IMP Report

Yifei Xu 11611209
Southern University of Science and Technology

1. Preliminaries

Given a social network G and a positive integer k, the *influence maximization* problem asks for k nodes (in G) whose adoptions of a certain idea or product can trigger the largest expected number of follow-up adoptions by the remaining nodes. This problem has been extensively studied in the literature, and the state-of-the-art technique runs in $O((k + l)(n + m) \log n/\varepsilon^2)$ expected time and returns a $(1 - 1/e - \varepsilon)$-approximate solution with at least $1 - 1/n^l$ probability.

1.1 Introduction[1]

Problem Statement. Given G, a diffusion model M , and a positive integer k, the influence maximization problem asks for a size-k node set S_k with the maximum expected influence $E[I(S_k)]$.

In my project, I implemented ***IMM***[1] to solve the problem.

1.2 Application

There are various applications of the influence maximization problem. In the real life, advertisement companies can find an optimal way to do advertising and reduce the cost.

2. Methodology

This part is to introduce the details of algorithm and the architecture of my program.

2.1 Notation

Notation	Description
$G(V, E)$	a social network G with a node set V and an edge set E
n, m	the numbers of nodes and edges in G, respectively
k	the size of the seed set for influence maximization
$P(e)$	the possibility of triggered edge ***e***
g	the distribution of social networks induced by the triggering distribution of each node (see the sampling algorithm)
R	the set of *RR* sets generated by ***IMM's*** sampling phase
$F_R(S)$	the fraction of *RR* sets in R that are covered by a node set S
θ	the number of *RR* sets in R
ε	related to the accuracy of the final result (negative correlation)

2.2 Data Structure

- *Worker*
  Pipeline processors
- *Network*
  Store the information of a network dataset.
  e.g. The weight of the edge e from a to b.
- *Node*
  Store the information of a vertex.
  e.g. The reverse neighbors of node v.

2.3 Model Design

As following are the steps of solving a carp problem.

1. Read and resolve the network in datasets.
2. Sampling the network and find a set of all ***RR*** sets.
3. Using **Greedy(CELF[2])** algorithm to select ***k*** nodes.

2.4 Details of Algorithms

NodeSelection : This method is to accelerate the process of greedy algorithm by using **CELF** and find the top-k seeds having the maximum influence.

Input: R : list, k : int
Output: S_k : set

$S_k \leftarrow \varnothing$
iteration $\leftarrow 0$
repeat
  Identify the vertex v that maximizes
  $F_R(S_k \cup v) - F_R(S_k)$
  # to ensure v is up-to-date
  if *iteration* = v_i **then**
    Insert v into S_k
  else
    $v_i \leftarrow v_i$+1
until *iteration* = k
return S_k

Sampling **:** This method is to generate a ***R*** , which

$$\lambda' = \frac{(2+\frac{2}{3}\varepsilon')\cdot\left(\log\binom{n}{k}+\ell\cdot\log n+\log\log_2 n\right)\cdot n}{\varepsilon'^2}. \quad (1)$$

$$\lambda^* = 2n\cdot\left((1-1/e)\cdot\alpha+\beta\right)^2\cdot\varepsilon^{-2}. \quad (2)$$

Input: G: graph, k: int, ε: float, l: float
Output: S : solution

$R \leftarrow \varnothing$
$LB \leftarrow 1$
Let $\varepsilon' \leftarrow \sqrt{2}\cdot\varepsilon$
for $i \leftarrow 1$ to $\log_2(n-1)$ ***do***
  $x \leftarrow n/2^i$
  $\theta_i \leftarrow \lambda'/x$, where λ' is as defined in (1)
  while $|R| \le \theta_i$ **do**
    Select a node v from G uniformly at random
    Generate an RR set for v, and insert it into R
  $S_i \leftarrow$ **NodeSelection**(R, k)
  if $n \cdot F_R(S_i) \ge (1+\varepsilon')\cdot x$ **then**
    $LB \leftarrow n \cdot FR(Si)/(1+\varepsilon')$
    break
$\theta \leftarrow \lambda^*/LB$, where λ^* is as defined in (2)
while $|R| \le \theta$ **do**
  Select a node v from G uniformly at random
  Generate an RR set for v, and insert it into R
return R

IMM : This method is to find the final result of the given network G

Input: G: graph, k: int, ε: float, l: float
Output: S_k : set

$l \leftarrow l \cdot (1 + \log_2/\log_n)$
$R \leftarrow$ ***Sampling***(G, k, ε, l)
$S_k \leftarrow$ ***NodeSelection***(R, k)
return S_k

3. Empirical Verification

This part is to introduce the performance of this algorithm on different datasets.

3.1 Dataset

- network dataset
- NetHEPT dataset
- Gnutella30 dataset

3.2 Performance Measure

I. **Performance**

Fig.1 is the performance on network datasets.
Fig.2 is the performance on NetHEPT datasets.
Fig.3 is the performance on NetHEPT datasets.

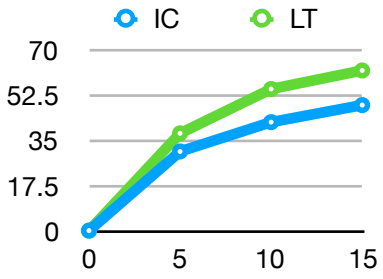

Fig.1

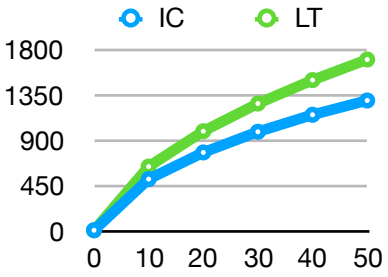

Fig.2

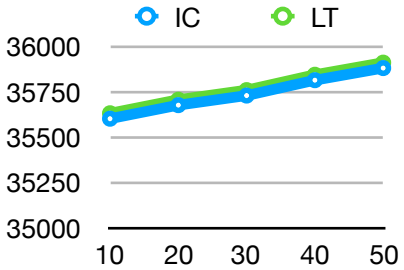

Fig.3

II. **Test Environment**

CPU：	Intel Core i7
Frequency：	2 GHz
# Cpu：	1
# Core：	4
Python version:	3.7

3.3 Hyperparameters

Hyperparameter	Value
ε	0.1
l	1

According to practice and **IMM**[1] essay, $\varepsilon = 0.1$ is suitable. This based on the accuracy of the final result and the running time.

3.4 Experimental

From Fig.1 and Fig.2, for the datasets network, this algorithm can find the best solution in most cases. But for datasets NetHEPT and val, the algorithm is able to find relatively good solutions but not the best solutions.

3.5 Conclusion

Advantages : The process of generating *RR* sets solutions is fast.

Disadvantages : When it comes to a network with all nodes that can active their neighbors every time, IMM algorithm can not find the result quickly and it may exceed the time limitation.

4. References

[1] 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.
[2] J. Leskovec et al. Cost-effective outbreak detection in networks. In KDD 2007.