part.

2.3.1 IMP

First reading the requirements of ISE problem, I first concluded the INPUT and OUTPUT format of IMP:

INPUT: A graph G(V, E) where V is the set of individuals and E is the set of relationships, an activation model m, a number k indicating the size of desired seed set, and a limited time budget t.

OUTPUT: A set S of nodes within V where S□ such that the final activations f (S) is maximized and |S| <= K .

Then I analyzed the main steps to solve this problem:

The main steps:

1. Use sys and getopt to read in commands, graph and seed limit k.
2. Initialized nodeList and set the attributes.
3. Start eight concurrent pools.
4. Recognize the required model and start calculation accordingly. The algorithm used in this step is similiar to the ISE algorithm, but there are some modifications and optimization. And in this report the details and pseudocoede is given in the next part.
5. The main **idea** is to record how many nodes each node influenced in the above process. The pseudocode and details of this algorithm is given in the later parts.
6. Choose the top k nodes ranked according to their influence.

2.4 Detail of Algorithms

2.4.1 ISE

Pre-handling:

Describe:

I used modules sys and opt to read in the instruction and saved the information in the file to my program. All the file context, seeds, model, time limit are saved for future use.

Pseudocode:

```
Read in file network.txt as t, seedlist as s, model as m, time limit as t
open file(n), get nodeCount and edgeCount
initialized nodeList
add each edge into the nodeList: add neighbor and value
```

Algorithm for model IC:

Describe:

First start with the given Seedlist. After this propagation finished, set the newly activated nodes as the newSeedlist and start propagation. Repeat until the genereated newSeedList is empty. The standard for activation is a random number and the threshold.

Pseudocode:

```
calculate_IC(timeout):
    global save, nodeList
      start = now.time
      while(True)
            ActivitySet = save
            initialize all nodes in nodeList
            set all seed.active to True
            while ActivitySet is not null
                  initialize newActivitySet
                  For all node in nodeList
                        try activate all neighbors with a random number
                        if activates
                                change neighbor's state
                              append this neighbor to newActivitySet
                        End if
                  End For
                    add number of nodes in newActivitySet to count
                    set ActivitySet to newActivitySet
               if now.time - start > timeout - 3
                    return ans_sum/ans_count
               End if

               ans_count++
               ans_sum += count
```

Algorithm for model LT:

Describe:

This process is similiar to IC. Only when trying to activate the neighbors, it useds the values of the edges and thresholds of the neighbors instead of the probability.

Pseudocode:

```
calculate_LT(timeout):
    global save, nodeList
      start = now.time
      while(True)
            ActivitySet = save
            initialize all nodes in nodeList
            set all seed.active to True
              while ActivitySet is not null
                  initialize newActivitySet
                  For all node in nodeList
                        try activate all neighbors by comparing adding edge to
the value and the threshold of neighbors
                        if activates
                                change neighbor's state
                              append this neighbor to newActivitySet
                        End if
                  End For
                    add number of nodes in newActivitySet to count
                    set ActivitySet to newActivitySet
              if now.time - start > timeout - 3
                    return ans_sum/ans_count
              End if

              ans_count++
              ans_sum += count
```

Output:

```
result stores the return values of all pools
for tmp in result
        add sum to total_sum
        add 1 to total_cnt
End For
Print(total_sum/total_cnt)
```

2.4.2 IMP

Pre-handling:

Describe:

I used modules sys and opt to read in the instruction and saved the information in the file to my program. All the file context, the number of seeds required, model, time limit are saved for future use.

Pseudocode:

```
Read in file network.txt as t, seedlist as s, model as m, time limit as t
open file(n), get nodeCount and edgeCount
initialized nodeList
add each edge into the nodeList: add neighbor and value
```

Algorithm for model IC:

Describe:

Each iteration uses a randomed seedlist with size k. Run ISE algorithm on this seedset for n times (This parameter is later discussed). Record how many nodes each node activated. Each activation is evaluated by probability.

Pseudocode:

```
while True
    randomly chooose k distinct nodes out of n nodes as seedlist
    start = now.time
        while(True)
            ActivitySet = seedlist
            initialize all nodes in nodeList
            set all seed.active to True
            while ActivitySet is not null
                For all node in ActivitySet
                    try activate all neighbors with a random number
                    if activates
                        change neighbor's state
                        append this neighbor to newActivitySet
                        this node.influence ++
                    End if
                End For
                set ActivitySet to newActivitySet
             if now.time - start > timeout - 3
                return
             End if
```

Algorithm for model LT:

Describe:

Each iteration uses a randomed seedlist with size k. Run ISE algorithm on this seedset for n times (This parameter is later discussed). Record how many nodes each node activated. Each activation is evaluated by sum of the values pointing to it from already activated nodes.

Pseudocode:

```
while True
    randomly chooose k distinct nodes out of n nodes as seedlist
    start = now.time
        while(True)
            ActivitySet = seedlist
            initialize all nodes in nodeList
            set all seed.active to True
            while ActivitySet is not null
                For all node in ActivitySet
                    try activate all neighbors with this value    #Difference 
with IC
                    if activates
                        change neighbor's state
                        append this neighbor to newActivitySet
                        this node.influence ++
                    End if
                End For
                set ActivitySet to newActivitySet
             if now.time - start > timeout - 3
                return
             End if
```

Output:

```
print out the number of the top k nodes in nodeList with highest influence value
```

3. Empirical Verification

3.1 Dataset

1. Network-5-IC/LT
2. NetHEPT-5-IC/LT
3. NetHEPT-50_IC/LT
4. NetHEPT_fixed
5. Dataset on the IMP Platform

3.2 Performance Measure

3.2.1 Test Environment

Python 3.6, Windows

3.2.2 Time

Since the problem was to approach the most precise answer within the time limit, and because the more iterations it ran, the more precise the results are, my algorithm was to use **time** module of python to keep track of the running time and make sure it finishes only seconds before time limit. So time is not taken into consideration regarding performance.

3.2.3 ISE performance

On the IMP platform, it passed all the test suites well and also all the tests after stage 1 finished.

3.2.4 IMP performance

I measured the performance by checking the result it gets after uploading to the IMP platform. However, the results are not as ideal as expected.

When running the network-5-IC test, the result is only 21.3272/30.6199. And when running the NetHEPT-50_IC test, the result was only 889.6641/1297.8358. (Best result referenced is the top result on the ranking list of IMP platform)

3.3 Hyperparameters

1. Parameter: n, which indicates the number of iterations ISE run

At first I set the the n to 10000, which means ISE run 10000 times and print out the average answer of all these 10000 answers. It worked well at first, but when the graph becomes larger, such as NetHEPT, this would time out. Then I set the n to 1000, the process of ISE can be finished but the results are not as ideal. So as last I completely abandoned this parameter and let the program run until the time budget is about to use up.

2. Parameter: X, which indicates the number of ISE each new seedless run

At first I set x to be 10, which means when a randomed seedlist is generated, ISE will run 10 times and add the numbers to these nodes' influence. However, this means less likely some nodes are chosen as seed, therefore damaging the final result of the algorithm. Then I set x to be 3, which means more seedlist can be generated and in total more fair to all nodes.

3.4 Experimanteal results

3.4.1 ISE

Time_limit: 60 seconds

Network	SeedSet	Model	Iteration	Result	Time
network.txt	seeds.txt	IC	8916220	5.015134	57.022938
network.txt	seeds.txt	LT	3247162	5.033380	57.028794
NetHEPT_fixed.txt	seeds.txt	IC	38368	21.01169	57.034299
NetHEPT_fixed.txt	seeds.txt	LT	14866	21.97481	57.037941

3.4.2 IMP

Test suite	X	Model	Result	Time
network.txt	2	IC	21.3272	58.228
network.txt	2	LT	37.5412	58.590
network.txt	5	IC	21.3272	58.343
network.txt	5	LT	37.3482	58.903
NetHEPT-50.txt	2	IC	889.6641	118.421
NetHEPT-50.txt	2	LT	982.5775	118.331
NetHEPT-50.txt	5	IC	867.0562	118.520
NetHEPT-50.txt	5	LT	982.2082	118.442

3.5 Conclusion

3.5.1 Advantages

My algorithm has the following advantages:

1. Instead of setting the number of iteration of our program, I used the time limit and let the program run until the time budget is about to use up. By this means, we guarantee that the program won't timeout and more iterations can be executed than the above mentioned mean.
2. I created a class named node to formulate this graph. A node has several attibutes such as neighbor list, value list, status such as current value and whether it's activated. After read in the file, we can interpret the file content and store all the information into the list of all the nodes we initiated.

3. I used Pool module in python and run eight pools simutaneously. So after the time-limit is up, we can obtain eight answers instead one. By adding eight answers together and divide eight, we can obtain a more precise answer by calculating more data.

3.5.2 Disadvantages

However, this algorithm still has some disadvantages:

1. When solving the seedlist of Influence Maximization Problem, I only considered the influence of nodes as discrete individuals. Because when I was counting their influence (the number of nodes each one activated), I add all the numbers together, regardless of the iteration or the chosen seedlist. However, after analyzing the influence maximization problem, we can easily find there are interactions between some nodes and the relationship between them is more complicated than we understand. For example, when applying other algorithms such as genetic algorithms or hill climbing algorithm, it would only change part of the seed set for a better conclusion. But in this algorithm, the seedlist is completely randomed at each iteration.

4. References

[1] D. Kempe, J. Kleinberg, E. Tardos. Maximizing the Spread of Influence through a Social Network. Cornell, June 2003.

[2] Z. Lu, W. Wu, Influence Maximization. Encyclopedia of Algorithms. October 2014.

[3] Chen W, Yuan Y, Zhang L, Scalable influence maximization in social networks under the linear threshold model. The 2010 international conference on data mining. Sydney, Australia. 2010.