Influence Maximization Problem Report

Using IMM to solve the problem

Kai WANG 11612909
School of Computer Science and Engineering
Southern University of Science and Technology
Email: 11612909@mail.sustc.edu.cn

***Abstract*—This is a report for the third project in AI class, the influence maximization problem (IMP). The problem IMP is a popular question which has been researched by lots of people. The challenge is how to find the optimal seeds with maximum influence in the shortest possible time. For solving this problem, I researched the method IMM. I learned a lot from this method and implemented part of it.**

***Index Terms*—IMP, ISE, IMM, LT, IC**

I. Preliminary

A. Problem Description

The influence maximization problem(CARP) can be described as follows: a social network G = (V, E), with a set of vertices denoted by V, a set of directed edges denoted by E.Each directed edge has a weight which represent the probablity of influence. There are two diffusion models, IC and LT used in this project which decide a actived node how to influence others. A solution to the problem is a seed set which has biggest influence in social network when actived them at first:.

B. Problem Applications

Nowadays, social network is becoming more valuable. How to influence or transmate some information to maximum numbers of people in lowest cost has got a lot attention. This project can be used in lots of scenes, for exemple:

- Advertising Stats
- Shops Recommended

II. Methodology

A. Notation

- **G**: a social network G with a node set V and an edge set E;
- **n**: number of nodes;
- **m**: number of edges;
- **k**: the size of the seed set for influence maximization;
- **RR**: research reachable set;
- **MAX**: 9999, a constraint factor used in programing;
- **process**:number of multiprocess
- **I(S)**:the influence of a node set S in a diffusion process on G;
- **Rtimes**:the times of ISE;

B. Data Structure

- **NETWORK**:In ISE, a two-dimensional list which stores all edges with t heir out-neighbors and weight;
- **NETWORK**:In IMP, a two-dimensional dictionary which stores all edges with their in-neighbors and weight;
- **Rset**: a list which stores RR sets;

C. ISE and diffusion modle design

Task one in this project is to find the maximum size of influenced nodes of a given seed set. This is also a function helps us to estimate the effect of the seed. There are two kinds of diffusion modles, IC and LT, which decide the way to spread influence, I will introduce them in detail soon.

IC, the independent cascade model, The IC model originates from the marketing literature [22,23], and it assumes that each edge e E is associated with a probability p(e) [0, 1]. For any node u and any of its outgoing neighbors v,if u is first activated at timestamp i, then it has p(¡u, v¿) probability to activate v at timestamp i + 1. In other words, whether or not u can activate v is independent of the history of diffusion before u is activated, and hence, the order of node activations does not affect the diffusion results.

LT, the linear threshold model. The LT model is another diffusion modle. For any node u and any of its in-neighbors v, u has a probality P(u), only when the sum of p(¡v, u¿) of all actived in-neighbor v of u bigger than P(u), u will be actived.

In ISE task, I apply Monte Carlo Method to simulate diffusion process. After $Rtimes$ times, find the mean value as the influence size.

D. IMP Model design

The first thing I need to do is to load the data in properly data structure. In IMM algorithm, for each node, we only need to consider about its in-neighbor nodes, so we need to store these messages. I use $NETWORKT$, a two-dimension dictionary. The keys are every nodes, and values are list of their in-neighber nodes and weights.

And then apply the IMM algorithm which can split to two parts,$sampling()$ and $nodeselection()$.

$sampling()$:This phase iteratively generates random RR sets and puts them into a set $Rset$, until a certain stopping condition is met.

For generating RR for node u, we need to use iteration method. Firstly, in all in-neighbor nodes v of u, choose them in probablity p(¡v,u¿) to actived and add them to RR. Then do the same thing on the new actived nodes until no nodes to be actived. Finally return RR.

$nodeselectin()$:This phase applies the standard greedy algorithm for maximum coverage to derive a size-k node set Sk that covers a large number of RR sets in $Rset$. It then returns Sk as the final result.

It worth to noting that, $nodeselectin()$ need to be optimized, because it will be really slow if we use the simple algorithm. The details will be showed in pseudocode.

And in $sampling()$, I use multiprocessing to generate $Rset$ as fast as possible.

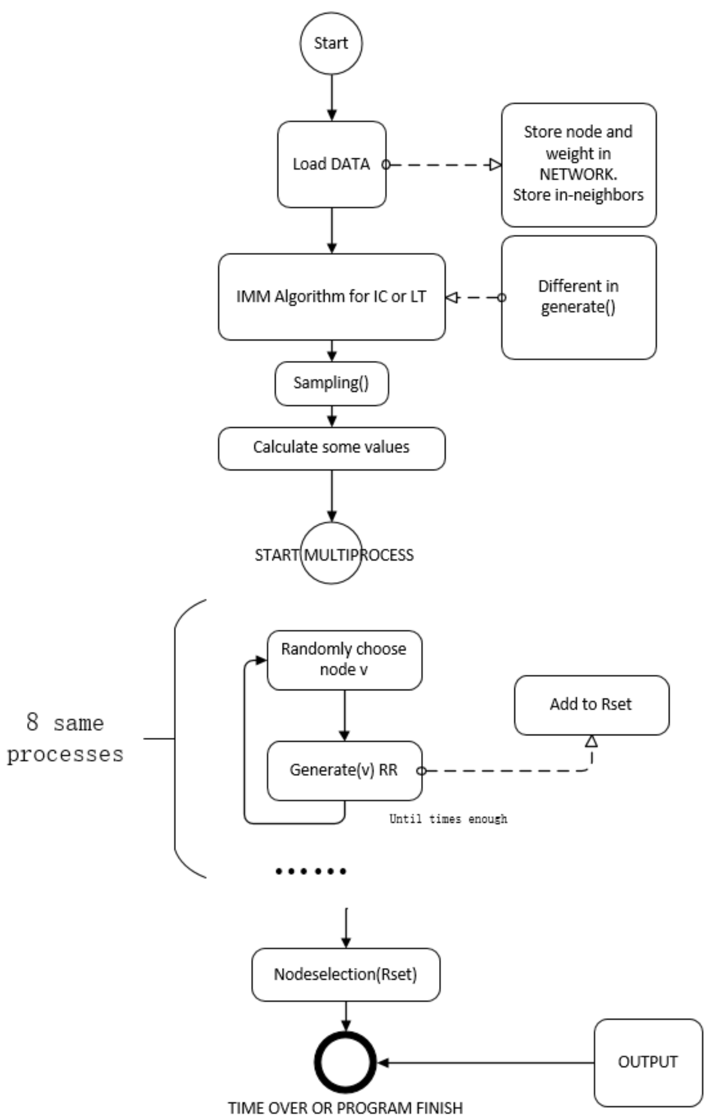

Fig. 1. Model Achitecture

E. Detail of algorithms

1-6 pseudocodes have showed the details of IMM algorithems. There are some necessary formulas used in IMM:

$$\alpha = \sqrt{l * \log n + \log 2}$$

$$\beta = \sqrt{(1 - 1/e) * (\log C(n, k) + l * \log n + \log 2)}$$

$$lambda_a = 2n * ((1 - 1/e) * \alpha + \beta)^2 * Epsilon^{-2}$$

$$lambda_b = \frac{(2 + \frac{2}{3} Epsilon)(\log C(n, k) + l \log n + \log \log_2 n)n}{Epsilon^2}$$

For different diffusion models, IC and LT, there are a little different in $generate()$ function. For IC, I use iteration method to add node to Rr under the probability given by weight $p(< uj, u >)$. But for LT, I suppose that there must be a node and only a node would be actived of an actived node's in-neighbers. So i just need to do iteration and randomly select one of the in-neighbors to add in RR.

I find that when the $NETWORK$ is a big network, $Rset$ will be a huge set. That makes program runing time too long, So I apply multiprocessing to help program to find the $Rset$.

Algorithm 1 ICsearch

Input: $SEED$
Output: $count$

```
active ← SEED
actived ← SEED
count += len(actived)
while active is not empty do
  newactive = []
  for node u ∈ active)) do
    for node v ∈ in-neighbor of u do
      if v ∈ actived then
        continue
      end if
      p = random(0,1)
      if p > p(< v, u >) then
        newactive.append(v)
        actived.append(v)
      end if
    end for
  end for
  active = newactive
  count += len(newactive)
end while
return count
=0
```

Algorithm 2 LT search

Input: $SEED$
Output: $count$

```
active ← SEED =0
```

Algorithm 3 LT search (continued))

Input: $SEED$
Output: $count$

$active \leftarrow SEED$
$actived \leftarrow SEED$
$count$ += len($actived$)
for node in $network$ **do**
  $pnow(node)$ = 0
  $threshold(node)$ = random(0,1)
end for
while $active$ is not empty **do**
  $newactive$ = []
  for node u $\in$ active)) **do**
    for node v $\in$ in-neighbor of u **do**
      if v $\in$ $actived$ **then**
        continue
      end if
      $pnow(u)$ = random(0,1) + $pnow(u)$
      if $pnow(u) > threshold(u)$ **then**
        newactive.append(v)
        actived.append(v)
      end if
    end for
  end for
  $active$ = $newactive$
  $count$ += len($newactive$)
end while
return $count$
=0

Algorithm 4 sampling

Input: $NETWORK$, K, $Epsilon$, l
Output: $Rset$

$Rset = []$
$LB = 1$
$Epsilon_{new} = 2^{1/2} * Epsilon$
for i = 1 to log2(n)-1 **do**
  $x = b/2^i$
  $theta = lambda_a/x$
  while $len(Rset) <= theta$ **do**
    $v \leftarrow randomlyfromNETWORK$
    $Rset+ = generate(v)$
  end while
  $Si, Fractioni = nodeselection(Rset)$
  if $n * Fractioni >= (1 + Eplison_{new}) * x$ **then**
    $LB = n * Fractioni/(1 + Eplison_new)$
    break
  end if
end for
$theta = lamba_b/LB$ =0

Algorithm 5 sampling (continued)

while $len(Rset) <= theta$ **do**
  $v \leftarrow randomlyfromNETWORK$
  $Rset+ = generate(v)$
end while
return Rset
=0

Algorithm 6 nodeselection

Input: $Rset$, k
Output: Sk,$fraction$

$Sk = []$
$Rdict = []$
$num = []$
$total = len(Rset), active = 0$
for i = 0 to total **do**
  for node j in Rset[i] **do**
    num[j] += 1
    Rdict[j].append(i)
  end for
end for
while len(Sk) ¡ k **do**
  $maxnumber = max(num)$
  $seed = num.index(maxnumber)$
  $active+ = maxnumber$
  $Sk.append(s)$
  for each node in $Rdict(seed)$, remove them form other $Rdict$ and num changed
end while
$fraction = active/total$
return Sk, fraction
=0

III. Empirical Verification

A. Dataset

All datasets from the platform: $network-5-IC$, $network-5-LT$, $NetHEOT-5-IC$, $NetHEPT-5-LT$, $NetHEPT-50-LT-bonus$, $NetHEPT-50-LT-bonus$

B. Performance measure

Given a time, look at the difference between the solution given at the end of the program and the optimal solution.

Test envirment is given by CARP-Oj-Platform.

C. Hyperparameters

$Epsilon = 0.07$
$l = 1$
$process = 8$

D. Experimental results

See table 1.

Amoung these six datasets, the solution is not very well but also far from worst. It performs well in little size network.

Algorithm 7 generate

Input: v, $NETWORK$
Output: RR

$Rnew = []$
for node u in $NETWORK[v]$ **do**
  $Rnew.append(u)$
  $RR = Rnew$
end for
while $Rnew! = []$ **do**
  $NEW = []$
  for node u in $Rnew$ **do**
    for node uj in $NETWORK[u]$ **do**
      p = random(0,1)
      if $p <= p(< uj, u >)]$ **then**
        if uj not in RR **then**
          $NEW.append(uj)$
          $RR.append(uj)$
        end if
      end if
    end for
  end for
  $Rnew = NEW$
end while
return RR
=0

Algorithm 8 IMM

Input: k, $NETWORK$, $Eplison$, l
Output: S

$Rset = sampling(NETWORK, k, Eplison, l)$
$S = nodeselectiion(Rset, k)$
return S
=0

The disadvantages of my program is that it doesn't perform well on big data sets. For big datasets, it will be really slow and have a relative bigger bias.

ACKNOWLEDGMENT

I would like to thanks my classmates who discussed algorithms with me which help me finish this project. And I also want to thanks for TA Yao Zhao who teached algorithms and guided me to research. Last I would like to thank forward to all the student assistances who will assess my codes and reports.

TABLE I
EXPERIMENTAL RESULTS

Dataset	time(s)	optimal	MySeed
$network - 5 - IC$	1.6	30.73	30.68
$network - 5 - LT$	1.4	37.54	36.2
$NetHEOT - 5 - IC$	20.6	324.16	317.09
$NetHEPT - 5 - LT$	17.0	392.98	392.96
$NetHEPT - 50 - LT - bonus$	40.6	1294.55	1298.10
$NetHEPT - 50 - LT - bonus$	32.87	1652.49	1702.00

REFERENCES

[1] Youze Tang, Yanchen Shi, Xiaokui Xiao, "Influence Maximization in Near-Linear Time: A Martingale Approach," Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015, 1539-1554.