Project IMP Report

Zhaoxu Zhang 11611308

***Abstract*—This is a project for CS303-Aritificial Intelligence, which is about a famous problem "Influence Maximization Problem". This project requires us to implement an algorithm to solve the IMP problem. Each student's program will be upload to an online platform and tested with multiple testsets.**

I. Preliminaries

A. Problem Description

The problem can be divided into two subproblems: ISE, IMP.

- ISE: To estimate the maximum $\sigma(S)$ with a given seed set S in a network using a certain stochastic diffusion model.
- IMP: To find a seed set S that maximizes $\sigma(S)$ in a network using a certain diffusion model, subject to S = k.

A social network is modeled as a directed graph $G = V$, E with nodes in V modeling the individual in the network and each edge $u, v \in E$ is associated with a weight $w, u, v \in [0, 1]$ which indicates the probability that u influences v. Let $S \subseteq V$ to be the subset of nodes selected to initiate the influence diffusion, which is called the seed set.

The stochastic diffusion models specify the random process of influence cascade from S, of which the output is a random set of nodes influenced by S. The expected number of influenced nodes $\sigma(S)$ is the influence spread of S. There are two diffusion models: Independent Cascade(IC) and Linear Threshold (LT) model.

- IC: A seed node activate its neighbors with the probability propotionally to the weight between them.
- LT: At the beginning, each node v selects a random threshold θ_V uniformly at random in range [0,1]. An inactive neighbor v for a active seed becomes activated if $\sum_{activated\, neighbors\, u} w(u, v) \geq \theta$

B. Problem Applications

This problem originates from viral marketing [1], where a company provides free samples of a product to a set of influential individuals in a social network, aiming to create a cascade of product adoptions via the word-of-mouth effect.

C. Software

This project is written by python 3.6 using IDE pycharm. The library include numpy, random, sys, time, multiprocessing, heapq, operator, math.

D. Algorithm

This project mainly use IMM[2] and CELF[3].

II. Methodology

A. Notation

- network: a n*n Graph containing the weight of each path
- invneighbors: a list with multiple length, containing nodes that have path to this node
- neighbors: a list with multiple length, containing neighbors for each node
- differential_model: 'IC' or 'LT'
- R: collection of all RR set
- RR: the collection of all node that might influence a node
- $F_R(S)$: the fraction of RR sets in R that are covered by a node set S

B. Data and data structure

In this project, I mainly use dict, list, heap and set data structure in python. In the following, I will demonstrate how I use data structure for each objects.

- network graph: I use adjacent table as this is quich to search a weight for a path O(1).
- neighbors list and invneighbors list: I use adjacent list. There always be situations that we need to go through all neighbors for one node. Because it is a parse gragh, using adjacent list to go throgh is both time-efficient and space-efficient.
- RR set: I use set. It is because that in IMM when we calculate $F_R(S)$ we need to check wether a node in a RR. Here we use set will shorten the checking time to O(1) instead of O(n) using list.
- maxmum queue: I also use heap in package 'heapq' to maintain a maximum queue in CELF.

C. Model Design

How to fomulate: First I need to understand what is ISE, IMP, IC and LT. Then I will decide to choose which algorithm. Then I need to decide which data structure to use and how to implement the algorithm. In the following, I will present how to solve IMP and ISE problem.

- ISE: For ISE problem, we need to estimate the max influence for a given seed set. I run the diffusion models for 10000 times and take the average value as the estimated value.
- IMP: For IMP problem, we need to find out a seed set with given size in order to achieve the max influence. Here I apply IMM algorithm to solve it. In the initialization process, we generate a collection of RRs which are generated by applying reverse diffusion model to a random node. Then we choosed the top k nodes that are covered most in those RRs as seed set. Here, everytime we add one node into seed set. We move all RR that contains this node and we can use CELF algorithm to accelerate this algorithm.

D. Detail of Algorithm

Here we represent the main algorithm for ISE and IMP.

- ISE
  - function used to calculate ISE

```
1: function ISE(network,seedset)
2:     sum = 0
3:     result = 0
4:     for i from 1 to 10000 do
5:         result = IC or LT
6:         sum += result
7:     end for
8: return sum/N
9: end function
```

  - function used for model IC to calculate estimation

```
1: function IC(network,seedset)
2:     activityset = seedset
3:     count = activityset.length
4:     while activity set is not empty do
5:         $newActivityset = \emptyset$
6:         for each seed in activitySet do
7:             for each inactive neighbor in seed do
8:                 seed activates its neighbors according to the weight
9:                 if neighbor is activated then
10:                    Refresh neighbor's state
11:                    newActivityset.add(neighbor)
12:                end if
13:            end for
14:        end for
15:        count = count + newActivityset.length
16:    end while
17: return count
18: end function
```

  - function used for model LT to calcualte estimation

```
1: function LT(network,invneightbors)
2:     Initialize each node with a threshold, if the threshold is 0, add this node into activityset
3:     count = activityset.length
4:     while activityset is not empty do newactivityset = $\emptyset$
5:         for each seed in Activity do
6:             for each inactive neighbors in seed do
7:                 w_total = sum of all weight for all activited neighbors of this node
8:                 if w_total$\leq$neighbors.thresh then
9:                     Refresh neighbor's state
10:                    newActivityset.add(neighbor)
11:                end if
12:            end for
13:        end for
14:        count = count+newactivityset.length
15:        activityset = newactivityset
16:    end while
17: return count
18: end function
```

- IMP
  - IMM method

```
1: function IMM(graph,k,$\varepsilon$,l)
2:     $l = l * (1 + log2/logn)$
3:     R = Sampling(G,k,$\varepsilon$,l)
4:     $S_k^*$ = NodeSelection(R,k)
5: return $S_k^*$
6: end function
```

  - functions used to generate R

```
1: function SAMPLING(graph,k,$\varepsilon$,l)
2:     Initialize a set R = $\emptyset$ and LB = 1
3:     $\varepsilon 1 = \varepsilon$
4:     for i = 1 to $log_2 n - 1$ do
5:         $x = n/2^i$
6:         $\lambda 1 = \frac{(2+\frac{2}{3}*\varepsilon')(log\binom{n}{k}+log(1/\sigma_3))}{\varepsilon'^2} * \frac{n}{x}$
7:         $\theta_i = \lambda 1/x$
8:         while $|R| \leq \theta_i$ do
9:             Select a node v from G uniformly at Random
10:            Generate an RR set for v, and insert it into R
11:        end while
12:        $S_i$ =NodeSelection(R)
13:        if n*$F_R(S_i) \geq (1 + \varepsilon 1) * x$ then
14:            LB = n*$F_R(S_i)/(1 + \varepsilon 1)$
15:            break;
16:        end if
17:        $\alpha = \sqrt{l * logn + log2}$
18:        $\beta = \sqrt{(1 - 1/e)(log\binom{n}{k} + l * logn + log2)}$
19:        $\lambda^* = 2n * ((1 - 1/e) * \alpha + \beta)^2 * \varepsilon^{-2}$
20:        $\theta = \lambda^*/LB$
21:        while $|R| \leq \theta$ do
22:            Select a node v from G uniformly at random
23:            Generate an RR set for v and insert it into R
24:        end while
25:    end for
26: return R
27: end function
```

  - function used to select k seed set using collection R

```
1: function NODESELECTION(R,k)
2:     $S_k^* = \emptyset$
3:     for i = 1 to k do Identify the node v that maximizes $F_R(S_k^*) - F_R(S_k^*)$ Insert v into $S_k^*$
4:     end for
5: return $S_k^*$
6: end function
```

III. EMPIRICAL VERIFICATION

A. Dataset

Here I use Gnutella30 dataset to test my program. In this dataset there is 36682 nodes and 88328 edges.

B. Performance Measurement

- Test envirment: MacOS Mojave version 10.14; Processor 2.3 GHz Intel Core i5 4 cores; Memory 8 GB 2133 MHz LPDDR3.
- How to measure: For ISE, I use the total time and the accuracy of the estimation to evaluate(however without a reference answer, I can't judge wether my answer is correct). For IMP, I use total time and the estimation influence of the output seed set to measure the performance.

C. Hyperparameters

For ISE, the parameters is the the size of iterations.This parameter decide the number of iteration which also connect with the speed. For IMP, the parameters is the input for IMM algorithm, which decide the size of R set. With a larger size ,we need spend more time to construct it.

In the paper, the author assigns following values to them:

ε	l	size
0.1	1	10000

I try to make these parameter to be smaller in order to maket the program faster. However, after being modified, the output is extramely unaccurate. At the same time, I added multiprocess in my program which accelerate it dramatically. Therefore, I decided not to modify the parameters.

D. Experimental Results

Here is the result on the dataset Gnutella30

IMP(k=5)	time/s	grade
IC	668.9	35632
LT	1.32	35589

ISE(seedsize=5)	time/s	grade
IC	80	35632
LT	213.6	35589

E. Conclusion

As a state-of-art algorithm, the IMM algorithm perform quite well.

In my point of view, such a great performance have strong connection with the data structure that we use. For example, using adjacent list is much better than adjacent matrix. It is also due to how we implement the algorithm. For example, in the NodeSelection method, after find out one point, we need to remove all RRs that contains that node. Here we can't immediately delete it. Instead, we can label it. However, there are also some disadvantage in this algorithm. For example, for 'IC' model, it's performance is not so good compared with 'LT' model, for which I don't know why.

IV. Acknowledgements

I want to thanks for Doc. Tang, Doc. Shi, Doc. Xiao who created IMM algorithm. I also want to thanks for TA Yao Zhao who clarify my confusion about this project. At last I would like to thank forward to all the student assistances who share themselve's experience and maintain the online judge system.

References

[1] P. Domingos and M. Richardson. Mining the network value of customers. In KDD, pages 57–66, 2001.

[2] Tang Y, Shi Y, Xiao X. Influence maximization in near-linear time: A martingale approach[C]//Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015: 1539-1554.

[3] J. Leskovec et al. Cost-effective outbreak detection in networks. In KDD 2007.