Cross-Network Learning With Fuzzy Labels for Seed Selection and Graph Sparsification in Influence Maximization

Xiao Shen, Sitong Mao, and Fu-lai Chung

***Abstract*—To maximize the influence across multiple heterogeneous networks, we propose an innovative cross-network learning model to study the influence maximization problem from two perspectives, namely, seed selection and graph sparsification. On one hand, we consider seed selection as a cross-network node prediction task, by leveraging the greedy seed selection knowledge prelearned in a smaller source network, to heuristically select the nodes most likely to act as seed for the target networks. On the other hand, we consider graph sparsification as a cross-network edge prediction problem, by adapting the influence propagation knowledge previously acquired in the source network to remove the edges least likely to contribute to influence propagation in the target networks. To address domain discrepancy, a fuzzy self-learning algorithm is proposed to iteratively train the prediction model by leveraging not only the fully labeled data in the source network, but also the most confident predicted instances with their predicted fuzzy labels in the target network. With such fuzzy labels, we can differentiate the confident levels of predictions generated by different self-training iterations, thus lowering the negative effects caused by less confident predictions. The performance of the proposed model is benchmarked with the popular influence maximization algorithms for seed selection; and also competed with several graph sparsification algorithms for inactive edge prediction. Experimental results on the real-world datasets show that the proposed cross-network learning model can achieve a good tradeoff between the efficiency and effectiveness of the influence maximization task in the target networks.**

***Index Terms*—Fuzzy domain adaptation, graph sparsification, influence maximization, negative transfer, self-training.**

I. Introduction

NOWADAYS, people tend to trust the information from their friends, relatives, and families more than that from general advertising media [1]. Thus, one promising marketing strategy for product promotion is to select a few most influential initial users to give them free samples and let them influence their friends through the word-of-mouth effect. Such an approach is referred to viral marketing [2]. Motivated by the idea of viral marketing, the influence maximization (IM) problem can be formulated as a discrete optimization problem [3], 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. Existing IM algorithms can be grouped into two families, namely, greedy algorithms and heuristic algorithms. Generally, greedy algorithms [3]–[6] are highly effective (i.e., achieving large influence spread) but with low efficiency (i.e., long running time). In contrast, heuristic algorithms [6]–[9] are highly efficient but generally fail to guarantee large influence spread.

Although the IM problem has been extensively studied, very little work addresses this problem in a cross-network scenario. Suppose that a company intends to promote a new product through multiple media (e.g., online social networks, email communication networks, telephone communication networks) by viral marketing. Then, if the company greedily selects initial users for each target network independently, it should be extremely time consuming. On the other hand, if the company heuristically selects initial users, it might be difficult to guarantee a high influence spread in a new target network, where the company lacks enough preliminary knowledge about which types of users should be most influential. To address such across-network IM problem more effectively and efficiently, we propose a cross-network learning (CNL) model to the leverage the knowledge prelearned in a smaller source network, where the IM task has been successfully completed, to help maximize the influence in the new larger target networks. Specifically, we address the cross-network IM problem from two perspectives, i.e., seed selection and graph sparsification.

On one hand, we consider seed selection for IM across multiple networks as a cross-network node prediction task. Here, the goal is to select the nodes most likely to act as seeds for each target network, by leveraging the greedy seed selection knowledge prelearned in a source network. Fig. 1 illustrates the main idea of the cross-network seed selection problem in IM. First, in a smaller source network G_S, one can run a standard IM greedy algorithm to select a set of k seed nodes. Then, based on the topological structures and labels of all the nodes in G_S, we can train a node prediction model P^N to learn what characterized nodes would be selected as seeds by the greedy algorithm. Then, the prediction model P^N is iteratively adapted to the larger

Manuscript received September 25, 2017; revised March 28, 2018, January 7, 2019, and June 28, 2019; accepted July 12, 2019. Date of publication July 25, 2019; date of current version September 1, 2020. This work was supported in part by Hong Kong Ph.D. Fellowship Scheme under Grant PF14-11856 and in part by UGC GRF Projects (PolyU 152228/15E and PolyU 152039/14E). *(Corresponding author: Fu-lai Chung.)*

The authors are with the Department of Computing, Hong Kong Polytechnic University, Hong Kong (e-mail: xiao.shen@connect.polyu.hk; sitong.mao@connect.polyu.hk; cskchung@comp.polyu.edu.hk).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2019.2931272

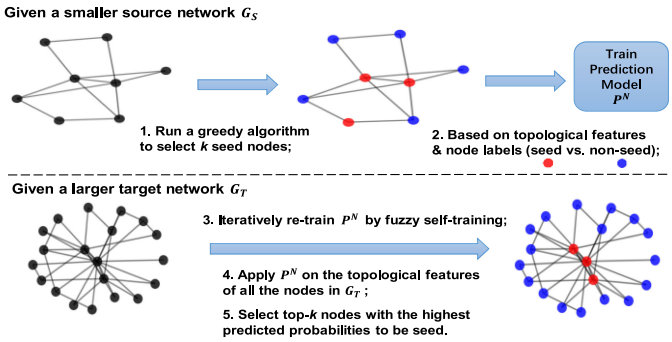


Fig. 1. Illustration of cross-network seed selection in IM. The red nodes indicate seed nodes, while blue nodes represent nonseed nodes.

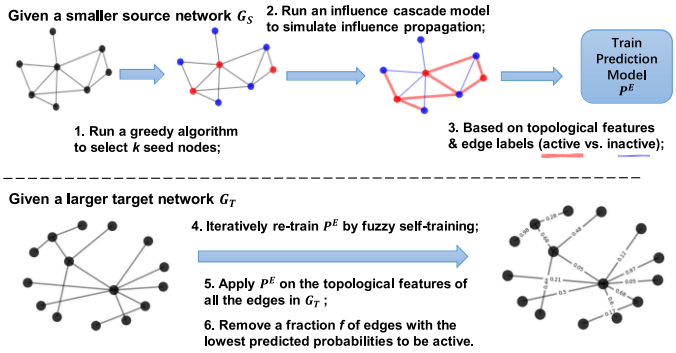


Fig. 2. Illustration of cross-network graph sparsification in IM. The red nodes indicate seed nodes, while blue nodes represent nonseed nodes. The red lines indicate active edges during influence propagation simulations, while blue lines represent inactive edges.

target network G_T to heuristically select the nodes most likely to act as seeds for IM. Since the proposed CNL model learns the knowledge from a highly effective greedy algorithm in the source network, it is more reliable to achieve a high influence spread in the target network, as compared to the conventional IM heuristic algorithms. In addition, since CNL heuristically selects seed nodes, it runs much faster than the greedy algorithms in the target network. Thus, the CNL model can achieve a good tradeoff between efficiency and effectiveness for seed selection in IM.

On the other hand, to tackle large-scale IM problem, some studies proposed to employ graph sparsification as a preprocessing step for IM, by removing a fraction of edges to make the original network become more concise and tractable. Previous IM-based graph sparsification algorithms only leverage the information in a single network, either using an unsupervised approach [10], [11] or requiring a log of past influence propagation traces in the given network [12]. Recently, we proposed a cross-network graph sparsification (CNGS) model [13], which is the first work to leverage the cross-network information to conduct graph sparsification for IM. We consider graph sparsification as a cross-network edge prediction task. The goal here is to remove the edges least likely to contribute to influence propagation in the target network, by leveraging the influence propagation knowledge prelearned in the source network. As illustrated in Fig. 2, to address the cross-network graph sparsification problem, we first simulate the influence propagation traces in a smaller source network G_S, by running an influence cascade model. After that, we can label all the edges in G_S as either active or inactive, where active edges indicate that the influence has actually been propagated through them to maximize the influence during the simulations. Then, based on the topological features and labels of all the edges in G_S, we can train an edge prediction model P^E to learn what characterized edges would be helpful for influence propagation in IM. While those unsupervised graph sparsification algorithms [10], [11] without any labeled data would fail to do so. Next, we iteratively adapt the prediction model P^E to a larger target network G_T to predict the probability of each edge to be active for influence propagation. By removing the edges least likely to be active, we can make 1) existing IM greedy algorithms run more efficiently; and 2) the loss of influence spread of the greedy algorithm as small as possible, in the sparse target networks.

The aforementioned cross-network seed selection and graph sparsification problem can be regarded as a domain adaptation task, which aims to transfer the knowledge prelearned from a source domain to assist in solving the same task in a target domain, in the condition that the source and the target domains share an identical feature space but have different data distributions [14], [15]. In our previous CNGS model [13], a self-training for domain adaptation (SEDA) algorithm [16] was directly employed to address domain discrepancy between different networks. SEDA iteratively leverages not only the fully labeled data in the source domain, but also the most confident predicted instances in the target domain with their pseudolabels for training. SEDA is a simple and efficient domain adaptation approach, however, it can easily cause negative transfer. This is because the predictions generated by different self-training iterations are with different levels of confidence. Directly utilizing binary pseudolabels to retrain the prediction model might cause negative effect on the prediction performance, especially when the target network predictions become not confident enough. How to reduce such negative effect for domain adaptation is challenging. Recently, the fuzzy techniques which can well capture the imprecise, uncertain, and vague information during knowledge transfer have been successfully employed by the domain adaptation algorithms to alleviate negative transfer [15], [17]–[22]. Motivated by this, in this article, we propose a CNL model to leverage a fuzzy self-training algorithm for domain adaption, which assigns fuzzy labels to the most confident predicted target network instances, when they are iteratively added to retrain the prediction model. With the provision of fuzzy labels, we can easily differentiate the confidence levels of the predictions generated by different self-training iterations during retraining, thus, the negative effects caused by such not confident enough predictions can be alleviated. The main improvement of the CNL model over our previous CNGS model [13] is the incorporation of fuzzy labels into the iterative SEDA algorithm to alleviate negative transfer. In addition, CNGS only addresses the cross-network graph sparsification problem, while the CNL model has been employed to address both cross-network seed selection and cross-network graph sparsification problems.

The contributions of this work can be summarized as follows.

1) We propose a CNL model to study two issues of cross-network IM problem, namely seed selection and graph sparsification, by viewing them as a cross-network node prediction and edge prediction task, respectively.
2) For seed selection, CNL leverages the greedy seed selection knowledge prelearned in a smaller source network, to heuristically select top-k nodes most likely to act as seeds for IM in multiple heterogeneous target networks.
3) For graph sparsification, CNL leverages the influence propagation knowledge previously acquired in a smaller source network to remove a fraction of edges least useful for influence propagation in multiple heterogeneous target networks.
4) To address domain discrepancy, a fuzzy self-training approach is proposed to iteratively adapt the prediction model to the target network, by utilizing fuzzy labels to handle prediction uncertainty so as to alleviate negative transfer during iterative retraining.
5) Extensive experiments on the real-world public datasets demonstrate that CNL can achieve a good tradeoff between efficiency and effectiveness of IM.

The rest of this article is organized as follows. Section II introduces the related work about IM algorithms and IM-based graph sparsification approaches. Section III formulates the cross-network seed selection and cross-network graph sparsification problem, respectively. Section IV presents the proposed CNL model empowered by fuzzy labeling. Section V discusses the experimental results on the real-world datasets. Section VI concludes this article.

II. Related Work

In this section, we review the standard IM algorithms and the graph sparsification algorithms developed for IM.

A. Influence Maximization

Domingos and Richardson [23] are the first to study influence propagation in a social network using a probabilistic algorithm. Then, Kempe *et al.* [3] proposed two pioneering influence cascade models, namely independent cascade (IC) model and linear threshold (LT) model [3]. In an influence cascade model, each node is associated with a status at a certain time, either active or inactive. In IC model, each edge is associated with an influence probability p, which is set to be a constant. Then, an inactive node becomes active if it is successfully influenced by any of its active neighbors independently. While in LT model, an inactive node would become active if the sum of the influence probabilities from all of its active neighbors exceeds a given threshold. Given a certain influence cascade model, the influence is propagated from seed nodes to all the other nodes in the network. The goal of the IM problem is to maximize the influence spread in the given network with the constraint that the seed set size is fixed to be a small value, i.e., k.

Existing algorithms to address the IM problem can be grouped into two families, namely greedy and heuristic methods. Originally, Kempe *et al.* [3] proposed a greedy hill-climbing approach to iteratively add a new node to the seed set which provides the largest marginal gain on the influence spread. This greedy algorithm can guarantee $(1 - 1/e)$ approximation of the optimal solution but requires a rather high computation cost. To improve the efficiency of the general greedy algorithm [3], the CELF [4] and CELF++ [5] greedy algorithms have been proposed to reduce the number of evaluations on the influence spread estimation, by exploiting the submodularity property of the influence spread function. In addition, Chen *et al.* [6] proposed a NewGreedyIC algorithm to employ a breadth first search (BFS) on a deterministic graph converted by the influence probabilistic graph so as to calculate the influence spread. Also, to efficiently estimate the influence spread, a pruned BFS method [24] and a static greedy algorithm [25] are proposed to reduce the number of Monte-Carlo simulations. Also, Wang *et al.* [26], [27] developed a community-based greedy algorithm to select the most influential nodes within each community rather than in the whole network. Instead of utilizing Monte-Carlo simulations to estimate the influence spread, some recent work [28]–[31] proposed to utilize the reverse influence sampling (RIS) method to select seed nodes based on reverse reachability tests. The idea of RIS [28] is to randomly sample a collection of reserve reachable (RR) sets from the given network and then select the set of nodes which can cover the maximum number of RR sets as the seeds. Although such sampling can guarantee up to $(1 - 1/e - \varepsilon)$ approximation, they incur a high memory overhead in practice [32]. Besides greedy algorithms, some heuristic algorithms are also developed to tackle the IM problem. Degree and centrality-based heuristics are the common metrics to estimate the influence of nodes in the network. To improve the pure degree heuristic, Chen *et al.* [6] proposed a DegreeDiscountIC algorithm to discount the degree contributed by the nodes already in the seed set. Luo *et al.* [9] proposed a PageRank-based heuristic which greedily selects seed nodes only from the nodes with high PageRank scores. Chen *et al.* [7] developed a PMIA algorithm to approximate the influence spread based on the maximum influence paths. Then, an IRIE algorithm [8] and an IPA algorithm [33] are proposed to reduce the high memory overhead incurred by PMIA. Besides, Tang *et al.* [32] designed a hop-based influence estimation algorithm to compute the influence spread up to two hops, the idea is similar to the time-constrained IM problem [34], [35].

Although the IM problem has been extensively studied in the literature, very little work focuses on the cross-network IM problem. In [36] and [37], the influence propagation across multiple aligned social networks has been studied, but some common users must be shared by different networks. Besides, Hu *et al.* [38] studies the cross-network IM problem in a more generalized scenario where the source and target networks do not need to share any common users. A transfer influence learning (TIL) method was proposed to transfer the influence across multiple networks, by viewing seed selection for IM as a node classification task. However, this TIL method [38] lacks the consideration of domain discrepancy, thus it is limited to be applied in multiple homogeneous subnetworks which are with similar sizes and extracted from the same original large network. In contrast, our proposed CNL model iteratively addresses the

domain discrepancy between the source and the target networks by a fuzzy self-training approach. Thus, the CNL model can be generalized to select seed nodes across multiple independent heterogeneous networks with varied sizes.

B. Graph Sparsification

Many real-world networks are with massive number of nodes and edges, hampering some promising IM greedy algorithms to work in practice. To tackle the large-scale IM problem, one can construct a more succinct representation of the original network by retaining fewer nodes or edges. Graph sparsification is one technique to construct a sparse network by retaining all the nodes in the original network, while removing a fraction of edges. Several graph sparsification algorithms have been developed as a preprocessing step for IM. For example, Wilder and Sukthankar [10] developed a Random Walk algorithm to preserve a subset of edges by minimizing the Kullback–Leibler divergence [39] between a random walk on the original network and the sparse network. Mathioudakis *et al.* [12] proposed a SPINE algorithm to detect the "backbone" of an influence network, by preserving the edges most important for influence propagation. However, the SPINE algorithm requires the input of not only the topology structure, but also a log of past influence propagation traces in the given network, which are impracticable to obtain in most real-world applications. Purohit *et al.* [40] proposed a COARSENET algorithm to merge a fraction of adjacent node pairs, by minimizing the difference of the first eigenvalue of the adjacency matrix between the original network and the coarsened network. Zhou *et al.* [41] proposed a brute force method to prune a subset of least important edges such that the overall graph connectivity can be best maintained. In addition, Lamba and Narayanam [11] proposed a model independent approach to remove the least informative edges, according to an overall ranking weighted by several topological features. To aggregate multiple feature rankings, they measured the Kendall Tau distances [42] between different rankings, and then assigned higher weight to more unique ranking.

Previous IM-based graph sparsification algorithms [10], [11], [40], [41] only leverage the unlabeled information in a single network. Thus, they might fail to know what characterized edges should be helpful for influence propagation in IM. Recently, we proposed a CNGS model [13] to leverage the influence propagation knowledge (i.e., edge prediction model) previously learned in the source network to carry out graph sparsification for multiple heterogeneous target networks. To address domain discrepancy, we adopted a SEDA algorithm [16] to iteratively retrain the prediction model, by leveraging not only the fully labeled edges in the source network but also the most confident predicted edges in the target network with their pseudobinary labels. However, binary labels fail to differentiate the degree of the membership to be active, for those most confident target network predictions generated by different self-training iterations. To improve this, in the CNL model, we propose a fuzzy self-training algorithm to assign fuzzy labels instead of binary labels to those most confident predicted instances in the target network when they are iteratively utilized to retrain the prediction model.

III. Problem Formulation

In order to achieve a good tradeoff between efficiency and effectiveness of IM, we propose to leverage the cross-network information to address the seed selection and graph sparsification tasks in IM. Let $G_S = (V_S, E_S)$ be a smaller source network with a set of nodes V_S and a set of edges E_S and $G_T = (V_T, E_T)$ be a larger target network with a set of nodes V_T and a set of edges E_T. Next, we formulate the cross-network seed selection and graph sparsification problem, respectively.

A. Cross-Network Seed Selection in IM

We consider cross-network seed selection as a cross-network node prediction task, with the goal of selecting the nodes most likely to act as seed in G_T by leveraging the greedy seed selection knowledge prelearned in G_S.

First, in G_S, we run an IM greedy algorithm to select k seed nodes multiple times. Then, a node $v_S^i \in V_S$ is labeled as seed if it is selected by the greedy algorithm at least one time; otherwise, v_S^i is labeled as nonseed. Next, based on $D_S = \{(x_S^i, y_S^i)\}_{i=1}^{|V_S|}$, where x_S^i and y_S^i represent the set of topological features and the label of node v_S^i, we can train a seed node prediction model P^N.

In G_T, we compute the same set of topological features (as in G_S) for all the nodes, i.e., $D_T = \{x_T^i\}_{i=1}^{|V_T|}$. Then, we iteratively retrain P^N by the proposed CNL algorithm and apply P^N on $\{x_T^i\}_{i=1}^{|V_T|}$ to predict $\{\hat{y}_T^i\}_{i=1}^{|V_T|}$, where $\hat{y}_T^i$ denotes the predicted probability of v_T^i to be labeled as seed. Finally, we rank $\{\hat{y}_T^i\}_{i=1}^{|V_T|}$ and heuristically select the top-k nodes with the highest predicted probabilities as seeds, denoted as $\hat{A}$. In addition, we run the same greedy algorithm (as in G_S) to select a set of k seed nodes in G_T, denoted as A, which is treated as the ground-truth seed selection result. The aim of cross-network seed selection is to make $\hat{A}$ as similar as possible to A such that the influence spread achieved by $\hat{A}$ can be approximately maximized in G_T.

B. Cross-Network Graph Sparsification in IM

We consider cross-network graph sparsification as a cross-network edge prediction task. Here, the goal is to remove the edges least likely to contribute to influence propagation in G_T, by leveraging the influence propagation knowledge prelearned in G_S.

In G_S, we first run an IM greedy algorithm to select k seed nodes. Then, we run an influence cascade model multiple times to simulate the influence propagation process induced by the k seed nodes. In an influence cascade model [3], [43]–[46] the influence is first propagated from the seed nodes to their inactive neighbors. Then, if a neighbor has been successfully influenced to become active, it can further influence its inactive neighbors with a specific probability. After simulations, we can label all the edges in G_S as either active or inactive as follows.

In an undirected network, an edge e^{ij} is labeled as active, if during at least one time of influence propagation simulation, node v^i successfully influences node v^j or node v^j successfully influences node v^i; otherwise, e^{ij} is labeled as inactive. In a directed network, an edge e^{ij} is denoted as active, iff node v^i

successfully influences node v^j, during at least one time of influence propagation simulation; otherwise, e^{ij} is labeled as inactive.

Next, based on $D_S = \{(x_S^{ij}, y_S^{ij})\}_{ij=1}^{|E_S|}$, where x_S^{ij} and y_S^{ij} represent the set of topological features and the label of edge e_S^{ij}, we can train an active edge prediction model P^E. In G_T, we compute the same set of topological features for all the edges, i.e., $D_T = \{x_T^{ij}\}_{ij=1}^{|E_T|}$. Then, we iteratively retrain P^E by the CNL algorithm and then apply P^E on $\{x_T^{ij}\}_{ij=1}^{|E_T|}$ to predict $\{\hat{y}_T^{ij}\}_{ij=1}^{|E_T|}$, where $\hat{y}_T^{ij}$ denotes the predicted probability of e_T^{ij} to be labeled as active. Next, we rank $\{\hat{y}_T^{ij}\}_{ij=1}^{|E_T|}$ and remove a fraction f of the edges with the least predicted probabilities to be active for influence propagation, denoted as $\hat{I}$. In addition, we define the ground-truth labels for all the edges in G_T, via the same approach as in G_S, and denote the set of ground-truth inactive edges in G_T as I. The goal of cross-network graph sparsification in IM is to make all the edges in $\hat{I}$ are indeed inactive, i.e., belonging to I. Thus, we would only remove the edges inactive for influence propagation in the target network, which makes the loss of influence spread as small as possible in the sparse target network.

It is worth noting that in the proposed CNL model, it is flexible to learn the greedy seed selection and influence propagation knowledge from any IM greedy algorithms and any influence cascade models in G_S. But the greedy algorithm and influence cascade model employed to define the ground-truth labels in G_T should be the same as in G_S.

IV. Proposed Algorithm

In this section, we briefly introduce several topological features adopted in the prediction model and then present the detailed framework of CNL model.

A. Topological Features

As shown in the literature [11], [38], [47], the following topological features can reflect the influence of a node in the network.

1) Degree. It calculates the number of edges adjacent to a node.
2) Weighted Degree. Different from degree, it calculates the number of edges adjacent to a node by taking the weight of each edge into consideration.
3) Eigenvector Centrality. It evaluates a node's influence in the scenario of information diffusion. A node with high eigenvector centrality indicates it is highly influential to spread the influence in the network [48].
4) HITS Hub. HITS algorithm [49] computes two values for each node, namely HITS authority and HITS hub. The authority and hub values of a node are estimated based on the incoming links and the outgoing links from the node, respectively.
5) PageRank Score. PageRank algorithm [50] was originally designed to rank page authority. By viewing each node as a page, it can be used to compute the ranking of nodes based on the structure of the incoming links to the nodes.
6) Clustering Coefficient. It reflects the fraction of a node's friends who are also friends with each other. A node with high clustering coefficient indicates that the node's neighbors are likely to be connected with each other.

On one hand, for seed node prediction, we employ the aforementioned topological features as node features. On the other hand, for active edge prediction, we assume that the likelihood of an edge to be active for influence propagation depends on the influence of two nodes connected by the edge. Thus, we construct edge features based on the average topological feature values of the two nodes on each edge. To make the feature values network independent, we ranked all the absolute values of each feature ascendingly and mapped them into [0, 1]. All these selected features can be measured efficiently by NetworkX[1], thus making the proposed CNL model more efficient for the IM task, as compared to the standard greedy algorithms. In addition, it is flexible to employ other informative topological features in the CNL model as long as they can effectively reflect a node's influence and also be efficiently computed in large-scale networks.

B. Cross-Network Learning (CNL) Model

We apply the CNL model to address both cross-network seed selection and cross-network graph sparsification problems. For seed selection, we treat seed nodes as positive class and non-seed nodes as negative class. For graph sparsification, we treat active edges as positive class while inactive edges as negative class. Then, with positive instances labeled as "1" and negative instances labeled as "0," a supervised learning method can be devised to train a node or edge prediction model via the logistic regression (LR) algorithm as follows:

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

where m_S denotes the number of training examples in D_S; $x_S^u = \{x_{S,j}^u\}_{j=1}^n$ is a feature vector representing the topological feature values of instance *u* in D_S,and *n* indicates the number of features; y_S^u is the label of instance *u* in D_S; $\theta = \{\theta_j\}_{j=1}^n$ is a weight vector denoting the importance degree of different typological features for the prediction task; $h_\theta(x_S^u) = P(y_S^u = 1|x_S^u;\theta)$ refers to the probability of instance *u* to be predicted as positive. After $\theta^* = \arg\ \min_\theta$ (1) has been learned in D_S, one can first leverage it to estimate the probability of an instance *v* to be positive in D_T as follows:

$$\hat{y}_T^v = \left(1+e^{-(\theta^*)^T x_T^v}\right)^{-1}. \quad (2)$$

Due to its simplicity and high efficiency, LR was adopted as the prediction algorithm in this article. However, the CNL model is flexible to work with other prediction algorithm, as long as the algorithm can work efficiently and also provide the probabilities for its predictions.

[1][Online]. Available: http://networkx.github.io/

1) Feature Incompatibility: Due to domain discrepancy, an identical feature might have different importance degree for the same prediction task in different networks. To address this, we measure the incompatibility of each feature X_j between D_S and D_T, following the approach in [16] as follows:

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

where $\mathrm{Pcc}(X_{S,j}, Y_S)$ represents the Pearson correlation coefficient (PCC) between the value of the *j*th feature and the label of all the instances in D_S; $\mathrm{Pcc}(X_{T,j}, \hat{Y}_T)$ indicates the PCC between the value of the *j*th feature and the predicted probability of all the instances to be positive in D_T. The smaller the incompatibility, the more similarly the feature performs between the source network and the target network. Based on the feature incompatibility measure, a regularization term is defined as follows:

$$R_1\left(\theta\right) = \sum_{j=1}^{n} IC\left(X_j\right)|\theta_j|. \tag{4}$$

By minimizing $R_1(\theta)$, lower absolute weights would be assigned to the features with larger incompatibility (i.e., perform more differently between D_S and D_T). In addition, a L_2 regularization is defined to prevent overfitting as follows:

$$R_2\left(\theta\right) = \frac{1}{2}\sum_{j=1}^{n} {\theta_j}^2. \tag{5}$$

By integrating the regularization terms (4) and (5) and the cost function (1), an overall loss function is developed as follows:

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

where $\lambda_1, \lambda_2 \geq 0$ are the tradeoff parameters to balance the effects of the regularizations (4) and (5). Next, gradient descent algorithm is employed to obtain the parameters minimizing the overall loss function (6), as follows:

$$\frac{\partial L\left(\theta\right)}{\partial \theta_j} = \frac{1}{m_S}\begin{cases} \sum_{u=1}^{m_S}[(h_\theta(x_S^u) - y_S^u)x_{S,j}^u & \text{if } \theta_j \geq 0 \\ \qquad + \lambda_1 IC(X_j) + \lambda_2\theta_j], & \\ \sum_{u=1}^{m_S}[(h_\theta(x_S^u) - y_S^u)x_{S,j}^u & \text{if } \theta_j < 0 \\ \qquad - \lambda_1 IC(X_j) + \lambda_2\theta_j], & \end{cases} \tag{7}$$

$$\theta_j = \theta_j - \alpha\frac{\partial L\left(\theta\right)}{\partial \theta_j} \tag{8}$$

where $\alpha > 0$ denotes the learning rate.

2) Iterative Self-Training: So far, we have considered the feature incompatibility between D_S and D_T. However, the training data are merely obtained from D_S. To make the prediction model also consider the training data from D_T, in our previous CNGS model [13], we adopted an SEDA algorithm [16] to leverage both the fully labeled data in the source network and the unlabeled data in the target network. Specifically, at each self-training iteration, the top-c most confident predicted active edges with the pseudo binary label "1" and the top-c most confident predicted inactive edges with the pseudo binary label "0," are moved from D_T to D_S to retrain the prediction model in the next self-training iteration.

But in the IM application, the number of seed nodes should be much smaller than that of nonseed nodes, and the number of active edges is generally smaller than that of inactive edges. To tackle such imbalanced data, in the CNL model, we propose to move the top-c most confident predicted positive instances, while the top-c' most confident predicted negative instances from D_T to D_S at each self-training iteration, where $c' = c * l$ and $l > 1$ denotes the ratio of the number of most confident predicted negative instances over that of positive instances moved from D_T to D_S.

3) Fuzzy Labels: SEDA is a simple and efficient domain adaptation approach, however, it is highly susceptible to negative transfer during the iterative retraining process. In the SEDA algorithm [16], the most confident predicted positive (or negative) instances generated by different self-training iterations are actually not with the equal likelihood to be positive (or negative). Thus, assigning binary pseudolabels to them would fail to differentiate the degree of their membership to be positive (or negative). Moreover, when the iteration becomes large or too large, the most confident predictions in the target network might become not confident enough. In such a case, if some wrongly predicted labeled instances in the target network are employed to retrain the prediction model, the prediction performance will be degraded. Fuzzy techniques which can well handle imprecision, uncertainty and vagueness during knowledge transfer have demonstrated high effectiveness in alleviating negative transfer in domain adaptation [15], [17]–[22]. Motivated by this, we exploit the use of fuzzy labels to alleviate the weakness of the SEDA algorithm [16], by differentiating the confidence levels of the predictions generated by different self-training iterations. Here, the fuzzy labels represent the degree of the membership of the instances predicted as positive. Note that in the SEDA algorithm [16], when the most confident predictions in the target network are iteratively added to D_S, they are also simultaneously removed from D_T. Intuitively, as the self-training iteration increases, the most confident predictions in D_T will become less confident. Thus, the fuzzy labels are devised with the goals of assigning higher membership of positive to the most confident predicted positive instances, while lower membership of positive to the most confident predicted negative instances, generated by the earlier self-training iterations. Specifically, at the *i*th iteration during the total number of *t* self-training iterations (i.e., $1 \leq i \leq t$ and $t > 1$), the fuzzy labels assigned to the top-c most confident predicted positive instances are defined as follows:

$$1 - \frac{1-\alpha}{t-1}\left(i-1\right) \tag{9}$$

where $0.5 < \alpha < 1$denotes the fuzzy label (i.e., predicted membership to be positive) assigned to the top-c most confident predicted positive instances, generated by the last (i.e., *t*th) self-training iteration. Note that if $\alpha = 1$, the fuzzy labels are equivalent to binary labels. And if $t = 1$, we cannot assign fuzzy

labels. On the other hand, the fuzzy labels assigned to the top-c' most confident predicted negative instances, generated at the *i*th self-training iteration are defined as follows:

$$\frac{1-\alpha}{t-1}(i-1). \tag{10}$$

For simplicity, in this work, we only differentiate the confidence levels of the predictions generated by different self-training iterations. While we treat all the top-c most confident positive instances (and all the top-c' most confident negative instances) generated within the same self-training iteration as equally confident.

Next, by iteratively moving the most confident predicted instances (i.e., both feature vectors and predicted fuzzy labels) from D_T to D_S, the prediction model can be iteratively updated based on not only all the fully labeled data in the source network, but also the newly added most confident prediction data in the target network. In addition, with the devised fuzzy labels, lower degree of membership would be assigned to less confident predicted instances. Thus, we can lower the negative effects caused by adding the less confident predicted target network instances into the training set. Finally, after *t* self-training iterations, for cross-network seed selection, we employ the latest learned node prediction model to select the top-*k* nodes with the highest predicted probability to be positive (i.e., act as seed for IM) in the target network. On the other hand, for cross-network graph sparsification, we leverage the latest learned edge prediction model to remove a fraction *f* of the edges predicted as least likely to be positive (i.e., active for influence propagation in IM) in the target network.

V. Experiments

A. Datasets

The performance of the proposed CNL model was tested for cross-network seed selection and cross-network graph sparsification tasks on four public real-world datasets, namely, NetHEPT,[2] Email-Enron,[3] Epinions,[4] and DBLP.[5] These datasets were frequently employed to evaluate the IM performance in the literature [5], [6], [10], [11], [24]. Both NetHEPT [6] and DBLP [51] datasets are collaboration networks, where each node represents an author and each link connecting two nodes indicates the coauthor relationship. The Email-Enron dataset [52] is an email communication network, where each node represents an email address and each link connecting two nodes indicates the existence of communication between them. The Epinions [53] dataset is a "who trust whom" online social network generated from the Epinions site. Table I gives some statistics of these datasets. To demonstrate the efficiency of the CNL model, we employed the smallest NetHEPT network as the source network, while the other three larger networks as the target networks.

[2][Online]. Available: https://www.microsoft.com/en-us/research/people/weic/

[3][Online]. Available: https://snap.stanford.edu/data/email-Enron.html

[4][Online]. Available: https://snap.stanford.edu/data/soc-Epinions1.html

[5][Online]. Available: https://snap.stanford.edu/data/com-DBLP.html

TABLE I
Statistics of Real-World Datasets

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

Algorithm 1: Cross-Network Learning (CNL).

Input: Source network $D_S = \{(x_S^u, y_S^u)\}_{u=1}^{m_S}$ with m_S labeled instances; Target network $D_T = \{x_T^v\}_{v=1}^{m_T}$ with m_T unlabeled instances; Number of self-training iterations: *t;* Number of most confident predicted positive instances moved from D_T to D_S at each iteration: c; Ratio of the number of most confident predicted negative instances over that of positive instances moved from D_T to D_S: l.
Output: Predicted probabilities of all the instances on D_T to be positive:$\{\hat{y}_T^v\}_{v=1}^{m_T}$.

1. $D'_T = D_T$;
2. On D_S, train a model to obtain $\theta^* = arg\ \min_{\theta} Eq.\ (1)$;
3. For $i = 1$:*t* iterations, do:
   a) Based on θ^*, apply (2) on D_T to predict $\{\hat{y}_T^v\}_{v=1}^{m_T}$;
   b) Based on $\{(x_S^u, y_S^u)\}_{u=1}^{m_S}and\{(x_T^v, \hat{y}_T^v)\}_{v=1}^{m_T}$, measure feature incompatibility between D_S and D_T, via (3);
   c) Rank$\{\hat{y}_T^v\}_{v=1}^{m_T}$and move top-$c$ most confident predicted positive instances with fuzzy labels from D_T to D_S:
   $D_S := D_S + \{(x_T^v, 1 - \frac{1-\alpha}{t-1}(i-1))|\hat{y}_T^v \in Top_c$ highest $\{\hat{y}_T^v\}_{v=1}^{m_T}\}$;
   $D_T := D_T - \{(x_T^v, 1 - \frac{1-\alpha}{t-1}(i-1))|\hat{y}_T^v \in Top_c$ highest $\{\hat{y}_T^v\}_{v=1}^{m_T}\}$;
   d) Rank$\{\hat{y}_T^v\}_{v=1}^{m_T}$and move to p-$(c*l)$ most confident predicted negative instances with fuzzy labels from $D_T$to D_S:
   $D_S := D_S + \{(x_T^v, \frac{1-\alpha}{t-1}(i-1))|\hat{y}_T^v \in \text{Top}_{(c*l)}$ lowest $\{\hat{y}_T^v\}_{u=1}^{m_T}\}$;
   $D_T := D_T - \{(x_T^v, \frac{1-\alpha}{t-1}(i-1))|\hat{y}_T^v \in \text{Top}_{(c*l)}$ lowest $\{\hat{y}_T^v\}_{u=1}^{m_T}\}$;
   e) On new D_S, retrain the model to obtain updated $\theta^* = arg \min_\theta Eq.(6)$;
   End for
4. Based on θ^*, apply (2) on D'_T to predict $\{\hat{y}_T^v\}_{v=1}^{m_T}$.

B. Implementation Details

In the experiments, the IC model [3] with the influence probability $p = 0.01$ was employed as the influence propagation model. In addition, to estimate influence spread, 10 000 Monte Carlo simulations were run, following the standard practice in the literature [3]–[6].

On one hand, for cross-network seed selection, the NewGreedyIC algorithm [6] was employed to select 100 seed nodes in the source network over 10 times to learn the node labels.

Next, in the CNL model, we set the weight of the feature incompatibility regularization as $\lambda_1 = 10$ and divided it by a factor of 1.1 after each self-training iteration, following the practice in [16]; and set the L_2-regularization weight as $\lambda_2 = 0.5$. For self-training process, we set $t = 5$, $c = 10$, $l = 30$ for all the datasets. It means that at each of the five self-training iterations, the top-ten most confident predicted seed nodes and top-300 most confident predicted nonseed nodes in the target network would be iteratively moved to the training set to retrain the node prediction model in the next self-training iteration. In addition, for fuzzy labels design, we set $\alpha = 0.8$, which indicates that at the last (i.e., 5th) self-training iteration, the top-ten most confident predicted seed nodes are with 80% of membership to be seed, while the top-300 most confident predicted nonseed nodes are with 80% of membership to be nonseed.

On the other hand, for cross-network graph sparsification, in the source network, we run the IC model [3] 1000 times to simulate the influence propagation traces induced by 50 seed nodes selected by NewGreedyIC [6] so as to learn the edge labels. Then, in the CNL model, λ_1 was set with the same value as that for cross-network seed selection. While $\lambda_2 = 1,\ 0.1$, and 1 were experimentally adopted for the Email-Enron, Epinions, and DBLP target network, respectively. For self-training process, we set $t = 3$, $c = 200$, $l = 30$, meaning that during each of 3 self-training iterations, the top-200 most confident predicted active edges and top-6000 most confident predicted inactive edges in the target network would be moved to the training set to iteratively update the edge prediction model. For fuzzy labels design, we conducted a grid search on $\alpha \in \{0.6,\ 0.7,\ 0.8, 0.9\}$ and consequently set $\alpha = 0.7,\ 0.7,$ and 0.8 for the Email-Enron, Epinions, and DBLP target network, respectively.

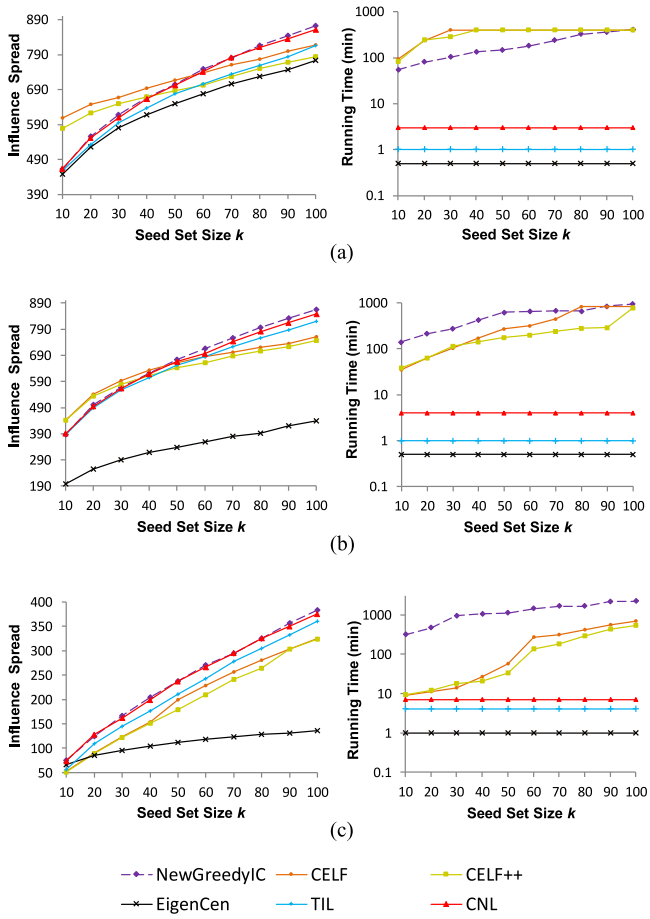


Fig. 3. Performance of CNL for seed selection for IM in three target networks. The higher the influence spread, the better the performance; while the shorter the running time, the better the performance. (a) Email-Enron. (b) Epinions. (c) DBLP.

C. Performance of CNL for Seed Selection

In this section, we empirically evaluate the effectiveness of the proposed CNL model for cross-network seed selection.

1) Baselines: First, the following IM algorithms we compete with are introduced.

1) *NewGreedyIC [6]:* It is a greedy IM algorithm which first converts the influence probabilistic graph into a deterministic graph and then employs a BFS to calculate the influence spread.
2) *CELF [4]:* It is a greedy IM algorithm which greatly reduces the number of evaluations on influence spread by exploiting the submodularity property.
3) *CELF++ [5]:* It further exploits the submodularity property to avoid unnecessary re-computations of marginal gains incurred by CELF.
4) *EigenCen [48]:* It heuristically selects the nodes with the highest eigenvector centrality to be seeds.
5) *TIL [38]:* It is most related to our proposed CNL model, which views seed selection for IM as a cross-network node classification task. However, it ignores the domain discrepancy between the source and the target networks.

2) Evaluation Metrics: For each of the compared IM algorithms, the same number of seed nodes from [10, 100] was assigned for each target network. We measure the cross-network seed selection performance from two perspectives, i.e., the performance w.r.t. IM in the target network and the performance w.r.t. cross-network node prediction. First, to measure the IM performance in the target networks, we adopted two evaluation metrics widely adopted in the IM literature [3]–[8], namely, influence spread and running time. The higher the influence spread, the better the performance; while the shorter the running time, the better the performance. Second, by considering seed selection as a node prediction task, we let both TIL and the proposed CNL model learn the greedy seed selection knowledge from the same greedy algorithm (i.e., NewGreedyIC) in the source network. Thus, in the target networks, we should check whether TIL and CNL could obtain the similar seed selection results w.r.t. the greedy algorithm which they learned the knowledge from. To evaluate it, we adopted the precision@k metric to measure the accuracy of their top-k seed node retrieval results. Here, the ground-truth of seed selection results in the target networks should be the k seed nodes selected by the same greedy algorithm (i.e., NewGreedyIC) as in the source network. The higher the seed node precision, the better the performance.

3) Performance Analysis: First, as shown in Fig. 3, CNL can always achieve a good influence spread almost matched with

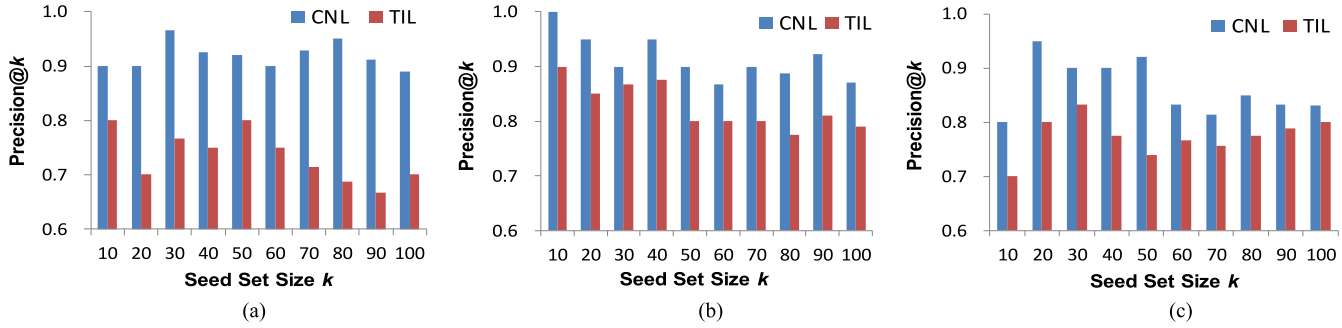


Fig. 4. Performance of CNL for seed node prediction in three target networks, in terms of the seed node precision at top-k retrieved most likely seed nodes. The higher the seed node precision, the better the performance. (a) Email-Enron. (b) Epinions. (c) DBLP.

NewGreedyIC in all the three target networks. In addition, as shown in Fig. 3(a), in the Email-Enron target network, both CNL and NewGreedyIC achieved lower influence spread than CELF and CELF++ when selecting no more than 40 seed nodes; while achieving higher influence spread than CELF and CELF++ when selecting at least 60 seed nodes. In the Epinions target network, as shown in Fig. 3(b), both CNL and NewGreedyIC achieved lower influence spread than CELF and CELF++ when the seed set size was smaller than 50, while achieving higher influence spread than CELF and CELF++ if at least 50 seed nodes were selected. While in the DBLP target network, as shown in Fig. 3(c), both CNL and NewGreedyIC achieved higher influence spread than CELF and CELF++ for any seed set sizes within [10, 100]. These results could be explained by the fact that in the experiments CNL learned the greedy seed selection knowledge from NewGreedyIC in the source network, thus CNL would tend to match with the influence spread of NewGreedyIC, rather than other greedy algorithms, such as CELF and CELF++. On the other hand, as shown in Fig. 3, the running time of CNL was much shorter than all the greedy algorithms in all the three target networks. For example, when selecting 100 seed nodes in the largest DBLP target network containing millions of edges, the running time of NewGreedyIC, CELF, and CELF++ was about 36, 11, and 9 h, respectively. While in the CNL model, it only took 4 min to measure all the topological features for all the nodes in the network and another 4 min to train the prediction model via five self-training iterations. Since the selected topological features can be measured efficiently and the time taken to train the prediction model via self-training approach was quite short, CNL can run much more efficiently than the greedy algorithms. In addition, as shown in Fig. 3, although the EigenCen heuristic was fastest among all the compared algorithms, it failed to achieve a high influence spread in all the three target networks. Also, we can see that although the greedy algorithms can achieve a high influence spread, the running time was extremely long. However, the proposed CNL model can obtain a good tradeoff between the efficiency and effectiveness of the IM problem, i.e., greatly saving the running time while still achieving a good influence spread almost matched with the greedy algorithm.

Next, we compare the performance of TIL and our CNL model. As shown in Fig. 3(a)–(c), in the Email-Enron and DBLP target networks, CNL always achieved higher influence spread than TIL for different seed set sizes between [10, 100]. In the Epinions target network, as shown in Fig. 3(b), CNL achieved higher influence spread than TIL when selecting more than 30 seed nodes. In addition, as shown in Fig. 4, CNL always achieved much higher seed node precisions than TIL, when retrieving any number of k seed nodes between [10, 100] in all the three target networks. Both the higher influence spread and precision of CNL over TIL demonstrates that the proposed fuzzy self-training approach can effectively address the domain discrepancy between the source and the target networks, for cross-network seed selection in IM.

Moreover, we can observe an interesting phenomenon of IM in different types of networks. As shown in Fig. 3, when the same number of seed nodes, say 10, was selected for the 3 types of target networks, the influence spread achieved by CNL in the DBLP collaboration network was only 74; in the Epinion trust social network, it was 390; while in the email communication network, it was 463. These reveal that the email communication and online social networks are much easier to spread the information effectively than the collaboration network.

D. Performance of CNL for Graph Sparsification

Next, the effectiveness of the proposed CNL model for cross-network graph sparsification will be reported.

1) Baselines: First, we introduce the following graph sparsification algorithms we compare with.

1) *Random Heuristic:* It randomly selects a fraction of edges to remove.
2) *RandomWalk [10]:* It removes a fraction of edges such that the Kullback–Leibler divergence between a random walk on the original network and the sparse network can be minimized.
3) *AggRanks [11]:* It removes a fraction of edges according to an overall ranking aggregated by multiple topological feature rankings. It assigns higher weights to more unique feature rankings during aggregation.
4) *CNGS [13]:* It is our previous proposed graph sparsification algorithm, which considers graph sparsification as a cross-network edge prediction task. It adopts the SEDA algorithm [16] to iteratively adapt the influence

propagation knowledge prelearned in a source network to the target network to remove a fraction of the edges least likely to be active for influence propagation. The newly proposed CNL model is based on CNGS, but the main difference is that we devise fuzzy labels in CNL to differentiate the prediction confidence levels of different self-training iterations.

2) Evaluation Metrics: For each of the compared graph sparsification algorithms, a fraction f of edges chosen from [10%, 90%] were removed to extract the sparse networks. Then, in both the original network and the sparse network, the NewGreedyIC algorithm [6] was employed to select the same number of 50 seed nodes to maximize the influence. Next, we evaluate the performance of the graph sparsification algorithms from two perspectives, namely the performance w.r.t. IM and cross-network edge prediction. To evaluate the performance of graph sparsification as a preprocessing step for IM, we adopted two metrics as in the related literature [10], [11], [13]. First, we measure how much influence spread will be lost in the sparse networks, as compared to that in the original network. The less the loss of influence spread, the better the performance. In addition, we check how much running time of the greedy algorithm (i.e., NewGreedyIC) could be saved in the sparse networks, as compared to that taken in the original network. Note that the running time of NewGreedyIC [6] depends on the number of edges in the target network, rather than the topological structures of the edges. Thus, after removing an equal fraction of edges by different graph sparsification algorithms, the save of running time of NewGreedyIC would be the same for different graph sparsification algorithms.

On the other hand, by viewing graph sparsification as a cross-network edge prediction task, AggRanks and the proposed CNGS and CNL models all aim to remove a fraction f of the edges most likely to be inactive (i.e., useless) for influence propagation. Thus, we employed the precision@f metric to examine the accuracy of the top-f(%) most likely inactive edges retrieved by the graph sparsification algorithms. To learn the ground-truth edge labels for the target networks, we run the same influence cascade model as in the source network (i.e., IC model [3]) to simulate the influence propagation traces. The detailed approach for edge label learning can be referred to Section III-B. The higher the inactive edge precision, the better the performance.

3) Performance Analysis: First, as shown in Fig. 5, among all the compared graph sparsification algorithms, CNL always achieved the lowest influence spread loss in all the sparse target networks with different edge removal fractions between [10%, 90%]. In addition, in the Email-Enron, Epinions, and DBLP target networks, if 40% of edges were removed by CNL, the influence spread was just reduced by 8.07%, 6.37%, and 0.84%, respectively, while the running time of NewGreedyIC can be greatly saved by 34.7%, 56.2%, and 29.2% in return, respectively. In addition, as a graph sparsification algorithm for IM, the proposed CNL model is with high efficiency and scalability. For example, even in the largest DBLP target network, the time taken to measure all the topological features and train the prediction model via three self-training iterations was only about 15 min. These reveal that the proposed CNL model indeed acts as an

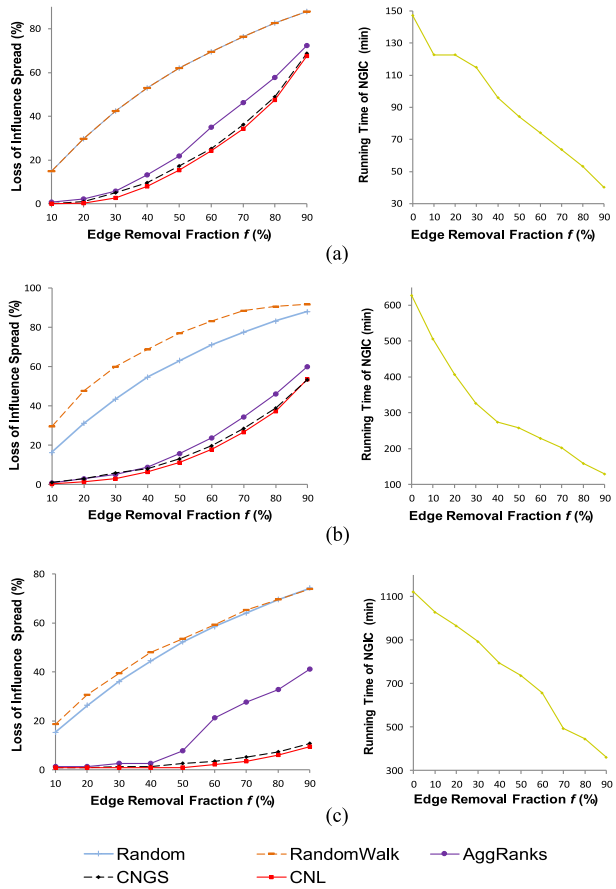


Fig. 5. Performance of CNL for graph sparsification as a preprocessing step for IM in three target networks. The lower the loss of influence spread, the better the performance. (NGIC is short for NewGreedyIC). (a) Email-Enron. (b) Epinions. (c) DBLP.

effective graph sparsification algorithm for IM, since it can obtain a good tradeoff between efficiency and effectiveness of IM, i.e., greatly speed up the greedy algorithm without causing a notable loss of influence spread in the sparse networks.

Second, we look at the performance of the random heuristic and RandomWalk algorithms. As shown in Fig. 5, in all the three target networks, the AggRanks and our CNGS and CNL algorithms can all achieve much lower influence spread loss than the random heuristic and RandomWalk algorithms. This could be explained by the fact that for the IM task, the influence tends to be propagated through those active edges which are adjacent to the highly influential nodes, rather than the randomly selected edges. Since the highly influential nodes are with discriminative topological features w.r.t. randomly selected nodes. The active edges should also be with discriminative topological structures w.r.t. inactive edges. Thus, the AggRanks, CNGS, and CNL algorithms based on the discriminative topological features can significantly outperform the random-based approaches.

Next, we compare the performance of AggRanks, CNGS, and CNL algorithms in the sparse target networks, in terms of the influence spread loss and inactive edge precision. As shown in Fig. 5, when removing less than 30% edges in the target networks, all these three algorithms would perform quite

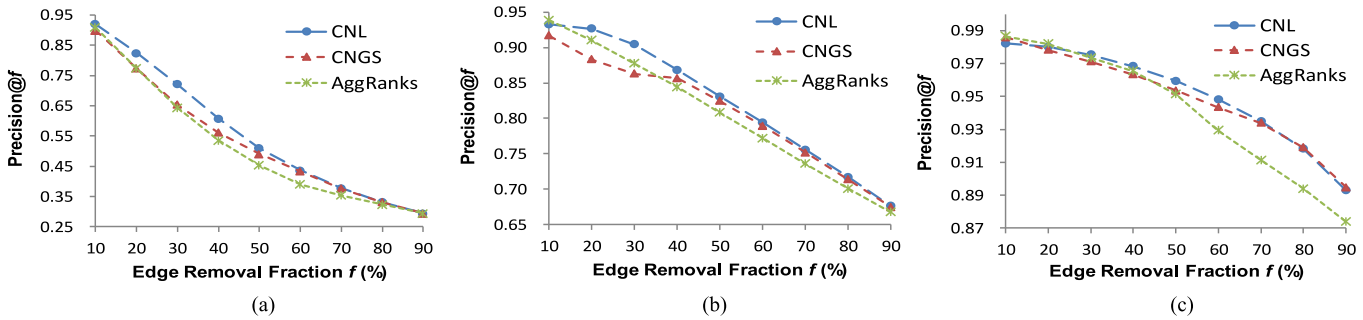


Fig. 6. Performance of CNL for inactive edge prediction in three target networks, in terms of the inactive edge precision at top-*f(%)* retrieved most likely inactive edges. The higher the inactive edge precision, the better the performance. (a) Email-Enron. (b) Epinions. (c) DBLP.

similarly, i.e., only lead to a little loss of influence spread in the sparse target networks. However, both CNL and CNGS can perform much better (i.e., achieve lower influence spread loss) than AggRanks, when the edge removal fraction is more than 30% in the three target networks. On the other hand, as shown in Fig. 6(a), CNL achieved higher inactive edge precision than AggRanks in the sparse Email-Enron target networks with the edge removal fraction between [10%, 70%]. And as shown in Fig. 6(b) and (c), CNL achieved higher precision than AggRanks when the edge removal fraction is more than 10% and 20% in the Epinions and DBLP target network, respectively. In addition, CNGS can also achieve higher inactive edge precision than AggRanks when the edge removal fraction is larger than 30% in the three target networks. For fair comparison, in the experiments, we let AggRanks, CNGS, and CNL all aggregate the same set of topological features to remove the edges least useful for influence propagation. But when learning the feature weightings for aggregation, AggRanks only leverages the topological information in a single network and assigns higher weight to more unique feature in an unsupervised manner. However, the more unique feature might not necessarily be more important for inactive edge prediction. In contrast to AggRanks, the proposed CNL as well as the CNGS models employ a semisupervised self-training approach to iteratively leverage the influence propagation knowledge prelearned in a source network to learn the feature weightings for the target network. Thus, the better overall performance of CNL and CNGS over AggRanks demonstrates the advantage of leveraging the cross-network information w.r.t. the single network information for inactive edge prediction in IM. Moreover, we can observe that CNL achieved slightly lower influence spread loss than CNGS (as shown in Fig. 5) and also higher inactive edge precision than CNGS (as shown in Fig. 6). This reflects the effectiveness of incorporating fuzzy labels into the SEDA algorithm to differentiate the confidence levels of the predictions generated by different self-training iterations so as to lower the negative effects caused by less confident target predictions.

Finally, let us look at some interesting differences when graph sparsification is applied to different types of networks. As shown in Fig. 5, when removing equal fraction of edges in the three target networks, the loss of influence spread is lowest in the DBLP target network. On the other hand, as shown in Fig. 6, when giving the same edge removal fraction, the inactive edge precisions are highest in the DBLP target network. These results reveal that the email communication (i.e., Email-Enron) and trust social networks (i.e., Epinions) show greater challenge to graph sparsification than the collaboration network (i.e., DBLP). This could be explained by our previous observation in seed selection that the email communication and trust social networks are much easier to spread the information than the collaboration network. In other words, the fraction of active edges should be smallest (i.e., most of the edges are inactive for influence propagation) in the DBLP collaboration network. Thus, even though we remove quite a large fraction (e.g., 70%) of edges in the DBLP network, the influence spread in the sparse network would not be substantially affected.

E. Sensitivity Analysis of Parameters

In this section, we analyze the sensitivity of the parameters, i.e., $\lambda_1, \lambda_2, t, c, l, \alpha$ in the CNL model. Fig. 7 shows the effect of these parameters on selecting 50 seed nodes by CNL in the DBLP target network. Also, Fig. 8 reports the sensitivity of the parameters for removing 30% of edges by CNL in the Email-Enron target network.

First, we analyze the effect of λ_1 on the performance of CNL, which denotes the weight of the feature incompatibility regularization (4). As shown in Figs. 7(a) and 8(a), $\lambda_1 \in \{10, 15\}$ can significantly increase both seed node and inactive edge precisions, as compared to $\lambda_1 = 0$. This demonstrates the effectiveness of incorporating (4) in CNL to assign lower absolute weights to the features which perform less similarly between the source and the target networks. While when $\lambda_1$becomes too large (i.e., 20), both the seed node and inactive edge precisions would slightly decrease. Thus, we set $\lambda_1 = 10$ in the proposed CNL model for both seed node and inactive edge prediction. In addition, as shown in Figs. 7(b) and 8(b), the performance of CNL is not sensitive to the value of λ_2, which indicates the weight of L_2 regularization (5) to prevent overfitting. It might be because that the feature incompatibility regularization (4) also implicitly prevents overfitting, thus making the effectiveness of (5) become less significant.

Second, we investigate the sensitivity of t, i.e., the total number of self-training iterations. As shown Fig. 7(c), the seed node precision keeps increasing as t increases until t reaches 7, while after that, the seed node precision slightly decreases. In addition,

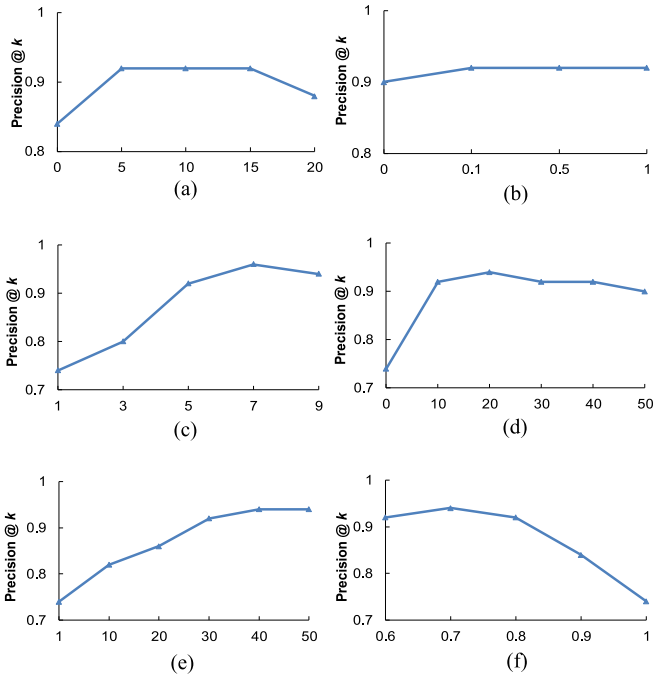


Fig. 7. Parameter sensitivity of CNL over $\lambda_1, \lambda_2, t, c, l, \alpha$ for seed node prediction in the DBLP target network when the seed set size k is 50. The default settings are $\lambda_1 = 10$, $\lambda_2 = 0.5$, $t = 5, c = 10, l = 30$, $\alpha = 0.8$. (a) Weight of feature incomp. reg.: λ_1. (b) Weight of L_2 reg.: λ_2. (c) # self-training iterations: t. (d) # pos. instances at each iteration: c. (e) Ratio of neg. over pos. instances: l. (f) Value of fuzzy label: α.

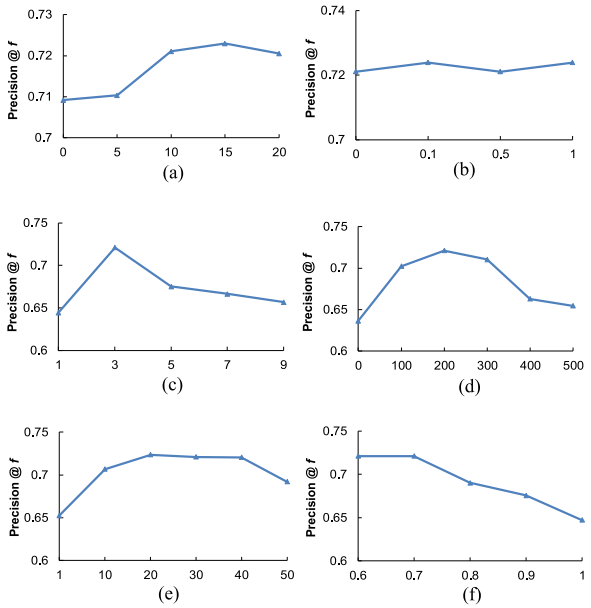


Fig. 8. Parameter sensitivity of CNL over $\lambda_1, \lambda_2, t, c, l, \alpha$ for inactive edge prediction in Email-Enron target network when the edge removal fraction f is 30%. The default settings are $\lambda_1 = 10, \lambda_2 = 1, t = 3, c = 200, l = 30$, $\alpha = 0.7$. (a) Weight of feature incomp. reg.: λ_1. (b) Weight of L_2 reg.: λ_2. (c) # self-training iterations: t. (d) # pos. instances at each iteration: c. (e) Ratio of neg. over pos. instances: l. (f) Value of fuzzy label: α.

note that the running time of CNL will increase as t increases. Thus, in order to achieve a good prediction performance and also save the running time, we fix $t = 5$ for seed node prediction on all the datasets. On the other hand, as shown in Fig. 8(c), $t = 3$ can significantly increase the inactive edge precision w.r.t. $t = 1$, however, when $t > 3$, the inactive edge prediction performance will be degraded. This is because the SEDA algorithm can easily cause negative transfer, i.e., when t becomes too large, the most confident predicted instances in the target network might become not confident enough. Then, if some wrongly predicted target network instances are utilized as the ground-truth training data to retrain the prediction model, the prediction performance will be declined. Thus, we fix $t = 3$ in the CNL model for cross-network graph sparsification in all the target networks.

Third, the sensitivity of c is investigated, which indicates the number of most confident predicted positive instances moved from the target network to the training set, at each iteration. As shown in Fig. 7(d), the positive values of $c \in \{10, 20, 30, 40, 50\}$ all achieve much higher seed node precision than $c = 0$. Also, as shown in Fig. 8(d), $c \in \{100, 200, 300\}$ yields much higher inactive edge precision than $c = 0$. These demonstrate the effectiveness of the self-training approach to leverage the target network data to retrain the prediction model. In addition, we assigned a larger value of c for inactive edge prediction than that for seed node prediction. This is because the number of edges should be much larger than that of nodes in a target network. Intuitively, when training an edge prediction model, we would require larger number of training examples (i.e., larger c) than that for node prediction.

Next, we investigate the sensitivity of l, which represents the ratio of the number of most confident predicted negative instances over that of positive instances moved from the target network to the training set. As shown in Fig. 7(e), $l \geq 1$ contributes to much higher seed node precision than $l = 1$. This is because the number of nonseed nodes is much larger than that of seed nodes. If we just add the equal number of most confident predicted seed nodes and nonseed nodes (i.e., $l = 1$) in the target network to the training set, the prediction model might fail to consider enough negative training examples from the target network. Thus, it is effective to incorporate the l parameter to address the imbalanced data condition for seed node prediction. In addition, as shown in Fig. 8(e), $l \geq 1$also leads to much better inactive edge prediction than $l = 1$. In addition, we observe from Fig. 8(d) and (e) that when c or l become too large, i.e., $c \geq 400$, $l = 50$, the inactive edge precision will significantly decrease. This is also caused by the weakness of the self-training approach as we explained above, i.e., utilizing not confident enough target network predictions to retrain the model would lead to negative effect on the prediction performance.

Finally, the effectiveness of incorporating fuzzy labels into the self-training algorithm is discussed. In the designed fuzzy labels, we used the parameter $\alpha$to denote the degree of membership to be positive assigned to the most confident predicted positive instances generated at the last self-training iteration. Note that $0.5 < \alpha < 1$indicates that we incorporate fuzzy labels, while $\alpha = 1$ indicates that we utilize binary labels. As shown in Figs. 7(f) and 8(f), $\alpha \in \{0.6, 0.7, 0.8, 0.9\}$ lead to both higher seed node and inactive edge precisions than $\alpha = 1$. This demonstrates that the fuzzy labels are indeed effective in alleviating negative transfer caused by the conventional SEDA algorithm for

both cross-network seed selection and graph sparsification. This is because with the provision of fuzzy labels, the confident levels of the target network predictions generated by different self-training iterations can be easily differentiated. Then, the negative effects caused by the less confident predicted instances can be lowered. In addition, as shown in Fig. 7(f), the performance of CNL w.r.t. seed node prediction is not sensitive to the value of α, when $\alpha \in \{0.6,\ 0.7,\ 0.8\}$. In contrast, as shown in Fig. 8(f), for inactive edge prediction, $\alpha \in \{0.6,\ 0.7\}$can achieve much higher precision than $\alpha = 0.8$. This could be explained by the fact that we utilize a larger number of target network instances (i.e., larger value of c) to train the inactive edge prediction model, as compared to that for seed node prediction. Such larger number of target network instances might be more likely to include the predictions which are not confident enough. Thus, for inactive edge prediction, we would assign lower confident level (smaller value of α) to those most confident target network predictions generated by the last self-training iteration.

VI. Conclusion

Although the IM problem has been extensively studied, very little work addresses this problem in a cross-network scenario. In this article, we propose an innovative cross-network learning approach to study two issues of cross-network IM problem, i.e., cross-network seed selection and cross-network graph sparsification. On one hand, we consider seed selection for IM as a node prediction task, with the goal of selecting the nodes most likely to act as seed for IM. On the other hand, we view graph sparsification as an edge prediction task, aiming to remove the edges predicted as least likely to contribute to influence propagation for IM. To achieve such goals, a CNL model is proposed to leverage the knowledge prelearned in a smaller source network to predict seed nodes and inactive edges for multiple heterogeneous target networks. To address domain discrepancy, lower weights would be assigned to the features which perform less similarly between the source and the target networks. In addition, a self-training for domain adaptation approach is employed to iteratively train the prediction model based on not only the fully labeled data in the source network, but also the most confident predicted labeled data in the target network. Moreover, to alleviate the negative effect of leveraging not confident enough predicted data in the target network, we propose a fuzzy self-training approach to further improve the cross-network prediction performance. We devise the fuzzy labels to represent the degree of the membership of the instances predicted as positive. With the help of fuzzy labels, we can capture prediction uncertainty in the target network, and differentiate the prediction confidence in different self-training iterations so as to reduce the negative effects of the less confident target network predictions on iterative retraining.

Experiments on the real-world datasets demonstrate that the proposed CNL model taking advantage of the fuzzy self-training approach can achieve good performance for both seed node prediction and inactive edge prediction in IM. On one hand, by leveraging the cross-network seed selection knowledge, the CNL model can achieve a good tradeoff between efficiency and effectiveness of IM, i.e., greatly save the required running time while still achieving a satisfactory influence spread comparable to the greedy algorithm in the target networks. On the other hand, by leveraging the cross-network influence propagation knowledge for graph sparsification, CNL would only remove the edges least useful for influence propagation, thus causing just a little loss of influence spread in the sparse target networks, while significantly speeding up the IM greedy algorithms in the sparse target networks.

In the future, we plan to leverage the knowledge prelearned from multiple source networks rather than a single source network, to make predictions over seed nodes and inactive edges for the target networks. In this article, the fuzzy labels have been incorporated into the self-training algorithm to alleviate negative transfer in cross-network predictions for IM. In the future, we plan to apply the proposed fuzzy self-training approach to other domain adaptation tasks, such as cross-domain image classification and cross-domain sentiment prediction.

References

[1] J. Nail, C. Charron, and S. Baxter, "The consumer advertising backlash," in Forrester Research and Intelliseek Market Research Report, vol. 137, 2004.

[2] I. R. Misner, *The World's Best Known Marketing Secret: Building Your Business With Word-of-Mouth Marketing*. Austin, TX, USA: Bard Press, 1999.

[3] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 137–146.

[4] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2007, pp. 420–429.

[5] A. Goyal, W. Lu, and L. V. Lakshmanan, "Celf++: Optimizing the greedy algorithm for influence maximization in social networks," in *Proc. Int. Conf. Companion WWW*, 2011, pp. 47–48.

[6] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 199–208.

[7] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 1029–1038.

[8] K. Jung, W. Heo, and W. Chen, "IRIE: Scalable and robust influence maximization in social networks," in *Proc. IEEE 12th Int. Conf. Data Mining*, 2012, pp. 918–923.

[9] Z.-L. Luo, W.-D. Cai, Y.-J. Li, and D. Peng, "A pagerank-based heuristic algorithm for influence maximization in the social network," in *Recent Progress in Data Engineering and Internet Technology*. New York, NY, USA: Springer, 2012, pp. 485–490.

[10] B. Wilder and G. Sukthankar, "Sparsification of social networks using random walks," in *Proc. Int. Conf. Social Comput.*, 2015. [Online]. Available: http://teamcore.usc.edu/people/bryanwilder/publications/wilder_socialcom_2015.pdf

[11] H. Lamba and R. Narayanam, "A novel and model independent approach for efficient influence maximization in social networks," in *Proc. Int. Conf. Web Inf. Syst. Eng.*, 2013, pp. 73–87.

[12] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen, "Sparsification of influence networks," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 529–537.

[13] X. Shen, F. Chung, and S. Mao, "Leveraging cross-network information for graph sparsification in influence maximization," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf.*, 2017, pp. 801–804.

[14] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

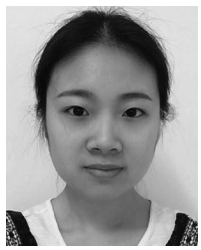
[15] S. Kumar, A. K. Shukla, P. K. Muhuri, and Q. D. Lohani, "Atanassov intuitionistic fuzzy domain adaptation to contain negative transfer learning," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2016, pp. 2295–2301.

[16] M. Chen, K. Q. Weinberger, and J. Blitzer, "Co-training for domain adaptation," in *Proc. 24th Int. Conf. Neural Inf. Process. Syst.*, 2011, pp. 2456–2464.

[17] F. Liu, J. Lu, and G. Zhang, "Unsupervised heterogeneous domain adaptation via shared fuzzy equivalence relations," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 6, pp. 3555–3568, Dec. 2018.
[18] Z. Deng, Y. Jiang, F.-L. Chung, H. Ishibuchi, and S. Wang, "Knowledge-leverage-based fuzzy system and its modeling," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 4, pp. 597–609, Aug. 2013.
[19] J. Shell and S. Coupland, "Fuzzy transfer learning: methodology and application," *Inf. Sci.*, vol. 293, pp. 59–79, 2015.
[20] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, and J. Lu, "Granular fuzzy regression domain adaptation in Takagi-Sugeno fuzzy models," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 2, pp. 847–858, Apr. 2018.
[21] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, and J. Lu, "Fuzzy regression transfer learning in Takagi-Sugeno fuzzy models," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 6, pp. 1795–1807, Dec. 2017.
[22] P. Qian *et al.*, "Cluster prototypes and fuzzy memberships jointly leveraged cross-domain maximum entropy clustering," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 181–193, Jan. 2016.
[23] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2001, pp. 57–66.
[24] N. Ohsaka, T. Akiba, Y. Yoshida, and K.-I. Kawarabayashi, "Fast and accurate influence maximization on large networks with pruned Monte-Carlo simulations," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 138–144.
[25] S. Cheng, H. Shen, J. Huang, G. Zhang, and X. Cheng, "Staticgreedy: Solving the scalability-accuracy dilemma in influence maximization," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.*, 2013, pp. 509–518.
[26] Y. Wang, G. Cong, G. Song, and K. Xie, "Community-based greedy algorithm for mining top-k influential nodes in mobile social networks," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 1039–1048.
[27] G. Song, X. Zhou, Y. Wang, and K. Xie, "Influence maximization on large-scale mobile social network: A divide-and-conquer method," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1379–1392, May 2015.
[28] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithm*, 2014, pp. 946–957.
[29] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2014, pp. 75–86.
[30] H. T. Nguyen, M. T. Thai, and T. N. Dinh, "Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2016, pp. 695–710.
[31] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 1539–1554.
[32] J. Tang, X. Tang, and J. Yuan, "Influence maximization meets efficiency and effectiveness: A hop-based approach," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, 2017, pp. 64–71.
[33] J. Kim, S.-K. Kim, and H. Yu, "Scalable and parallelizable processing of influence maximization for large-scale social networks," in *Proc. IEEE 29th Int. Conf. Data Eng.*, 2013, pp. 266–277.
[34] T. N. Dinh, H. Zhang, D. T. Nguyen, and M. T. Thai, "Cost-effective viral marketing for time-critical campaigns in large-scale social networks," *IEEE/ACM Trans. Netw.*, vol. 22, no. 6, pp. 2001–2011, Dec. 2014.
[35] W. Chen, W. Lu, and N. Zhang, "Time-critical influence maximization in social networks with time-delayed diffusion process," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 1–5.
[36] Y. Shen, T. N. Dinh, H. Zhang, and M. T. Thai, "Interest-matching information propagation in multiple online social networks," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 1824–1828.
[37] Q. Zhan, J. Zhang, S. Wang, S. Y. Philip, and J. Xie, "Influence maximization across partially aligned heterogenous social networks," in *Proc. Pac.-Asia Conf. Knowl. Discovery Data Mining*, 2015, pp. 58–69.
[38] Q. Hu, G. Wang, and S. Y. Philip, "Transferring influence: Supervised learning for efficient influence maximization across networks," in *Proc. Int. Conf. Collaborative Comput.: Netw. Appl. Worksharing*, 2014, pp. 45–54.
[39] S. Kullback, "Letter to the editor: The Kullback-Leibler distance," *Am. Statistician*, vol. 41, pp. 340–341, 1987.
[40] M. Purohit, B. A. Prakash, C. Kang, Y. Zhang, and V. Subrahmanian, "Fast influence-based coarsening for large networks," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 1296–1305.
[41] F. Zhou, S. Mahler, and H. Toivonen, "Simplification of networks by edge pruning," in *Bisociative Knowledge Discovery*. New York, NY, USA: Springer, 2012, pp. 179–198.
[42] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 613–622.
[43] T. Leung and F. Chung, "Persuasion driven influence propagation in social networks," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining* 2014, pp. 548–554.
[44] B. Liu, G. Cong, D. Xu, and Y. Zeng, "Time constrained influence maximization in social networks," in *Proc. IEEE 12th Int. Conf. Data Mining*, 2012, pp. 439–448.
[45] N. Barbieri, F. Bonchi, and G. Manco, "Topic-aware social influence propagation models," in *Proc. IEEE 12th Int. Conf. Data Mining*, 2012, pp. 81–90.
[46] X. Yi, X. Shen, W. Lu, T. S. Chan, and F. L. Chung, "Persuasion driven influence analysis in online social networks," in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 4451–4456.
[47] M. Newman, *Networks*. Oxford, U.K.: Oxford Univ. Press, 2010.
[48] M. E. Newman, "The mathematics of networks," in *The New Palgrave Encyclopedia Economics*, vol. 2. Basingstoke, U.K.: Palgrave Macmillan, 2008, pp. 1–12.
[49] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.
[50] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford Info Lab, Stanford, CA, USA, Tech. Rep. 422, 1999.
[51] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 181–213, 2015.
[52] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Int. Math.*, vol. 6, no. 1, pp. 29–123, 2009.
[53] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the semantic web," in *Proc. 2nd Int. Conf. Semantic Web Conf.*, 2003, pp. 351–368.

Xiao Shen received the double B.Sc. degrees from the Beijing University of Posts and Telecommunications, Beijing, China, and Queen Mary University of London, London, U.K., in 2012, the M.Phil. degree from the University of Cambridge, Cambridge, U.K., in 2013, and the Ph.D. degree from Department of Computing, Hong Kong Polytechnic University, Hong Kong, in 2019.

She is currently a Postdoc with Hong Kong Polytechnic University. She received the Hong Kong Ph.D. Fellowship. Her research interests include feature representation learning, deep learning, transfer learning, and data mining in complex networks.



Sitong Mao received the B.Eng. degree in computer science from the University of Xia Men, Xiamen, China, in 2015. She is currently working toward the Ph.D. degree with the Department of Computing, Hong Kong Polytechnic University, Hong Kong.

Her research interests include deep learning and transfer learning.

Fu-lai Chung received the B.Sc. degree from the University of Manitoba, Winnipeg, MB, Canada, in 1987, and the M.Phil. and Ph.D. degrees from the Chinese University of Hong Kong, Hong Kong, in 1991 and 1995, respectively.

He is currently an Associate Professor with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His current research interests include deep learning, transfer learning, adversarial learning, social network analysis and mining, recommendation systems, and big data learning.

Dr. Chung also serves on program committees of top international conferences, including IEEE ICDM, AAAI, and ICPR. He has published widely in prestige international journals, including the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON FUZZY SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and *Pattern Recognition and Neural Networks*.