Proximity-Aware Degree-Based Heuristics for Influence Maximization Problem*

Research Article

Maryam Adineh[1] Mostafa Nouri-Baygi[2]

Abstract: The problem of influence maximization is selecting the most influential individuals in a social network. With the popularity of social network sites and the development of viral marketing, the importance of the problem has increased. The influence maximization problem is NP-hard, and therefore, there will not exist any polynomial-time algorithm to solve the problem unless P = NP. Many heuristics are proposed for finding a nearly good solution in a shorter time. This study proposes two heuristic algorithms for finding good solutions. The heuristics are based on two ideas: 1) vertices of high degree have more influence in the network, and 2) nearby vertices influence on almost analogous sets of vertices. We evaluate our algorithms on several well-known data sets and show that our heuristics achieve better results (up to 15% in the influence spread) for this problem in a shorter time (up to 85% improvement in the running time).

Keywords: Degree Centrality, Heuristic Algorithm, Independent Cascade Model, Influence Maximization

1. Introduction

Interactions of people in a social network provides a lot of information about their behavior and the structure of the social graph. It has also made the social network a good platform for spreading information, beliefs, innovations, and so on. One of the most important applications of the spread of influence in social networks is viral marketing.

Consider a company that wants to market its product in a social network. A simple and low-cost approach is to select a subset of individuals to offer the product, so they will encourage their friends to buy it. This behavior is like spreading a virus in a society. The important part of this type of marketing is the initial selection of the most influential individuals. This problem is known as influence maximization problem.

Influence maximization problem was first introduced by Domingos and Richardson [1, 2]. Kempe et al. [3] formally defined the problem and proved that it is NP-hard. They also introduced two monotone and submodular diffusion models for the spread of influence, namely independent cascade model and linear threshold model. An immediate result proved by Kempe et al. [3] was that a greedy hill climbing algorithm approximates the solution within 63% of the optimal solution for these models.

Because the greedy algorithm runs a simulation several thousand times to find the marginal influence of each vertex, which is a time-consuming process, many heuristics are proposed to improve its performance. Although the heuristics have reduced the running time, they are still time-consuming for large-scale networks, which is the case for most social networks. On the other hand, degree-based heuristics are very fast even on large-scale networks. Although they do not guarantee the quality of the solution, they still find good solutions for the problem.

This study proposes two degree-based heuristics with very short running time which improve the results of previous degree-based heuristics. As it will be illustrated by the experiments, the quality of the results produced by our algorithms are very close to the quality of the results produced by the greedy algorithm, while their running time is very small and close to other degree-based heuristics.

This paper is an extended version of the paper [4] presented at the 8th International Conference on Computer and Knowledge Engineering (ICCKE 2018). The current version contains mathematical foundations of our techniques and rigorous descriptions of the algorithms. Furthermore, in this version we conduct a thorough evaluation and comparison of our algorithms with the best and state of the art algorithms for the problem.

The remainder of this paper is organized as follows. In Section 2, related works are reviewed. A formal definition of the problem is described in Section 3. Section 4 proposes heuristics and presents the experimental results. Finally, we conclude the paper in Section 6.

2. Review of related works

Influence maximization problem was formally defined by Kempe et al. [3] and proved to be NP-hard. They proposed a greedy hill climbing algorithm that yields a solution within $1-1/e-\varepsilon$ factor of the optimal solution for two models they introduced for influence propagation. In the above approximation ratio, e is the base of the natural logarithm, and ε, which can be any positive real number, is the error of the Monte Carlo simulations. Picking a small value for ε increases the running time, while taking a large value for it reduces the quality of the result. In the algorithm by Kempe et al., the most influential vertices are selected by their estimated marginal influence. Since estimated marginal influence is computed by a large number of simulations, the algorithm is not efficient.

In order to improve the efficiency of the computations, many studies have been conducted. Leskovec et al. [5] proposed Cost-Effective Lazy Forward (CELF) optimization that reduces the computation cost of the influence spread using the sub-modularity property of the objective function.

* Manuscript received May 30 2020, Revised, November 29, 2021; Accepted: January, 17 2022.
[1]. Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.
[2]. Corresponding author. Assistant Professor, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.
Email: nouribaygi@um.ac.ir

Chen et al. [6] proposed new greedy algorithms for independent cascade and weighted cascade models. They made the greedy algorithm faster by combining their algorithms with CELF. They also proposed a new heuristic, named degree discount, which produces results of quality close to the greedy algorithm while being much faster than that and performing better than the traditional degree and distance centrality heuristics.

In order to avoid running repeated influence propagation simulations, Borgs et al. [7] generated a random hypergraph according to the reverse reachability probability of vertices in the original graph and selected k vertices that cover the largest number of vertices in the hypergraph. They guarantee $1-1/e-\varepsilon$ approximation ratio of the solution with probability at least $1 - 1/n^l$. Later, Tang et al. [8, 9] proposed TIM and IMM to cover the drawbacks of Borgs et al.'s [7] algorithm and improved its running time.

Bucur and Iacca [10] and Krömer and Nowaková [11] used genetic algorithms for the influence maximization problem. Weskida and Michalski [12] used GPU acceleration in their genetic algorithm to improve its efficiency.

There are some community-based algorithms for the influence maximization problem that partition the graph into small subgraphs and select the most influential vertices from each subgraph. Chen et al. [13] used H-clustering algorithm and Manaskasemsak et al. [14] used Markov clustering algorithm for community detection. Song et al. [15] divided the graph into communities, then selected the most influential vertices using a dynamic programming algorithm.

3. Problem definition

In this section we formally define influence maximization problem and the independent cascade diffusion model.

We consider a social network as an undirected graph G = (V,E) where V is the set of individuals of size n, and E is the set of relationships of size m. In this study we describe the algorithms for undirected graphs, but it is trivial to extend the results to directed graphs. We also assume that G is unweighted, even though we can easily generalize the methods to the weighted case, where the weight of an edge (u,v) denotes the probability of the influence propagation from u to v. Clearly, the edge weights must be a value in the range $[0,1]$.

For each vertex u and an integer $h > 0$, let $N_{\leqslant h}(u)$ denote the set of vertices of distance at most h from u in G. We call $N_{\leqslant h}(u)$ the set of multi-hop neighbors of u.

For a set $S \subseteq V$ of vertices selected as the seed set to initiate the influence prorogation, let $I(S)$ denote the influence spread by S, i.e., the expected number of the influenced vertices, given S the initial seed set.

3.1. Diffusion Model

There are many diffusion models for the influence propagation process [16]. In this paper we focus on the Independent Cascade Model (ICM). In the independent cascade model, for each edge (u,v), a newly activated vertex u can activate v with probability $p_{u,v} \in [0,1]$.

The diffusion process is as follows. Let S_i be the set of newly activated vertices in timestamp i. In timestamp $i + 1$ each vertex $u \in S_i$ has a chance to activate each of its inactive neighbors. Once u tried to activate its neighbor v, whether it succeeds or not, u will not try to activate v in later steps. Furthermore, each activated vertex remains active in all subsequent timestamps. This process terminates when no more activation is possible.

3.2. Influence maximization problem

In influence maximization problem, given a graph G, a constant k and a diffusion model M, we are asked for a set S of k vertices with the maximum influence spread, $I(S)$. In this paper, we focus on the independent cascade model as M, and leave extending the algorithms to other models in future studies.

4. Proposed algorithms

In this section we describe our heuristics for influence maximization problem under the independent cascade model.

As mentioned above, although the greedy algorithm and its variants guarantee the solution in terms of the influence spread, they are very time consuming, especially for large scale social networks. On the other hand, degree centrality heuristics do not guarantee the quality of the solution, but may produce solutions of high quality in much smaller time. As a result, we propose two novel heuristics based on degree centrality which demonstrate more influence spread in comparison to similar algorithms.

Algorithm 1 MaximumDegree

Input: $G(V, E)$: social network graph; k: size of the result set; p: propagation probability in the ICM.
Output: S: seed set.

```
S ← ∅
for i ← 1 to k do
    u ← arg max_{v∈V\S} d_v   ▷ Find the vertex u with the highest degree
    S ← S ∪ {u}
return S
```

Degree centrality heuristics select k vertices with the highest degrees as the most influential vertices, because individuals with more relationships may have more influence spread in the network. The pseudo code of the maximum degree method is given in Algorithm 1.

A variant of this method, which is called single discount and was proposed by Chen et al. [6], decreases the degree of neighbors of each selected seed. For example, when u is selected as a seed, the degree of each neighbor v is decreased according to the number of edges they have in common. Although these heuristics usually find suitable candidates as seeds, they are not good enough. The reason is that in social networks normally high degree vertices are close to each other and influence on almost similar sets of vertices.

To select better seed sets, Chen et al. [6] proposed degree discount which decreases degrees of vertices according to the expected number of adjacent active vertices and the amount of influence propagation probabilities they have. Although the influence spread of degree discount is improved, it does not work well since it considers only direct

neighbors and multi-hop influence spreads are not considered at all.

The main reason that degree centrality heuristics cannot keep up with greedy algorithms is that in social networks, vertices with high degrees are usually close to each other. Suppose that two adjacent vertices *u, v* have the maximum degrees in the input graph. When we select *u* as the first seed, with a high probability *v* will also be activated by *u*. Therefore, there will not be much benefit from selecting *v* as another seed. This will be amplified when the propagation probability, $p_{u,v}$, is higher. A similar case can be explained for multi-hop neighbors. In the following sections, we propose two heuristics to handle these situations properly.

4.1. Removing neighbors

In the first heuristic, called NeighborsRemove, we iteratively select *k* vertices with the highest degrees. But to avoid selecting vertices with rather similar influence spread, in each step, we remove multi-hop neighbors of the selected seed from the list of candidates for subsequent steps. More precisely, in the first iteration, we select the vertex *u* with the maximum degree as the first seed. Since the multi-hop neighbors of *u* will be directly influenced by *u*, even though they may have high degrees, we remove them from the list of candidates and select next seeds from the remaining vertices. This process terminates when *k* seeds are selected.

In each step, when *u* is selected as a seed, we remove its multi-hop neighbors at distance of most *h*, $N_{\leqslant h}(u)$, with a breadth-first search starting from *u*. An important parameter here is *h*, the maximum level at which the visited vertices in the breadth-first search is removed.

It is easy to see that when the distance between a seed vertex *u* and another vertex *v* increases, the probability that *v* is activated by *u* decreases dramatically. This amount is equivalent to the product of the activation probabilities of the edges in the path from *u* to *v*.

In our experiments, like most of the works in the literature, we assume the activation probability of each edge constant, and equal to *p*. Based on this assumption, the value of *h* is dependent only on *p*. According to our experiments on several data-set, which are reported in Appendix A, the appropriate value for *h* is computed by $\lfloor 12\sqrt{p} \rceil$. The notation $\lfloor x \rceil$ here means rounding *x* to the nearest integer. The pseudocode of the method is given in Algorithm 2.

Algorithm 2 NeighborsRemove

Input: $G(V, E)$: social network graph; k: size of the result set; p: propagation probability in the ICM.
Output: S: seed set.

```
S ← ∅
V' ← V                          ▷ Make a copy of the vertices of G.
for i ← 1 to k do
    u ← arg max_{v∈V'} d_v
    S ← S ∪ {u}
    V' ← V' \ N_{≤⌊12√p⌉}(u)     ▷ Remove from V' all nodes of distance at most ⌊12√p⌉ from u.
return S
```

4.2. Decreasing degree

The second heuristic for the influence maximization problem is called DegreeDecrease. Similar to the NeighborsRemove heuristic, the main idea here is to select vertices with the highest degrees. But to avoid selecting vertices with rather similar sets of influenced vertices, in each step, we reduce the priority of selecting vertices close to the selected seed. In each step, when a vertex *u* is selected as the seed, the amount of the reduction in the priority of each vertex *v* is calculated according to the number of different paths from *u* to *v*, and their lengths. In the following, a more detailed description of the algorithm is given.

In the beginning, the priority of selecting each vertex *u*, denoted by *u.priority* is equal to the degree of *u*. As the first seed, we therefore select the vertex s_0 with the maximum degree. Then for each vertex $v \in N_{\leqslant h}(s_0)$, we decrease the *v.priority* to reduce the chance of *v* being selected as subsequent seeds. In the second step, the vertex with the highest *u.priority* is selected as the second seed. This process continues until *k* vertices are selected as the seed set.

The probability of activating *v* by *u* is decreased as the length of the path from *u* to *v* increases. In addition, this probability increases as the number of paths from *u* to *v* increases. Therefore, the larger the number of paths or the smaller the path length from a vertex *u* to a multi-hop neighbor *v*, the more reduction is applied on *v.priority* when *u* is selected as a seed. This is to reduce the chance of selecting vertices close to *u* as subsequent seeds.

Figure 1 shows two different paths from *u* to *v*. In each path, there is a possibility of *v* being activated by *u*. The probability of activation of *v* from the lower path $u \to v_1 \to v$ is greater than the upper path $u \to v_2 \to v_3 \to v$. In the lower path, *v* will be activated when both edges (u,v_1) and (v_1,v) propagate the influence, which happens with probability p^2, while in the upper path, the probability is p^3, since three edges need to cooperate to propagate the influence.

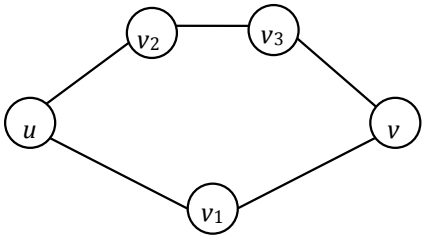

Figure 1. Two different activation paths from *u* to *v*

Suppose when a vertex *u* is selected as a seed, for each neighbor *v* of *u*, we decrease *v.priority* by a value that depends on $f(p)$, which is a function of *p*, the propagation probability of edges. We call $f(p)$ the path reduction coefficient. In Figure 1. *v* may be activated by the lower path only if both edges (u,v_1) and (v_1,v) propagate the influence. Thus, we decrease *v.priority* for this path by a value that depends on $f^2(p)$. Similarly for the upper path, we decrease *v.priority* by a value that depends on $f^3(p)$, because the length of the path is 3.

The influence propagation through the two paths in Figure 1 are independent, so if we denote by *A* (respectively

B) the event of the influence propagation through the lower (resp. upper) path, the probability of the influence propagation through either of paths is equal to

$P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

Events *A* and *B* are independents, so for the probability of the influence propagation through both paths we have $P(A \cap B) = p^5$. Given that $p<1$ is a small value, $P(A \cap B)$ is negligible compared to $P(A)$ and $P(B)$. Therefore, to make computations simple, we can find the required reduction amount in *v.priority* for each path independently, and then simply sum up those values.

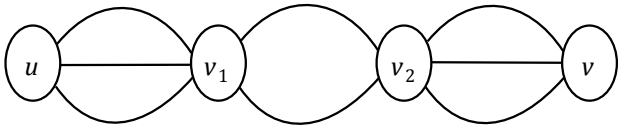

Figure 2. There are 3·2·3 different paths from *u* to *v*.

In Figure 2 there are 3·2·3 different paths from *u* to *v*. The length of each path is 3, and based on the above arguments, each path reduces a value from *v.priority* that depends on $f^3(p)$, which totally sum up to $3 \cdot 2 \cdot 3 \cdot f^3(p)$. For ease of processing, we introduce a recurrence relation. Let *v.decrease* denote the value to be reduced from *v.priority* for vertex *v*. The value of *v.decrease* for vertex *v* in Figure 2 can be written as

$v.decrease = v_2.decrease \cdot c(v_2,v) \cdot f(p)$.

In the above recurrence relation, $c(v_2,v)$ denotes the number of edges in G from v_2 to *v*. The intuition is that in v_2*.decrease* we take into account both the number of different paths from *u* to v_2 and the path reduction coefficient for those paths, $f^2(p)$. Therefore, it is enough to multiply v_2*.decrease* to the number of edges from v_2 to *v* and the path reduction coefficient to determine *v.decrease*.

We need to characterize $f(p)$ and the base case of the recurrence relation to be able to update *v.priority* for each vertex *v*. We choose two constant values α and β, whose exact value will be determined by further experiments, and define the functions based on these values. For the function $f(\cdot)$, we opt a linear function as $f(p) = \beta \cdot p$, and for the base case of the recurrence relation, we write $u.decrease = \alpha$, where *u* is the current seed vertex.

Since the value of *v.decrease* reduces as the distance of *u* to *v* grows, after enough hops, *v.decrease* gets close to 0 and we can stop further reduction process from *v.priority*. Based on our experiments, which are reported in Appendix A, we selected $\varepsilon = 0.1$ as the threshold value for priority reduction. When the value of *v.decrease* falls below ε, we stop further priority reduction propagation through *v*. If we select a small value as the threshold, the number of levels at which the breath-first search is performed is increased, and then the running time. On the other hand, choosing a large value reduces the number of levels of the breadth-first search, degrades the algorithm to normal maximum degree heuristic, and decreases the accuracy.

Since the vertices with high influence in social networks usually have high degrees, choosing $\varepsilon = 0.1$ results in both high accuracy and low running time. In addition, based on the experiments which are reported in Appendix A, the suitable value selected for α and β are 50 and 10, respectively. The pseudocode of the method is given in Algorithm 3.

Algorithm 3 DegreeDecrease

Input: $G(V, E)$: social network graph; k: size of the result set; p: propagation probability in the ICM.
Output: S: seed set.

$S \leftarrow \varnothing$
$V' \leftarrow V$ ▷ Make a copy of the vertices of G.
for each $v \in V'$ **do**
    $v.priority \leftarrow v.degree$ ▷ Initially, the priority of each vertex is equal to its degree.
for $i \leftarrow 1$ to k **do**
    $u \leftarrow \arg\max_{v \in V'} v.priority$ ▷ Select the vertex with the highest priority.
    $S \leftarrow S \cup \{u\}$
    $V' \leftarrow V' \setminus \{u\}$
    Set Q as an empty queue.
    for each $v \in V'$ **do**
        $v.visited \leftarrow$ **false**
    $u.visited \leftarrow$ **true** ▷ Perform a BFS starting from u.
    add u to Q
    $u.decrease \leftarrow \alpha$
    while Q is not empty **do**
        $v \leftarrow$ extract the vertex with the minimum priority from Q
        if $v.decrease > \varepsilon$ **then**
            for each $w \in v.adj \cap V'$ **and not** $w.visited$ **do**
                $w.visited \leftarrow$ **true**
                add w to Q
                $w.decrease \leftarrow v.decrease \cdot c(v, w) \cdot \beta \cdot p$ ▷ Compute the amount to be decreased from the priority of w.
                $w.priority \leftarrow w.priority - w.decrease$
return S

5. Experiments

In this section, we analyze and report the results of the experiments performed on the proposed heuristic algorithms and some previous algorithms using several real-life data-sets to evaluate the effectiveness of the new methods. We show that our maximum degree heuristics outperform previous degree-based heuristics in terms of the spread of the influence, while output a solution of quality close to the approximation algorithms.

5.1. Experimental settings

We evaluate our implementation on three data-sets which are commonly used in related researches, including [6]. The first data-set is NetHEPT with the number of vertices $n = 15233$ and the number of edges $m = 58891$. The second data-set is

NetPHY with n = 37154 and m = 231584. These two networks are collaboration graphs crawled from arXiv[1] website, in High Energy Physics – Theory section and Physics section, respectively. The third data-set is Epinions from Stanford Large Network Dataset Collection website [17], which is a who-trust-whom online social network of a general consumer review site with n = 75879 and m = 508837. All the above data-sets can be downloaded from the code repository of this paper[2].

We compare our algorithms represented by NeighborsRemove and DegreeDecrease with four algorithms named SingleDiscount [6], DegreeDiscount [6], TIM [8] and IMM [9] that are available by their authors. All algorithms are implemented in C++ and compiled with GCC 6.2.1 and are run on a system with an Intel Core i7–3820 @ 3.60GHz and 32GB memory.

5.2. Running times and influence spread analysis

Figure 3 and 4 show running times and influence spreads of different algorithms under independent cascade model on NetHEPT data-set for p = 0.01 and p = 0.1, respectively. Similar results are shown for NetPHY and Epinions data-sets in Figures 5, 6, 7, 8.

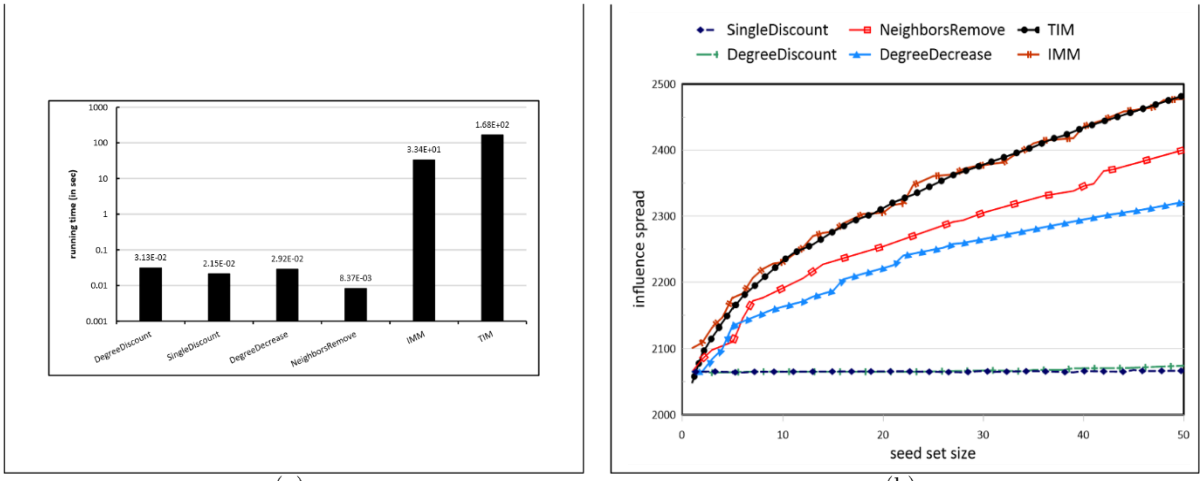

Figure 3. Running times (a) and influence spreads (b) of algorithms on NetHEPT under independent cascade model (p = 0.01, k = 50)

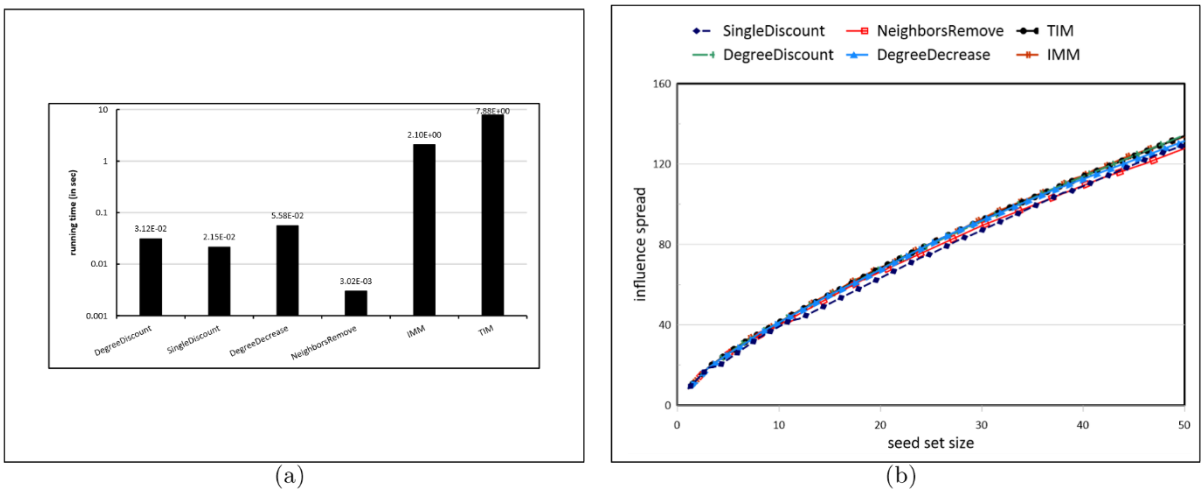

Figure 4. Running times (a) and influence spreads (b) of algorithms on NetHEPT under independent cascade model (p = 0.1, k = 50)

[1] https://arxiv.org
[2] https://github.com/Maryam-Adineh/InfluenceMaximization

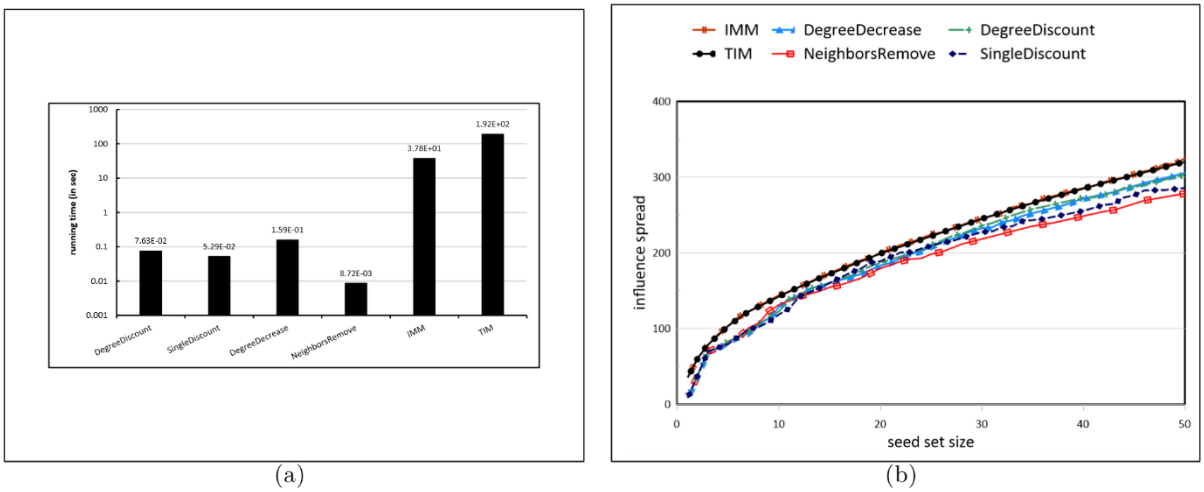

Figure 5. Running times (a) and influence spreads (b) of algorithms on NetPHY under independent cascade model (p = 0.01, k = 50)

As stated before, we see from the running times charts that the degree centrality heuristics are much faster than TIM and IMM. The running time of DegreeDecrease is usually close to DegreeDiscount and SingleDiscount, while NeighborsRemove is usually faster than all other algorithms. Sometimes the running time of NeighborsRemove is about 15% of the running time of the next fastest algorithm.

It can be seen from the influence spread charts that although the proposed algorithms show their superiority for large values of p, compared to DegreeDiscount and SingleDiscount heuristics, they still work well even for p = 0.01 and return solutions of quality close to the quality of solutions of TIM and IMM.

The effectiveness of our algorithms especially for larger values of p is because in those cases the influence of a seed vertex increases on its multi-hop neighbors. Therefore, there will be less advantage from selecting vertices close to the previous seeds. This is exactly one of the main ideas we follow in our proposed algorithms. Our strategy is to avoid selecting vertices with high probability of being influenced.

As it can be seen from the charts, for example Figure 6(b), the influence propagated by the results of our algorithms is sometimes about 15% more than the influence propagated by the results of DegreeDiscount and SingleDiscount, while the running times are less than or almost equal to their running times.

Figure 9 shows the influence spreads of different algorithms under independent cascade model for different values of p. As it can be seen, from p = 0.12 on, the influence spread of our algorithms significantly increases, compared to all other algorithms, both degree centrality heuristics and greedy algorithms.

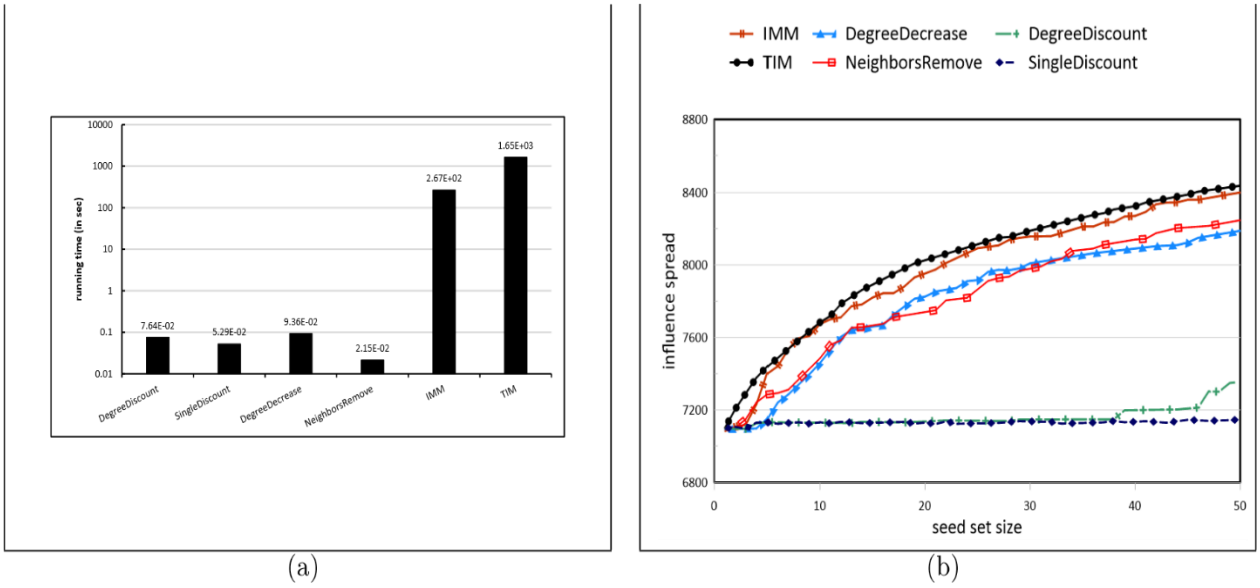

Figure 6. Running times (a) and influence spreads (b) of algorithms on NetPHY under independent cascade model (p = 0.1, k = 50)

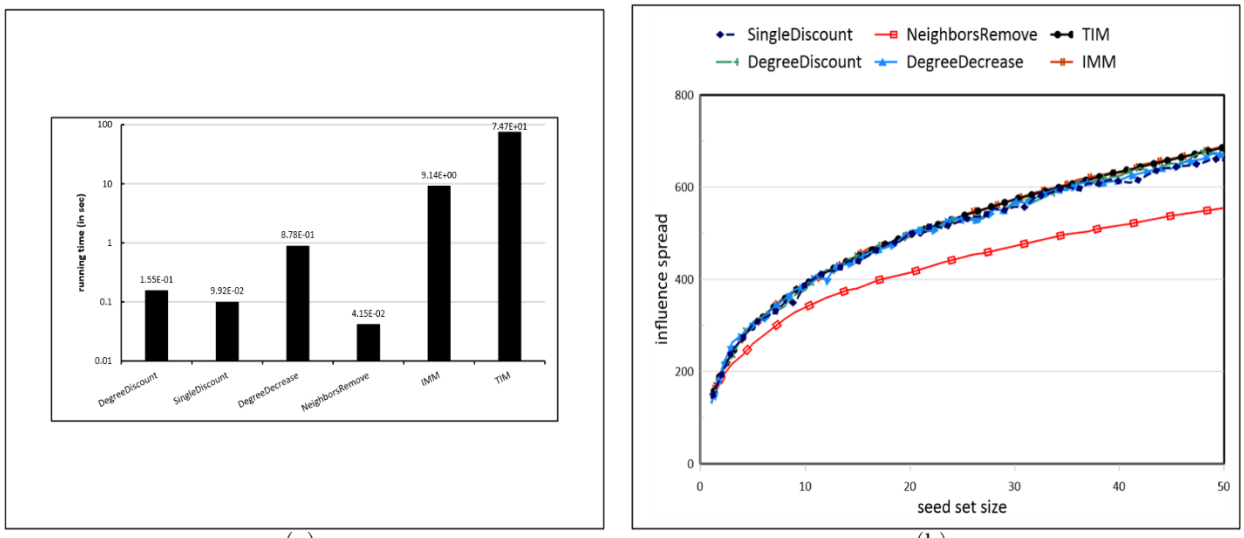

Figure 7. Running times (a) and influence spreads (b) of algorithms on Epinions under independent cascade model (p = 0.01, k = 50)

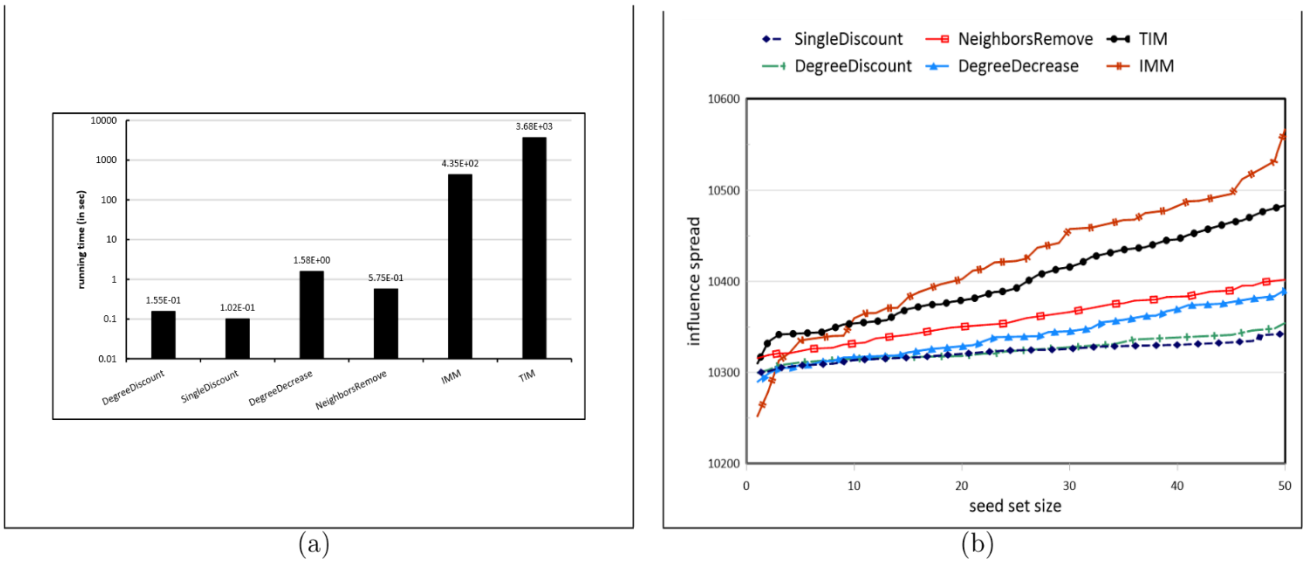

Figure 8. Running times (a) and influence spreads (b) of algorithms on Epinions under independent cascade model (p = 0.1, k = 50)

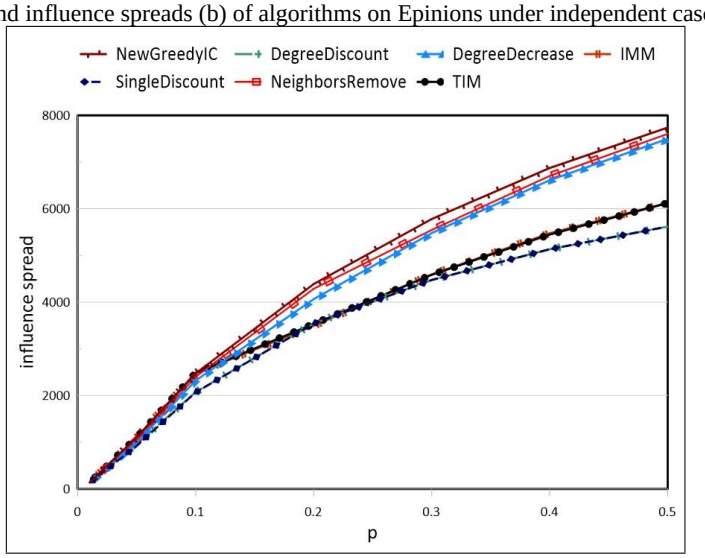

Figure 9. Comparison of influence spreads of different algorithms on NetHEPT under independent cascade model for different values of p

5.3. Ranking Similarity Analysis

In this section, we evaluate different influence maximization algorithms in terms of the similarity of the results to the results of the algorithm in Kempe et al. [3]. The similarity between two ranking methods, denoted by $F(k)$, represents the amount of similarity between the results of the methods and is defined as

$$f(k) = \frac{L(k) \cap L'(k)}{k}$$

where $L(k)$ and $L'(k)$ are the set of top-k nodes in the two ranking methods.

For the method of Kempe et al. [3], which we use as the true ranking, we consider the result of 20,000 times Monte Carlo simulations and compare the results of other algorithms with this ranking based on the ranking similarity.

In Figure 10, we see the comparison of ranking similarities on NetHEPT. Figure 10(a) shows that for p = 0.01, IMM and DegreeDiscount have the most ranking similarity to the true ranking. It can also be seen that the results of NeighborsRemove and DegreeDecrease have high similarity to the true ranking in the beginning but as the value of k increases the similarity decreases compared to other methods. On the other hand, for p = 0.1, Figure 10(b) shows that DegreeDecrease and especially NeighborsRemove have greater ranking similarity to the true ranking than SingleDiscount and DegreeDiscount, which proves the effectiveness of our methods for larger values of p.

Figure 11 shows the comparison of ranking similarities on NetPHY. In Figure 11(a) and 11(b), IMM has the closest ranking to the true one among all methods in the beginning, but as the result size increases, the difference between its ranking and the true ranking tends to increase. However, other methods show a different behavior. The ranking similarities of all methods are zero at first, and then with the growth in the result size the values tends to increase. As Figure 11(b) shows, DegreeDecrease and NeighborsRemove have better ranking in comparison with SingleDiscount and DegreeDiscount for larger values of p.

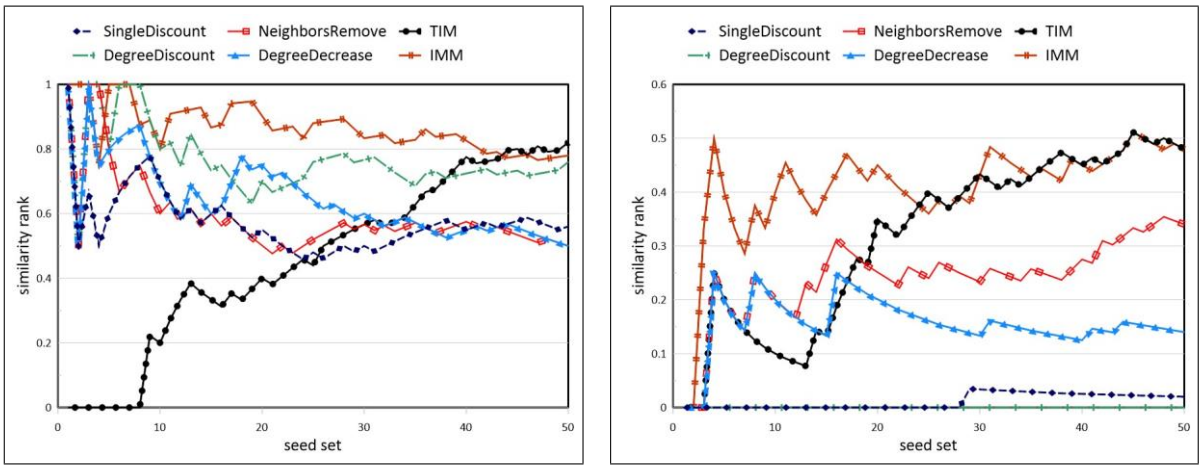

Figure 10. Rank similarity comparison on NetHEPT for p = 0.01 (a) and p = 0.1 (b)

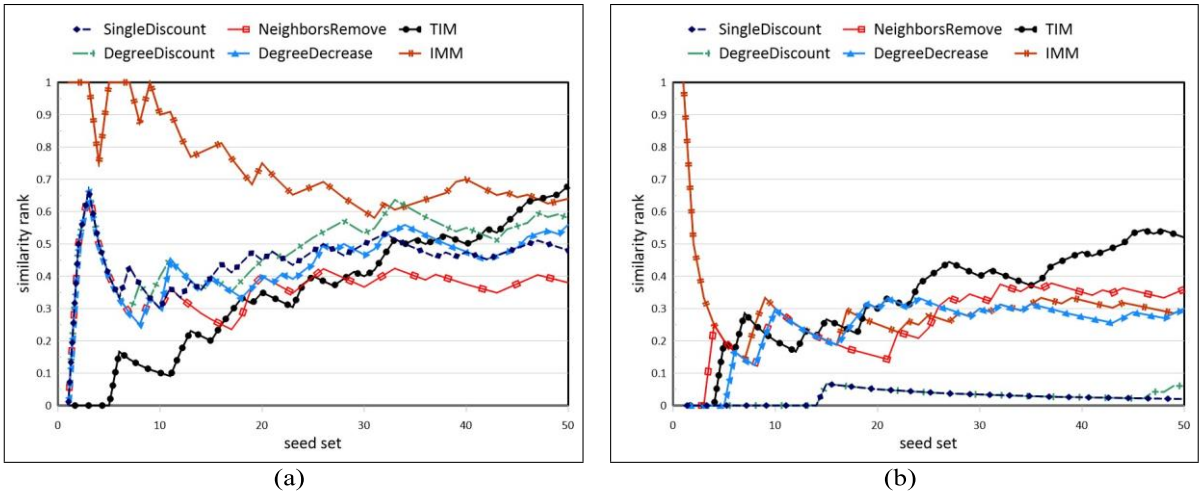

Figure 11. Rank similarity comparison on NetPHy for p = 0.01 (a) and p = 0.1 (b)

6. Conclusion

In this study we proposed two maximum degrees based heuristics for influence maximization problem under the independent cascade model. These heuristics take into account the idea that the vertices of high degree are close to each other in social networks. Experiments show that our heuristics outperform previous degree centrality heuristics in terms of the spread of influence in the network.

Since the algorithms that guarantee the quality of the outputs are very time-consuming on large-scale networks, finding heuristics which have small running time and producing solutions of good quality is so desirable. While the influence spread of the results produced by our proposed algorithms are close to the outputs generated by the approximation algorithms, the algorithms run in a much shorter time.

In future work, we will examine the maximum-degree based heuristics for other cascade models. Moreover, we are looking for more accurate strategies to improve the spread of influence with small running time.

7. References

[1] P. Domingos and M. Richardson, "Mining the network value of customers", *in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 57–66, ACM, (2001).

[2] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing", *in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 61–70, ACM, (2002).

[3] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network", *in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146, ACM, (2003).

[4] M. Adineh and M. Nouri-Baygi, "Maximum degree based heuristics for influence maximization", *in 2018 8th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 256–261, Oct (2018).

[5] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks", *in Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 420–429, ACM, (2007).

[6] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks", *in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 199–208, ACM, (2009).

[7] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time", *in Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 946– 957, SIAM, (2014).

[8] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency", *in Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 75–86, ACM, (2014).

[9] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach", *in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1539– 1554, ACM, (2015).

[10] D. Bucur and G. Iacca, "Influence maximization in social networks with genetic algorithms", *in European Conference on the Applications of Evolutionary Computation*, pp. 379–392, Springer, (2016).

[11] P. Krömer and J. Nowaková, "Guided genetic algorithm for the influence maximization problem", *in International Computing and Combinatorics Conference*, pp. 630–641, Springer, (2017).

[12] M. Weskida and R. Michalski, "Evolutionary algorithm for seed selection in social influence process", *in Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pp. 1189–1196, IEEE, (2016).

[13] Y.-C. Chen, W.-Y. Zhu, W.-C. Peng, W.-C. Lee, and S.-Y. Lee, "Cim: Community-based influence maximization in social networks", *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, No. 2, p. 25, (2014).

[14] B. Manaskasemsak, N. Dejkajonwuth, and A. Rungsawang, "Community centrality-based greedy approach for identifying top-k influencers in social networks", *in International Conference on ContextAware Systems and Applications*, pp. 141–150, Springer, (2015).

[15] G. Song, X. Zhou, Y. Wang, and K. Xie, "Influence maximization on large-scale mobile social network: a divide-and-conquer method", *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, No. 5, pp. 1379–1392, (2015).

[16] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network", *Theory of Computing*, vol. 11, no. 4, pp. 105–147, 2015.

[17] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection", http://snap. stanford.edu/data, (2014).

Appendices

Appendix A: Parameter Selection

In this section, we show the results of our experiments to choose the value of parameters of the algorithms. All the experiments are performed to select 50 seeds in the selected network. Three parameters in Degree Decrease are needed to be determined: α, β and ε. We have run several experiments on NetPHY data-sets for $p = 0.01$ and $p = 0.1$ with different values for α and β to find the best combination of values. The influence spreads are shown in Table 1 and 2. Based on the results of the experiments, we find the best selection as $\alpha = 50$ and $\beta = 10$.

The next parameter in Degree Decrease is ε. In Table 3 and 4 the influence spreads and the running time of Degree Decrease for different values of ε are reported. From the results and taking into consideration the fact that selecting a large value for ε may decrease the accuracy of the algorithm on other data-sets, we select the threshold value as $\varepsilon = 0.1$.

Table 5 represents the influence spread of NeighborsRemove for different values of h and p. The most influence in each row, which has been written in bold, shows the best value for h. According to the results we suggest $h = \lfloor 12\sqrt{p} \rceil$, as stated before.

Table 1. The inluence spread of Degree Decrease for different values of α and β on NetPHY for p= 0.01

β	α				
	10	20	30	40	50
2	268.418	273.005	276.168	287.087	287.077
3	273.309	277.849	286.867	290.409	291.459
5	277.808	288.308	293.268	292.866	298.408
7	281.396	293.107	291.946	300.539	301.49
10	287.748	294.812	300.445	301.515	305.889

Table 2. The inluence spread of Degree Decrease for different values of α and β on NetPHY for p= 0.01

β	α				
	10	20	30	40	50
2	7146.63	7333.55	7444.47	7548.64	7625.16
3	7341.11	7615.09	7737.2	7789.06	7840.71
5	7702.44	8005.31	8069.25	8089.31	8114.76
7	7911.9	8090.64	8134.55	8150.63	8154.23
10	8065.06	8100.42	8159.4	8177.86	8189.77

Table 3. The inluence spread of Degree Decrease for different values of ε

ε	p					
	NetHEPT		NetPHY		Epinions	
	0.01	0.1	0.01	0.1	0.01	0.1
0.01	127.67	2322.71	272.47	7636.74	673.76	10382.9
0.05	127.74	2318.49	273.19	7633.28	672.76	10382.8
0.1	128.34	2322.83	275.97	7638.84	674.85	10385.4
0.5	128.45	2320.71	272.32	7635.79	674.19	10384.2
1	127.92	2321.34	274.67	7636.55	670.25	10381.4
3	127.52	2320.71	280.41	7632.02	671.55	10382.1
5	129.43	2322.55	283.63	7635.15	671.91	10383.6
7	128.59	2322.82	283.77	7632.83	660.47	10382.7
10	128.57	2319.72	284.38	7630.14	661.04	10383.6

Table 4. The running time of Degree Decrease for different values of ε.

ε	p					
	NetHEPT		NetPHY		Epinions	
	0.01	0.1	0.01	0.1	0.01	0.1
0.01	0.0735	0.031	0.212	0.090	0.785	0.893
0.05	0.060	0.031	0.156	0.090	0.785	0.889
0.1	0.050	0.031	0.112	0.091	0.498	0.887
0.5	0.043	0.031	0.095	0.091	0.498	0.890
1	0.041	0.031	0.087	0.092	0.179	0.889
3	0.034	0.031	0.076	0.090	0.179	0.890
5	0.031	0.031	0.074	0.091	0.179	0.897
7	0.031	0.031	0.072	0.091	0.137	0.891
10	0.031	0.031	0.0717	0.090	0.137	0.891

Table 5. The influence spread of Neighbors Remove for different values of h and p

data-set	p	h						
		1	2	3	4	5	6	7
NetHEPT	0.01	**127.57**	101.89	82.66	74.80	72.17	70.54	69.7185
	0.05	981.78	1056.55	**1047.98**	1012.89	994.64	985.25	975.54
	0.1	2081.51	2279.97	2361.26	**2399.12**	2388.78	2357.75	2333.27
NetPHY	0.01	**280.41**	219.11	146.14	106.47	97.79	96.74	96.14
	0.05	3832.77	4155.09	**4331.41**	4288.06	4213.6	4210.28	4204.85
	0.1	7366.74	7799.77	8107.84	**8247.58**	8213.54	8208.30	8207.57
Epinions	0.01	**553.70**	283.80	227.55	226.97	230.99	226.901	227.036
	0.05	5724.05	5689.48	**5690.16**	5686.31	5687.40	5683.12	5685.07
	0.1	10348.3	10426.5	10409.4	**10400.9**	10395.1	10399.5	10396.4