Influence Maximization in Social Networks under Linear Threshold Model with Negative Opinion

Zilong Guo

May 27, 2018

Abstract

Influence maximization, which is first formulated by Kempe, Kleinberg and Tardos, is the problem that find a subset of nodes in a social network so that their total influence reach the maximum under a certain influence cascade model. In this paper, I first extend the classic linear threshold model defined by Kempe et al. and take the commonly appeared negative opinions into consideration. The model maintains some important properties such as submodularity and its behaviors of the negative opinions are well corresponded to those in real life. Then, I propose a efficient heuristic algorithm to overcome the severe scalability problem encountered by greedy algorithm, which can reduce the running time from days to minutes without losing much of the performance. In addition, I also show that the commonly ignored evolution algorithm can outperform greedy algorithm both in time complexity and performance, while none of other algorithm could accomplish this as far as I know.

Keywords: influence maximization, social networks, negative opinion, linear threshold model

1 Introduction

Nowadays, with the development of social network sites and applications like Facebook, Twitter, Wechat, etc., people are connected more widely and closely. One can have friends living in the opposite side of the earth and communicate with them in every second. As a result, viral marketing, a strategy that promote products through existing social networks, is getting more and more attention by business companies. In fact, together with the personalized recommendation, advertising to the right person has become a critical and pervasive problem for the industrial circle.

To tackle with the problem, Kempe et al. formulated this as the influence maximization problem [1] and proposed two basic models, the independent cascade (IC) model and the linear threshold (LT) model. Informally speaking, both model regard the social networks as directed or indirected graph G=(V, E), where V and E represent the set of nodes and edges in the graph respectively. The problem is to find an optimal set of k node (seeds) such that their influence, which is computed by some propagation rules, to the whole graph is the largest. The influence is usually defined as the number of activated nodes.

Those two model take a different view on how the influence is spread. IC model defines the propagation as a probabilistic process, in which each activated node has a probability to activate its neighbor nodes. On the contrary, LT model focus on the quantitative properties of spreading. Each nodes in LT model have a amount of influence to its neighbors and a node is activated if the total influence of its neighbors reach a threshold. In real world, those two view both stands. For example, you may want to buy a new iPad when a friend of you bought one and recommended it to you or when many of you friends all bought one and you feel that you need one too.

This problem has been applied in all kinds of works especially for viral marketing and a lot of research about it has been conducted [2, 3, 10, 8, 11, 5, 4, 9]. However, most of them are concentrated on IC model and ignore a critical fact that bad opinions can also propagate in the social networks. As far as I know, only one heuristic algorithm for LT model was proposed [4] and only two paper consider the negative opinion [6, 7]. As a contrast, negative opinions are ineligible in real world. For instance, when we shopping on Taobao, an e-shop platform of Alibaba, the rates of commendation of goods are vital to the decision of the customers. Usually, few people will take the goods with a commendation rate under 90% into consideration. As a result, the shopkeepers will use all kinds of method to avoid any negative comment include but not limited to small refund and small gift.

In fact, the influence of negative opinions has long been studied by sociologists [12, 13, 14]. Researchers have shown that negative opinions usually have a much stronger impact on people than positive opinions. And people tend to believe the bad one when both negative and positive opinions appear. So, it is reasonable and inevitable to cover both positive and negative opinions into the influence maximization problem.

In this paper, I first extend the classic linear threshold model to the linear threshold model with negative opinion (LT-N), in which negative opinions emerge and spread together with the positive opinions. Briefly speaking, when a node is activated, it has a probability q to have a good impression while has a probability $(1-q)$ to get annoyed. When a node is negatively activated, the influence to its neighbor reversed. A node is activated when the sum of influence (both positive and negative) of its neighbors exceed its threshold. Formal description will be given in Sect 3.

The LT-N model reflects some real world phenomena well. First, there must be a start point for the opinions spread in the network. And negative opinions often come from bad experience after buying something caused by product defects, transport accidents or picky customers etc. So, it is reasonable to model that there is a probability that the nodes turn negative after being activated. Besides, a small amount of negative comments for one merchandise won't prevent one from buying it totally. A product with enough large proportion of positive comments can still be a good choice. So, in LT-N model, both positive and negative opinions are considered when activated a node.

Second, I will give some analysis about LT and LT-N model and shows that both of them are NP-hard problems. Some important properties especially submodularity are proved. And as a result, a greedy algorithm holds an 1-1/e approximation.

However, greedy algorithm faces a severe efficiency problem and is not scalable. Fortunately, [4] shows that we can compute the influence in linear time in a directed acyclic graphs (DAGs). Inspired by this, we can construct a local DAG for every nodes in G and compute their influence in the local DAG. Then we can use the influence in local DAG as an approximation or heuristic function for the influence originally evaluated by Monte-Carlo simulation in greedy algorithm. Incorporating the effects of negative opinion, I propose a modified LDAG algorithm LDAG-N and show that it achieves a significant improvement in time complexity with some tricks, which could reduce the running time for some graph from days to minutes.

What's more, I will show that the commonly ignored evolution algorithm (EA) [16, 15] could outperform greedy algorithm both in time complexity and total influence, while all of the algorithm designed for influence maximization problem regard greedy algorithm as a upper-bound and none of them exceed its performance. Together with the LDAG-N, evolution algorithm can be very fast and efficient at the same time.

All of those three algorithm will be formally discussed in Sect 4. In the end, some extensions of the model and the effects and social meaning of q are discussed in Sect 6.

In short, the paper is organized as follow: Sect 2, related works; Sect 3, model definition and analysis; Sect 4, influence computation algorithms; Sect 5, experiments; Sect 6, extensions; Sect 7, conclusion.

2 Related Work

Influence maximization was first studied by Domingos and Richardson [17, 18]. They modeled the problem as a Markov random field. Then, in 2003, Kempe et al. gave a discrete optimization formulation to this problem and dug out a dozen of properties including the submodularity, which result in a 1-1/e approximation greedy algorithm. After they gave the elegant definition, researchers are mainly concentrated in two aspect of this problem: give better algorithm [2, 3, 8, 4, 5, 9] and extend the two basic model [6, 19].

For general greedy algorithm, Leskovec et al propose a trick called "cost-efficient lazy-forward" (CeLF) and achieve a acceleration up to 700 times [2]. Goyal A et al. further improved CeLF to Celf++ [3] and gained an additional 55% speed up. However, even CeLF++ is still not capable of running in a graph with millions of points. It still takes hours to compute in NetHEPT dataset, which has 15k nodes and 37k edges.

Then, many heuristic algorithm are proposed in order to replace the time consuming MC simulations in greedy algorithm and under-performing degree and centrality heuristic function. Maximum influence arborescence (MIA)[5], which takes the advantage of local tree structure, and degree discount, which taking neighbors' degree into consideration, proposed by Chen et al. are the most famous and efficient ones. However, all of them are for IC model. In fact, only two algorithms are proposed for LT model: Sharply-value-based SPIN [20] and heuristic algorithm LDAG [4].

Although heuristic algorithms are usually efficient enough, some researchers proposed some randomized algorithm [9] to achieve better efficiency.

As for negative opinions, only [6] and[7] had discuss them in influence maximization problem. However, [7] failed to reflect real-world phenomena and [6] didn't consider the mixing effect of positive and negative opinions.

What's more, evolutionary algorithm is powerful tool for discrete optimization problems with extreme large search space. For example, the state-of-the-art result of image classification on benchmark datasets(CIFAR-10 and ImageNet) are achieved by architecture designed by evolution [21]. However, it gets less and less attention these days for more and more people pay their attention to deep learning.

3 Linear Threshold Model With Negative Opinion

I first give a formal description about LT-N model, and then discuss some properties of the model.

3.1 Model Definition

Social networks are modeled as a directed graph $G = (V, E)$, where V and E represents the set of nodes and edges in the graph respectively. Each edge in G has a weight w. For any in-activated vertex u, v that $(u, v) \in E$, $0 < w(u, v) \leq 1$. And for any in-activated node v, $\sum_{u:(u,v)\in E} w(u, v) \leq 1$. And each node v has a threshold λ_v which $0 < \lambda_v \leq 1$. Beside, a satisfaction probability q is introduced as the probability that one nodes get positive opinion after being activated. If an activated node gets negative opinion, its influence turn negative.

The dynamic of influence spread is as follow: in time step t, for all the in-activated neighbor of nodes that is activated in $t - 1$, compute the influence of their activated neighbors. Denote $PN(v)$ and $NN(v)$ as the positive and negative activated neighbor of v If for node v, $\sum_{u:u\in PN(v)} w(u, v) - \sum_{u:n\in NN(v)} w(u, v) \geq \lambda_v$, then v is activated. More specifically, v is positive activated with a probability q and negative activated with a probability $1 - q$.

The problem is to find a subset of nodes S with a cardinality of k so that the number of positive activated nodes $Inf(S)$ reach the maximum.

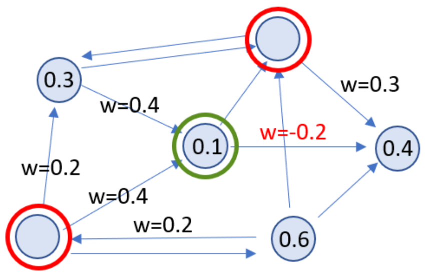

Figure 1: A simple example

3.2 Model Analysis

Even the influence maximization in original LT model is proved to be NP-hard. Luckily, we have a important property called submodularity in IC and LT model. Formally speaking, we says function f is submodular if for any set $S \subseteq T \subseteq V$ and node $u \in V\ T$, $f(S \cup u) - f(S) \geq f(V \cup u) - f(V)$. And IC and LT model also have monotonicity for $f(S) \leq f(T)$.

Claim 1 LT-N is monotone and submodular when $q > 0.5$.

However, for LT-N is a model about quantity and threshold, it is hard for us to analyze it directly. So, we need to have a alternative view about this problem.

Claim 2 The influence spread in LT-N equivalent to the following process: in time step t, if a node v has a new positive activated nodes, randomly select one edge with each edge (u, v) such that u not in $A_{1:t-1}$ with a possibility

$$\frac{w(u,v) * max(0, \frac{\sum_{u:u\in A^p_t} w(u,v) - \sum_{u:n\in A^n_t} w(u,v))}{\sum_{u:u\in A^p_t} w(u,v)})}{1 - max(0, \sum_{u:u\in A^p_{1:t-1}} w(u,v) - \sum_{u:n\in A^n_{1:t-1}} w(u,v))} \tag{1}$$

where A_t is the nodes that are activated in time-step t and A^p, A^n represent the positive and negative activated nodes. (Note that there is a possibility that no edge is selected) If the selected $u \in A$ and is positively activated, then keep the edge. And assign v as negative node with probability $1 - q$. Otherwise, wait for next new activated neighbor.

Note that in time step t, for a in-activated node v, the probability of v being activated in time step $t + 1$ is

$$\frac{max(0, \sum_{u:u \in A_t^p} w(u,v) - \sum_{u:n \in A_t^n} w(u,v))}{1 - max(0, \sum_{u:u \in A_{t-1}^p} w(u,v) - \sum_{u:n \in A_{t-1}^n} w(u,v))} \tag{2}$$

which is exactly the probability that the selected edge is adjacent to A^p. So, we successfully transfer the origin problem into a connectivity problem in a special random graph X, which is the end state of the random process.

Lemma 1 $Inf(S)$ for a fixed outcome X ($Inf_X(S)$) is submodular.

This lemma is proved by Kempe et al [1]. We can simply see that comparing with adding v to S where $S \subset T$, the newber of newly reachable nodes from adding v to T is at most equal, for some of the nodes is already reachable from T.

Finally, we have

$$Inf(S) = \sum_X P(X) * Inf_X(S) \tag{3}$$

where $P(X)$ is the probability of outcome X. A linear combination of submodular function still holds sumodularity.

As for monotonicity, we can see from the process above. Considering one activating nodes v and $q = 0.5$, it is activated positively with a probabililty of 0.5. So, the one step influence of v is expected to be 0. And as we can see from (2), the state of v has a linear influence on its neighbor, which still have zero expectation on whether it is activated. So by induction, the expected influence of adding v is 0. However, it's apparent that the influence grows as q gets bigger. So Inf is monotone.

Lemma 2 For any monotone and submodular set function f with $f(\emptyset) = 0$, the greedy algorithm is $1 - 1/e$ approximation.

Proof was shown in[23]. And as a result, greedy algorithm is guaranteed to have a $1 - 1/e$ approximation result.

4 Influence Computation

4.1 Greedy Algorithm

The greedy algorithm is quite straightforward. The idea is select the node that maximize the additional influence in each step. And the influence is evaluated by Mont-Carlo simulations. Each simulation is O(m) where m is te number of edges. And in each round, all the n nodes need to be simulated for R times. So, the time complexity is $O(knRm)$.

Algorithm 1 Greedy Algorithm

```
1: $S = \emptyset$
2: for i=1 to k do
3:     $u = argmax_u Inf(S \cup \{u\})$
4:     $S = S \cup \{u\}$
5: end for
6: return $S$
```

4.2 LDAG-N

To overcome the inefficiency of greedy algorithm, heuristic functions are used to replace the simulation. In the following, I will first show that the influence of a activated nodes decays exponentially. So, it is reasonable to use the local influence as an approximation of the global influence. And then, algorithm 2 shows that the influence in DAG can be compute in linear time. Sect 4.2.2 illustrate the whole algorithm. Sect 4.2.3 gives the time complexity. What's more, a trick for complexity optimization is also shown in Sect 4.2.3.

4.2.1 Influence in DAGs

As discussed above, LT-N model can be viewed by a probabilistic problem. Here, we explore it further. Let $P_a(v)$ be the probability that v is activated in a DAG D given the seed set S. We can see that

$$\begin{aligned} P_a(v) =& max(0, \sum_{u \in N_{in}(v)} q P_a(u) * w(u,v) \\ & - \sum_{u \in N_{in}(v)} (1-q) P_a(u) * w(u,v)) \\ =& (2q - 1) \sum_{u \in N_{in}(v)} P_a(u) * w(u,v) \end{aligned} \tag{4}$$

where $N_{in}(v)$ is the set of in neighbors of v and for $q > 0.5$. And $P_a(u)$ for $u \in S$ are initialized to be 1. It is easy to see that this equation holds true from a probability view.

We can also see that

$$Inf(S) = \sum_{v \in V} qP_a(v) \tag{5}$$

Noticing that $P_a(v)$ decays exponentially by constantly products $(2q-1)$ and $P_a(u)$ as v goes further from S. So, the influence far from S can be ignored. As a result, only consider the local influence becomes reasonable.

With the formulation (4), we can use dynamical programming to compute all $P_a(v)$ in DAG in a topological order and add them up to get the influence.

Algorithm 2 DAG influence

```
Require: D, S, x
1: $\forall u \in D$, $P_a(u) = 0$
2: $\forall u \in S$, $P_a(u) = 1$
3: $\rho$ = TOPO-SORT(D, S)
4: for v in $\rho$ do
5:     $P_a(v) = (2q-1)\sum_{u \in N_{in}(v)} P_a(u) * w(u,v)$
6: end for
7: return $P_a(x)$
```

4.2.2 LDAG-N algorithm

Now that we could compute the influence in DAGs efficiently, the problem comes to how to construct a local DAG in the general graph. [4] formulate this problem as **MAX-LDAG problem**: given a graph $G(V,E,w)$, a node $v \in V$ and a threshold $\theta > 0$, find a DAG $D = (X,Y,w)$, such that (a) D is a subgraph of G;(b) $v \in X$; (c) $I(u,v) \geq \theta$ for all $u \in X$ where $I(u,v)$ is the influence probability from u to v; (d) $\sum I(u,v)$ is the maximum.

Unfortunately, this problem is also NP-hard. And we can only use a greedy algorithm without any approximation guarantee. In the greedy algorithm, we constantly add the neighbor nodes of D with the maximum influence probability until none of the nodes satisfies the threshold. So, the whole LDAG-N algorithm comes as follow:

In algorithm 4, for the DAGs are local DAGs, adding a node to S doesn't mean all the influences are changed. Only the DAGs that include u need to be update. However, this is still not efficient enough. I will briefly talk about a trick for updating influence in below.

Algorithm 3 Find DAG

```
Require: G, v, $\theta$
1: $X = Y = \emptyset$
2: $\forall u \in V$, $I(u,v) = 0$
3: $I(v,v) = 1$
4: while $max_{u \in V \ X} I(u,v,) \geq \theta$ do
5:     $x = argmax_{u \in V \ X} I(u,v,) \geq \theta$
6:     $Y = Y \cup (x,u) | u \in X$
7:     $X = X \cap x$
8:     for u in $N_{in}(x)$ do
9:         $I(u,v) += w(u,x) * I(x,v)$
10:    end for
11: end while
12: return $D(X,Y)$
```

Algorithm 4 LDAG-N without trick

```
1: $S = \emptyset$
2: for v in V do
3:     D(v) = FIND_DAG(G, v, $\theta$)
4:     Inf(v) = $\sum_{u:v \in DAG(u)}$ DAG_INF(D(u), $\{v\}$)
5: end for
6: for i=1 to k do
7:     $u = argmax_u Inf(v)$
8:     $S = S \cup u$
9:     for $v : u \in D(v)$ do
10:        Inf(v) = $\sum_{u:v \in DAG(u)}$DAG_INF(D(u), $S + \{v\}$)
11:    end for
12: end for
13: return $S$
```

4.2.3 Algorithm complexity

Let n_θ, m_θ denote the max number of nodes and edges in local DAGs. To compute the local DAGs, we need $O(nm_\theta)$ time. And in each round, about n_θ nodes need to recompute their influence. Each node need a $O(n_\theta m_\theta)$ time for computing DAG influence for n_θ nodes. So the total time complexity is $O(nm_\theta + kn_\theta n_\theta m_\theta)$

Although this algorithm reduce the complexity significantly by only focus on the local DAG, it is still not efficient enough. So, here is a trick.

If we expend the formulation (4), we can find that $P_a(v)$ is a linear combination of linear combination for $P_a(u)$ the influence from u to v can be expressed as $P_a(u)\alpha$ where α is a coefficient to show the linear influence caused by u. If we maintain all the $P_a(u)$ and $\alpha(u, v)$ explicitly, we can use $P_a(u)$ and $\alpha(u, v)$ update influence efficiently. We can reduce the time for updating from $O(n_\theta m_\theta)$ to $(m_\theta logn)$ for $logn$ is the overhead of adjust the heap used for argmax, which is hidden behind of Inf recalculation in the algorithm without trick.

4.3 Evolutionary algorithm

Evolutionary algorithm is a framework of discrete optimization, which is inspired by theory of evolution. Despite its simplicity, it works well in some tasks. So, EA is used here to get over the $1 - 1/e$ bound of greedy algorithm. For EA, three important components are included: (a) population (b) mutation; (c) crossover; (d) selection.

population The population is the set of possible S.

mutation There is a chance of change some nodes $s \in S$ in reproduction.

crossover Select a pair of parents S_1, S_2. Take the first i s in S_1 and last $k - i$ s in S_2 to form a new S.

selection Only the top Ss will be kept for the next generation.

What's more, from the 700 times speed-up achieved "lazy-forward" [2], we can know that even if the influence of a certain node decreases when S expend, most of the selected nodes have a strong influence when S is empty. So, we can test influence for all the nodes at the beginning, and let the population select only from the top nodes, which will significantly reduce the search space.

Algorithm 5 Evolutionary Algorithm

```
 1: for v in V do
 2:     compute Inf(v)
 3: end for
 4: elite_nodes = TOP(V)
 5: population = INIT_POPULATION(elite_nodes) //generate population randomly
 6: for i=1 to rounds do
 7:     population += REPRODUCE(elite_nodes) // include crossover and mutation
 8:     SORT(population)
 9:     population = TOP(population)
10: end for
11: return population[0]
```

Note that no appointed method to compute influence. So this algorithm is applicable to both Mont-Carlo simulation and heuristic functions.

EA has no guarantee of approximation bound and running time bound. But it is the only algorithm that view the k nodes as whole instead of a gradual process of adding nodes, which means that it have the chance to handle the interaction within S. And the experiments prove its efficiency.

5 Experiment

5.1 Experiment setting

Before conducting experiment for different algorithms, some setting should be determined. This includes way of generating weight, simulation rounds for Greedy algorithm and simulation-based EA, q and threshold for LDAG-N.

weight There are two popular way of generating weights on edges: random and uniform, where uniform means giving each in-edge a weight of 1/in-degree. Greedy algorithm is test on a co-author graph of 1000 authors on arXiv physic section with different weights. The result figure 2 shows that they don't influence the result much. So, [illegible]

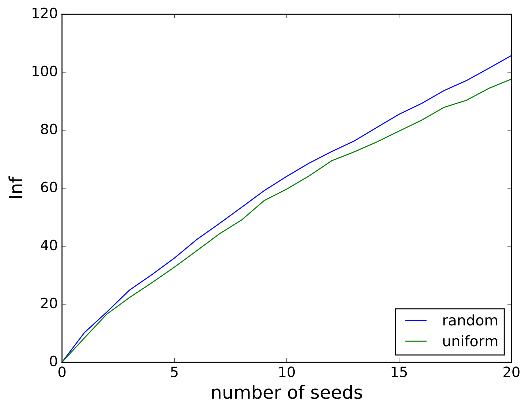

Figure 2: Differnece of Weights

simulation rounds The outcomes of simulation have a enormous variance. Take one result of EA in ca-GrQc [22] for $k = 30$ as a example. The result could range from 100 to more than 600. And the mean of the simulations are shown in figure 3. Kempe et al. recommend to take average of 10000 simulations. However, due to the computation resource, I finally choose to simulate 200 times.

q and threshold q and threshold are 0.9 and 1/640 as

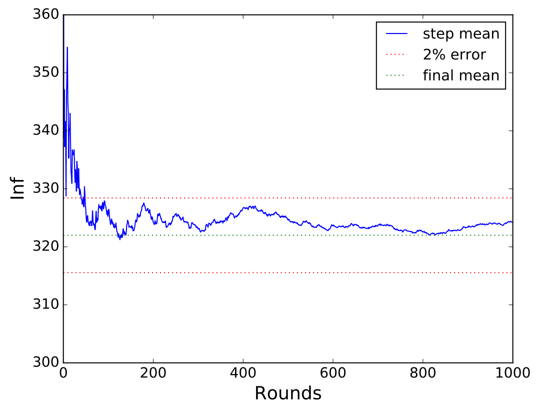

Figure 3: Mean of Simulations

default. the effect of q is discussed in extension.

5.2 Experiment for algorithms

A random graph with 30 nodes, 120 edges, arXiv physic co-author graph and the full ca-GrQc are used in the following experiment. And all the figures are about ca-GrQc for it is real and the largest (5424 nodes and 14496 undirected edges) among these three dataset. Random algorithm (choose nodes randomly) and degree algorithm (choose the nodes with largest out-degree) are also testes as a comparison. All the algorithm except EA are tested for k range from 1 to 30. And EA is only evaluated for k=10, k=20 and k=30. The result is shown in figure 4

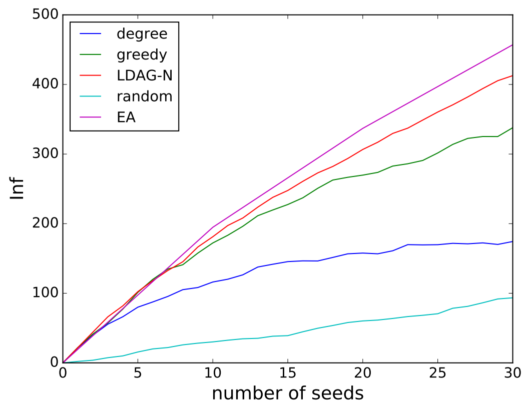

Figure 4: Performance

We can see that random and degree algorithm don't work well. Greedy algorithm ranks first until $k = 8$. And it become a little bit unstable when $k > 15$. This is because of the insufficient simulation (See Extension).

LDAG-N works well for all the k. However, for any tested k, EA always gets the best result. Although insufficient simulation could also influence its performance, however, different from greedy algorithm, all the nodes are select at once. So, the bias won't accumulate and the error is expected to be within 2%.

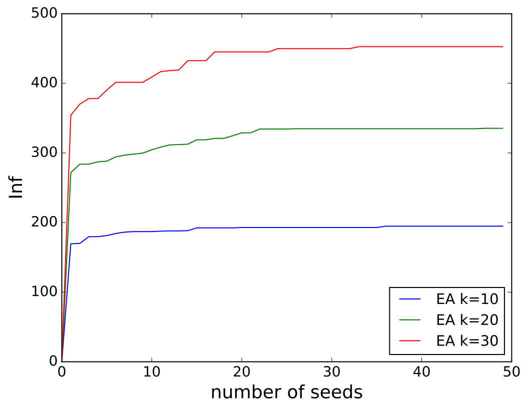

Figure 5: Convergence of EA

Before having a look at the running time, let us first give a glance to the running curve of EA. As shown in figure 5, all of the experiment is almost converge before round 25. So the running time of EA is the time for first 25 rounds.

Algorithm	Time/min	Relative Time
Greedy	2246.3	1
EA	148.6	0.066
LDAG-N	1.38	0.00062

Table 1: Time for k=30

As for the running time, greedy algorithm is almost unbearable. Considering it high performance, the running time for EA is much better. It seems that only LDAG-N is scalable to millions of nodes (Note that its time complexity is linear to n if m_θ remains the same). However, EA is easy to compute in parallel. So it is still applicable to millions of nodes if more CPU are used.

All of the experiment is running on a personal computer with one intel core i7-4790k using Python.

6 Extension

In this section, I will give some more brief analysis to the influence of simulation for greedy algorithm and negative opinions in LT-N model.

First, let us have a look at figure 4. The curve of greedy algorithm become fluctuate when $k > 10$. This is because the high variance make some bad nodes looks good within these 200 simulations and chosen. However, in latter rounds, the bad can't influence a large amount of nodes, so it reduce the performance of all the latter round. What's worse, this poor performance is accumulated in each round. So, the performance gap is widened again and again. With the restriction of computer resources, a small experiment for 20, 200 and 400 simulations is conducted and the result is shown in figure 6 This phenomena is also verified by Chen et al in [4]. When the number of simulation reach 20000, the performance of simulation-based greedy algorithm exceed all the heuristic ones in LT model.

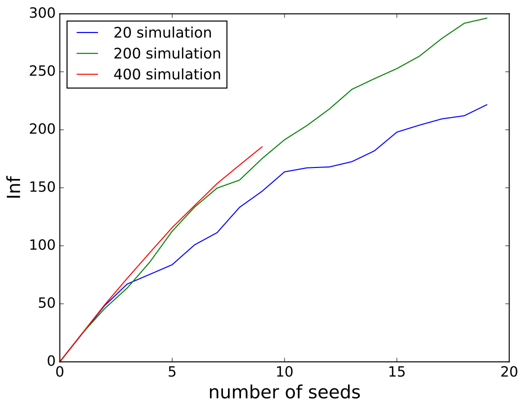

Figure 6: Influence of simulation

Experiment has been conducted about different q.

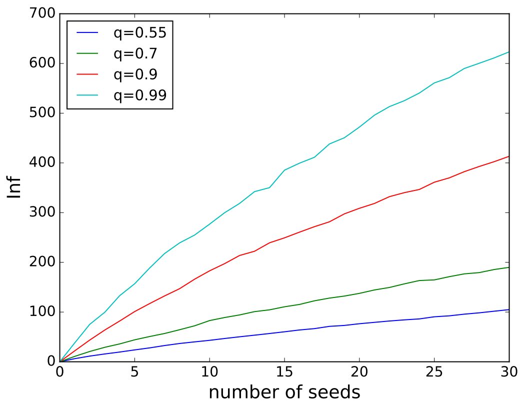

Figure 7: Influence of q

The result is in great accordance with our expectation. Influence goes down as q goes to 0.5.

However, what if the negative opinion have a weight other than the opposite number of positive opinion? Generally, the monotonicity no longer holds in the model for the linearity of the influence no longer holds. As a result, greedy algorithm won't give any approximation guarantee. This shows that the LT-N model have a balance in model complexity, expressiveness and tractability.

What's more, what if there is a negative threshold so that one node is sure to propagate negative opinion when the total weight of negative opinion exceed the threshold? This model is yet to be analysis.

7 Conclusion

In this paper, a Linear threshold model with negative opinion for influence maximization is proposed, which has a strong connection with the real world advertising and capture the character of the mental activity of customers. This model keeps the important properties of monotonicity and submodularity, which result in a 1-1/e approximation guarantee for greedy algorithm. Then, to tackle with the efficiency and quality problem, LDAG-N and EA are proposed or used in this problem. Both of them outperform greedy algorithm in speed, EA achieve the best performance and LDAG-N is scalable to larger dataset.

Acknowledgement

This project is a course project in Wireless course given by Prof. Wang and Prof. Fu. Thanks teachers and TAs for helping me discover and study this topic.

References

[1] Kempe D, Kleinberg J, va Tardos. Maximizing the spread of influence through a social network[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2003:137-146.

[2] Leskovec J, Krause A, Guestrin C, et al. Cost-effective outbreak detection in networks[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2007:420-429.

[3] Goyal A, Lu W, Lakshmanan L V S. CELF++: optimizing the greedy algorithm for influence maximization in social networks[C]// International Conference Companion on World Wide Web. ACM, 2011:47-48.

[4] Chen W, Yuan Y, Zhang L. Scalable Influence Maximization in Social Networks under the Linear Threshold Model[C]// IEEE International Conference on Data Mining. IEEE Computer Society, 2010:88-97.

[5] Chen W, Wang C, Wang Y. Scalable influence maximization for prevalent viral marketing in large-scale social networks[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2010:1029-1038.

[6] Chen W, Collins A, Cummings R, et al. Influence Maximization in Social Networks When Negative Opinions May Emerge and Propagate[J]. Journal of China University of Petroleum, 2010:379-390.

[7] Ma H, Yang H, Lyu M R, et al. Mining social networks using heat diffusion processes for marketing candidates selection[C]// ACM Conference on Information and Knowledge Management. ACM, 2008:233-242.

[8] Chen W, Wang Y, Yang S. Efficient influence maximization in social networks[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2009:199-208.

[9] Hu Q C, Zhang Y, Xu X H, et al. A new approach for influence maximization in complex networks[J]. Acta Physica Sinica, 2015, 64(19).

[10] Wang Y, Cong G, Song G, et al. Community-based greedy algorithm for mining top-K influential nodes in mobile social networks[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2010:1039-1048.

[11] Kempe D, Kleinberg J, va Tardos. Influential Nodes in a Diffusion Model for Social Networks[C]// International Colloquium on Automata, Languages, and Programming. Springer, Berlin, Heidelberg, 2005:1127-1138.

[12] Baumeister R F, Bratslavsky E, Finkenauer C, et al. Bad is stronger than good[J]. Review of General Psychology, 2001, 5(4):477-509.

[13] Rozin P, Royzman E B. Negativity Bias, Negativity Dominance, and Contagion[J]. Personality & Social Psychology Review, 2001, 5(4):296-320.

[14] Taylor S E. Asymmetrical effects of positive and negative events: The mobilization-minimization hypothesis.[J]. Psychological Bulletin, 1991, 110(1):67-85.

[15] Vikhar P A. Evolutionary algorithms: A critical review and its future prospects[C]// International Conference on Global Trends in Signal Processing, Information Computing and Communication. IEEE, 2017:261-265.

[16] Bck T. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms[M]. Oxford Univ. Pr, 1998.

[17] Domingos, Pedro, Richardson, et al. Mining the network value of customers[J]. 2001.

[18] Richardson, Matthew, Domingos, et al. Mining knowledge-sharing sites for viral marketing[J]. 2002.

[19] Chen W, Lu W, Zhang N. Time-Critical Influence Maximization in Social Networks with Time-Delayed Diffusion Process[J]. Chinese Journal of Engineering Design, 2012, 19(5):592-598.

[20] Narayanam R, Narahari Y. A Shapley Value-Based Approach to Discover Influential Nodes in Social Networks[J]. IEEE Transactions on Automation Science & Engineering, 2010, 8(1):130-147.

[21] Real E, Aggarwal A, Huang Y, et al. Regularized Evolution for Image Classifier Architecture Search[J]. 2018.

[22] J. Leskovec, J. Kleinberg and C. Faloutsos. Graph Evolution: Densification and Shrinking Diameters. ACM Transactions on Knowledge Discovery from Data (ACM TKDD), 1(1), 2007.

[23] Nemhauser G L, Wolsey L A, Fisher M L. An analysis of approximations for maximizing submodular set functionsI[J]. Mathematical Programming, 1978, 14(1):265-294.