Information Diffusion and Influence Maximization of Social Networks

Kemiao Huang, 11610728, *Undergraduate, CSE*

***Abstract*—Influence Maximization(IM), which selects a set of k users (called seed set) from a social network to maximize the expected number of influenced users (called influence spread), is a key algorithmic problem in social influence analysis. Due to its immense application potential and enormous technical challenges, IM has been extensively studied in the past decade. In this work, the Independent Cascade (IC) model and Linear Threshold (LT) model are generated and use efficient CELF method to find the seeds which maximize the influence. The implementation of this work shows competitive performance on the given dataset.**

***Index Terms*—Social networks, greedy algorithms, heuristic algorithms, diffusion models, influence maximization**

I. Preliminaries

This work aims to solve the problem of influence maximization (IM). As a key algorithmic problem in information diffusion research, it has been extensively studied recently due to its potential commercial value.

A. Problem Description

IM aims to select a set of k users in an online social network, aka. seed set with the maximum influence spread, i.e., the expected number of influenced users through the seed set in information diffusion is maximized.

B. Problem Application

A well-known application of IM is viral marketing[1], where a company may wish to spread the adoption of a new product from some initially selected adopters through the social links between users. Besides viral marketing, IM is also the cornerstone in many other important applications such as network monitoring[2], rumor control[3], [4], and social recommendation[5].

II. Methodology

A. Notations

1) G: social graph
2) V: vertices set
3) E: edges set
4) k: max size of seed
5) n: number of cores used in multiprocessing
6) e: tolerant error in evaluation

B. Data Structures

1) Priority Queue: To efficiently implement CELF, the max heap is used for insert and pop vertices with its margin influence.

2) Dictionary: The python dict is used for storing the neighbours.

3) Set: The set structure are used to store and add the activated vertices.

C. Model Design

Firstly build the graph with vertices and its neighbours with weights. For ISE, the seeds are diffused by independent cascade (IC) model and linear threshold (LT) model with several times of random tries. For IMP, CELF are used. When every vertex are included in the seed set, IC or LT are used for influence estimation. To realize the parallel computation, the multiprocessing pool are used to simulate the influence diffusion model for each process.

D. Algorithms

1) Independent Cascade Model: When a vertex u gets activated, initially or by another vertex, it has a single chance to activate each inactive neighbour *v* with the probability proportional to the edge weight *w(u,v)*. Afterwards, the activated vertices remain its active state but they have no contribution in later activations. The weight of the edge (*u*,*v*) is calculated as *w(u,v)* = $\frac{1}{d_{in}}$, where d_{in} denotes the in-degree of vertex *v*.

Algorithm 1 Independent Cascade

```
Input: $G, seed$
Output: influence spread of seed
  initialize $S \leftarrow seed$, $activated \leftarrow seed$
  while $S \neq \varnothing$ do
    $new\_S \leftarrow \varnothing$
    for $active \in S$ do
      for $inactive \in$ inactive neighbours of $active$ do
        tries to activate $inactive$ by $actuve$
        if succeed to activate then
          add $inactive$ to $new\_S$
        end if
      end for
    end for
    add $new\_S$ to $activated$
    $S \leftarrow new\_S$
  end while
  return $|activated|$
```

2) Linear Threshold Model: At the beginning, each vertex *v* selects a random threshold θ_v uniformly at random in range [0,1]. An inactive vertex *v* becomes activated if $\sum_{activeneighbouru} w(u,v) \geq \theta_v$. The weight of the edge *(u,v)* is calculated as $w(u,v) = \frac{1}{d_{in}}$, where d_{in} denotes the in-degree of vertex *v*.

3) Evaluation: To get the precise influence spread of the seed, we should implement the two diffusion models for large number of times. To stop the program after the result converged, a counter is used to check the times for current results which obey the error e.

4) Greedy Algorithm Optimized with CELF: One of the most notable work in improving the greedy algorithm is [2], where submodularity is exploited to develop an efficient algorithm called CELF, based on a ”lazy-forward” optimization in selecting seeds. The idea is that the marginal gain of a vertex in the current iteration

Algorithm 2 Linear Threshold

```
Input: $G, seed$
Output: influence spread of seed
    initialize $S \leftarrow seed$, $activated \leftarrow seed$
    set a random threshold for each vertex
    while $S \neq \varnothing$ do
        $new\_S \leftarrow \varnothing$
        for $active \in S$ do
            for $inactive \in$ inactive neighbours of $active$ do
                $w \leftarrow$ total weights of incoming active neighbours of $inactive$
                if $inactive.threshold < w$ then
                    add $inactive$ to $new\_S$
                end if
            end for
        end for
        add $new\_S$ to $activated$
        $S \leftarrow new\_S$
    end while
    return $|activated|$
```

Algorithm 3 Evaluation

```
Input: $G, model, seed$
Output: influence spread of seed after estimating for many times
    initialize $sum \leftarrow 0$, $rounds \leftarrow 0$, $count \leftarrow 0$
    set $N$ processors for pool
    $result \leftarrow$ sample output of $model$ with $seed$
    while $count < breakcount$ or time out do
        run $model$ with $seed$ asynchronously in pool
        $sum \leftarrow sum+$ output of model
        $rounds =\leftarrow rounds + 1$
        $current \leftarrow sum/rounds$
        if $|current - result| < e$ then
            $count \leftarrow count + 1$
        else
            $count \leftarrow 0$
        end if
        $result \leftarrow current$
    end while
    return $result$
```

cannot be better than its marginal gain in the previous iterations. CELF maintains a table $\langle u, \Delta_u(S)\rangle$ sorted on $\Delta_u(S)$ in decreasing order, where S is the current seed set and $\Delta_u(S)$ is the marginal gain of u w.r.t S. $\Delta_u(S)$ is re-evaluated only for the top vertex at a time and if needed, the table is resorted. If a vertex remains at the top, it is picked as the next seed. Leskovec et al.[2] empirically shows that CELF dramatically improves the efficiency of the greedy algorithm. To control the running time, the program should exit before time out. As a heuristic way to solve the time limitation problem, the algorithm will stop evaluation and add the first $k - |s|$ vertices in the max heap to the seed set S when the program is about to end.

III. Empirical Verification

A. Dataset

Considered that there are enough testing data for this project, I didn't use the other datasets from the website. I only used datasets that the course provided.

Algorithm 4 CELF

```
Input: $G, model, k$
Output: expected seed set
    initialize seed set $S \leftarrow \varnothing$, max heap $Q \leftarrow \varnothing$, $v.flag \leftarrow 0$ for each $v \in V$
    for $v \in V$ do
        $v.mg \leftarrow$get_influence$(G, v)$
        add $v$ to $Q$ by $v.mg$ in descending order
    end for
    while $|S| < k$ do
        $u \leftarrow Q.pop$
        if $u.flag = |S|$ then
            $S \leftarrow S + u$
            $Q \leftarrow Q - u$
        else
            $u.mg \leftarrow$ get_influence$(G, S + u)$ - get_influence$(G, S)$
            $u.flag \leftarrow |S|$
            insert $u$ into heap $Q$
        end if
    end while
    return $S$
```

B. Performance measurement

For ISE, the less time cost of the evaluation which can output the influence spread with tolerant error is considered as the better performance. For IMP, the larger influence spread of the seed set output by searching and evaluation not exceeding limit time is considered as the better performance.

C. Hyperparameters

To save the time when the evaluation result converges, the ”tolerant error coefficient and” the ”max count” are defined.

For IMP, the cost time of computing influence spread for each iteration in CELF should be much smaller. Therefore, the ”max rounds” for IMP is defined.

1) *Tolerant Error Coefficient:* 0.0001 - 0.01
2) *Max Count:* 5
3) *Max Rounds:* 50 - 1000

D. Experiment Results

The test result on my Dell laptop is shown in the tables. According to the result, the IC diffusion model is nearly half faster than LT model.

TABLE I: ISE Result (time limit: 60s)

	model	seed	error	count	time
network	IC	5	0.0001	5	1.27
network	LT	5	0.0001	5	1.81
NetHEPT	IC	50	0.0001	5	19.3
NetHEPT	LT	50	0.0001	5	39.8

TABLE II: IMP Result (time limit: 60s)

	model	seed	error	count	rounds	time	result
network	IC	5	0.0001	5	1000	2.50	30.55
network	LT	5	0.0001	5	1000	4.84	36.97
NetHEPT	IC	50	0.01	5	50	58.2	1005.1
NetHEPT	LT	50	0.01	5	50	59.3	1351.9

E. Conclusion

The IC model and LT model are both very classic diffusion model in IMP. The value of influence spread are different due to the mathematical influence probability is different. LT model is considered to have a larger spread than IC model. The greedy algorithm to realize IM is greatly improved by CELF method. Although there are some papers have proposed improved CELF like CELF++ and Lv_CELF, the practical performances are not better than classic CELF in this project. There are still a lot of space for improvement of my algorithm. The multiprocessing can be used better and the algorithm for CELF can be replaced as other more optimized algorithms.

IV. References

[1] P. Domingos and M. Richardson, ”Mining the network value of customers,” in KDD, 2001, pp. 5766.

[2] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, ”Cost-effective outbreak detection in networks,” in KDD, 2007, pp. 420429.

[3] C. Budak, D. Agrawal, and A. El Abbadi, ”Limiting the spread of misinformation in social networks,” in WWW, 2011, pp. 665674.

[4] X. He, G. Song, W. Chen, and Q. Jiang, ”Influence blocking maximization in social networks under the competitive linear threshold model,” in SDM, 2012, pp. 463474.

[5] M. Ye, X. Liu, and W.-C. Lee, ”Exploring social influence for recommendation: A generative model approach,” in SIGIR, 2012, pp. 671680.