IMP Report

Xinhuai Deng 11610320
School of Computer Science and Engineering
Southern University of Science and Technology
Email:11610320@mail.sustc.edu.cn

1. Preliminaries

In this section, the IMP description and application will be mentioned.

1.1. Problem Description and application

Influence Maximization Problem(IMP) originates from Viral Marketing, where a company provides free samples of a product to several influential individuals in a social network so that they can market the product to others. It is a way of Data mining and it has a lots of applications such as products maketing and advertising. For example, in facebook, we can use this algorithm to find some influential users and pay money for them to advertise products. In 2003, Kempe et al. [3] formulate influence maximization as an NP-hard combinatorial optimization problem on the two pioneering diffusion models, namely, Independent Cascade (IC) and Linear Threshold (LT). And they design an approximation algorithm using greedy providing $(1 - 1/e)$ approximations.

This problem is described as (G, k). Given a social network G and a positive integer k. Then the output is the k influential individuals that will maximize the number of influenced nodes (which also called influence spread) in G through some special diffusion model. And in following section, I will introduce detail of two fundamental diffusion model named IC and LT.

2. Methodology

In this section, the frequently used notation will be demonstrated first. And thr key data structure used to solve this problem will be discussed. And the final part are the model design and detail of algorithms.

2.1. Notation

The frequently used notations used in this report are shown in table 1,

2.2. Key Data Structure

As introduced before, IMP are basically composed of a graph $G = (V, E)$. And for python 3.6, I use a dict-of-dict structure to represent the graph. It is actually a attributed graph with the probablity $p(e)$ as the attribute. So for the dict-of-dict data structure, a exmaple is shown below table2. In this example, $p(i, j)$ is equivalent to $p(e)$. And for convenience to generating the RR sets, I also stored the transposition G^T with all edges reversed.

And I store all the RR sets in $\mathcal{R}$. RR set is kept in set data structure and $\mathcal{R}$ is as a list of sets because $\mathcal{R}$ should be mutable.

TABLE 1. FREQUENTLY USED NOTATIONS.

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
$d_{in}i$	indegree of node i
$I(S)$ or $I_G(S)$	the influence of a node set S in a diffusion process on G.
IMM	a IMP algorithm which give a $(1-1/e-\varepsilon)$ approximation in near-linear time proposed by Tang et al [4].
LT	Linear Threshold diffusion model
IC	Independent Cascade diffusion model
RR set	The reserve reachable set
R_u	a random RR set motivated by node u
$\mathcal{R}$	the set of RR sets generated by *IMM*'s sampling phase
$\mathcal{F}_{\mathcal{R}}(S)$	the fraction of RR sets [4] in R that are covered by a node set S
$p(e)$	the probability that a edge will be taken when diffusing

TABLE 2. DICT-OF-DICT EXAMPLE

key	{key:probablity}
i	{j : $p(i, j)$}
	{k : $p(i, k)$}
	{l : $p(i, l)$}

2.3. Model Design and Details of Algorithm

First of all, in order to solve IMP, I read serveral papers. And I implement a improved greedy algorithm based on [3] named CELF++ [2]. This algorithm can provide $(1 - 1/e)$ accurancy. Since it is just a greedy algorithm with a heap structure, I just show a basic greedy frame in Algorithm1. However, because evaluating $I(S)$ is $\#P$-hardness [5], when the G is very large, it takes days to generate a seed set [3]. So I must change to another algorithm. Finally, I choose algorithm which has been categorized as a sketch-based algorithm in [5] called IMM [4].

Algorithm 1 Greedy(G, k)

```
the seed number $k$ and the social network $G = (V, E)$. $S$: the seed set $S \leftarrow \emptyset$ $i \leftarrow 1, ..., k$ $u \leftarrow$ argmax$_{u \in (V-S)}$( $I(S \cup u)$-$I(S)$ ) S ← S ∪ $u$ **return** S
```

And now I will introduce the main steps and the main algorithms I implemented in IMM. At first, IMM contains two steps:

1) Do sampling for G to generate a $\mathcal{R}$ which contains many RR sets.And a RR set contain the influential nodes in one "sampling".
2) Finding Max-Coverage: Use the greedy algorithm for the Max-coverage problem [6] to find a seed set S that covers the maximum number of RR sets and return S as the solution. Note that the standard greedy algorithm for Max-Coverage will just derive a $(1 - 1/e)$ approximate solution.

As shown in Algorithm2, we firstly do sampling to get all the RR sets stored in $\mathcal{R}$ and then select the k influential nodes from $\mathcal{R}$. And these two parts will be explained in succession.

Algorithm 2 IMM$(G, k, \varepsilon, \ell)$

```
$\ell$ = $\ell$·(1 + log2/log$n$) $\mathcal{R}$ ←Sampling$(G, k, \varepsilon, \ell)$ $S \leftarrow$ NodeSelection$(\mathcal{R}, k)$ **return** S
```

2.3.1. Sampling.

Generally speaking, sampling is a way to narrow the node range which could be influential. And this is done by generate a RR set [1]. Here is the definition of RR set.

Definition 1 (REVERSE REACHABLE SET). Let v be a node in G, and g be a graph obtained by removing each edge e in G with 1 p(e) probability. The reverse reachable (RR) set for v in g is the set of nodes in g that can reach v. (That is, for each node u in the RR set, there is a directed path from u to v in g.)

Before illustrating how RR set is be generated, I would like to show the details of two fundamental diffusion models. Because the way of generating RR sets varies from different diffusion models.

Definition 2 (Independent Cascade). Independent Cascade(IC) is a widely-used fusion model. It considers a vertex v is activated by each of its incoming neighbors independently by an influence probability $p(u, v)$. And this probablity is calculated as $1/d_{in}v$ where the $d_{in}v$ is the in-degree of v. Each node can only used to active its neighbours only right after the timestamp it is actived, i.e., if u is actived by a node at timestamp i, it can only active its neighbors at timestamp i+1.

Definition 3 (Linear Threshold). The second one is LT. Linear Threshold Model(LT) is also a seminal diffusion model. The basic idea is that a vertex is activated only if the sum of its incoming actice neighbours' influencial power is large than a threshold value. And each node will have a random threshold. And the influencial power for one incoming is calculated as the same as the propagation probablity which is $1/d_{in}v$ for a node v.

And the key concept of Sampling is that pick a node u randomly and find all the nodes that may have influence to u. These nodes is just the RR set. The whole procedure is shown in Algorithm3.

Algorithm 3 RR(G)

```
$G = (V, E)$ a RR set $u$ ←a node picked randomly form G RR ← ∅ RR.append($u$) RR is not empty $i \leftarrow$ RR.pop(0) **each** $j$ points to $i$ Activate $j$ with probablity p$(i, j)$ Append $j$ to RR if $j$ is actived **return** all the nodes that have been added into RR.
```

And for the IC model, we set the propagation probability p(i, j) to 1/d, where d denotes the number of edges that share the same ending point with i. While for the LT model, p(i, j) is construsted as following steps: (i)we first assign each incoming neighbor of i including j a random number between [0,1]. (ii)Normalize these random number so that they sum up to one. (iii) p(i, j)=1 if $1/d_{in}i$ is larger than the random number of j, otherwise p(i, j)=0.

And the whole procedure of Sampling is just set some special count to control the total times of calling RR to guarantee a theoretical accurancy. Because if the total number of RR sets is too small, then the solution NodeSelection phase will not be very good. However, if the number becomes very large, the algorithm will spend a lots of time. So Tang et al. [4] proposed IMM which use a martingale approach to solve this problem. And my sampling phase is totally the same as Sampling in [4].

2.3.2. NodeSelection.

The algorithm corresponds to the standard greedy approach for the maximum coverage problem [6], which guarantees that $\mathcal{F}_{\mathcal{R}}(S)$ is at least $(1 - 1/e)$ times the fraction of RR sets covered by any size-k node set. And the details are shown in Algorithm4.

Algorithm 4 NodeSelection$(\mathcal{R}(S), k)$

```
the seed number $k$ and the social network $G = (V, E)$. $S$: the seed set $S \leftarrow \emptyset$ $i \leftarrow 1, ..., k$ $u \leftarrow$ argmax $_{u \in (V-S)}$ ( $\mathcal{F}_{\mathcal{R}}(S \cup u) - \mathcal{F}_{\mathcal{R}}(S)$ ) S ← S ∪ $u$ **return** S
```

3. Empirical Verification

This part is about the performance and the testing method of my program. In the following section, I will give a detail explaination of the data set and the perfomance. After that, I will analyse the reason why it works like that.

3.1. Date set

Beside the data set provided by OJ platform, I also test my program, I also use the data mentioned in [4] to judge whether my implementation is good or not. The first network is from the full paper list of the ”Physics” section of the e-print arXiv, denoted as NetPHY. And the second network is from the full user list of the Amazon, denoted as Amazon. And the third network is download from QQ group denoted as DBLP. The detail information are shown in table3.

TABLE 3. DATASETS.

Name	n	m
NetHEPT	15.2K	32.2K
NetPHY	37.2k	231.6k
DBLP	425.7K	15.8K
Amazon	548.5K	13.5K

3.2. Perfomance Measure

To evaluate my program performance, for the data set provided by platform, I just submit my code and compare my algorithm’s infuence spread and running time with others. While for the other data set, I first test the running time in my own computer and compute the influence spread. Because these are all widely used data sets, I can easily compared my influence spread with the results shown in different papers. And I don’t care about the exact number of the influence spread, Instead, I just to compare whether my results is at the same level with all others’ algorithm in different papers. And it shows that all of my results can meet a (1-1/e) approximation. So I will focus on the running time upon different data sets instead of the influence spread. And here is the information about my laptop.

TABLE 4. BASIC INFOMATION ABOUT MY LABTOP

CPU	Intel(R)Core(TM) i3-4005U
Frequence	@1.70Hz @1.70Hz
RAM	4GB
System	x64 Windows7

3.3. Hyperparameters

In IMM, two parametes should be set. One is ε and the other is ℓ. And IMM will returns a $(1-1/e-\varepsilon)$- approximate solution with at least $1 - 1/n^{\ell}$ probability. And I set ε as 0.5 and ℓ as 1, which is recommanded in [4].

3.4. Experimental Results

The results tested in my laptop shows in the following table5 (all value are rounded to integer).

TABLE 5. EXPERIMENTAL RESULTS

Data set	k	spread(IC)	spread(LT)	running time
Amazon	5	82	102	41s(IC,LT)
Amazon	50	630	842	35s(IC),36s(LT)
DBLP	5	117	177	24s(IC,LT)
DBLP	50	651	987	25s(IC,LT)
NetHEPT	5	321	390	1s(IC),2s(LT)
NetHEPT	50	1286	1670	3s(IC,LT)
NetPHY	5	314	731	6s(IC),8s(LT)
NetPHY	50	1687	2794	11s(IC),15s(LT)

3.5. Conclusion

This paper implement IMM proposed by Tang et al [4], an influence maximization algorithm runs in O((k + ℓ)(n + m)log n/ε^2) expected time and returns a $(1 - 1/e - \varepsilon)$-approximation with at least 1-1/n^{ℓ} probability under the IC and LT diffusion model. As stated in [4], IMM has the same approximation guarantee and time complexity as the state of the art, but achieves higher empirical efficiency with a novel algorithm design based on martingales. And my experimental results also show that IMM consistently outperforms the states of the art in terms of computation efficiency.

References

[1] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, ”Maximizing social influence in nearly optimal time.” In *SODA*, pages 946–957, 2014.

[2] A. Goyal, W. Lu, and L. V. S. Lakshmanan, ”CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Ntworks.” In *WWW*, pages 47–48, 2011.

[3] D. Kempe, J. M. Kleinberg, and É. Tardos. ”Maximizing the spread of influence through a social network,” In *KDD’03*, pp. 137146, ACM New York, NY, USA, 2003.

[4] Y. Tang, Y. Shi, and X. Xiao, ”Inuence maximization in near-linear time: A martingale approach,” In *Proceedings of SIGMOD International Conference on Management of Data”*, pp. 1539-1554, ACM, 2015.

[5] Y. Li, J. Fan, Y. Wang and K. Tan, ”Influence Maximization on Social Graphs: A Survey,” In *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 10, pp. 1852-1872, 1 Oct. 2018.

[6] S. Khuller, A. Moss, and J. S. Naor, ”The budgeted maximum coverage problem,” In *Information Processing Letters*, vol. 70, no. 1, pp. 3945, 1999.