Algorithm Project Report

Jincao Zhu and **Wenzheng Tao** and **Zhenduo Wang** *

Abstract

We propose a computation lightweight heuristic algorithm based on "high degree" for social influence maximization problem. And we implement the proposed algorithm and compare its performance with naive high-degree algorithm and existing simulation-based greed algorithm proposed in (Kempe et al., 2003). The heavy computation requirement of the greedy algorithm makes it unusable for large graph despite its good performance in term of number of final activation nodes. Our performance measurement is done by apply social network dataset from https://snap.stanford.edu.

1 Introduction

Social network analysis has drawn a lot attention for the past decades. Initially sociologist have start to study how social networks is formed by and how people are connected. One of the interesting problems is influence maximization problem, which emphasis on how individuals take effects on information spreading in a connected social network. And this is a typical application to leverage social network analysis for marketing purpose. Nowadays social networks like Facebook, Twitter, Instagram are used for spreading information like news, ideas, advertisements. And also for marketing purpose, for example researchers also want to study how to maximize the influence of advertisements by choose proper individual targets that maximize the number of people it can influenced via spreading through the social networks . Social networks normally can be modeled as graph structure model, vertices/node represent an individual and edge represents the relationship between individuals. The problem is to find a initial set of vertices that can maximize the number of vertices they can reach with certain propagation model.

In this work we are interested in a specific influence maximization problem (Kempe et al., 2003). This problem has been proven an NP-hard. And a simulation-based greedy algorithm is also proposed by (Kempe et al., 2003), but still constrained to only able to apply to small graph due its computation overhead. Our solution is to literately choose one node with highest degree in the subgraph composed with nodes that not influenced by a choose initial target set. and add it to the initial target set.

2 Problem Statement

Max influence in Social network is an interesting topic of SNA (social network analysis). This problem is formally defined in (Kempe et al., 2003). Given a Graph N (V,E), and choose an initial k number of vertices set A_0, what is the maximum number of influence we can achieve? In (Kempe et al., 2003), two Basic Diffusion Models are considered. Specifically, Linear Threshold and Independent Cascade models. In this work, we considering only Independent cascade models for simplicity. In this model, initially start with a set of active nodes A_0, in each step a active nodes v will have a chance to flip his any inactive neighbor w into active one with probability P_{vw}. In the whole process node v only have a single chance to active any of his inactive node w, meaning he cannot make any further attempts no matter he is successful in flip w or not. And the process stops when there is no more activation are possible (Kempe et al., 2003). This problem is NP-hard, even the greedy algorithm proposed in (Kempe et al., 2003) is computation costing. For each it-

*Authors' names are listed in alphabetic order.

eration of the greedy algorithm, it simulates the influence of each inactive node, and choose the nodes brings maximum influence. Thus the simulating number needed is proportion to the size of the graph in each iteration.

We proposed a heuristic algorithm to reduce the computation load while still achieve acceptable approximation accuracy. Our algorithm choose initial target nodes by choose the nodes with highest degree in the sub-graph of potential less influenced nodes. We only do simulation when calculating this sub-graph.

3 Algorithm in Paper

The major algorithm in this paper (Kempe et al., 2003) is the greedy hill-climbing strategy. This is a local search algorithm. One of the contributions of this paper is that it shows this natural greedy strategy obtains a solution that is provably 63% of optimal for several classes of models. More specifically, the optimal solution for influence maximization can be efficiently approximated, using this greedy algorithm, to within a factor of $(1 - 1/e - \epsilon)$, in both the Linear Threshold and Independent Cascade models.

And the paper stated the fact, with real experiments, that hill-climbing is always within a factor of at least 63% of optimal for this problem. With respect to this algorithm, the paper also showed that this algorithm significantly outperforms strategies based on targeting high-degree or 'central' nodes.

The greedy hill-climbing algorithm is defined as below: First we denote the final activated population by $\sigma(A)$, while A refers to the set of nodes that are activated at the initial stage. At first step of the algorithm, we set A to be empty and add nodes into A one by one, until the number of nodes added reached the size of A, i.e. k, for k-node set of maximum influence. The greedy algorithm makes the rule for choosing node to add in as below: every time select the node $a \in (V - A)$ so that $\sigma(A+a)-\sigma(A)$ is the highest, where $(V-A)$ denote the set of all the nodes that $\notin A$.

This algorithm seeks and employ the optimal choice while adding the nodes into A one by one. And using an analysis framework based on submodular functions, the paper could result in the approximation guarantees mentioned above. The submodular function f satisfies $f(S \cup \{v\} - f(S) \geq f(T \cup \{v\}) - f(T)$. This characteristic

Algorithm 1 Greedy hill-climbing

1: $A = \emptyset$
2: **for** iter = $1 : k$ **do**
3: $\quad a = \arg\max_a (\sigma(A \cup a) - \sigma(A))$
4: $\quad \triangleright$ maximizing the spread of chosen set.
5: $\quad A = A \cup a$
6: **return** A

matches the greedy algorithm that every time the algorithm seeks the maximum marginal gain.

4 Baseline Algorithms

In this chapter we will cover the baseline algorithms introduced in the original paper.

4.1 High-degree Heuristic

The high-degree heuristic is based on the belief that nodes with high degrees are more likely to influence more nodes and thus have a larger influencial spread over the network. The high-degree chooses the first k nodes in a descending order of the degree of nodes. This heuristic is intuitively reasonable and it actually achieves very good performances and is ranked 2nd over all algorithms and heuristics in the original paper. Since the high-degree heuristic does not involve any runtime simulation and ranking, it is extremely time efficient.

A potential problem of the high-degree heuristic is that it only takes into account the role of exerting influence of high-degree nodes, rather than that of being influenced. In fact, the high-degree nodes also have a higher chance of getting influenced. For this reason, we believe that there is still room for improvement for high-degree heuristic in terms of spreading the source of influence.

5 Proposed Algorithm

As discussed above, the greedy algorithm is too computationally expensive, while the high-degree heuristic fails to exploit the distribution of high-degree nodes in the network. Our goal is to balance the two major downsides of both approaches and come up with an algorithm that is both accurate and fast. To this end, our proposed algorithm is designed to combine the forte of both approaches, and we call it greedy high-degree.

We agree with the original paper that with the Independent Cascade model, the influence measure $\sigma(\cdot)$ can be proved submodular and thus has

good properties favoring greedy approaches. Inspired by that, in our algorithm, we try to find the node set with highest degrees in a similar greedy manner. Let N be the node set, our algorithm works as follows:

Algorithm 2 Greedy high-degree

```
S = ∅
for iter = 1 : k do
    Pick n_1 as the highest degree node in G
    S = S ∪ {n_{r_i}}
    I = Spread(G, S)
    G = G \ I
return S
```

To summarize our algorithm, we repetitively choose the node n_i with the highest degree in the remaining graph, which is defined as the whole graph subtracting the current spread. Then we add the node n_i to S and simulate the new spread and update the remaining graph. We initialize the algorithm with a singleton set $S = \{n_1\}$ containing only one node with the highest degree in the network, and repeat until $|S| = k$.

There is an interesting relation between the two greedy algorithms, namely the paper's and ours. Consider that in the paper's algorithm, if the greedy choice generated by the simulation process agrees with the highest degree node in the remaining graph, then our algorithm will obtain exactly the same result as the paper's algorithm. From this point of view, our algorithm can also be seen as a simplification for the simulation-pick process in the original paper.

Comparing with the greedy algorithm in the paper, our algorithm is computationally cheaper in that we do not iteratively find the greedy choice for the maximum marginal influence gain, rather we simply pick the next node which we believe will bring the max influence gain. This will save a lot of time since the time complexity of finding the node for max influence gain is determined by the size of the whole network, which is usually large. Our algorithm needs only one query to generate the greedy choice, and thus the size of network is factorized off in time complexity.

It is not easy to explicitly prove how our algorithm solves the repetition issue in the high-degree heuristic. However our algorithm is intuitively better in the sense that we simultaneously maximize the degree of source nodes and minimize the overlap among them.

In the experiment chapter, we will show that our algorithm achieves reasonably good performance while being time efficiency, compared to the greedy algorithm and high-degree heuristic.

6 Experiments

6.1 Datasets

We conduct two simulation experiments on two datasets and compare the three algorithms discussed above. The first dataset we use is from (Leskovec et al., 2007a). The dataset is a network of Quantum Cosmology collaboration containing 5242 nodes and 14496 edges. The second dataset is from (Yin et al., 2017), (Leskovec et al., 2007b), which is a network of email network from a large European research institution containing 1005 nodes and 25571 edges. Note that the two datasets have significantly different edges to node ratios (2.77 vs. 25.44), which reflects the edge density of the two networks. Another major difference in statistics we find is their 90-percentile effective diameters (7.6 vs. 2.9), which also tells us how dense the nodes are connected and clustered. In fact, we characterize the first network which has fewer edges in average as sparse network and the second as dense network. We make comparisons among algorithms on both of them and show that these intrinsic natures of network have various impacts on the performances of the algorithms.

6.2 Results and Analysis

We show the results as line charts of target set size to final spread size. (See Fig. 1 - 4) In the experiments, we take the average of 20 simulations to reduce the impact of randomness. Also, we set the activation probability to be lower in order to get a reasonable spread size. This trick is also used in the original paper.

In Fig. 1 & 2, we see the algorithm in the paper achieves the best performances on sparse network. Our algorithm is slightly worse than the papers algorithm, but it is better than the naive high-degree algorithm. Our algorithm does better as the target size increases. Our explanation for this is that the repetition part in high-degree algorithm increases as the target size increases, thus our algorithm can show its power better when the target size is larger.

In Fig.3 & 4, we see that our algorithm achieves the best performances on dense networks. Also it is surprising that the greedy algorithm becomes

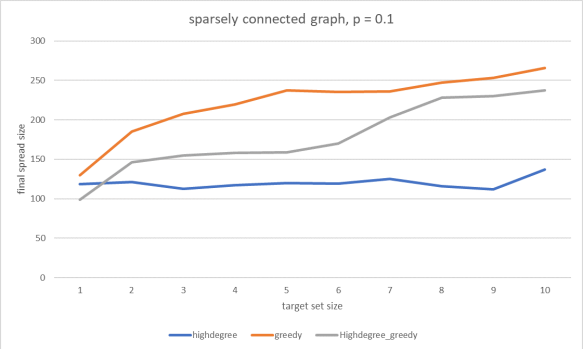

Figure 1: Results on sparse network, activation probability = 0.1

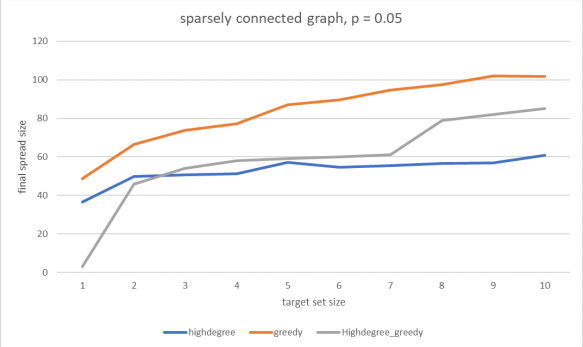

Figure 2: Results on sparse network, activation probability = 0.05

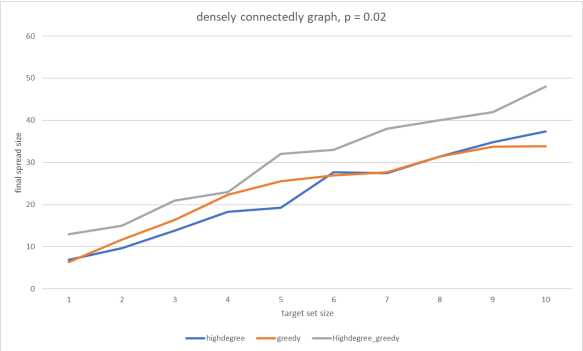

Figure 3: Results on dense network, activation probability = 0.02

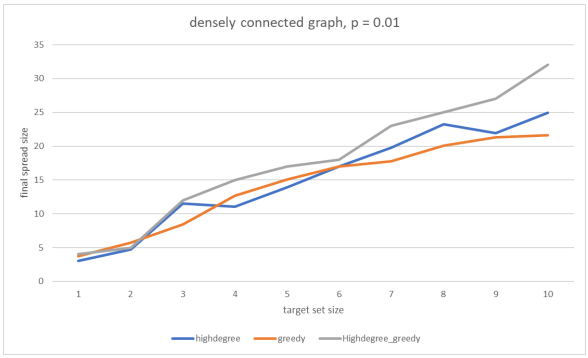

Figure 4: Results on dense network, activation probability = 0.01

the worst algorithm when the target size is large. We consider two possible reasons for this result. One is that the result still has a tiny chance to be wrong because of the high randomness of simulation. (although we use the result from the average of 20 simulations) A more likely reason is that the dataset is too small (notice that the 90-percentile diameter of this network is only 2.9), and the greedy algorithm in the paper may not work very well. From the results on dense networks, we believe that there exist certain cases where degree-based algorithm that uses graph natures can perform better than purely simulation based greedy algorithms.

Dataset	sparse (N=5242)		dense (N=1005)	
$P_{activate}$	p = 0.05	p = 0.1	p = 0.01	p = 0.02
HD	.005	.013	.008	.011
GHD (ours)	.50	.74	.256	.241
G (paper)	303	947	49.5	69.1

Table 1: Average running time to get 10 nodes for the max spread. $P_{activate}$ denotes the probability limit of activating neighbors. HD denotes the high-degree algorithm, GHD for the greedy high-degree, and G for greedy.

Time Analysis. Assume that we work on a network of size N and target set of size k. We assume that the Independent Cascade simulation has time complexity as a function $\mathcal{O}(IC(N,k))$. Given these assumptions, the high-degree algorithm has time complexity of $\mathcal{O}(IC(N,k))$ since it directly generates the solution and simulates once to get the spread. The greedy high-degree algorithm has time complexity of $\mathcal{O}(k \cdot IC(N,k))$ since it greedily pick a node k times and simulates after each

picking. The greedy algorithm in the paper has time complexity of $\mathcal{O}(Nk \cdot IC(N,k))$, since it simulates for N nodes before picking the best one for each of the k greedy choices.

In Table.1, we show the running time comparisons of the three algorithms on both the sparse and dense dataset. We divide the running time for algorithms (high-degree and greedy) that needs multiple simulations by their simulation rounds for fair comparisons. The high-degree algorithm is the fastest, and the greedy algorithm is notably expensive. Our algorithm is in between the two algorithm. Combining with the performance figures, we think our algorithm is significantly faster than the paper's algorithm without sacrificing too much power, and it outperforms the naive high-degree algorithm without committing too much time.

7 Conclusion

In this work, we design a new degree-based greedy algorithm for maximizing the influence spread of networks. We conduct experiments on two datasets with different characteristics to compare our algorithm with naive high-degree method and the simulation-based greedy algorithm in the original paper (Kempe et al., 2003). Our algorithm greatly reduces the time compared to the algorithm in the paper while maintains most of the power. When the edges becomes dense, our algorithm does even better than the algorithm in the original paper.

8 Explorations and Future Works

On defining the greedy high-degree algorithm, we come up with another two alternative algorithms.

Algorithm 3 Greedy high-degree ALT1

$S = \emptyset$
$R = rank(G)$
for iter = $1 : k$ **do**
    Pick the first n_1 in R **and** not in I
    $S = S \cup \{n_{r_i}\}$
    I = Spread(G, S)
return S

The first alternative is that we keep the degree rank static and iteratively pick the next one that is not included in the current spread, hence we do not update the remaining graph. The second alternative is that we do multiple simulations in each round and update the spread by thresholding. Both of them are different from our proposed algorithm in terms of how the greedy choice is generated.

Algorithm 4 Greedy high-degree ALT2

$S = \emptyset$
for iter = $1 : k$ **do**
    Pick n_1 as the highest degree node in G
    $S = S \cup \{n_{r_i}\}$
    for j = $1 : M$ **do**
        I_j = Spread(G, S)
    I = Threshold$(I_1, ..., I_M)$
    $G = G \setminus I$
return S

In experiments we find that the first alternative is worse than our proposed algorithm. We explain this by the fact that constructing the remaining graph allows us to exploit more of the dynamic structure of the network. The second alternative fails to improve our proposed algorithm although it commits more time. So far we cannot give a convince explanation for this phenomenon. Hopefully more datasets and experiments can provide more aspects to this in the future.

Acknowledgement

We use two github repositories blackliquid/ SocialInfluence and nd7141/ influence-maximization for comparison means for our own implementation of the algorithm in the paper.

References

David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 137–146.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007a. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1(1):2.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007b. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1(1):2.

Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 555–564.