Influence Maximization Problem Report

Zunyao Mao 11811620
Department of Computer Science and Engineering
Southern University of Science and Technology
11811620@mail.sustech.edu.cn

I. Preliminaries

A. Problem Description

The problem of maximum influence is a classic problem in social science. The purpose of the problem is to study the process of information dissemination in social networks.In computer science,IMP can be modeled as a directed graph to simulate a specific social network and calculate a subset of nodes of size k so that this subset has the greatest influence. In this directed graph, nodes represent users, and the weight of directed edges represents the influence of one node to another node. Kempe et al. has already show that IMP is NP-hard.Therefore, it is necessary to use computing method to simulate this process,which is called Influence Spread Estimation(ISE). In this project, we need to construct ISE and IMP model and implement alogorithm to solve them.

1) Software: This project is written in *Python 3.8* with IDE of *PyCharm*.

2) Algorithm: Monte Carlo method is used in ISE and Influence Maximization via Martingales algorithm(IMM) is used in IMP.

B. Problem Applications

The application scenarios for maximizing influence are very rich, including viral marketing, recommendation systems, information diffusion, time detection, expert discovery, link prediction, etc. For example, if a company wants to promote its own products and hopes to use viral marketing methods to select a small number of people to let them try the products they need to promote for free. When the selected users (seed nodes) respond to the products When you are satisfied, you have to recommend the product to your colleagues and friends through the Internet, so that more people can understand and finally buy the product. How to find out these people to try out the products to maximize the number of people who finally buy the products is the core issue that the company needs to consider.

II. Methodology

A. Notation

- $G = (V, E)$: a Directed graph representing the network graph,G with node set V and edge set E
- n, m: the number of nodes and edges in G, respectively
- $w(e)$: the weight of edge e,range from [0,1]
- k: the number of seed node required in IMP
- RR: reverse reachable set
- R: the set of randomly generated RR

B. Data Structure

- **activitySet**: the nodes being activated in some phase of spreading, implemented as a python list
- **seedNodes**: the initial seeds to activate the whole graph, implemented as a python list
- S_k: the output set in IMP, implemented as a python list

C. Model Design

1) Two kinds of diffusion models:

- Independent Cascade (IC) model: In the IC model, each of directed edges $e(u, v) \in E$ in the network graph have a corresponding probability value $p(e) \in [0, 1]$. Intuitively speaking, p(e) indicates that when node u is activated, node u passes through the edge e(u, v) has w(e) probability to activate node v independently. The dynamic propagation process is as follows: At t=0, a pre-selected initial set S_0 is activated first, while other nodes are inactive. This initial set of nodes is called the seed node set. For any time t1, use every edge e of node in S_0, try to "activate" the other inactive node with probability w(e).Then use the new batch of active nodes to do next iterate process until node node can be activate.
- Linear Threshold (LT) model: In the linear threshold model, each directed edge e(u, v) There is a weight w(e) [0,1] on E. As mentioned above, w(e) reflects the influence of node u at node v among neighbors.Each node v also has a Impact threshold v [0,1], Uniformly and randomly selected within the enclosure. once confirmed in the spread the threshold will no longer change. Like the independent cascade model, at t=0 At any time, only the nodes in the seed set S 0 are activated. At every time t 1, every active node $v \in activeSet$needs to interate its out edge,trying to activate its neighbors. The neighbors will be activated when the linear weighted sum has reached its threshold value. When there are no new nodes at a certain moment to be activated, the propagation process ends.

2) Influence Spread Estimation: In order to estimate the influence of certain seed node set, Monte Carlo method is used in my implementation. The basic idea of the Monte Carlo method is that:Using the given seed set to simulate the spread process for N times; If N is large enough relativelythen the average result can be seen as expected value of real spread process.

IC Sampling: Firstly use given seed set of nodes, and put them into a queue called $activeSet$.Dequeue and traverse

each node's out edges try to activate its inactive neighbors at a probability w(e). If the neighbor is activated, enqueue it to $activeSet$.One time of sample process will terminate when there are no nodes in $activeSet$.For algorithm detail, see Algorithm 1.

LT Sampling: Firstly we set every node in the graph a threshold, and put initial seed set of nodes into a queue called $activeSet$.Then we dequeue traverse each node's out edges and add the weight of edge to the active value of neighbor node.If the neighbor node's active value is greater than its threshold, the node is activated, and we enqueue into $activeSet$.One time of sampling will terminate when there are no nodes in activeSet.For algorithm detail, see Algorithm 2.

3) Algorithm to solve IMP: The alogrithm I implemented is derive from Influence Maximization via Martingales(IMM)[1].The algorithm is based on greedy strategy.Let Set S be a node set in network Graph G, and function $F(S)$ is the expected influence of set S.If we find a function F, such that for any subset $S \subseteq V$ and its Any superset $T(S \subseteq T \subseteq V)$ and for any element $u \in V$,f satisfies $f(S \cup u) - f(S) \geq f(T \cup u) - f(T)$. Then we can iteratively find the node with the largest marginal effect.

DEFINITION 1 (Reverse Reachable Set). Let v be a node in V . A reverse reachable (RR) set for v is generated by first sampling a graph g from G, and then taking the set of nodes in g that can reach v. A random RR set is an RR set for a node selected uniformly at random from V .

4) RR sampling of IMM[1]: It can be proved that for a randomly set R of RR_s, if node V is contained in the most number of RR_s, then node V is most likely to be the most influential nodes.Moreover Function $F_R(S)$: the number of RR convered by node set of R fits property of submodular function.So we use such function to do node selection.

5) Node Selection of IMM : In the node selection, the input is set R(contains plenty of reverse reachable sets) and given seeds number k. We need to find the most influential k nodes which "cover" the most number of RR_s in the R set. Initially let $S_k = \emptyset$. For each turn, find the node that can cover most of RR in R, and then add this node to S_k. After that, delete those coveredRR from R.(because we need to fit the property of submodular function).Iterate the process for k times. Then return the set S_k,which is the most influential k nodes we want.

D. Detail of Design

1) IC Sampling: in Algorithm 1.

2) LT Sampling: in Algorithm 2.

3) Generate RR: There are little difference between sampling and RR generation.Just change the seed node to be a randomly chosen set ,and conduct the algorithm in a reversed graph.

4) GetR: Phase 1 of IMM algorithm,for detail in Algorithm3.This function generates a RR set R.

5) Node Selection: Phase 2 of IMM. See algorithm 4.

6) IMM_modify: in Algorithm 5.

Algorithm 1: IC mode Sampling

Input: $seedNodes$: Given node seeds
$graph$: Given Input Graph

Output: $count$: Number of nodes being activated

$activeSet = seedNodes$
$count = 0$
while *$activeSet$ is not empty* **do**
  $newActiveSet = \emptyset$
  for *node v in $activeSet$* **do**
    for *edges e(v, u) adjacent to v* **do**
      if *u is not activated and $weight(e) >$ random number of range [0,1]* **then**
        add v into $newActiveSet$
      end
    end
  end
  clear $activeSet$
  $count+ =$length of $(newActiveSet)$
  add all v from $newActiveSet$ into $avtiveSet$
end

Algorithm 2: LT mode Sampling

Input: $seedNodes$: Given node seeds
$graph$: Given Input Graph

Output: $count$: Number of nodes being activated

$activeSet = seedNodes$
Init every node v in gragh g with a threshold value randomly from [0,1]
Init every node v's activate value to 0
$count = 0$
while *$activeSet$ is not empty* **do**
  $newActiveSet = \emptyset$
  for *node v in $activeSet$* **do**
    for *edges e(v, u) adjacent to v* **do**
      if *u is not activated* **then**
        add weight e to v's activate value
        if *v's activate value $> v's$ threshold* **then**
          add v into newActiveSet
        end
      end
    end
  end
  clear $activeSet$
  $count+ =$length of $(newActiveSet)$
  add all v from $newActiveSet$ into $avtiveSet$
end

Algorithm 3: Get R

Input: $graph$: Input Graph
$time\_limit$: time limit of program
$space\_limit$: space limit of program

Output: $setR$: set of RR_s

$setR = \emptyset$
$time\_cost = 0$
while $time\_cost < time\_limit$ *and size of setR <* $space\_limit$ **do**
  uniformly choose a node v from graph
  RR=get RRs by choosing IC mode or LT mode
  add RR into $setR$
end
return R

Algorithm 4: Node Selection

Input: $setR$: set of RR_s
k: Size of output seeds

Output: S_k: the output set in IMP
$S_k = \emptyset$
for *loop time from 1 to* k **do**
  select a node v that covers the most number of RR_s
  add v to S_k
  remove those RR_s which is covered by v from R
end

III. Empirical Verification

A. Dataset

The following data set are the graph I used to test my program's performance.

- **NetHEPT**: Data from sakai site of CS303A, which contains 62 nodes and 159 edges.
- **MyRandomData**:To avoid exceeding time and memory limitsI generated some datasets, node range is [50,10 0000], edge range is [0,100 0000] and k range is [1,1000],which ensures that my program would not throw error when encountering some unusual datasets.

Algorithm 5: IMM_modify

Input: $graph$: Input Graph
k: Size of output seed
$time\_limit$: time limit of program
$space\_limit$: space limit of program

Output: S_k: the output set in IMP
R <=getRRs($nodeNum, graph, model,$ time_limit,space_limit)
S_k <=node_selection($setR, k, nudeNum$)
output the S_k

B. Performance

In ISE, to make the result more accurate, I have my program to make the most use of time. The program will terminate when there are 6 seconds left, which is enough for the overhead to output result and garbage collection.Iterate time is about 540,000 when 15233 nodes and 32235 edge is contained in gragh and time limit is 60s.

In IMP, there are mainly two parts in IMM algorithm: sampleRR part and nodeSelection part. When the amount of calculation is the largest(10 0000 nodes, 100 0000 edges and 1000 node seed set), about 60000 RR_s will be generated in 60s, and node selection process will cost about 10-20s, when time limit is 120s.Detailed result can be seen in table 1.

C. Hyperparameters

In ISE, the most important hyperparameter is N, which is iterate time when doing Monte Carlo Sampling. However, in online judge environment, so a constant t(the rest of time before timeout) is required instead of N.Through try and error, I find t=6s is enough for my program avoid time out.

In IMP, the most important two hyperparameters are the time limit and space limit of sample RR.Intuitively,if we get more RR, the result will be more accurate.However, too large size of set R will cost too much time and space, which increase the risk of exceeding time and space limit.To find the balance between accuracy and performance, I did many experiments.It turns out that 512MB of size R and 50% ratio of time in time limit to sample RR is safe enough for online judge. So I take these two parameters as my final program.

D. Experimental Results

The table below contains some typical experimental result. Table1 is pressure test, we can see that when edge number is very large(10 times of node number), the RR number we can get reduced very rapidly, but we can still handle it.The time left is enough in case my program throws error when memory or time runs out.

Graph with 1,000,000 nodes, k=1000,time limit = 120				
edgeNum	Model	getRR time(s)	node selection time(s)	RR number
250000	IC	61.91	16.96	1075372
250000	LT	61.71	11.51	896382
1000000	IC	63.80	8.51	58658
1000000	LT	64.13	11.75	16348

TABLE I
Pressure Test Result

Table2 is the result of my code run in online judge, we can see that it is very close to the result of rank 1.

E. Conclusion

The advantage of my project is that in addition to time limit, I did a lot of experiment to adjust space limit so that time out issue are less likely to happen.The disadvantage of

data set in OJ			
data set	my result	rank1 result	difference rate
NetHEPT-50-IC	1297.9041	1298.7102	0.06%
NetHEPT-50-LT	1701.6728	1701.9953	0.02%
NetHEPT-500-IC	4331.4476	4333.7530	0.05%
NetHEPT-500-LT	5587.1591	5588.4460	0.02%

TABLE II
ACCURACY TEST RESULT

my project is that the result may be more accurate if time is further exploited.

To sum up, The Influence Maximization problem is a very classic and valuable problem.In this project, I learned valuable experience of solving a problem in reality.The process is of great significance—modeling problem as a graph structure, using simulation to solve an NP-hard problem, analyzing and implementing a greedy strategy algorithm and doing the performance and correctness test.It really let me use the knowledge I learned.

ACKNOWLEDGMENT

I would like to thank Ms.*Zhao* for her patient and thoughtful guidance.Without her guidance, I would have much more difficulty to understand the wonderful algorithm principle of IMP.

REFERENCES

[1] Y. Tang, Y. Shi, and X. Xiao. Influence Maximization in Near-Linear Time:A Martingale Approach. In ACM SIGMOD International Conference, pages 1539-1554, 2015.