Leveraging Cross-Network Information for Graph Sparsification in Influence Maximization

Xiao Shen, Fu-lai Chung, Sitong Mao
Department of Computing, Hong Kong Polytechnic University
Kowloon, Hong Kong
xiao.shen@connect.polyu.hk, cskchung@comp.polyu.edu.hk, sitong.mao@connect.polyu.hk

ABSTRACT

When tackling large-scale influence maximization (IM) problem, one effective strategy is to employ graph sparsification as a pre-processing step, by removing a fraction of edges to make original networks become more concise and tractable for the task. In this work, a Cross-Network Graph Sparsification (CNGS) model is proposed to leverage the influence backbone knowledge pre-detected in a source network to predict and remove the edges least likely to contribute to the influence propagation in the target networks. Experimental results demonstrate that conducting graph sparsification by the proposed CNGS model can obtain a good trade-off between efficiency and effectiveness of IM, i.e., existing IM greedy algorithms can run more efficiently, while the loss of influence spread can be made as small as possible in the sparse target networks.

KEYWORDS

Influence Maximization; Graph Sparsification; Cross-network; Domain Adaptation; Feature Incompatibility; Self-training

1 INTRODUCTION

Motivated by the idea of viral marketing, the influence maximization (IM) problem has been extensively studied. Domingos *et al.* [1] were the first to study influence propagation in a social network using a probabilistic algorithm. Then, Kempe *et al.* [2] formulated IM as a discrete optimization problem, i.e., to select k seed nodes (i.e. initial users) in a given network such that the expected number of nodes influenced by the k seeds (i.e. influence spread) is as large as possible, under a certain influence cascade model. A number of promising greedy algorithms [2-5] have been proposed to tackle the NP-hard IM problem.

However, many real-world networks are with massive number of nodes and edges, hampering existing IM algorithms to work in practice. To tackle this problem, one effective approach is to employ graph sparsification as a pre-processing step for IM, by removing a fraction of edges to make the original networks more concise and tractable. Several graph sparsification algorithms have been developed for IM. For example, Wilder *et al.* [6] developed a Random Walk algorithm to preserve a subset of edges by minimizing Kullback-Leibler (KL) divergence between a random walk on the original network and the sparse network. Mathioudakis *et al.* [7] presented a SPINE algorithm to detect the "backbone" of an influence network, by preserving the edges important for influence propagation based on a log of past influence propagation traces. This SPINE algorithm, however, is impracticable for the networks containing nodes with large in-degree [7]. Zhou *et al.* [8] proposed a brute force method to prune a subset of the least important edges for weighted graphs such that the overall graph connectivity can be best maintained. Lamba *et al.* [9] proposed a model independent approach to remove the least informative edges, via aggregating multiple topological feature rankings, weighted by the Kendall Tau distances between different rankings. In addition, instead of maximizing the influence in a single network, recently Hu *et al.* [10] proposed a Transfer Influence Learning (TIL) method to study the cross-network IM problem, by viewing seed selection for IM as a node classification task. However, this TIL method just directly leveraged a classifier trained in a source network to select seed nodes for multiple homogeneous target networks with similar sizes, without considering the domain discrepancy issue.

All the existing graph sparsification algorithms [6-9] developed for IM only leveraged the information in a single network. Motivated by [10], we propose a Cross-Network Graph Sparsification (CNGS) model to leverage the cross-network information to conduct graph sparsification. Here, we consider graph sparsification as an edge prediction problem, with the goal of removing the edges least likely to contribute to influence propagation. To detect the influence backbone, the SPINE algorithm [7] requires a log of past influence propagation traces in a specific network, which are difficult to obtain in many real-world datasets. With respect to this issue, our proposed CNGS model leverages the knowledge pre-learned in a smaller source network, to detect the influence backbone for multiple heterogeneous target networks with larger sizes. Firstly, in a source network, we simulate the influence propagation traces by running an influence cascade model. Then, one can label each edge as active or inactive, where active edges indicate that they actually make contribution to the influence propagation during the simulations. We assume active edges should be with discriminative topological structures w.r.t. inactive edges, as in [9].

SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan.
 ISBN 978-1-4503-5022-8/17/08…$15.00.
DOI: http://dx.doi.org/10.1145/3077136.3080646

However, unlike [9] which learns relative weights based on the distances between different feature rankings in a single network, the CNGS model utilizes a cross-network learning approach to transfer the feature weightings learned from a source network to multiple heterogeneous target networks. To address the domain discrepancy issue, a feature incompatibility measure [11] is incorporated to impose stronger effects to the features which perform more similar for backbone detection between the source and the target networks. Also, a semi-supervised approach [11] is utilized to iteratively train the model based on not only the fully labeled data in the source network, but also the most confident prediction data in the target network. After the domain discrepancy has been reduced, we adapt the CNGS model to the target networks to predict the probability of each edge to be active for influence propagation. Then, by removing the edges least likely to be active, 1) existing IM greedy algorithms can run more efficiently in the sparse networks; and 2) the loss of influence spread in the sparse networks can be made as small as possible. To the best of our knowledge, the proposed CNGS model is the first work to study the cross-network graph sparsification problem for IM. Experiments on four real-world datasets show the capability of the CNGS model.

2 CROSS NETWORK GRAPH SPARSIFICATION (CNGS) MODEL

Firstly, we briefly introduce several criteria widely used to capture the topological structures of a node in a given network, as in the literature [9-10]. **1) Degree.** It calculates the number of edges adjacent to a node. **2) Weighted Degree.** It computes the degree by considering the weight of each edge adjacent to a node. **3) Eigenvector Centrality.** It evaluates a node's influence in the scenario of information diffusion. **4) HITS Hub**. It is estimated based on the outgoing links from a node. **5) PageRank Score**. It is computed based on the structure of incoming links to a node. **6) Clustering Coefficient**. It reflects the fraction of a node's friends who are also friends with each other. In the CNGS model, we build edge features based on the average topological feature values of two nodes on each edge, as in [9]. To make the feature values to be network independent, we normalize all the absolute values to [0, 1]. Note that it is also flexible to employ other informative features in the CNGS model as long as they can be efficiently computed in the large-scale networks.

The CNGS model is presented in Algorithm 1. Firstly, in a source network, we run an influence propagation model *l* times to simulate the influence propagation traces, induced by the *k* seed nodes selected by a greedy algorithm. Note that the CNGS framework is model independent, meaning that it can work with any existing IM greedy algorithms and any influence propagation models. After simulations, one can label each edge as active or inactive. In an undirected network, if node *u* successfully influences node *v*, or node *v* successfully influences node *u*, then edge (*u,v*) is denoted as "active". While in a directed network, edge (*u,v*) is denoted as active, if and only if node *u* successfully influences node *v* during the influence propagation simulations. Then, in the source network, with active edges labeled as "1" and inactive edges labeled as "0", a supervised learning method can be devised to train a logistic regression (LR) model with the following cost function:

$$J(\theta) = -\frac{1}{m_S}\sum_{u,v} I_{u,v}\begin{bmatrix} y_S^{(u,v)} log(h_\theta\left(x_S^{(u,v)}\right)) + \\ (1-y_S^{(u,v)}) log(1-h_\theta\left(x_S^{(u,v)}\right))\end{bmatrix} \quad (1)$$

where m_S is the number of edges in the source network *S*; $I_{u,v}$ is an indicator to represent whether node u and v are directly connected by an edge, if they are directly connected, $I_{u,v}=1$, otherwise, $I_{u,v}=0$; $x_S^{(u,v)}$ and $y_S^{(u,v)}$ represent the feature vector and true label of edge (*u,v*) in *S*, respectively; $\theta = \{\theta_j\}_{j=1}^{n}$ denotes a vector of weights of the n topological features for active edge prediction. After the θ minimizing (1) has been learned in *S*, one can firstly estimate the probability of an edge (*u,v*) to be active in the target network *T*, using the following formula:

$$\hat{y}_T^{(u,v)} = (1+e^{-\theta^T x_T^{(u,v)}})^{-1} \quad (2)$$

However, the same feature might have different level of relative importance in different networks. Thus, we further adopt a self-training for domain adaptation (SEDA) algorithm [11] to measure the incompatibility of each feature X_j between *S* and *T*, as below:

$$IC(X_j) = \left(1 - Pcc(X_{S,j}, Y_S) Pcc(X_{T,j}, \hat{Y}_T)\right) \quad (3)$$

where $Pcc(X_{S,j}, Y_S)$ denotes the Pearson correlation coefficient (PCC) between the *j*-th feature and the label of all the edges in *S*; $Pcc(X_{T,j}, \hat{Y}_T)$ indicates PCC between the *j*-th feature and the predicted probability of all the edges in *T*. The smaller the incompatibility, the more similar the feature performs between the source and the target networks. Then, a feature incompatibility based regularization term can be defined as below:

$$R_1(\theta) = \frac{1}{2}\sum_{j=1}^{n} IC(X_j)\,{\theta_j}^2 \quad (4)$$

Via minimizing $R_1(\theta)$, stronger weights will be assigned to the features with smaller incompatibility between *S* and *T*. Also, to prevent overfitting, a L_2-regularization term is defined as below:

$$R_2(\theta) = \frac{1}{2}\sum_{j=1}^{n} {\theta_j}^2 \quad (5)$$

By incorporating (4) and (5) into (1), an overall loss function can be developed as:

$$L(\theta) = J(\theta) + \frac{\lambda_1}{m_S}R_1(\theta) + \frac{\lambda_2}{m_S}R_2(\theta) \quad (6)$$

where $\lambda_1, \lambda_2 \geq 0$ are the parameters to balance the effects of the two regularization terms. Next, gradient descent algorithm can be utilized to find the θ minimizing loss function (6), as follows:

$$\frac{\partial L(\theta)}{\partial \theta_j} = \frac{1}{m_S}\sum_{u,v}\left[I_{u,v}\left(h_\theta\left(x_S^{(u,v)}\right) - y_S^{(u,v)}\right)x_{S,j}^{(u,v)} + \lambda_1 IC(X_j)\theta_j + \lambda_2\theta_j\right] \quad (7)$$

$$\theta_j = \theta_j - \alpha\frac{\partial L(\theta)}{\partial \theta_j} \quad (8)$$

where $\alpha > 0$ denotes learning rate.

So far, we have considered the feature incompatibility between *S* and *T*. However, the training data of the prediction model are merely obtained from *S*. To make the prediction model also consider the data in the target network, a self-training algorithm [11] is employed to iteratively move the top-*c* most confident prediction data (i.e. both feature vectors and predicted labels) from *T* to *S*. By this end, the parameters can be iteratively updated based on not only the fully labeled data in the source network, but also the newly added most confident prediction data in the target network. After *t* iterations, we leverage the latest learned model to predict and remove the edges least likely to be active in the target network, and consequently making the loss of influence spread in the sparse target networks as small as possible.

Algorithm 1: CNGS

Input: Source network $D_S = \{(x_S^{(u,v)}, y_S^{(u,v)})\}$ with m_S labeled edges; Target network $D_T = \{x_T^{(u,v)}\}$ with m_T unlabeled edges; Self-training iteration *t;* Top-*c* most confident prediction; Edge removal fraction *f*.

Output: Predicted probability of all the edges in D_T to be active $\{\hat{y}_T^{(u,v)}\}$.

1. $D_T' = D_T$
2. In D_S, train a model to obtain $\theta^* = argmin_\theta\ (1)$;
3. for *t* iterations, do:
   a) Based on θ^*, apply (2) on D_T to predict $\{\hat{y}_T^{(u,v)}\}$;
   b) Measure feature incompatibility via (3);
   c) Based on $\{\hat{y}_T^{(u,v)}\}$, move top-*c* most confident predicted active edges, i.e., $\left\{(x_T^{(u,v)}, 1) | \hat{y}_T^{(u,v)} \in Top_c\ highest\ \{\hat{y}_T^{(u,v)}\}\right\}$ and top-*c* most confident predicted inactive edges, i.e., $\left\{(x_T^{(u,v)}, 0) | \hat{y}_T^{(u,v)} \in Top_c\ lowest\ \{\hat{y}_T^{(u,v)}\}\right\}$ from D_T to D_S;
   d) In new D_S, train a model to obtain $\theta^* = argmin_\theta\ (6)$;
   end for
4. Based on updated θ^*, apply (2) on D_T' to predict $\{\hat{y}_T^{(u,v)}\}$, and remove a fraction f of edges with the lowest predicted probability to be active.

3 EXPERIMENTS

3.1 Datasets

Table 1: Statistics of Real-World Datasets

Dataset	Type	Nodes	Edges
NetHEPT	Undirected	15233	31398
Email-Enron	Undirected	36692	183831
Epinions	Directed	75879	508837
DBLP	Undirected	317080	1049866

The proposed CNGS model was tested on four real-world public datasets, namely, NetHEPT, Email-Enron, Epinions and DBLP. Both NetHEPT and DBLP datasets are co-authorship networks; the Email-Enron dataset is an email communication network; and the Epinions dataset is a trust social network. Table 1 gives some statistics of the datasets. To make CNGS more efficient, the smallest network, NetHEPT was employed as the source network, while the other three larger networks, i.e., Email-Enron, Epinions, DBLP were employed as the target networks.

3.2 Implementation Details

In the experiments, we set $\lambda_2 = 1$; $\lambda_1 = 10$ and divided by a factor of 1.1 after each iteration, following [11]. And we set *t=3, c=200* for self-training. It means that during the 3 self-training iterations, the top-200 most confident predicted active edges and top-200 most confident predicted inactive edges in the target networks were iteratively moved to the model training set. In the source network, the Independent Cascade (IC) model [2] (influence probability *p*=0.01 in the experiments) was employed to simulate the influence backbone, induced by 50 seed nodes selected by the NewGreedy algorithm [4]. Then, for graph sparsification in the target networks, the edges with the fractions chosen from [0.1, 0.9] were removed to extract the sparse networks. Next, both NewGreedy and IC were employed again in each sparse network to maximize the influence induced by the same number of seeds (50 in our experiments).

The performance of CNGS was compared with several graph sparsification algorithms, including 1) **Random heuristic** which randomly selects a fraction of edges to remove; 2) **RandomWalk** algorithm [6] which minimizes KL divergence between a random walk on the original network and the sparse network; and 3) **AggRanks** algorithm [9] which aggregates multiple topological feature rankings, weighted by the Kendall Tau distances between different rankings. We evaluate the performance of the algorithms based on how much influence spread will lose in the sparse networks. The less the influence spread loses, the better the performance. In addition, we are interested in how much running time could be saved in return, by utilizing the same greedy algorithm in the sparse networks.

3.3 Performance of CNGS

As shown in Figure 1, in the Email-Enron target network, if 40% of the edges were removed by CNGS, the influence spread was just reduced by 8.8%, while the running time of NewGreedy can be greatly saved by 47% in return. Similarly, in the Epinions target network (as shown in Figure 2), if 40% of the edges were removed, CNGS managed to just reduce the influence spread by 6.8% but save the running time by 39%. Moreover, in the DBLP target network (as shown in Figure 3), even if 80% of the edges were removed by CNGS, the influence spread was just reduced by 5.1%, while the running time can be greatly saved by 69%. These results demonstrate that the CNGS model indeed acts as an effective graph sparsification algorithm for IM, since it can greatly reduce the running time of greedy algorithm, while still achieve satisfactory influence spread in the sparse networks.

In addition, we can observe that CNGS outperformed (i.e. lowest influence spread loss) all the baselines in all the sparse networks. Note that both CNGS and AggRanks aim to aggregate multiple topological features to remove the least likely active edges, while the superior performance of CNGS over AggRanks reflects the benefit of leveraging cross-network information to

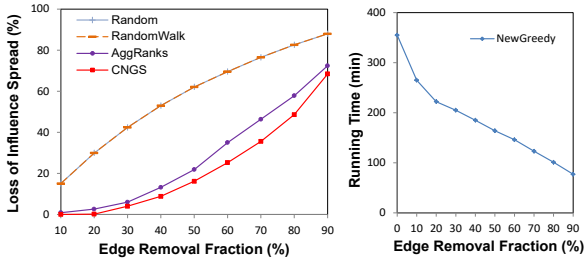

Figure 1: Performance of CNGS in Email-Enron target network.

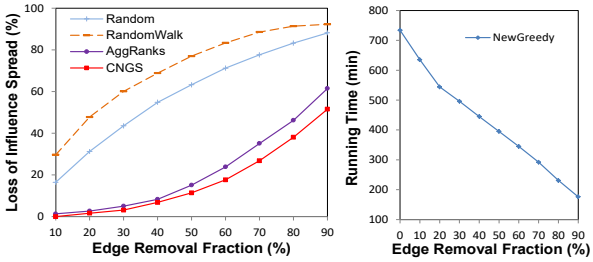

Figure 2: Performance of CNGS in Epinions target

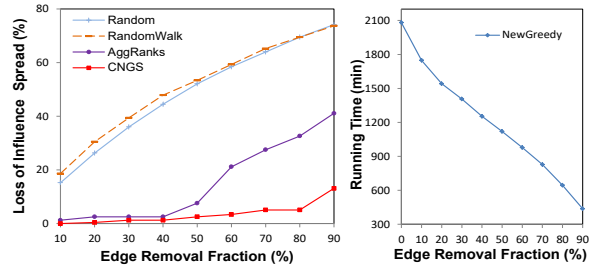

Figure 3: Performance of CNGS in DBLP target network.

learn the feature weightings for active edge prediction. AggRanks is an unsupervised method and tends to assign higher weights to more unique features in a specific network. However, the more unique features might not necessarily be more important for active edge prediction. In contrast to AggRanks, CNGS utilizes a semi-supervised approach to leverage the cross-network information to learn more useful feature weightings for active edge prediction. In addition, we can see that both CNGS and AggRanks achieved much higher influence spread than random heuristic and RandomWalk. This could be explained that for the IM task, the influence tends to be propagated through those edges adjacent to the highly influential nodes (i.e. active edges), rather than the randomly selected edges. Since the influential nodes are with discriminative topological features w.r.t. the random nodes [9]. The active edges should also be with discriminative topological structures w.r.t inactive edges. Moreover, the CNGS model is with high efficiency and scalability. For example, even in the largest target network (i.e., DBLP containing millions of edges), the time taken to measure all the topological features and train the prediction model via 3 self-training iterations was only 10 minutes, which is extremely short as compared to the save of running time of the greedy algorithm in the sparse networks.

4 CONCLUSIONS

To obtain a good trade-off between the effectiveness and efficiency of large-scale IM problem, one approach is to conduct graph sparsification as a pre-processing step to make existing IM greedy algorithms more feasible to work on the sparse networks. In this work, we proposed an innovative cross-network learning approach to study the cross-network graph sparsification problem for large-scale IM. A CNGS model was developed to leverage the influence backbone knowledge pre-learned in a smaller source network, to predict and remove the edges least likely to contribute to influence propagation in multiple heterogeneous target networks. To reduce domain discrepancy, a feature incompatibility measure and a self-training approach have been adopted. Experimental results show that conducting graph sparsification by CNGS will not cause notable loss of influence spread in the sparse networks, while obtaining the advantage of saving a lot of running time. Although in this paper, we only focus on graph sparsification for the IM scenario, however, the CNGS framework could also be transferrable to other cross-network node or link ranking/filtering tasks.

ACKNOWLEDGMENTS

This work was supported by Hong Kong PhD Fellowship Scheme (No. PF14-11856) and PolyU project (No. 152228/15E).

REFERENCES

[1] P. Domingos, and M. Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[2] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[3] J. Leskovec, A. Krause, and C. Guestrin. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[4] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

[5] A. Goyal, W. Lu, and L. V. S. Lakshmanan. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th International Conference Companion on World Wide Web*, 2011.

[6] B. Wilder, and G. Sukthankar. Sparsification of Social Networks Using Random Walks. In *Proceedings of International Conference on Social Computation (SocialCom)*, 2015.

[7] M. Mathioudakis, F. Bonchi, and C. Castillo. Sparsification of influence networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011.

[8] F. Zhou, S.Mahler, and H. Toivonen. Simplification of networks by edge pruning. In *Bisociative Knowledge Discovery*, 2012.

[9] H. Lamba, and R. Narayanam. A novel and model independent approach for efficient influence maximization in social networks. In *Proceedings of International Conference on Web Information Systems Engineering*, 2013.

[10] Q. Hu, G. Wang, and P. S. Yu. Transferring influence: Supervised learning for efficient influence maximization across networks. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, 2014.

[11] M. Chen, K. Q.Weinberger, and J.Blitzer. Co-training for domain adaptation. In *Advances in Neural Information Processing Systems*, 2011.