4-2013

Dynamic label propagation in social networks

Juan DU
Singapore Management University, juandu@smu.edu.sg

Feida ZHU
Singapore Management University, fdzhu@smu.edu.sg

Ee Peng LIM
Singapore Management University, eplim@smu.edu.sg

DOI: https://doi.org/10.1007/978-3-642-37450-0_14

Citation

DU, Juan; ZHU, Feida; and LIM, Ee Peng. Dynamic label propagation in social networks. (2013). *Database Systems for Advanced Applications: 18th International Conference, DASFAA 2013, Wuhan, China, April 22-25, 2013. Proceedings, Part II*. 7826, 194-209. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/1733

Dynamic Label Propagation in Social Networks

Juan Du, Feida Zhu, and Ee-Peng Lim

School of Information System,
Singapore Management University
{juandu,fdzhu,eplim}@smu.edu.sg

Abstract. Label propagation has been studied for many years, starting from a set of nodes with labels and then propagating to those without labels. In social networks, building complete user profiles like interests and affiliations contributes to the systems like link prediction, personalized feeding, etc. Since the labels for each user are mostly not filled, we often employ some people to label these users. And therefore, the cost of human labeling is high if the data set is large. To reduce the expense, we need to select the optimal data set for labeling, which produces the best propagation result.

In this paper, we proposed two algorithms for the selection of the optimal data set for labeling, which is the *greedy* and *greedyMax* algorithms according to different user input. We select the data set according to two scenarios, which are 1) finding top-K nodes for labeling and then propagating as much nodes as possible, and 2) finding a minimal set of nodes for labeling and then propagating the whole network with at least one label. Furthermore, we analyze the network structure that affects the selection and propagation results. Our algorithms are suitable for most propagation algorithms. In the experiment part, we evaluate our algorithms based on 500 networks extracted from the film-actor table in freebase according to the two different scenarios. The performance including input percentage, time cost, precision and f1-score were present in the results. And from the results, the greedyMax could achieve higher performance with a balance of precision and time cost than the greedy algorithm. In addition, our algorithm could be adaptive to the user input in a quick response.

1 Introduction

The problem of label propagation has in recent years attracted a great deal of research attention [12, 17, 4], especially in the setting of social networks where an important application of it is to better understand the elements of the network, such as user profiles [8]. As user profiles are often represented by node labels denoting their interests, affiliations, occupations, etc, it is therefore desirable to know the correct labels for as many nodes as possible. However, in real-life social network applications, complete label information of the entire network is rare due to users' privacy concern and unwillingness to supply the information. Consequently, label propagation has been widely used to derive from the known labels of a subset of nodes the unknown ones of the other nodes for the rest of the network [1]. The underlying assumption is the well-observed phenomenon of "homophily" in social networks, i.e., users with strong social connections tend to share similar social attributes.

To trigger the label propagation process over a social network, we need to first acquire the correct labels for an initial set of nodes, which we called a *seed set*. As the acquisition cost of the labels is usually high, e.g., by human labeling and verification, the goal for label propagation in these settings is usually to find as small a seed set as possible such that the knowledge of these node labels would maximize the label propagation. A seemingly similar problem is the classic influence maximization problem, the goal of which is to find as small a set of nodes as possible to initiate certain adoption (e.g., products, innovation, etc.) such that it will trigger the maximum cascade, which has been the focus of many influential research works including Kleinberg's [6].

However, it is important to note the critical difference between our problem and the influence maximization problem. Our label propagation problem has an extra dimension of complexity as a result of the uncertainty of the labels assigned to the seed set. In the influence maximization problem, the labels to be assigned to the seed set are mainly for status indication which are known a priori — if a node is chosen as a seed to initiate the adoption, its label is set as "active", otherwise, its label remains as "inactive". The challenge is to find the right set to assign the initial "active" labels to maximize the cascade. On the other hand, in our label propagation problem, labels represent categorical attributes the values of which remain undecided until specified by users, i.e., for each node in the seed set, technically, users can specify any chosen label from the label universe. The challenge in identifying the right set is not only to study the network structure but to consider all possibilities of label assignment as well.

This important difference between the two problems also suggests that, in our problem setting, a dynamic model of seed set computation based on step-wise user input could be more suitable. Instead of computing the seed set all at once, we in fact should compute the seed set one node at a time based on user input for the next label. As shown in Example 1, different label revealed at each step could lead to drastically different propagation result.

Example 1. In Figure 1, suppose in Step 1 the network is initialized with only node 4 labeled as "A" and the propagation method is the majority voting algorithm such that each node gets the label of the majority of its neighbors. Depending on which node and its label is known in Step 2, we would get entirely different final propagation result. If in Step 2 we know node 6 with label "B", the propagation can not proceed, and if node 5 is further known with the same label "B", the result will be as shown in the right-upper network. On the other hand, if in Step 2 node 1 is revealed with label "A", the network will be fully propagated with label "A". Yet, if in Step 2 node 1 is labeled as "B" instead, then more nodes' labels need to be known in order to continue the propagation.

Therefore, in this paper, we propose the dynamic label propagation problem, which is to find, incrementally based on user input, the optimal seed set to

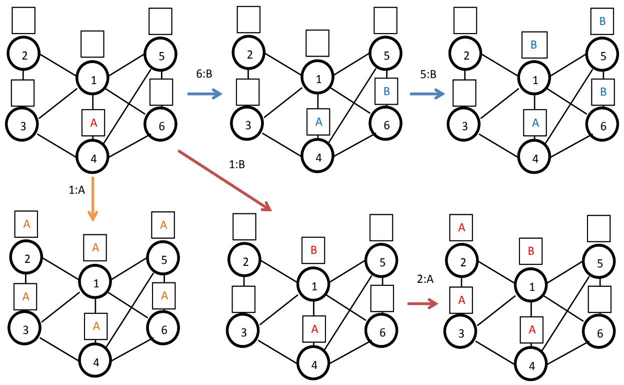


Fig. 1. Examples of different propagation results by dynamic label input

propagate the entire network. A closely related problem is to find the optimal k-size seed set where k is a user-specified parameter, e.g., budget constraint. We show that both problems are NP-hard, and present a greedy algorithm which gives good empirical results. We propose four evaluation criteria and compared different propagation models. To explore the connection between the actual label distribution and the network structure properties, we show the propagation results for some widely-used network measures including density, modularity and single node number. Our empirical results on a real-world data set demonstrate the effectiveness and efficiency of our proposed greedy algorithm.

The rest of the paper is organized as follows. We first introduce some popular propagation algorithms and the relation to our algorithm of finding the optimal given label set in Section 2. And in Section 3 we provide the details of our algorithms. Some network structure analysis that will affect the selection and propagation are shown in Section 4. And we evaluate the algorithms in Section 5. The related work is introduced in Section 6 and finally our work is concluded in Section 7.

2 Problem Formulation

2.1 Problem Definition

We denote a labeled network as $G = (V, E, L)$, where V, E and L represent the non-empty sets of nodes, edges, labels respectively. Given a labeled network $G = (V, E, L)$, there exists an Oracle function $\mathcal{O}_G : V \to L$ such that given a query of any node $v \in V$, $\mathcal{O}_G(v) \in L$, which simulates user input on the node labels. We assume initially no labels are know for any node of G, and each node could obtain a label of L during the label propagation, which could get updated during the process. However, we also assume that labels obtained from the Oracle will never change.

We begin by defining the notion of a *propagation scheme* as follows. The idea is that, given a set of nodes whose labels are known initially, a propagation scheme defines the set of the nodes each of which would obtain a label by the end of the label propagation process. The propagation scheme is defined as a function to achieve the greatest generality since the exact choice of the propagation algorithm would depend on the nature of the application. We leave the detailed discussion of the propagation scheme to subsequent parts of the paper.

Definition 1. ***[Propagation Scheme]*** *Given a labeled network* $G = (V, E, L)$*, an Oracle function* $\mathcal{O}_G$ *and a* $S \subseteq V$ *such that for each* $v \in S$*,* $\mathcal{O}_G(v)$ *is known, a* propagation scheme *is a function* $P : 2^V \to 2^V$ *such that* $P(S) \subseteq V$ *and for each* $v \in P(S)$*,* v *would obtain a label by the end of the label propagation process.*

The question of the greatest interest to users is the *Minimum Label Cover (MLC)* problem which is to find the smallest node set to obtain labels initially such that the subsequent propagation could cover the whole network, i.e., assign labels for every node. A closely related problem is the *K-set Label Cover (KLC)* problem in which we are interested in how much of the network we can at most cover if we know the labels of K nodes, which is useful for applications in which a budget is given to acquire the initial labels. These two problems are related in that a solution to the KLC problem would also give a solution to the MLC problem. Notice that in both problem settings, the Oracle to reveal the node labels is not available to the algorithm to find the seed set. In contrast, in our dynamic problem definitions later, the Oracle is available at each step to answer label queries.

Definition 2. ***[Minimum Label Cover (MLC)]*** *Given a labeled network* $G = (V, E, L)$ *and a propagation scheme* $P(.)$*, the* Minimum Label Cover *problem is to find a node set* $\mathcal{S}$ *of minimum cardinality, such that the label propagation as defined by* $P(.)$ *would cover the entire network, i.e.,* $\mathcal{S} = \underset{|S|}{\operatorname{argmin}}\{S|P(S) = V\}$*.*

Definition 3. ***[K-set Label Cover (KLC)]*** *Given a labeled network* $G = (V, E, L)$*, a propagation scheme* $P(.)$ *and a positive integer* K*, the* K-set Label Cover *problem is to find a node set* $\mathcal{S}^K$ *of cardinality* K *such that the label propagation as defined by* $P(.)$ *would achieve the maximum coverage of the network, i.e.,* $\mathcal{S}^K = \underset{|P(\mathcal{S})|}{\operatorname{argmax}}\{\mathcal{S}||\mathcal{S}| = K\}$*.*

We are now ready to define our dynamic label propagation problem, which essentially is to solve MLC and KLC incrementally given user input at each step.

Definition 4. ***[Dynamic Minimum Label Cover (DMLC)]*** *Given a labeled network* $G = (V, E, L)$ *, an Oracle function* $\mathcal{O}_G$ *and a propagation scheme* $P(.)$*, the* Dynamic Minimum Label Cover *problem is to find a node sequence of minimum length,* $\mathcal{S} = (v_1, v_2, \ldots, v_{|\mathcal{S}|})$*, such that the label propagation as defined by* $P(.)$ *would cover the entire network, i.e.,* $\mathcal{S} = \operatorname{argmin}_{|S|}\{S|P(S) = V\}$*.*

Definition 5. *[**Dynamic K-set Label Cover (DKLC)**] Given a labeled network $G = (V, E, L)$, an Oracle function $\mathcal{O}_G$, a propagation scheme $P(.)$ and a positive integer K, the* Dynamic K-set Label Cover *problem is to find a node sequence of length K, $\mathcal{S} = (v_1, v_2, \ldots, v_K)$, such that the label propagation as defined by $P(.)$ would achieve the maximum coverage of the network, i.e., $\mathcal{S}^K = \underset{|P(\mathcal{S})|}{\mathrm{argmax}}\{\mathcal{S}||\mathcal{S}| = K\}$.*

2.2 Complexity Analysis

In this section we give some complexity analysis of the varied problem settings, mostly based on known hardness results with some quite straightforward problem reductions. The detailed proofs are omitted due to space limit. First it is not hard to see the NP-hardness of the MLC problem as a result of the following theorem from [6].

Theorem 1. *[**6**] The influence maximaization problem is NP-hard for the Linear Threshold model.*

In our definition of the MLC problem, if we set the propagation scheme to be the function which corresponds to the Linear Threshold model as described in [6], and our label set L to be the set containing only a single label, then the status of a node whether or not it has acquired this label would map exactly to the status of being "active" or "inactive" as in the Linear Threshold model in [6]. Therefore, the influence maximization problem is indeed a sub-problem of the MLC problem. Due to Theorem 1, we have the following theorem for the MLC problem.

Theorem 2. *The MLC problem is NP-hard.*

As we can solve the MLC problem in polynomial time by systematically try a sequence of increasing values of K for the corresponding KLC problem, Theorem 2 implies that the KLC problem is also NP-hard.

Corollary 1. *The KLC problem is NP-hard.*

By similar argument, if we set our label set L to be the set containing only a single label to match exactly the status of a node being "active" or "inactive" as in the Linear Threshold model in [6], the having the Oracle available will not lend additional information as in this case the label, which is actually status, is known a priori. As such, the static versions of the problem are actually sub-problems of the dynamic versions. We therefore also have the following results by similar argument.

Theorem 3. *The DMLC problem is NP-hard.*

Corollary 2. *The DKLC problem is NP-hard.*

Since both versions of the dynamic label propagation problems are NP-hard, we resort to heuristic algorithms. In particular, we develop a greedy algorithm which will be detailed in Section 3. In [6], it has been shown that such a greedy hill-climbing algorithm would give an approximation to within a factor of $(1-1/e-\epsilon)$ for Linear Threshold model. It is worth noting that in this paper we are not limited to the Linear Threshold model, as we will discuss in the following. Unfortunately, the approximation bounds are not known for the greedy algorithm in models with other propagation methods, e.g., K-nearest neighbor algorithm, which we would like to explore in our future study.

2.3 Propagation Models

We present a discussion of some widely-used propagation models focusing on their applicability in our problem setting.

K-nearest Neighbor Algorithm. K-nearest neighbor algorithm (KNN) is a method for classification, while in label propagation, it is also widely used. The idea of KNN is that the node will be labeled as the same label as his nearest top-K nodes' labels. The distance of two nodes could be measured by different factors like SimRank [5], which measures the structural-context similarity. In this case, the selection prefers the nodes that are more similar to others.

Linear Threshold Model. Linear threshold model is widely used in information diffusion. Given a set of active node, and a threshold θ for each node, at each step, an inactive node will become active if the sum of the weights of the edges with active neighbors exceeds the threshold θ. Similar to this process, during the label propagation, a node will accept the label if the sum of the weights of the edges with neighbors by this label exceeds the threshold θ. In the linear threshold model, the selection prefers the nodes with higher degree and higher edge weights. We call it majority voting in the following sections to differentiate the propagation with information diffusion.

Independence Cascading Model. Independence cascading model is another widely used model in information diffusion and was also deeply discussed in [6] along with linear threshold model. When a node v becomes active in step t, it is given a single chance to activate each currently inactive neighbor w with a predefined probability. In addition, if v succeeds, then w will become active in step $t+1$; but whether or not v succeeds, it cannot make any further attempts to activate w in subsequent rounds. Obviously, in the label propagation scenario, node v should be able to propagate its labels out at any steps rather than only once. And therefore, this model is not suitable for label propagation.

Supervised Learning Algorithm. Supervised learning algorithms use the nodes with existing labels to train the classifier and then propagate to the unlabeled nodes, like Support Vector Machine(SVM) and Hidden Markov Model.

These algorithms need a certain number of labeled nodes as training dataset to train the model first. However, in our case, the labeling of the nodes is unknown and need to be adaptive to the user input in a quick response, and thus the supervised learning algorithms are not quite suitable.

3 Seed Node Detection Algorithm

3.1 Design Ideas

The complexity of the formation of set $\mathcal{S}$ is $O(2^n - 1)$, where n is vertex number. In addition, the selection also needs to consider the situation of nodes with different labels, which consequently will decrease the performance. So before propagation on the incomplete network, we need to employ some techniques to simplify it first. And according to the different characteristics of various networks, the techniques might be varies. Here, we introduce two approaches: pruning and clustering.

Pruning. In social network like twitter, there are some users who have many followers such as celebrities, film stars and politicians. We call these users as "Hub users". When the label stands for affiliations rather than interests, the propagation will fail due to the existence of these "Hub users". In addition, the normal users who do not know each other off-line will decrease the performance of propagating affiliations as well. And thus, we need to prune some users before propagation under different circumstances.

Besides, in social network, some nodes are isolated due to a lot of reasons such as they are puppets or new-comers. In this situation, the degree of these nodes in a certain target network is usually small. If these spam nodes are not essential in the specific scenario, then it could be pruned. Since different pruning techniques will be employed according to different label propagation scenarios, so the modification of the network will not affect the propagation result significantly.

By pruning techniques, we could not only remove noise nodes to increase precision, but also decrease the number of nodes in the network. And thus the computation time according to $O(2^n - 1)$ will be reduced.

Clustering. To reduce the complexity, another step is to divide the network into several subgraphs. However, a question is that if the network could be clustered well and then the minimum set $\mathcal{S}$ will be inferred by randomly choose a node in each cluster directly. Actually, the clustering approach of previous research works cannot achieve best results, which consequently leads the wrong labeling by choosing only one node. And therefore, the idea is that, before propagation, we just do a roughly clustering on the network. For each cluster, we select a minimum set $\mathcal{S}'$, and then union all the $\mathcal{S}'$. During the combination, the nodes those could be propagated by the others in $\bigcup \mathcal{S}'$ will be deleted. Actually, the selection after clustering might not be the optimal one compared to that on the original network. However, to deal with large networks, it works when considering time cost.

Finding Set $\mathcal{S}$. To select set $\mathcal{S}$, there are two approaches. The first one is to give the final result $\mathcal{S}$ directly in off-line mode, and the other one is to add the node to $\mathcal{S}$ online. The main difference between these two selection processes is that the second one is more dynamic. In the off-line modes, as long as the prediction of one node's label is different from the actual one, the selection according to the propagation result might be changed, as shown in Example 1. And thus, it needs to pre-compute all the situations for the nodes with different labels, which is impractical for large and complex networks. On contrary to the off-line mode, the online one picks up the nodes dynamically according to the user input. Once a node's label was given by users, the selection considers the current network states. In another words, in each iteration i, the selection depends on the network structure and the labels constructed in $(i-1)$'s iteration rather than the initial network state. The details of the algorithm will be shown as follows.

3.2 Algorithm

We propose a greedy algorithm to select the set $\mathcal{S}$ dynamically, which is called *G-DS*. In each iteration, we pick up a node which maximizes the propagation coverage. The measurement of the maximization considers different labels. Suppose in the i's iteration, the existing labels in the network are the set $\mathcal{L} = l_1, l_2, ..., l_k$. And then, for each unknown vertex v, we calculate the increase coverage "*Cov*" by v labeled as l_x as

$$Cov(v, l_x) = \frac{\#known\ label\ nodes}{\#total\ nodes}, \tag{1}$$

and the probability "P" that v to be labeled as l_x according to the current network status in the $(i-1)$'s iteration is

$$P(v, l_x) = \frac{\#nodes\ labeled\ as\ l_x\ in\ v's\ neighbors}{\#v's\ neighbors}, \tag{2}$$

and finally sum up $Cov * P$ for each label to get the average coverage *AvgCov* as Equation 3. In each iteration, we pick up the vertex with the largest *AvgCov* score. The *G-DS* algorithm is shown in Algorithm 1.

$$Score = AvgCov(v) = \sum_{l_x \in L \cup l_{new}} Cov(v, l_x) * P(v, l_x). \tag{3}$$

However, the performance of the *G-DS* algorithm is low. The time complexity is $O(n^3)$ in the worst case, where n is the node number in the network. Each time to choose a node, it needs to calculate the score for each unlabeled nodes with different existing labels. In order to improve the efficiency, we propose a semi-greedy algorithm, called *GMax-DS*. *GMax-DS* algorithm is similar to the *G-DS* algorithm. However, instead of calculating the *AvgCov* score for all the situations, it only considers the most possible label for node v according to the current network states as

Algorithm 1. *G-DS Algorithm*

```
Require: $G = (V, E, L), k$
Ensure: $|\mathcal{K}| == k$ or $|\mathcal{K}| == |V|$
 1: $\mathcal{S} = \emptyset$, $\mathcal{K} = \emptyset$, $L = \emptyset$
 2: while $|\mathcal{K}| < k$ or $|\mathcal{K}| < |V|$ do
 3:     start the $i$'s iteration
 4:     for each $v \in (V - \mathcal{K})$ do
 5:         for each $l_x \in L$ do
 6:             compute $AvgCov(v)$ according to the $(i-1)$'s iteration
 7:         end for
 8:     end for
 9:     $\mathcal{S}$.add(max(AvgCov(v)))
10:     input the label $\mathcal{O}_G(v)$ for the node with the max$(AvgCov(v))$ score
11:     propagate the network by $\mathcal{S}$, update $\mathcal{K}$
12:     add l' to L if L does not contain it
13: end while
14: return $\mathcal{S}$
```

in Equation 4. For example, for vertex v, its neighbors have labels like l_1,l_2 and l_3, among which, l_1 occurs most, and thus the score is calculated as $Cov(v, l_1)$.

$$Score = Cov(v, l_{max}) = Cov(v, l_{max}) * P_{max}. \quad (4)$$

In the *GMax-DS* algorithm, we replace the score in Algorithm 1 line 6 with the one in Equation 4. Since we just consider the label with the highest probability during the calculation, the computation cost will be significantly decreased. And the time complexity of *GMax-DS* in the worst case is $O(n^2)$.

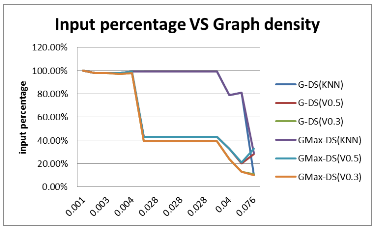
4 Social Network Structure

In real cases, the label distribution is related to the network structures. And therefore, the network structure will also affect the performance of our algorithm. So in this section, we present some structure features in social networks which might influence the performance of the selection. Actually, the performance is related to the propagation method as well. And thus, all the comparisons are based on the same propagation method. We randomly extracted 15 networks from film-actor table in FreeBase[1], and compared the performance based on two simple propagation methods, KNN and majority voting algorithms. The results based on different network attributes are shown in Figure 2.

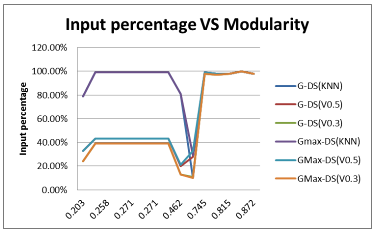
4.1 Graph Density

If the graph is less dense, then it indicates that the nodes are not well connected to others. Usually, the nodes with the same labels are more coherent. The propagation methods propagate the labels to a node from its neighbors or similar

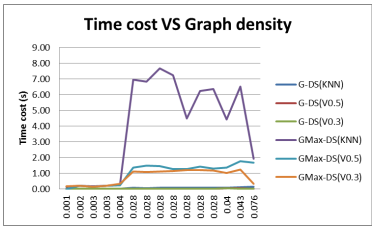
[1] http://www.freebase.com/view/film/actor



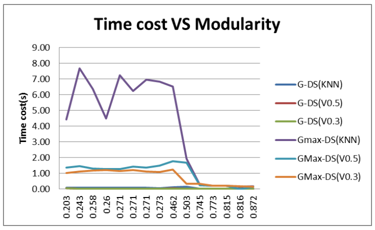
(a) Input (%) on density



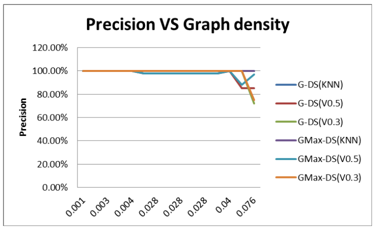
(b) Input (%) on modularity



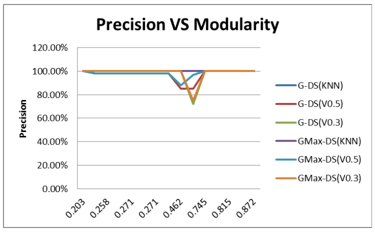
(c) Time cost on density



(d) Time cost on modularity



(e) Precision on density



(f) Precision on modularity

Fig. 2. Performance according to the *G-DS* and *GMax-DS* algorithms by KNN and Majority voting propagation methods under different network structures (The different methods are shown in different colors. Note that some networks are with similar properties and thus their points meet on the graph.)

nodes. However, in a sparse network, it is hard to propagate the labels. As in Figure 2.(a), (c) and (e) shows, the performance will arise linearly when the density increases.

4.2 Modularity

Modularity[10] measures the strength of a division of a network into modules. Networks with high modularity have dense connections between the nodes within the modules but sparse connections between nodes in different modules. So according to the definition of modularity, a network with higher modularity requires less input for labels. In addition, it is much easier for labels to be propagated within the modules rather than across the modules, and therefore increases the precision. The results are shown in Figure 2.(b), (d) and (f), which indicates that the performance will also arise when modularity increases. (The modularity score is calculated based on [2].)

4.3 Single Node Number

Actually, according to the analysis of graph density and modularity, the tendency of input percentage, time cost and precision shown in Figure 2 should be linearly. However, there is some exceptions. We further looked into these networks and found that these graphs include many one-degree nodes, which we also mentioned in Section 3. And here, if we just propagate these nodes from the only neighbor they connect to is unsafe. So here we will just pick up these one-degree nodes and add them to the input set, which increases the input percentage and the final precision in our result.

The reasons why we choose density and modularity as the attributes we further looked into is that: 1) the network structures they present affect the propagation performance directly, and 2) they are related to some other attributes like average degree, cluster coefficient, etc. However, there might be some other factors. And due to the limitation of the pages, we do not enumerate all the attributes here.

5 Experiment

5.1 Dataset

We utilized the film-actor table from FreeBase. In a network, the nodes indicate the actors and the edges stand for the relations that these two actors appeared in the same film. The labels for the node are the films that the actor performed within the network. We randomly extracted 500 networks from FreeBase for different actors' networks. The descriptions of the 500 networks are shown in Table 1. Furthermore, in Figure 3, we present a propagation result for a 131-vertex network from the set we select by *GMax-DS* and KNN, where the color indicates different labels. The size of the seed set is 13 and the precision is around 84.2%, which strongly illustrates that our algorithm works in real case.

Table 1. Description for the 500 networks extracted from freebase film-actor table

	Node	Edge	Density
#Minimum	2	81	0.94
#Medium	13	148	2.15
#Max	458	6937	6.21
#Average	24.50	377.51	1.84

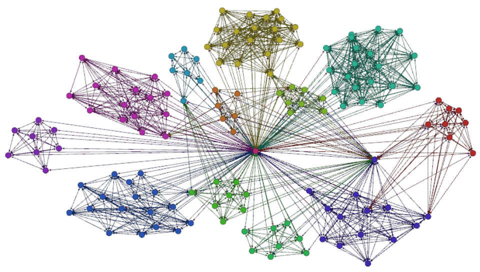


Fig. 3. Case study of the propagation result: the size of set $\mathcal{S}$ is 13, and the vertex number is 131. The precision is 84.2%.

5.2 Experiment Setup

We compared the *G-DS* and the *GMax-DS* algorithm according to different performance measurements. And based on the two scenarios we discussed in section 1, the comparison also included the two scenarios, which are the selection of the minimal set and the size-k set. The propagation algorithms we chose here are KNN and majority voting. In addition, we also compare our algorithm with the naive off-line one, which is to check all the possible seed sets and pick up the best one. "TK" stands for the selection of top-k nodes while "A" is to cover the whole network. And "GA" indicates the *G-DS* algorithm while "GM" means the *GMax-DS* algorithm. In addition, "K" and "V" indicate the propagation algorithms respectively, "K" is KNN and "V" is the majority voting with different thresholds as 0.3 and 0.5.

5.3 Experiment Result

Time Cost. Since our algorithm needs to be adaptive to the user input, so the time cost for the selection should be limited. Once the labels for a node are decided, our algorithm needs to pick up another node into $\mathcal{S}$ for input quickly. So, in our experiments, we evaluate the time cost in both scenarios. The result is shown in Figure 4, and notice that the time is normalized to log. Mostly our algorithm spent only less than 0.0001 seconds to select the data set for input. The only one extreme case is larger than 1. Since *G-DS* algorithm need to consider all the possible labels in the selection, it takes more time than *GMax-DS* algorithm when the network is with more labels.

F1-score under KMLC. Considering the scenario of selecting the size-k set for input, we evaluate the f1-score where k is equal to 3, 5 and 10. The result is given in Figure 5. When using the KNN propagation algorithm, the f1-score could be mostly beyond 90%. On contrary, by majority voting algorithms with threshold as 0.3, the f1-score is around 70% in average. In addition, comparing *G-DS* with *GMax-DS* algorithms, *GMax-DS* outperforms *G-DS* algorithm in both propagation algorithms.

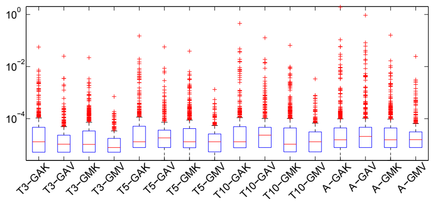


Fig. 4. Time cost (normalized to log) of the selection

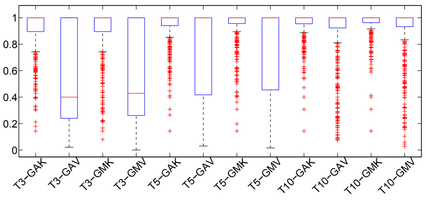


Fig. 5. F1-score of selecting top-K nodes

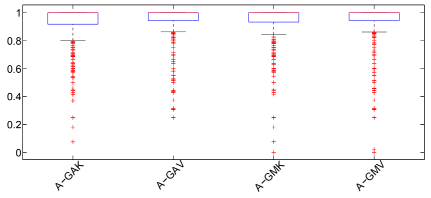


Fig. 6. Precision of covering all the nodes with at least one label

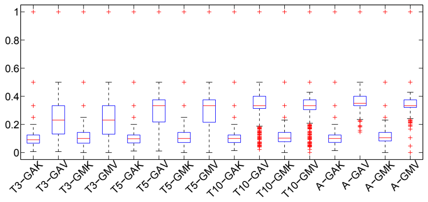


Fig. 7. Input percentage of covering all the nodes with at least one label

Precision under MLC. To pick up the minimal set, we evaluate the precision score and the results are shown in Figure 6. The median number mostly reaches 100% and the lower bound of the precision is around 80%, which indicates that our selection performs well to ensure the precision independence of the propagation algorithms.

Input Percentage under MLC. Actually, the precision is also related to the input percentage. When the input percentage is higher, then the precision will consequently be higher. So we further looked into the input percentage by different propagation methods with *G-DS* and *GMax-DS* algorithms. The evaluation result is illustrated in Figure 7. In general, the input percentage is less than 40%, and the average value for KNN and majority voting is 10% and 30% respectively. Some values are even smaller as 1 or 3. In this experiment, we could find that according to different propagation methods, the input percentage could varies, which has already been discussed in Section 2. In addition, by KNN, the input percentage is around 10% in average and the maximum value is around 25%. In most real cases, this number of input is acceptable.

Compared to the Naive Selection. Furthermore, we also compared our *GMax-DS* algorithm to the force brute selection. The results are shown in Table 2. The propagation method here is KNN. We might see from the table that the precision and the input percentage of our algorithms are mostly the same as the naive one. However, considering the time cost, different with our algorithm, the naive one will increase exponentially when the vertex number increases. On contrary, ours grow linearly and is under control.

Table 2. Performance comparison between naive algorithm and GMax-DS algorithm based on the networks with different vertex number

		5	10	15	19	20
Input percentage	Naive	20%	10%	7%	5%	10%
	GMax-DS	20%	10%	7%	5%	15%▲
Precision	Naive	100%	100%	100%	100%	100%
	GMax-DS	100%	100%	100%	100%	95%▼
Time cost(s)	Naive	0.1	8	**789**	**7882**	**18379**
	GMax-DS	0.015	0.124	0.063	0.156	0.327

From the above experiments, we might infer that the time cost of the *GMax-DS* algorithm is less than that of the *G-DS* algorithm. And in general, the time cost is limited to an acceptable value, which could be adaptive to the user input. In addition, to find the size-k set $\mathcal{S}$, even the size is quite small as 3, some network could also be propagated well and achieve higher f1-score. However, it would be better if k increases. To select the set $\mathcal{S}$ to cover the whole network, the precision could achieve higher even the input percentage is small.

6 Related Work

To our knowledge, there is no work on *dynamic* label propagation in social network. However, there is some researches in information diffusion to find the most influential user sets, which is similar to our problem to some extents. Both are propagated from neighbors. But the difference is that, in information diffusion, the status of a node is usually active or in active[9]; while in label propagation, the node might have multiple labels. In information diffusion, the status is not intrinsic like retweeting the posts [11], while for label propagation, a node's label like affiliation is intrinsic and will not changed according to different network structures. And thus, the problem in label propagation is more complicated than that in information diffusion.

In information diffusion, one of the most widely used algorithms to find the most influential nodes is the greedy algorithm. David Kempe [6] proposed a greedy algorithm to maximize the spreading of influence through social network first. He proved that the optimization problem of selecting the most influential nodes is NP-hard and provided the first provable approximation guarantees for efficient algorithm. The algorithm utilized the submodular functions to ensure finding the most influential nodes in each iteration. Later, based on Kempe's work, Yu Wang [13] proposed a new algorithm called Community-based Greedy algorithm for mining top-K influential nodes to improve the efficiency in large graphs.

In addition, there are some other attributes to measure the role of the nodes in the network, like the degree centrality, closeness centrality, betweenness centrality, eigenvector centrality, etc. [14] measured the node's importance in the network respectively in different aspects. And some papers also compared different measures. For example, Kwak et al. [7] looked into three measurements - number of followers, page-rank and number of retweets, and drew a conclusion that the finding of influential nodes differs by different measurements. As well as Kwak's work, [3] and [15] also compared different measures of influence like number of mentions, a modified page-rank accounting for topics, also found that the ranking of the influential nodes depends on the influence measures. In our problem of label propagation, the selection of the data set for input also differs by utilizing different propagation methods. And our selection algorithm should be adaptive to various propagation methods.

7 Conclusion

In this paper, we proposed the *G-DS* and *GMax-DS* algorithms to select the optimal seed set to maximize the propagation performance. Due to the label complexity, our algorithm could adjust itself dynamically according to the various user inputs. In addition, we further analyzed various network structure attributes since they are related to the label distribution and will affect the selection directly. Our empirical evaluations on real-world FreeBase data set demonstrated the effectiveness and efficiency of our algorithm in terms of input percentage, time cost, precision and f1-score.

Acknowledgments. This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

[1] Bakshy, E., Hofman, J., Mason, W., Watts, D.: Identifying influencers on twitter. In: Fourth ACM International Conference on Web Seach and Data Mining, WSDM (2011)

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

[3] Cha, M., Haddadi, H., Benevenuto, F., Gummadi, K.: Measuring user influence in twitter: The million follower fallacy. In: 4th International AAAI Conference on Weblogs and Social Media (ICWSM), vol. 14, p. 8 (2010)

[4] Gregory, S.: Finding overlapping communities in networks by label propagation. New Journal of Physics 12(10), 103018 (2010)

[5] Jeh, G., Widom, J.: Simrank: a measure of structural-context similarity. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 538–543. ACM (2002)

[6] Kempe, D., Kleinberg, J., Tardos, É.: 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)

[7] Kwak, H., Lee, C., Park, H., Moon, S.: What is twitter, a social network or a news media? In: Proceedings of the 19th International Conference on World Wide Web, pp. 591–600. ACM (2010)

[8] Lampe, C., Ellison, N., Steinfield, C.: A familiar face (book): profile elements as signals in an online social network. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 435–444. ACM (2007)

[9] Myers, S., Zhu, C., Leskovec, J.: Information diffusion and external influence in networks. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 33–41. ACM (2012)

[10] Newman, M.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences 103(23), 8577–8582 (2006)

[11] Sun, E., Rosenn, I., Marlow, C., Lento, T.: Gesundheit! modeling contagion through facebook news feed. In: Proc. of International AAAI Conference on Weblogs and Social Media, p. 22 (2009)

[12] Wang, F., Zhang, C.: Label propagation through linear neighborhoods. IEEE Transactions on Knowledge and Data Engineering 20(1), 55–67 (2008)

[13] Wang, Y., Cong, G., Song, G., Xie, K.: Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1039–1048. ACM (2010)

[14] Wasserman, S., Faust, K.: Social network analysis: Methods and applications, vol. 8. Cambridge University Press (1994)

[15] Weng, J., Lim, E., Jiang, J., He, Q.: Twitterrank: finding topic-sensitive influential twitterers. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, pp. 261–270. ACM (2010)

[16] Xie, W., Normal, E., Li, C., Zhu, F., Lim, E., Gong, X.: When a friend in twitter is a friend in life. In: Proceedings of the 4th International Conference on Web Science, pp. 493–496 (2012)

[17] Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Tech. rep., Technical Report CMU-CALD-02-107, Carnegie Mellon University (2002)