Transferring Influence: Supervised Learning for Efficient Influence Maximization across Networks

Qingbo Hu*, Guan Wang†, Philip S. Yu*
*University of Illinois at Chicago, {qhu5, psyu}@uic.edu
†LinkedIn Co., gwang@linkedin.com

***Abstract*—How to maximize influence through social networks is a key challenge behind many important applications in real life. For instance, marketers are interested in how to use limited resource to promote a new product as widely recognized by consumers. In recent years, researchers have conducted numerous studies to conquer this intriguing problem in single network scenario. In terms of the scale of achieved influence, the best solution is a greedy algorithm based on time-consuming Monte Carlo (MC) simulation. However, it is not scalable to large-scale social networks or the scenario of targeting multiple networks. We propose an innovative Transfer Influence Learning (TIL) method based on the study on three real networks, as well as statistics on network features of results generated by the greedy algorithm. The proposed method uses supervised learning technique to efficiently maximize influence across multiple networks. Once having the result of the greedy algorithm in one network, the TIL algorithm can avoid using MC simulation completely on other networks, which enables the algorithm to run very fast. The experiments show that the proposed TIL algorithm is able to generate a diffusion with closed scale comparing to the result of the greedy algorithm within a much faster time, while outperforms some other state-of-art heuristic algorithms.**

***Keywords*—Data Mining, Social Network, Viral Marketing, Supervised Learning**

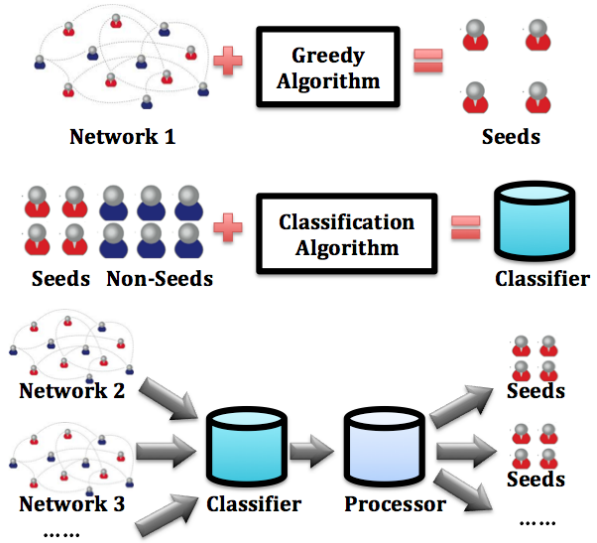


Fig. 1: Transfer Influence Learning (TIL) Framework

I. Introduction

Viral marketing is an efficient marketing strategy that aims to use limited budget to reach out as many customers as possible. The success of viral marketing mainly depends on the word-of-mouth effect through social networks initiated by the customers who favor the products. Therefore, the key problem is how to identify the most influential users as **seeds** and convince them to promote the product. Such problem is defined as the *Influence Maximization Problem* [13], [17].

In order to systematically study the Influence Maximization Problem, David Kempe et al. proposed general diffusion models, *Linear Threshold (LT) Model* and *Independent Cascade (IC) Model*, to describe how a piece of information is disseminated through social networks. Under the assumption of the IC or LT model, the Influence Maximization Problem can be formulated as a constrained maximization problem: given an integer, say k, how to pick k seeds so that the expected number of influenced users can be maximized at the end of influence propagation process. Therefore, some researchers also refer this problem as **k-seeds Maximization Problem** [21]. The Influence Maximization Problem is proved to be NP-hard under both LT and IC models, and the objective function (expected number of influenced users) is submodular. As a result, in terms of the scale of achieved influence, the state-of-art solution is a greedy algorithm proposed in [17], which guarantees a constant factor lower bound. At every step, the algorithm adds the node who generates the largest expected marginal gain of influenced users into the seed set, until the number of seeds reaches the constraint (k). Unfortunately, under both LT and IC models, calculating the expected number of influenced users by using an arbitrary seed set is proved to be #P-hard [6], [8]. As a result, the greedy algorithm proposed in [17] needs to use Monte Carlo (MC) simulation to approximate this number, which largely limits its scalability.

Previously, researchers proposed numerous improved solutions to compensate the disadvantage of the greedy algorithm [6]–[8], [16], [20]. These algorithms can efficiently shorten the running time of the greedy algorithm by reducing the number of calls on MC simulation [16], [20] or shrinking the length of influence path to get an approximated result [7]. However, they still cannot avoid using MC simulation entirely. This makes the algorithms still implausible on the situation when we have multiple disjoint networks on which we want to maximize the influence. In real life, we may often encounter the situation of targeting at multiple networks: a real

COLLABORATECOM 2014, October 22-25, Miami, United States

DOI 10.4108/icst.collaboratecom.2014.257260

social network is usually huge (hundreds of millions of nodes and millions of edges) and the marketing budget is limited. Therefore, instead of applying viral marketing to the entire network directly, many companies prefer to first partite it into subgraphs based on users' relationship then target at some selected subgraphs. In this case, applying some state-of-the-art algorithms on these subgraphs one by one seems not efficient anymore. Unlike previous solutions, this paper designs an innovative framework, ***Transfer Influence Learning (TIL)***, to utilize the results of the greedy algorithm on one network to maximize the influence on other networks. After we run the greedy algorithm on one network, the proposed framework is able to completely avoid MC simulations on other networks, which guarantees its efficiency in the setting of multiple target networks. To put it simple, the algorithm trains a classifier based on the results of the greedy algorithm and use it to directly decide whether a node should be selected. Fig. 1 is a sketch of the designed framework. Since the study of classification algorithms is very mature, the true challenge in the proposed framework is to choose appropriate node features. First, unlike other classification task is able to use any information of a node, we are limited to only network-structure related features, since the Influence Maximization Problem is a graph based optimization problem. In addition, the feature values should be transformed to a network-independent form in order to guarantee the trained classifier works universally on different networks. At last, previous works have demonstrated that some single features, such as degree centrality, is able to empower efficient heuristic algorithms [6], [17]. We need to ensure the selected features are able to combine the results of the greedy algorithms well to generate lager-scale diffusions comparing to heuristic algorithms.

Based on three real coauthor networks, we fully analyze 12 most common network-structure related features of a node and their correlation to the results of the greedy algorithm. We have also explored the heuristic algorithms empowered by these features. After feature selection, we combine 4 of them with a logistic regression classifier in the proposed TIL framework. Later experiments have shown the TIL is able to generate a close-scale diffusion to the greedy algorithm, and significantly outperforms any other heuristic solution. The experiments also show that the TIL is considerably scalable in a multiple network scenario comparing to the greedy algorithm. We summarize our major contributions as follows:

- As far as we know, this paper is the first article to study the problem of using supervised learning to efficiently maximize influence across multiple networks.
- We propose a new framework, ***Transfer Influence Learning (TIL)***, which uses supervised learning based on the result of the greedy algorithm on one network to guide the seed selection on other networks.
- We fully analyzed 12 basic heuristic algorithms by using different network features. Moreover, based on statistics, we further made a simple, yet efficient improvement for basic heuristics. The same insight is also applied to the proposed TIL method. The improved heuristics and several other previous algorithms are served as baselines to evaluate the proposed algorithm.
- Based on experiments on real-world networks, we have illustrated that the proposed algorithm is able to run in constant time like heuristic algorithms yet still generate a similar sized diffusion like the greedy algorithm.

The rest of paper is organized as follows: Section II introduces related work, which covers necessary backgrounds, as well as previous works of other researchers. Section III introduces the datasets we have used, as well as statistical discoveries on network features which support the proposed algorithm. Section IV discusses the performance of 12 different heuristic algorithms and the improvement of them. These heuristics are used as baselines to evaluate the proposed approach. Section V introduces the proposed Transfer Influence Learning framework and the experimental evaluation of it. At last, Section VI is the conclusion of this article.

II. Related Work

Influence Maximization Problem is first studied by Domingos et al. in [13] and formulated by David Kempe et al. in [17]. Previous research has made important progress in the study of this topic, including the proposition of general diffusion models. General diffusion models are used to simplify the description of the influence propagation process through social networks. Although many works attempt to design more accurate diffusion models [5], [23], the most frequently used ones nowadays are still Independent Cascade (IC) Model and Linear Threshold (LT) Model due to their simplicity and generality. The IC Model was first proposed in [17]. It views the influence between nodes as independent trials with certain probabilities to succeed. To be more specific, once a node accepts a piece of information (called active node), it is given one chance to activate each of its inactive neighbors (nodes who have not accepted the information). When the trial is successful, the activated neighbor will further attempt to activate its own inactive neighbors, in which case causes a cascade spreading the information. The probabilities to succeed are modeled as parameters and only depend on each pair of users.

Unfortunately, the Influence Maximization Problem is proved to be NP-hard under the LT and IC model. In terms of the scale of final influenced users, the best known approach to solve it is the greedy algorithm proposed in [17]. The greedy algorithm utilizes the *monotonocity* and *submodularity* of the objective function of the Influence Maximization Problem, which guarantees that the result has a constant factor lower bound of around 63% (proved by Cornuejols et al. in [9], [22]). However, the original greedy algorithm is not scalable to multiple-network scenario, since it needs to use Monte Carlo (MC) simulation to approximate the expected number of influenced nodes for any seed set. Naturally, many researchers have attempted to develop faster algorithms. For example, CELF [20] uses a priority queue to restore the computed marginal gains of each node. At each time, CELF only needs to update a few nodes at the top before making the correct decision on the next seed to select. This idea largely reduces the number of calls on MC simulation while still achieve the same scale of diffusion as the original greedy algorithm. Another improvement of CELF is called CELF++ [16], which can further

reduce the number of MC simulation calls. Other related works include the approaches proposed by Wei Chen et al. in [6]–[8]. For example, MIA [6] uses reduced subnetworks of the original network by ignoring pathes with insignificant probabilities. The result calculated by MC simulation on these subnetworks is much cost-effective and can be used to approximate the result of the original network. However, in order to reach the balance between performance and efficiency, the parameter of this approach needs to be adjusted accordingly, which may be network-dependent. The major differences between the proposed Transfer Influence Learning (TIL) algorithm in this paper and previous work lie in the following aspects: (1) instead of one target network, TIL addresses the problem of maximizing influence across multiple networks; (2) we treat the Influence Maximization Problem as a classification task and use trained classifiers to directly select nodes based on their features. This allows the proposed algorithm to utilize the result of the greedy algorithm on one network and completely avoid using MC simulation on other networks. Therefore, the proposed TIL algorithm is very competible with respect to the running time. Our experimental evaluation demonstrates that the TIL algorithm also generates seeds which can initiate a wide spread of the information.

III. Description of Datasets

A. Network Generation and Preprocess

In this paper, our work is mainly based on networks generated from a 1GB dataset, which is retrieved from DBLP[1]. The dataset contains 1,397,240 articles published in the computer science area until the year of 2011. More particularly, the dataset includes important information related to each paper, such as the names of authors, published date and etc.

To begin with, we construct a coauthor network based on all these papers. After we remove those authors who do not have any coauthor relationship (no connection at all), we obtain a refined network containing 866,055 nodes and 4,944,850 edges. Since our research is based on the Independent Cascade (IC) model, we also record the coauthor times of each pair of authors in order to calculate edge weights in the network. More specifically, as defined by [17], for any two connected authors, say u and v, the weight of their edge ($w_{u,v}$) is assigned as follows, where t is the coauthor times of two authors, and p is a small probability:

$$w_{u,v} = 1 - (1-p)^t \tag{1}$$

Similar to the original definition of the IC model, we apply a uniform probability to the value of p. For our network, $p = 5\%$ is used in Eq. (1). In the second step, since our problem setting is maximizing influence on multiple disjoint networks, we use a fast clustering algorithm, Graclus [10]–[12], to further partite the original network into 100 small subnetworks. The majority of the content in this paper is based on three networks with similar sizes from the obtained 100 subnetworks: network 1 contains 12,651 nodes with 38,163 edges connecting them. The number of nodes for network 2 and 3 are 10,945 and 11,402, respectively, and the number of edges are 32,961 and 32,004, respectively.

B. Node Features Generation

The features associated with each node in the network are all derived from the network structure itself and generated by Gephi[2]. Moreover, in order to guarantee that the proposed algorithm is universally applicable to all networks, instead of directly using the absolute values of the features, we transform all the values to network-independent format, such as the corresponding rank in the network. As a result, we have selected 12 most common features that are constantly used in the study of social networks. The reason why we choose such 12 features is that they almost thoroughly represent the structure-related characteristics of a node, and they are usually precomputed and stored before any specific data mining task. Moreover, generating these features is much less time-consuming than directly running the greedy algorithm on the network. We list and introduce all of them as follows:

1. **Degree Centrality.** Degree centrality is the number of edges that a node has. In this paper, since the networks we have are all undirected, there is no difference between in-degree and out-degree. However, it is easy to extend the content in the paper to a directed network scenario by introducing two types of degrees.
2. **Weighted Degree.** Unlike degree centrality, weighted degree also takes the weight of each edge into consideration. To be more specific, weighted degree of a node u is defined as $\sum_v w_{u,v}$, where v is a neighbor of u.
3. **Eccentricity.** Eccentricity of a node is the greatest geodesic distance between this node to any other node, where geodesic distance between two nodes is the length of the shortest path connecting them. Therefore, with a smaller eccentricity value, the node may be easier to reach other nodes.
4. **Closeness Centrality.** Similar to the definition of eccentricity, the closeness centrality of a node is also calculated from its geodesic distance to any other nodes. However, instead of using the greatest value like eccentricity, closeness centrality takes the average value of the geodesic distance from the node to others.
5. **Betweenness Centrality.** According to Wikipedia[3], betweenness centrality "is equal to the number of shortest paths from all vertices to all others that pass through that node". We generated this feature by Gephi, which uses the algorithm introduced in [2].
6. **Authority Score.** Authority score is the result generated by HITS algorithm [18]. During the calculation, we set the stopping threshold of HITS algorithm to be 10^{-6} for all networks. We do not include the hub score into our study because of two reasons: (1) in an undirected network, authority score and hub score are exactly the same; (2) authority score is the metric that measures the "reputation" of a node, which is more likely to be the characteristic of an "influencer".

[1] http://www.informatik.uni-trier.de/∼ley/db/

[2] http://gephi.org/

[3] http://www.wikipedia.com

7. **PageRank Score.** Similar to the HITS algorithm, PageRank [3] also provides the ranking information of the "importance" of nodes. We set the stopping threshold of PageRank to be 10^{-6} and set the probability that a surfer will randomly restart its walking to be 0.85.
8. **Modularity Class Size.** For each network, we apply a modular community detection algorithm [1] to further discover the inner community structure of the network. Modularity class size is the number of nodes in the same community that an individual is clustered into. The value of resolution is set to be 1.0.
9. **Component Size.** We find different Weakly Connected Components (WCCs) in each network. Of course, since our networks are all undirected, WCC is simply just connected component. Component size is a feature recording the number of nodes in the same connected component that a node belongs to.
10. **Clustering Coefficient.** According to [14], "the clustering coefficient of a node A is defined as the probability that two randomly selected friends of A are friends with each other". In this paper, we use the method proposed in [19] to compute this metric.
11. **Number of Triangles.** Triadic closure is a very important principle in social science and related research fields. A node's number of triangles feature is the number of triangles of which vertices contain the node. This is an indicator of whether the neighborhood of a node is saturated, as well as the neighborhood's compactness.
12. **Eigenvector Centrality.** Eigenvector centrality is also a frequently used measurement that evaluates a node's influence. We set the number of iterations to be 10,000 in Gephi to generate this feature.

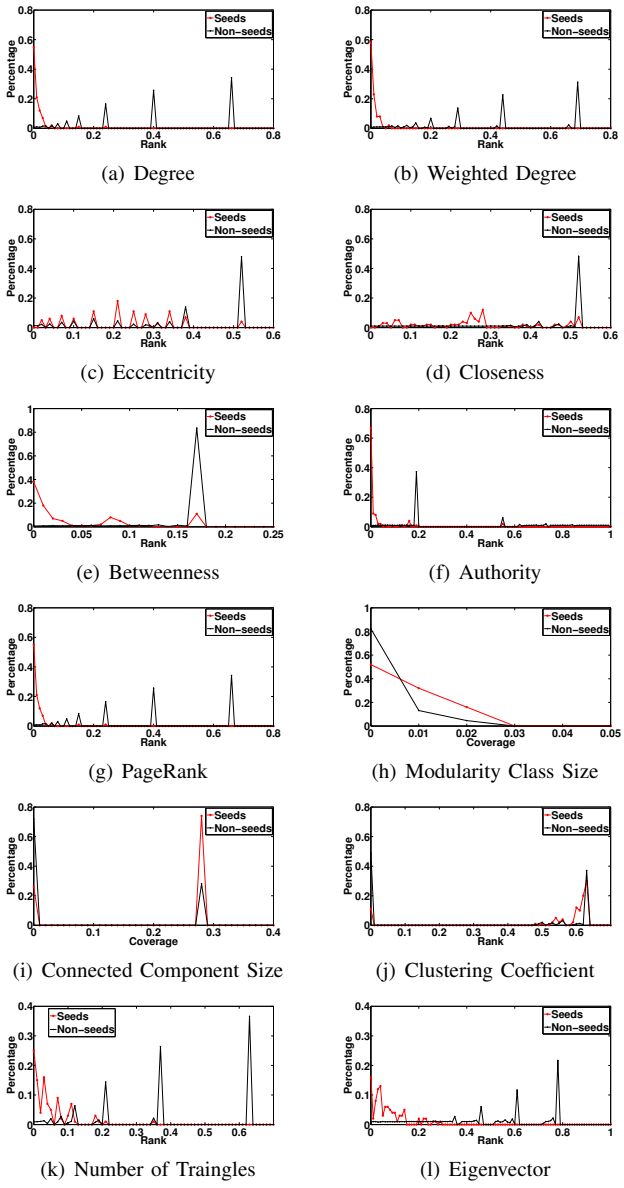
As we stated before, after we obtain the absolute value of each feature, we need to transform them to network-independent format. The reason to do this is that if we apply a classification algorithm to learn the connection between node features and the result of the greedy algorithm, such transformation will guarantee that the learned rules are still applicable on a different network. For example, if we use absolute feature values, an obtained decision tree classifier may have an inner node of whether the degree is greater than a certain value, say 3. Although the value of 3 is a good threshold to divide users in one network, it may be meaningless to another one, since all the nodes in that network may have a degree greater than 3. In this case, the trained classifier becomes useless. Compare to the absolute values, network-independent format such as the rank of a value in the entire network seems to be able to overcome such problem. Thus, for the feature 1$\sim$7 and 10$\sim$12, we replace the numerical values by the percentages they rank in a **descending** order (a.k.a. $\frac{rank}{\#nodes}$). In other words, the greater a value is, the higher it ranks in the network, and the transformed value, therefore, tends to be smaller. As for the feature 8 and 9, on the other hand, the absolute values are replaced by the percentage of the coverage of the community/component to the whole network. We dealt with these two features differently due to the nature of them (sizes of sub-networks). As a result, if a node stays in a community/component with larger size, the transformed result tends to be larger as well.

C. Statistics of Seed Sets of the Greedy Algorithm

In order to demonstrate the possibility of using supervised learning to perform efficient seed selection, we use statistical methods to display the connection between a node's features and whether it will be selected by the greedy algorithm of the Influence Maximization Problem. Intuitively, one may think that such connection is insignificant, since at each step, the greedy algorithm involves a more complicated consecutive selection process: it maximizes the expected marginal gain with respect to the current seed set. Such idea is very different from that simply using the features of a node to decide whether it should be selected as a seed. However, we think the existence of such connection is already powerful enough to support a brand new seed selection algorithm. Many previous works have shown that simple heuristics [17] or heuristics with slight improvement [7], albeit not as good as the greedy algorithm, may still generate a diffusion with the scale closed to the greedy algorithm. Such results are indirect evidence that hints the greedy algorithm may prefer nodes with "good" characteristics. Such selected nodes may overlap the seeds generated by heuristic algorithms. Therefore, the combination of node features and the result of the greedy algorithm may generate an efficient method to guide future seed selection.

In order to test our conjecture, we decide to use statistical analysis based on the three coauthor networks to display such connection. We first use the greedy algorithm with MC simulations (developed by Amit Goyal et al. in [15]) on three networks to generate 100 seeds. We label the seeds "1" (positive class) and non-seeds "0" (negative class), respectively. Then, we draw the probability mass functions on different features for the two classes. Because of the space limit of the paper, we only display the figures generated from the network 1 in Fig. 2. The statistics on other networks are similar. The X axis in Fig. 2 is the value of each transformed feature. The Y axis is the percentage of seeds/non-seeds having a certain value of the feature.

As we can see in Fig. 2, the probability mass functions of many features are very different for seeds and non-seeds. For most features using ranking percentage transformation, seeds tend to have smaller values than non-seeds. Such observation is very notable in features such as degree, weighted degree, PageRank score and etc. The peaks of seeds usually accumulate on small values, while peaks appear on large values for non-seeds. In other words, the seed chosen by the greedy algorithm tends to have higher degree/weighted degree/PageRank score and etc. than a non-seed. Therefore, it is not surprising to see some heuristic algorithms using centralities such as degree or PageRank may also generate considerably good diffusion comparing to the greedy algorithm. The exception of ranking percentage transformed features is the clustering coefficient. Our statistical results support that the greedy algorithm prefers nodes with lower ranks on the clustering coefficient, which means smaller absolute values. One possible explanation to such phenomenon is that a node with smaller value of clustering coefficient is more likely to have higher degrees, and thus



(a) Degree (b) Weighted Degree

(c) Eccentricity (d) Closeness

(e) Betweenness (f) Authority

(g) PageRank (h) Modularity Class Size

(i) Connected Component Size (j) Clustering Coefficient

(k) Number of Traingles (l) Eigenvector

Fig. 2: Feature Statistics of Seeds and Non-Seeds

it has the potential to influence more nodes. However, with the increase of degrees, the probability for every pair of neighbors being friends decreases intuitively, which results in the value of the clustering coefficient to be smaller.

The statistics on most ranking percentage transformed features are consistent with our understanding of the nature of the features. For example, nodes with higher degrees also have the potential to influence more nodes, so it is more likely to be selected by the greedy algorithm. However, the statistics on eccentricity and closeness are surprisingly against the intuition. The results demonstrate that the greedy algorithm prefers nodes with higher ranked eccentricity and closeness, which are the nodes having large values of eccentricity and closeness. This phenomenon is not easy to understand, since intuitively, a node with shorter (average or maximal) distance to other nodes is closer to the "center" of the network. Thus, such node should also be preferable for the greedy algorithm. However, the truth is not like this. In our opinion, we believe unlike degree centrality, distance centrality is not very indicative of a node's influence, since the influence to other nodes decreases exponentially with the increase of distance. Therefore, although a node may have a small value of eccentricity or closeness, it still can not assure that it will have a great influence on those nodes which are only two or three hops away. This may also be the reason why previous studies like [17] have shown that distance-based heuristics are usually not as promising as degree-based heuristics.

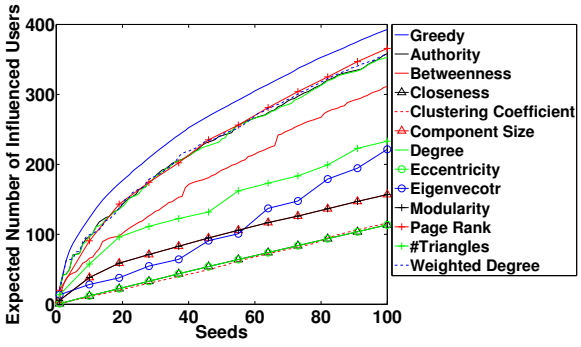
For features using the coverage percentage transformation (modularity class and component size), seeds generated by the greedy algorithm usually have larger values than non-seeds, which means the greedy algorithm also prefers to select nodes that belong to a large modularity class/component. This phenomenon is easy to understand, since such nodes have the potential to influence more other nodes in the same class/component. Generally speaking, the statistical observation is consistent in all three networks. In the next section, we will introduce heuristic solutions to Influence Maximization based on extracted features and use experiments to evaluate them. The improved version of such heuristics are served as one of the baselines to evaluate the proposed method.

IV. Heuristic Algorithms

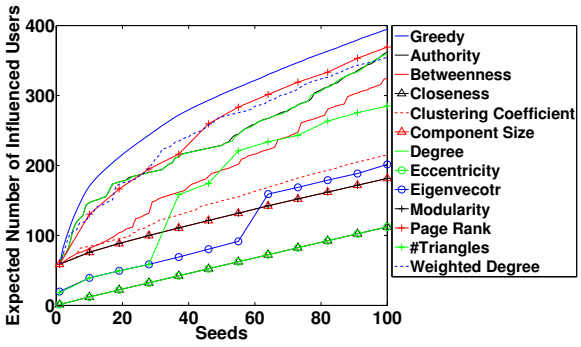
A. Basic Heuristics

Based on different features we have extracted in the previous section, we develop basic heuristic algorithms by simply using the order of nodes according to the transformed feature values. Whether to use the descending or ascending order to select seeds is usually straightforward according to the definition of the feature. For instance, there is no doubt that we should prefer nodes with higher degrees and greater authority scores (a.k.a. smaller values after ranking percentage transformation). Moreover, the statistics of the seeds of the greedy algorithm also support these judgements. The only exceptions are eccentricity and closeness centrality. Intuitively, we should prefer lower ranked nodes, since they have smaller values of eccentricity or closeness. However, the statistics on seeds of the greedy algorithm suggest us to take the opposite way. As a result, we compare these two algorithms first and found out that the heuristic based on our intuition is slightly better. As a result, we decide to use the descending order of the transformed values for eccentricity and closeness (a.k.a. choose smaller absolute values).

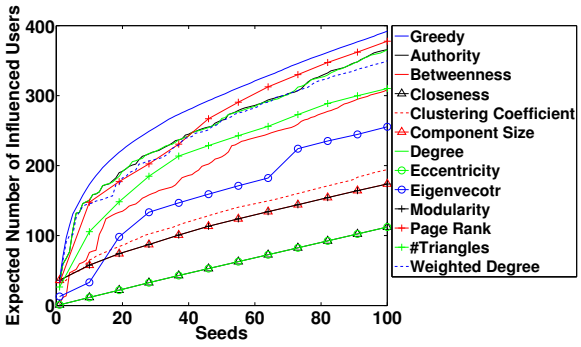
To sum up, for features in the network-independent format, on the one hand, we use the ascending order of the transformed values to select seeds for all features except for five of them: modularity class size, component size, clustering coefficient, eccentricity and closeness. On the other hand, for these five features, the heuristic algorithm should prefer to choose seeds in the descending order. In the evaluation, for each feature, we use the heuristics to select 1$\sim$100 seeds and calculate



(a) Network1



(b) Network 2



(c) Network 3

Fig. 3: Performance of Heuristic Solutions on DBLP Networks

the expected values of the number of influenced users by using them as seeds. The expected number of influenced users is computed by using 10,000 iterations of simulation of the influence process under the IC model. Fig. 3 shows the comparison of the 12 basic heuristics on three coauthor networks.

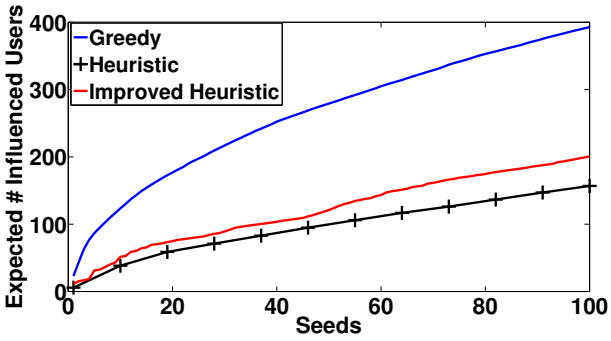
Through comparison, we can see that the difference between the degree and weighted degree centrality is not significant, which means whether to consider the weight of each edge does not change the result too much. By further examining the overlap of seeds selected by these two heuristics, we found out that 67% of their seeds are the same for network 1. This percentage for network 2 and 3 are 71% and 75%, respectively. Such observation suggests that a high degree node usually may also have a high weighted degree. In addition, we found out that the seeds of eccentricity and closeness heuristics overlap completely in all three networks, and the same situation also happens in heuristics using modularity and component size. Such results tell us for a seed set whose size is much smaller than the network, there is a high probability that some heuristics are the same. Although random picking nodes when there are multiple candidates having the same rank may differentiate two heuristics, our further experiment shows that this does not change the algorithm's performance too much. Moreover, we discover that PageRank heuristic is constantly better than others in all three networks, which is consistent with the findings in [6].

B. Improved Heuristics

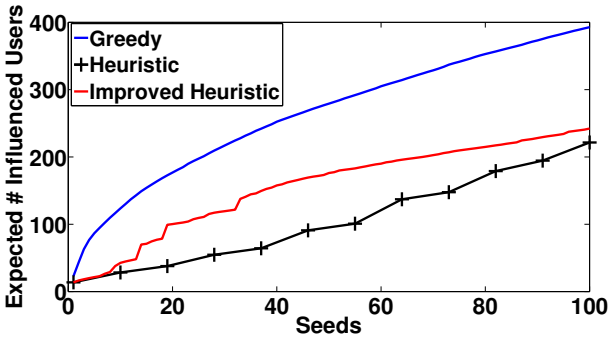
Through examining the ***neighborhood overlap*** [14] between each two seeds selected by the greedy algorithm, we found out that 98.04% of all seed pairs in the network 1 do not share any common neighbors. For the network 2 and network 3, these percentage numbers are even larger (98.3% and 98.61%, respectively). Therefore, a natural thought to obtain a seed set with better quality is that at each step of increasing the seed set, we should not select nodes which have a large overlap on neighbors with any node in the current seed set. The similar idea empowers the Degree Discount Algorithm proposed in [7]. However, unlike [7] only has degree heuristic, we have 12 different heuristics, which is not easy to compute an optimal "discount" for each of them. As a result, in order to obtain better heuristic algorithms, we impose a stronger restriction on the 12 different heuristics, where we do not allow any two selected seeds to have any common neighbor. In other words, for each heuristic algorithm, we still select seeds according to the order of feature values. The only difference is that after we add a new seed, in the future selection, we will not consider those nodes which share common neighbors with it. This rule is also adopted in the proposed Transfer Influence Learning (TIL) algorithm to obtain better seed set, which is going to be introduced in the next section.

Our experiments show that such improvement on heuristics is not equally effective for every heuristic. The improvement on most heuristics can increase the expected number of the influenced nodes less than 10 for a seed set of size 100. However, for the component size, eigenvector and number of triangles, the enhancement is more obvious. The improvement on these 3 heuristics on network 1 is shown in Fig. 4, and the results on other networks are similar. To our surprise, the effect of this refinement step on the heuristic using component size is much larger than the heuristic using modular class size. We should notice that before such refinement, these two algorithms are identical to each other in the previous section. Such difference suggests that the coverage of the connected component, to which a node belongs, may better illustrate the node's influence than the coverage of modularity class.

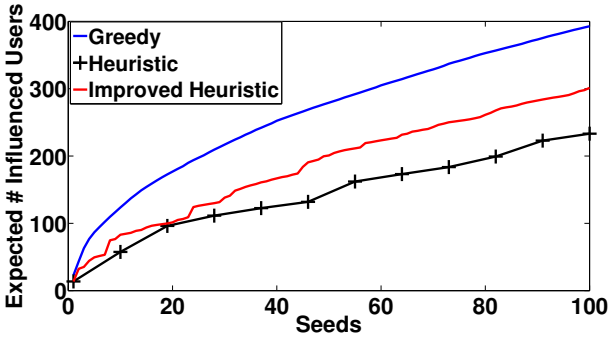
In the next section, we will introduce the method which combines the result of the greedy algorithm and heuristic



(a) Component Size



(b) Eigenvector



(c) # Triangles

Fig. 4: Improvements for Heuristics: Black curves with plus markers are the basic heuristics. Red curves are the improved heuristics. Blue curves are the results of the greedy algorithm.

features to obtain an efficient approach for the Influence Maximization Problem when targeting on multiple networks.

V. Transfer Influence Learning (TIL) Framework

On the one hand, the advantage of the greedy algorithm to solve the Influence Maximization Problem is that it can generate a large-scale diffusion, but the disadvantage is that it wastes too much time on MC simulation. On the other hand, heuristic algorithms using node features have the advantage of extremely fast running time, yet the generated diffusion is not as good as the greedy algorithm. In this section, we introduce a new method that uses supervised learning technique to train classifiers based on the features of nodes and the result of the greedy algorithm. The trained model can efficiently learn the hidden patterns of the greedy algorithm's preference, and thus can be used to guide the seed selection in other networks. The proposed Transfer Influence Learning (TIL) method is built according to the result of the greedy algorithm, yet it only uses the nodes' features to guide future seed selection. As a result, it provides a tradeoff between the efficiency of heuristic algorithms and the effectiveness of the greedy algorithm.

	Network 1	Network 2	Network 3
Feature #1	**Weighted Degree**	**Weighted Degree**	**Weighted Degree**
Feature #2	**PageRank Score**	**Authority Score**	**Authority Score**
Feature #3	**Degree**	**Degree**	**Degree**
Feature #4	**Authority Score**	**PageRank Score**	**PageRank Score**
Feature #5	Clustering Coefficient	# Triangles	Clustering Coefficient
Feature #6	Betweenness	Betweenness	# Triangles

TABLE I: Most Important Features in Networks (Ordered by the Importance)

A. Model Description

Feature Selection. For the purpose of simplifying the model, we are not using all 12 features for supervised learning. Therefore, we apply feature selection techniques to find informative features in the classification task, which should also be universally acceptable for all networks. We use the feature evaluation based on information gain to find most important features for the three networks. Table I lists the top 6 important features for each network. The ones in bold are the common features that exist in all three networks. We can see that the results of the evaluation of different features do not change too much for different networks, and the top 4 features are the same for all networks. Therefore, these 4 features might fulfill the requirement to be used in the supervised learning task.

Resampling Imbalanced Datasets. Another problem we need to deal with is the imbalanced training set. As we introduced before, since the greedy algorithm takes a long time to compute, for each network, we only generated 100 seeds. As a result, the positive class (a.k.a. seeds) contains very few samples comparing to the negative class. We apply resampling techniques to create a balanced dataset before applying any classification algorithm. We use random downsampling to reduce the major class and oversampling to increase the minor class until both of them reach the size of half of the original dataset. We also tried more complicated resampling approaches, such as SMOTE [4]. However, the advantage of using advanced resampling is not very significant. Moreover, considering that advanced resampling approaches usually take a long time to obtain the results, we think it is not cost-efficient to use them. Our experimental evaluation demonstrates that the simple sampling method is already good enough to obtain an efficient seed selection algorithm.

Training Classifiers and Seed Selection. For the selected four features, we decide to use the logistic regression algorithm to train classifiers. This is not only because of the excellent performance and generality of this discriminative model, but

also it can output a numerical score for each testing instance that represents how "confident" the classification result is. We sort all nodes in descending order of their scores generated by the classifier to obtain a candidate list. The order of this list is exactly the order how we select seeds. Of course, as we discussed before, when we add the current top ranked node into the seed set, we remove all other nodes sharing common neighbor(s) with it from the candidate list. We continue to select seeds until the size of seed set meets the problem constraint. Notice that here we accept one of the fundamental assumptions of the Influence Maximization Problem: the targeting network has much more nodes than the seeds, which assures that after removing nodes having common neighbor(s), the candidate list is still non-empty. Algorithm 1 presents the TIL framework.

Algorithm 1 Transfer Influence Learning (TIL) Framework

Input:
G_1: The social network from which the model is learnt.
S_1: The seed set of the greedy algorithm on G_1.
G_2: The social network we want to maximize influence.
k: The size of the required seed set in G_2
Output: S_2: Seed set for G_2 whose size is k.
1: Create a balanced training set T from G_1 and S_1.
2: Apply a classification algorithm on T and obtain the classifier C.
3: Feed nodes in G_2 to C and sort them according to the outputs of C. Let the obtained list be L.
4: S_2=$\varnothing$
5: **while** ($|S_2| < k$) **do**
6: $S_2 = S_2 \cup \{x|x$ is the top node in $L\}$
7: Remove all nodes who share common neighbor(s) with x out of L
8: **end while**
9: **return** S_2

B. Evaluation on Three Coauthor Networks

In order to evaluate the efficiency of Algorithm 1, we use each coauthor network and its 100 seeds selected by the greedy algorithm to train the model and use it to select seeds on each of the other two networks. In order to be precise, for each comparison algorithm, we run it for 50 times on each network and the reported results are the average values.

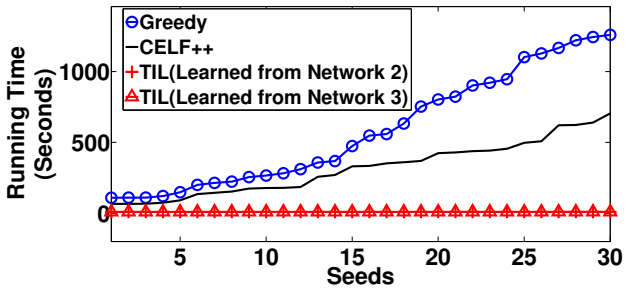
To show how much efficiency on the running time that the TIL algorithm can achieve, we have compared the average running time of the TIL to the greedy algorithm [17] and another state-of-art solution: CELF++ [16]. CELF++ algorithm is able to generate almost the same result as the original greedy algorithm, but runs much faster. Notice that for the network from which the TIL is learnt, since we need to run the greedy algorithm on it, the running time of the TIL method on that network will be exactly the same to the greedy algorithm. Therefore, the comparison focuses on the running time on those networks at which the trained model is targeting.

When we compute the running time of the TIL algorithm, we consider time of both training the classifier to select seeds and generating four types of node features that are used in the training task. For example, if we adopt the TIL from network 1 to network 2, the total time used on network 2 will be the sum of the running time of the following three steps: (1) generating four kinds of node features for all nodes in network 1 and 2; (2) training a classifier from the selected seeds in network 1; (3) use the classifier to select the seeds on network 2. We display the running time of using each algorithm to select 1$\sim$30 seeds on three networks in Fig. 5. We omit the running time of heuristic algorithms in this experiments, since they can finish in constant time for all sized seed set, which makes them indistinguishable with the TIL in Fig. 5. As one can see in Fig. 5, comparing to the original greedy algorithm and CELF++, the TIL algorithm is more scalable. The size of the seed set does not affect the efficiency of the TIL algorithm, since the TIL selects seeds similar to heuristic algorithms. In fact, the TIL can be done in less than 10 seconds on a target network that has around 10,000 nodes (the size of three coauthor networks).

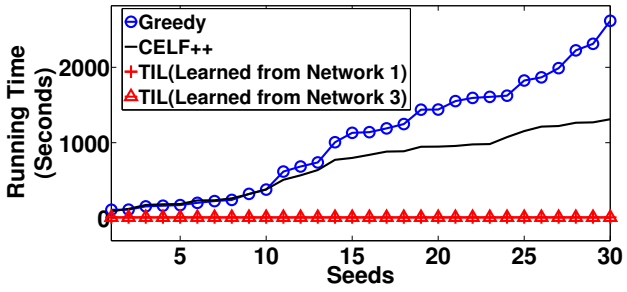
Next, we use experiments to show that although the TIL runs in constant time like heuristic algorithms, its selected seeds can generate much wider diffusion comparing to the seeds generated by heuristic ones. For each network, we use the seeds selected by different algorithms to simulate the diffusion process under the IC model. The expected number of final active users for each algorithm takes the average of 10,000 times of simulation results. Besides the original greedy algorithm and CELF++, we add two other heuristic baselines. The first one is the best results that can be generated by improved heuristic algorithms in the Section IV and Degree Discount heuristic [7]. Fig. 6 demonstrate the expected numbers of active users of 1$\sim$50 seeds selected by different algorithms. We choose 50 as the largest number of seeds to demonstrate the results because after 50 seeds, all algorithms are stable, a.k.a. adding one more seeds will only result the expected number of active users to increase less than 2. Moreover, notice that CELF++ and the greedy algorithm usually generate the same group of seeds, which makes their curves identical in the plots. Moreover, as we introduced before, these two greedy search based algorithms actually serve as an upper bound for other algorithms. The proposed TIL algorithm is able to generate a close-scale diffusion to the upper bound, and significantly outperforms the two heuristic baselines. Combining with the running time displayed in Fig. 5, we conclude that the TIL provides a tradeoff between the heuristic algorithm's efficiency and the greedy algorithm's effectiveness.

VI. Conclusion

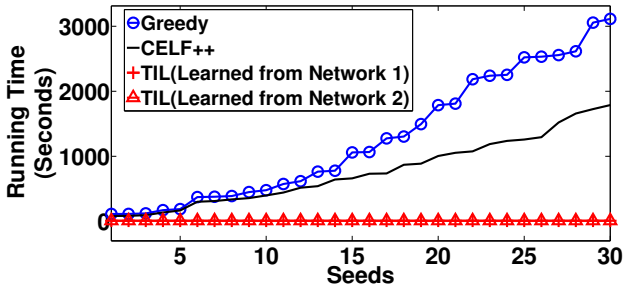
This paper confirms the possibility of using supervised learning technique to solve the Influence Maximization Problem in the scenario of multiple target networks. Based on three real coauthor networks, we use a statistical method to show how the difference between seeds and non-seeds of the greedy algorithm reflects on different node features. Without losing generality, we have tested 12 most frequently used features in the classification task. The statistics on them support that the difference between seeds and non-seeds of the greedy



(a) Network 1

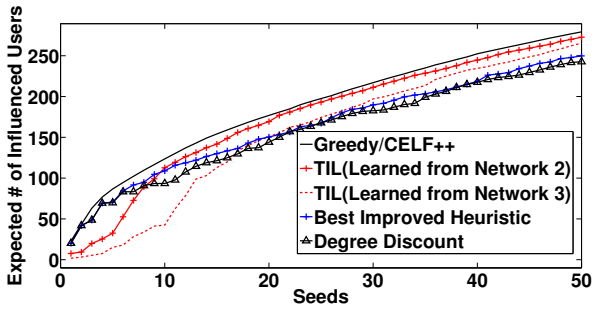


(b) Network 2

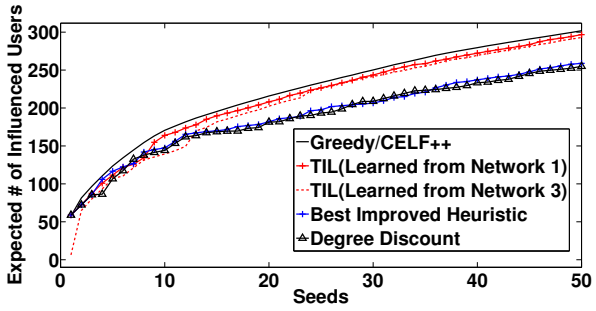


(c) Network 3

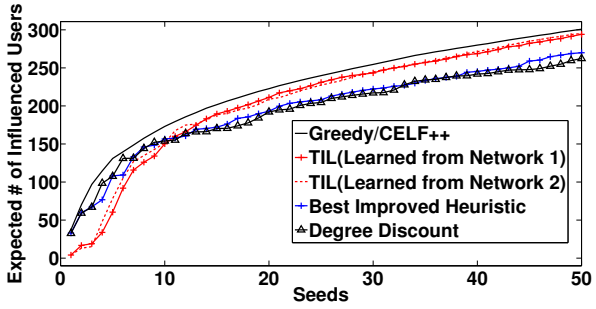
Fig. 5: Running Time of the Transfer Influence Learning Algorithm on Three Networks



(a) Network 1



(b) Network 2



(c) Network 3

Fig. 6: Evaluation of the Transfer Influence Learning Algorithm on Three Networks

algorithm can be reflected. In other words, although the greedy algorithm uses a hill-climbing method to select seeds dynamically, the statistics show that some characteristics of static node features cause a node more probable to be selected. Before introducing the proposed Transfer Influence Learning (TIL) algorithm, we have examined the performance of the heuristics of the 12 features, as well as how to improve them by a simple technique. At last, we have introduced the proposed algorithm, which uses a logistic regression classifier and four types of node features to solve the Influence Maximization Problem. Experimental evaluation has shown that the TIL is much more efficient comparing to some state-of-art greedy algorithms and more effective comparing to heuristic algorithms. This indicates that the TIL is preferable to be employed to maximize influence in a scenario of multiple homogenous target networks with similar sizes. However, how to extend the proposed TIL framework to the problem setting of heterogeneous networks and target networks of varied sizes still remains an unexplored and challenging task, and we intend to leave it for future work.

ACKNOWLEDGMENT

This work is supported in part by NSF through grants CNS-1115234, and OISE-1129076, and US Department of Army through grant W911NF-12-1-0066.

REFERENCES

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[2] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[4] N. V. Chawla, K. W. Bowyer, and et al. Smote: synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[5] W. Chen, A. Collins, and et al. Influence maximization in social networks when negative opinions may emerge and propagate. Technical report, Microsoft Research Asia, October 2010.

[6] W. Chen, C. Wang, and et al. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD '10*, pages 1029–1038, 2010.

[7] W. Chen, Y. Wang, and et al. Efficient influence maximization in social networks. In *KDD '09*, pages 199–208, 2009.

[8] W. Chen, Y. Yuan, and et al. Scalable influence maximization in social networks under the linear threshold model. In *ICDM '10*, pages 88–97. IEEE Computer Society, 2010.

[9] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management Science*, 23(8):789–810, 1977.

[10] I. Dhillon, Y. Guan, and et al. Kernel k-means: spectral clustering and normalized cuts. In *KDD '04*, pages 551–556, 2004.

[11] I. Dhillon, Y. Guan, and et al. A fast kernel-based multilevel algorithm for graph clustering. In *KDD '05*, pages 629–634, 2005.

[12] I. Dhillon, Y. Guan, and et al. Weighted graph cuts without eigenvectors a multilevel approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(11):1944–1957, 2007.

[13] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD '01*, pages 57–66, 2001.

[14] D. Easley and J. Kleinberg. *Networks, crowds, and markets*, volume 8. Cambridge University Press, 2010.

[15] A. Goyal, F. Bonchi, and et al. A data-based approach to social influence maximization. *VLDM '11*, 5(1):73–84, 2011.

[16] A. Goyal, W. Lu, and et al. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *WWW '11*, pages 47–48. ACM, 2011.

[17] D. Kempe, J. Kleinberg, and et al. Maximizing the spread of influence through a social network. In *KDD '03*, pages 137–146, 2003.

[18] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[19] M. Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theoretical Computer Science*, 407(1):458–473, 2008.

[20] J. Leskovec, A. Krause, and et al. Cost-effective outbreak detection in networks. In *KDD '07*, pages 420–429, 2007.

[21] C. Long and R. C.-W. Wong. Minimizing seed set for viral marketing. In *ICDM '01*, pages 427–436, 2011.

[22] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[23] G. Wang, Q. Hu, and P. Yu. Influence and similarity on heterogeneous networks. In *CIKM '12*, pages 1462–1466, 2012.