Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Community preserving adaptive graph convolutional networks for link prediction in attributed networks

Chaobo He [a,b], Junwei Cheng [a,b], Xiang Fei [c], Yu Weng [a], Yulong Zheng [a], Yong Tang [a,b,*]

[a] School of Computer Science, South China Normal University, Guangzhou 510631, Guangdong, China
[b] Pazhou Lab, Guangzhou 510335, China
[c] Department of Computing, Coventry University, Coventry CV15FB, UK

ARTICLE INFO

Article history:
Received 21 December 2022
Received in revised form 17 April 2023
Accepted 21 April 2023
Available online 27 April 2023

Keywords:
Link prediction
Network embedding
Community detection
Graph convolutional networks
Attributed networks

ABSTRACT

Link prediction in attributed networks has attracted increasing attention recently due to its valuable real-world applications. Various related methods have been proposed, but most of them cannot effectively utilize community structure, neither can they well fuse attribute information and link information to improve the performance. Inspired by our empirical observations on how community structure affects the generation of links, we propose a novel Community Preserving Adaptive Graph Convolutional Networks (CPAGCN) method to tackle the link prediction task in attributed networks. Specifically, CPAGCN is composed of two core modules: network embedding and link prediction. Network embedding module utilizes AGCN to seamlessly fuse link information and attribute information to obtain node representations, which are simultaneously driven to preserve community structure via an appropriate community detection model. Taking these node representations as the input, link prediction module applies multilayer perception (MLP) to directly learn the prediction scores for potential links. Through combining the graph reconstruction loss with the prediction loss to train AGCN and MLP jointly, CPAGCN can learn node representations that are more beneficial to predicting links. To verify the effectiveness of CPAGCN, we conduct extensive experiments on six real-world attributed networks. The results demonstrate that CPAGCN performs better than several strong competitors in link prediction. The source code is available at https://github.com/GDM-SCNU/CPAGCN.

© 2023 Elsevier B.V. All rights reserved.

1. Introduction

In the era of big data, complex networks with attribute information are increasingly common. For instance, users in online social networks have demographic attributes (e.g., age, gender and occupation) and textual features extracted from their public comments and posts, while authors in co-authorship networks can be associated with keywords found in their publications. These networks, known academically as attributed networks or graphs [1,2], contain richer information (both attribute and link information) and hold greater potential value. As a result, attributed networks have garnered significant attention from industry practitioners and academic researchers alike. Various related research topics have emerged, such as the link prediction discussed in this paper.

Link prediction aims to infer the missing or future links between nodes based on their attributes and the currently observed links [3–6]. Effective link prediction is not only helpful to understand the evolution of networks [7,8], but also has many important applications in other fields, such as friend recommendation in online social networks [9–11], knowledge graph completion [12,13] and drug–drug interactions prediction [14,15].

Recently, various methods for link prediction in attributed networks have been proposed and can be roughly divided into three categories: similarity-based methods, learning-based methods and network embedding-based methods. Similarity-based methods compute the similarity scores of pairwise nodes based on their link structure or attributes, and suppose that two nodes with higher scores are more likely to establish a link. How to compute the similarity between nodes is the key of this type of methods, and some simple yet effective local or global node similarity metrics are widely adopted, including Common Neighbors (CN), Preferential Attachment (PA), Adamic–Adar (AA), Resource Allocation (RA), Katz index, rooted PageRank (PR), SimRank (SR) and Random Walk with Restart (RWR), which have been comprehensively surveyed in [5,6]. Learning-based methods firstly learn a model using the training set constructed from currently observed links, and then directly apply the learned model to predict the likelihood of potential links between nodes. They can further

* Correspondence to: School of Computer Science, South China Normal University, Guangzhou 510631, Guangdong, China.
E-mail addresses: hechaobo@m.scnu.edu.cn (C. He), jung@m.scnu.edu.cn (J. Cheng), aa5861@coventry.ac.uk (X. Fei), wengyu@m.scnu.edu.cn (Y. Weng), yulongzheng@m.scnu.edu.cn (Y. Zheng), ytang@m.scnu.edu.cn (Y. Tang).

https://doi.org/10.1016/j.knosys.2023.110589
0950-7051/© 2023 Elsevier B.V. All rights reserved.

be grouped as: classification model-based methods [16,17], probabilistic model-based methods [18,19] and deep learning-based methods [20,21]. Having the features of aforementioned two types of methods, network embedding-based methods design embedding models to learn low-dimensional node representations, which are treated as the input of the subsequent link prediction task. The link prediction can be accomplished via specific vector similarity metrics or classification models, such as well-known support vector machine and multi-layer perception. In theory, all network embedding methods (e.g., methods reviewed in [22–25]) are applicable to link prediction in attributed networks, especially those which can fuse attribute information and link information, such as TADW [26], ASNE [27] and VGAE [28].

As previously mentioned, current techniques for link prediction in attributed networks concentrate on discovering and exploiting multiple factors that influence link generation to enhance performance, including attribute proximity and structural proximity. Effective fusion of these two types of proximities has been shown to improve the performance. Besides, structural proximity should be more thoroughly exploited due to its diversity. However, most methods tend to focus on microscopic proximities, such as low-order and second-order proximities [25], and overlook mesoscopic community structure. A community is a tightly bound network module in which nodes within the same community are more densely interconnected than those outside of it [29,30]. By analyzing real attributed networks (see Section 2), we have observed that nodes within the same community are more likely to be linked, particularly when they share multiple communities. This has prompted us to incorporate community structure into our approach, with the aim of further enhancing link prediction performance in attributed networks.

Based on the above analysis, in this paper we develop a method named CPAGCN for link prediction in attributed networks via Community Preserving Adaptive Graph Convolutional Networks. CPAGCN is based on the framework of network embedding, which can learn node representations that are more beneficial to link prediction. Specifically, these representations not only fuse attribute and microscopic structural proximities, but also more importantly, preserve mesoscopic community structure. Our major contributions are summarized as follows:

- We conduct empirical observations on real attributed networks to investigate the relationship between the generation of links and community structure. The results motivate us to further boost the performance of link prediction by utilizing community structure.
- We devise CPAGCN in an end-to-end manner, which is composed of a network embedding module and a link prediction module. These two modules are respectively implemented by using Adaptive Graph Convolutional Networks (AGCN) and Multi-Layer Perception (MLP). By introducing attention mechanism, AGCN can seamlessly fuse attribute and structural proximities to obtain node representations.
- We specially incorporate the community structure into network embedding via the combination of AGCN and community detection model. By designing the unified optimization framework, network embedding module can generate node representations that are very beneficial to the link prediction.
- We conduct extensive experiments on six real-world attributed networks. The results show that CPAGCN not only outperforms state-of-the-art methods in terms of the accuracy metrics, but also is robust and efficient enough.

The rest of this paper is organized as follows. In Section 2 we experimentally investigate the effect of community structure on the generation of links. The details on the proposed method CPAGCN and extensive experimental results are presented in Sections 3 and 4, respectively. We briefly review the related work in Section 5 and finally give our conclusions in Section 6.

Table 1
Statistics of datasets. n: number of nodes, m: number of links, r: number of attributes, d: number of communities, $density = \frac{2m}{n(n-1)}$.

Dataset	Network type	n	m	r	d	*density*
Cora	Citation	2708	5429	1433	105	0.0015
PubMed	Citation	19,717	44,338	500	37	0.0002
Facebook 1684 (Fb1684)	Social	792	14,024	15	17	0.0448
Facebook 1912 (Fb1912)	Social	755	30,025	29	46	0.1055
Computer science (CS)	Co-authorship	21,957	96,750	7800	18	0.0004
Engineering (Eng)	Co-authorship	14,927	49,305	4800	16	0.0004

2. Empirical observations

In this section, we firstly describe datasets used here, and then present our data analysis of the relationship between the generation of links and community structure.

2.1. Datasets

We select six real-world attributed networks as datasets to make the empirical analysis, including two widely used benchmark networks Cora[1] and PubMed[1], and four networks presented in [31]. All of these networks have explicitly defined ground-truth communities, as depicted in Fig. 1, and their statistical information is listed in Table 1. The sparsity of the network increases as the density value decreases in the table. For Cora and PubMed, we have obtained their ground-truth community partitions by applying the Louvain algorithm [32] instead of the associated classification labels. This is because we believe that the number of ground-truth communities is different from that of classification labels. Furthermore, the ground-truth communities in Cora and PubMed are non-overlapping, whereas those in the other four networks may overlap to some extent. Specifically, the ratios of overlapping communities with at least one other community are 52.9%, 97.8%, 100% and 100% in Fb1684, Fb1912, CS, and Eng networks, respectively.

2.2. Observations

For every network, we first respectively calculate the link probability of a pair of nodes in the same community (denoted as In_Pro) and the link probability of a pair of nodes in different communities (denoted as Out_Pro). The results are shown in Table 2. As we can see, on Fb1684 and Fb1912 networks, In_Pro is much higher than Out_Pro. On four relatively sparser networks: Cora, PubMed, CS and Eng, In_Pro is still higher than Out_Pro, although they are both very low. These mean that nodes in the same community is more possible to establish the links.

We further conduct an investigation of the relationship between the link probability of a pair of nodes, and the number of their shared communities on four networks with overlapping community structure, namely Fb1684, Fb1912, CS, and Eng. The probability of a link between nodes is plotted against the number of their shared communities in Fig. 2. In order to improve the clarity, all probabilities are normalized for each network, resulting in minimum and maximum values of 0 and 1, respectively. The probability curves display an upward trend as the number of shared communities increases, indicating that the likelihood of a

[1] https://linqs.soe.ucsc.edu/data.

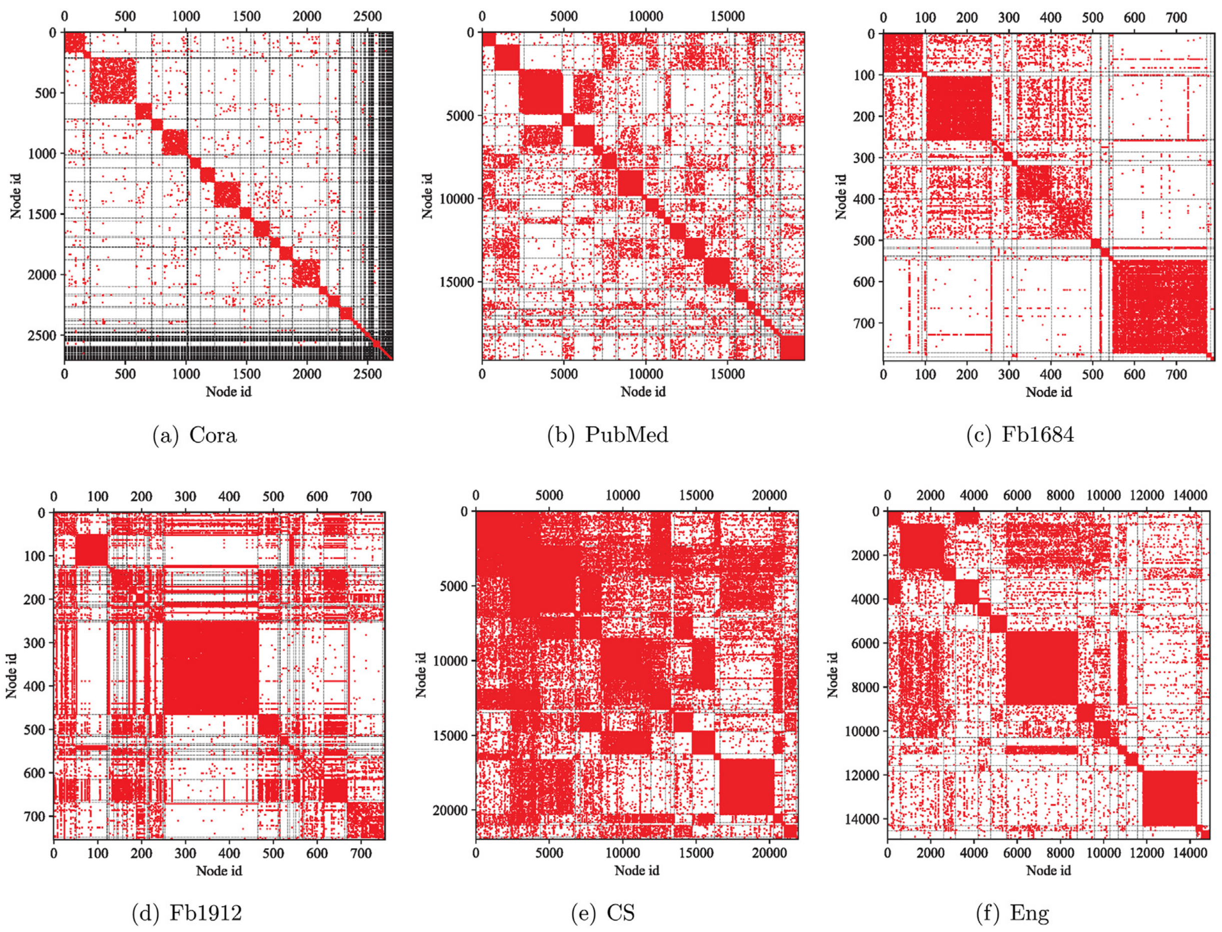


Fig. 1. Real-world attributed networks with ground-truth communities. The red points denote the links between nodes. The squares on the diagonal are the communities. The more red points between communities, the more likely communities are to be overlapping.

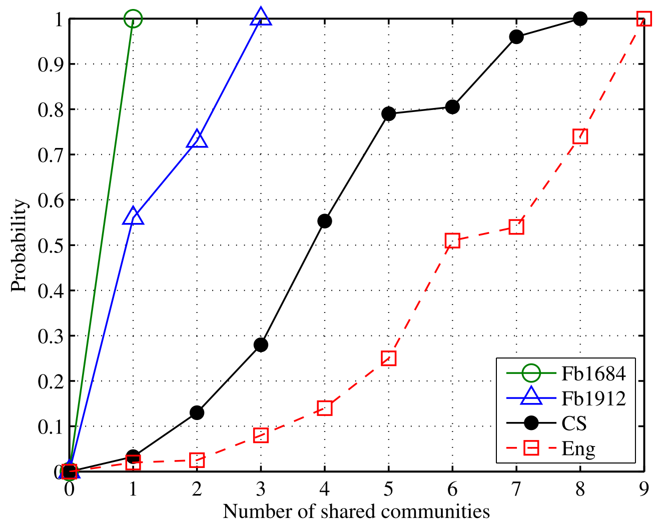


Fig. 2. The link probability with respect to the number of shared communities.

Table 2
The link probability comparison of a pair of nodes in the same community (In_Pro) and in different communities (Out_Pro).

Dataset	In_Pro	Out_Pro
Cora	0.007	3.3e−4
PubMed	0.003	2.7e−4
Fb1684	0.225	0.012
Fb1912	0.431	0.022
CS	0.003	1.9e−5
Eng	0.002	1.4e−5

link between nodes is positively correlated with the number of communities they have in common.

The above observations indicate that if two nodes share one or even more communities, they are very likely to be linked. Many intuitive examples in real life can support this. For example, people belonging to one or multiple common organizations are prone to establish friendships, and researchers working together in one or several common teams often have the cooperative relationships. These findings give us the inspiration that community memberships can affect the generation of links between nodes. Apart from attribute proximity and microscopic structural proximity, community structure should also be considered in the link prediction. This is also the major motivation of developing our method CPAGCN.

3. Methodology

In this section, we first make the problem statement concerned in this paper, and then describe our proposed method CPAGCN in detail. Throughout this paper, we denote matrices by bold uppercase letters. For a given matrix $\mathbf{Z}$, its ith row vector, (i, j)th entry and transpose are denoted as $\mathbf{Z}_i$, $\mathbf{Z}_{ij}$ and $\mathbf{Z}^T$, respectively.

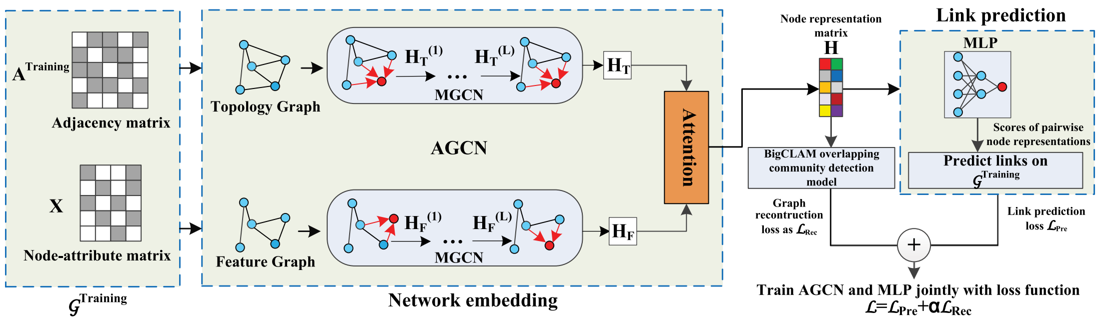

$\mathbf{A}^{\text{Training}}$
Adjacency matrix
X
Node-attribute matrix
$\mathcal{G}^{\text{Training}}$
Topology Graph
$\mathbf{H}_{\mathbf{T}}^{(1)}$ … $\mathbf{H}_{\mathbf{T}}^{(L)}$
MGCN
$\mathbf{H}_{\mathbf{T}}$
AGCN
Feature Graph
$\mathbf{H}_{\mathbf{F}}^{(1)}$ … $\mathbf{H}_{\mathbf{F}}^{(L)}$
MGCN
$\mathbf{H}_{\mathbf{F}}$
Attention
Network embedding
Node representation matrix
H
BigCLAM overlapping community detection model
Graph recontruction loss as $\mathcal{L}_{\text{Rec}}$
Link prediction
MLP
Scores of pairwise node representations
Predict links on $\mathcal{G}^{\text{Training}}$
Link prediction loss $\mathcal{L}_{\text{Pre}}$
Train AGCN and MLP jointly with loss function
$\mathcal{L}=\mathcal{L}_{\text{Pre}}+\alpha\mathcal{L}_{\text{Rec}}$

Fig. 3. The framework of CPAGCN. Note here that $\mathbf{A}^{\text{Training}}$ and $\mathbf{X}$ are respectively the illustrative examples of the corresponding adjacency matrix and node-attribute matrix of the input training graph $\mathcal{G}^{\text{Training}}$.

3.1. Problem statement

Like most existing link prediction methods, we model the given attributed network as an undirected and unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ denotes the nodes set with a size of n, $\mathcal{E} = \{e_{ij}|v_i \in \mathcal{V} \wedge v_j \in \mathcal{V}\}$ denotes the links set with a size of m, $\mathbf{A} = [\mathbf{A}_{ij}]^{n\times n}$ is the adjacency matrix and $\mathbf{X} = [\mathbf{X}_{ip}]^{n\times r}$ is the node-attribute matrix. If $e_{ij} \in \mathcal{E}$, $\mathbf{A}_{ij} = 1$, and $\mathbf{A}_{ij} = 0$ otherwise. Multiple links and self-loops are not allowed here. For the given attributes set $Y = \{y_1, y_2, \ldots, y_r\}$ with a size of r, if v_i has attribute y_p, $\mathbf{X}_{ip} = 1$, and $\mathbf{X}_{ip} = 0$ otherwise. $\mathbf{A}$ and $\mathbf{X}$ are respectively utilized to represent the link information and the attribute information of $\mathcal{G}$.

Generally, let $\mathcal{U}$, $\mathcal{E}$ and $\mathcal{U} - \mathcal{E}$ respectively denote the sets of all possible links, observed links and non-observed links, and the goal of link prediction is to find the links that may appear in the future from $\mathcal{U} - \mathcal{E}$. In practice, the future link information is often unknown, which makes it hard to evaluate the prediction performance. In view of this, we partition $\mathcal{E}$ into two parts randomly according to a preassigned proportion: a training set $\mathcal{E}^{\text{Training}}$ and a test set $\mathcal{E}^{\text{Test}}$. Their corresponding adjacency matrices are respectively denoted as $\mathbf{A}^{\text{Training}}$ and $\mathbf{A}^{\text{Test}}$. Accordingly, the problem concerned here is to learn a link prediction model based on $\mathcal{G}^{\text{Training}} = (V, \mathcal{E}^{\text{Training}}, \mathbf{A}^{\text{Training}}, \mathbf{X})$ and evaluate its performance on $\mathcal{G}^{\text{Test}} = (V, \mathcal{E}^{\text{Test}}, \mathbf{A}^{\text{Test}}, \mathbf{X})$.

3.2. Overview of CPAGCN

To incorporate community structure, we select network embedding-based framework to design CPAGCN. Unlike most existing network embedding-based methods, we do not treat network embedding and link prediction as the separate tasks, but we connect them together organically via the joint training strategy. This can facilitate network embedding to generate more targeted node representations.

Following the above idea, we devise the framework of CPAGCN depicted in Fig. 3. As we can see, CPAGCN comprises two main modules: network embedding and link prediction. Network embedding takes $\mathcal{G}^{\text{Training}}$ as the input, and applies AGCN (Adaptive GCN) to learn the node representation matrix $\mathbf{H} \in \mathbb{R}^{n\times d}$, where d denotes the dimension of node representation. In order to make $\mathbf{H}$ preserve community structure, we apply the BigCLAM overlapping community detection model [31,33] to reconstruct $\mathcal{G}^{\text{Training}}$, and this causes a graph reconstruction loss $\mathcal{L}_{\text{Rec}}$. As the downstream module, link prediction takes $\mathbf{H}$ as the input, and then use a Multi-Layer Perception (MLP) to compute the scores between the representations of nodes. By treating these scores as the probabilities of the existences of links between nodes in $\mathcal{G}^{\text{Training}}$, we obtain the prediction loss $\mathcal{L}_{\text{Pre}}$. In order to obtain the optimal $\mathbf{H}$, AGCN and MLP are jointly trained via the unified loss function $\mathcal{L}$ composed of weighted $\mathcal{L}_{\text{Rec}}$ and $\mathcal{L}_{\text{Pre}}$.

3.3. Network embedding using AGCN

Considering that the generation of links can also be affected by attribute information or link information or both, we utilize the network embedding module to adaptively fuse these two types of information to obtain node representations. To this end, we devise the following AGCN. Specifically, for attribute information, we firstly calculate the attribute similarity matrix $\mathbf{S} = [\mathbf{S}_{ij}]^{n\times n}$ among n nodes, where $\mathbf{S}_{ij}$ is defined as:

$$\mathbf{S}_{ij} = \frac{\mathbf{X}_i \cdot \mathbf{X}_j}{\|\mathbf{X}_i\|\|\mathbf{X}_j\|}. \tag{1}$$

Then we choose top k similar node pairs for each node to set edges, so as to construct a k-nearest neighbor (kNN) graph as the feature graph $\mathcal{G}_{\mathbf{F}} = (\mathcal{V}, \mathcal{E}_{\mathbf{F}}, \mathbf{A}_{\mathbf{F}}, \mathbf{X})$, where $\mathcal{E}_{\mathbf{F}}$ and $\mathbf{A}_{\mathbf{F}}$ respectively denote the corresponding links set and binary adjacency matrix. Finally, we utilize a multi-layer GCN with Mean aggregator (MGCN) to learn hierarchical representation $\mathbf{H}_{\mathbf{F}i}^{(l)}$ of each v_i in $\mathcal{G}_{\mathbf{F}}$ as follows:

$$\mathbf{H}_{\mathbf{F}i}^{(l)} = \sigma(\mathbf{W}^l \text{Mean}(\{\mathbf{H}_{\mathbf{F}i}^{(l-1)}\}\{\mathbf{H}_{\mathbf{F}j}^{(l-1)}, \forall v_j \in \mathcal{N}_{\mathbf{F}}(i)\})), \tag{2}$$

where $\mathbf{W}^l$ denotes weight matrix at the lth layer of MGCN, $\mathcal{N}_{\mathbf{F}}(i)$ denotes the neighbors of v_i in $\mathcal{G}_{\mathbf{F}}$, $\sigma(\cdot)$ is the activation function, and Mean$(\cdot)$ is used to aggregate node representations from neighbors of v_i and itself by averaging component-wise elements among vectors. Note that for $\forall v_i$ in $\mathcal{G}_{\mathbf{F}}$, we set its initial node representation $\mathbf{H}_{\mathbf{F}i}^{(0)}$ to be $\mathbf{X}_i$. Following these steps above, we can finally obtain node representation $\mathbf{H}_{\mathbf{F}} \in \mathbb{R}^{n\times d}$ of $\mathcal{G}_{\mathbf{F}}$.

As for link information, we directly use the original input graph $\mathcal{G}$ as the topology graph $\mathcal{G}_{\mathbf{T}}$. Then the node representation $\mathbf{H}_{\mathbf{T}} \in \mathbb{R}^{n\times d}$ of $\mathcal{G}_{\mathbf{T}}$ can be obtained in the same way as in $\mathcal{G}_{\mathbf{F}}$. To adaptively fuse $\mathbf{H}_{\mathbf{F}}$ and $\mathbf{H}_{\mathbf{T}}$, we use the attention mechanism to learn their corresponding importance $(\alpha_{\mathbf{F}}, \alpha_{\mathbf{T}})$ as follows:

$$(\alpha_{\mathbf{F}}, \alpha_{\mathbf{T}}) = \text{Attention}(\mathbf{H}_{\mathbf{F}}, \mathbf{H}_{\mathbf{T}}), \tag{3}$$

where $\alpha_{\mathbf{F}}, \alpha_{\mathbf{T}} \in \mathbb{R}^{n\times 1}$. For $\mathbf{H}_{\mathbf{F}}$, we firstly transform it through a nonlinear transformation, and then use one shared attention matrix $\mathbf{Q} \in \mathbb{R}^{1\times d'}$ to learn its attention value $\omega_{\mathbf{F}} \in \mathbb{R}^{n\times 1}$ as follows:

$$\omega_{\mathbf{F}} = \tanh(\mathbf{H}_{\mathbf{F}}\mathbf{W}^T + \mathbf{B})\mathbf{Q}^T, \tag{4}$$

where $\mathbf{W} \in \mathbb{R}^{d'\times d}$ and $\mathbf{B} \in \mathbb{R}^{n\times d'}$ are respectively the weight and bias matrices. In this way, we can also obtain the attention value $\omega_{\mathbf{T}} \in \mathbb{R}^{n\times 1}$ of $\mathbf{H}_{\mathbf{T}}$. Using the softmax function we normalize $\omega_{\mathbf{F}}$ to get $\alpha_{\mathbf{F}}$:

$$\alpha_{\mathbf{F}} = \text{softmax}(\omega_{\mathbf{F}}) = \frac{\exp(\omega_{\mathbf{F}})}{\exp(\omega_{\mathbf{F}}) + \exp(\omega_{\mathbf{T}})}. \tag{5}$$

Similarly, $\alpha_{\mathbf{T}} = \text{softmax}(\omega_{\mathbf{T}})$. Let $\boldsymbol{\alpha}_{\mathbf{F}} = \text{diag}(\alpha_{\mathbf{F}}) \in \mathbb{R}^{n\times n}$ and $\boldsymbol{\alpha}_{\mathbf{T}} = \text{diag}(\alpha_{\mathbf{T}}) \in \mathbb{R}^{n\times n}$, the final node representation $\mathbf{H}$ can be obtained by combining weighted $\mathbf{H}_{\mathbf{F}}$ and $\mathbf{H}_{\mathbf{T}}$:

$$\mathbf{H} = \boldsymbol{\alpha}_{\mathbf{F}}\mathbf{H}_{\mathbf{F}} + \boldsymbol{\alpha}_{\mathbf{T}}\mathbf{H}_{\mathbf{T}}. \tag{6}$$

3.4. Community preserving network embedding

To make $\mathbf{H}$ preserve community structure simultaneously, we use BigCLAM overlapping community detection model to constrain $\mathbf{H}$ in reconstructing $\mathcal{G}^{\text{Training}}$. Firstly, we treat d as the number of communities, select $\text{ReLU}(x) = \max(0, x)$ as the activation function of AGCN to ensure the non-negativity of $\mathbf{H}$ and normalize every row vector $\mathbf{H}_i$ in $\mathbf{H}$ with L_2 norm, i.e., $\mathbf{H}_i = \mathbf{H}_i/\|\mathbf{H}_i\|_2$. Through these operations, $\mathbf{H}$ can be treated as the community affiliations matrix and $\mathbf{H}_i$ represents the strength distribution that v_i belongs to d communities. Next, following the graph generation idea of BigCLAM, we respectively define the probabilities of $e_{ij} \in \mathcal{E}^{\text{Training}}$ and $e_{ij} \notin \mathcal{E}^{\text{Training}}$ as

$$P(e_{ij} \in \mathcal{E}^{\text{Training}}) = 1 - \exp(-\mathbf{H}_i\mathbf{H}_j^T), \tag{7}$$

$$P(e_{ij} \notin \mathcal{E}^{\text{Training}}) = \exp(-\mathbf{H}_i\mathbf{H}_j^T). \tag{8}$$

These definitions above suggest that: for any two nodes v_i and v_j, the more similar their community affiliations representations $\mathbf{H}_i$ and $\mathbf{H}_j$, the more likely they will be linked. This also means that community structure can be used to generate the graph. To this end, the generation of $\mathcal{G}^{\text{Training}}$ is modeled as

$$\begin{aligned} P(\mathcal{G}^{\text{Training}}|\mathbf{H}) = & \prod_{e_{ij}\in\mathcal{E}^{\text{Training}}} P(e_{ij} \in \mathcal{E}^{\text{Training}}) \\ & \times \prod_{e_{ij}\notin\mathcal{E}^{\text{Training}}} P(e_{ij} \notin \mathcal{E}^{\text{Training}}). \end{aligned} \tag{9}$$

Maximizing $P(\mathcal{G}^{\text{Training}}|\mathbf{H})$ is able to reconstruct the link structure of $\mathcal{G}^{\text{Training}}$ based on $\mathbf{H}$. Therefore, $1/P(\mathcal{G}^{\text{Training}}|\mathbf{H})$ is regarded as the reconstruction loss $\mathcal{L}_{\text{Rec}}$ of $\mathcal{G}^{\text{Training}}$ and applied to train AGCN. For ease of computation, we use the following log-likelihood of $1/P(\mathcal{G}^{\text{Training}}|\mathbf{H})$ as $\mathcal{L}_{\text{Rec}}$:

$$\begin{aligned} \mathcal{L}_{\text{Rec}} = & -\sum_{e_{ij}\in\mathcal{E}^{\text{Training}}} \ln(1 - \exp(-\mathbf{H}_i\mathbf{H}_j^T)) \\ & + \sum_{e_{ij}\notin\mathcal{E}^{\text{Training}}} \mathbf{H}_i\mathbf{H}_j^T. \end{aligned} \tag{10}$$

Real-world attributed networks are often very sparse, which means that the second term may dominate the loss and even largely increase the time complexity. To alleviate these problems, from $\{e_{ij}|e_{ij} \notin \mathcal{E}^{\text{Training}}\}$ we can randomly sample the same number of links as $\mathcal{E}^{\text{Training}}$. By minimizing $\mathcal{L}_{\text{Rec}}$ to train AGCN, AGCN becomes goal-oriented and its optimization process can drive it to learn node representations preserving community structure.

3.5. Link prediction

Similar to most network embedding based methods, we predict an unobserved link by estimating the probability of its existence. This can be achieved by computing the score of the representations of its incident nodes, such as the most common similarity score. Here, to obtain this score in an automatic learning manner, we adopt a Multi-Layer Perception (MLP) which takes the concatenation of a pair of node representations as the input and produces the scalar score. More specifically, for a pair of nodes (v_i, v_j), we first concatenate their node representations $\mathbf{H}_i$ and $\mathbf{H}_j$, and then feed the result into the following MLP to obtain the score s_{ij}:

$$s_{ij} = \text{MLP}(\mathbf{H}_i \parallel \mathbf{H}_j). \tag{11}$$

Let $\mathcal{E}^{\text{Training}}$ be the positive samples $\mathcal{E}^{\text{P}}$ and the same number of links extracted from non-observed links be the negative samples $\mathcal{E}^{\text{N}}$, then we can use MLP to respectively predict the score of every link in $\mathcal{E}^{\text{P}}$ and $\mathcal{E}^{\text{N}}$, and the prediction loss $\mathcal{L}_{\text{Pre}}$ can be computed as:

$$\mathcal{L}_{\text{Pre}} = -\frac{1}{|\mathcal{E}^{\text{P}}| + |\mathcal{E}^{\text{N}}|}\Big(\sum_{e_{ij}\in\mathcal{E}^{\text{P}}} \ln s_{ij}^{\text{P}} + \sum_{e_{pq}\in\mathcal{E}^{\text{N}}} \ln(1 - s_{pq}^{\text{N}})\Big). \tag{12}$$

By minimizing $\mathcal{L}_{\text{Pre}}$, MLP is optimized to learn the scores of pairwise node representations in a nonlinear way. This can be expected to perform better than some widely used linear methods, such as the dot product method that directly computes $\mathbf{H}_i\mathbf{H}_j^T$ as s_{ij}.

3.6. The unified loss function

As described above, network embedding and link prediction modules are connected through the intermediate node representations. To further help network embedding learn the node representations that are more beneficial to the subsequent link prediction task, we jointly train AGCN and MLP in these two modules instead of training them separately. To this end, we combine the graph reconstruction loss $\mathcal{L}_{\text{Rec}}$ with the prediction loss $\mathcal{L}_{\text{Pre}}$ to construct the following unified loss function:

$$\mathcal{L} = \mathcal{L}_{\text{Pre}} + \alpha\mathcal{L}_{\text{Rec}}, \tag{13}$$

where α is a positive hyper-parameter used for adjusting the contribution of $\mathcal{L}_{\text{Rec}}$, and its effect on the performance will be analyzed in our experiments. By minimizing $\mathcal{L}$ to jointly train AGCN and MLP, they can be expected to enhance each other, and meanwhile CPAGCN becomes an end-to-end deep learning method for link prediction in attributed networks.

3.7. Algorithm and complexity analysis

To analyze the efficiency of CPAGCN in theory, we briefly outline the main operation process of CPAGCN in Algorithm 1.

Algorithm 1: CPAGCN link prediction algorithm

```
Input: Training attributed networks $\mathcal{G}^{\text{Training}}$,
       Number of communities $d$, Maximum iterations MaxIter,
       $k$, $\alpha$;
Output: Node representation matrix $\mathbf{H}$, learned parameters in
        AGCN and MLP;
1 Construct the feature graph $\mathcal{G}_{\mathbf{F}}$ based on attributed inforamtion;
2 Initialize AGCN and MLP respectively;
3 for $t = 1...MaxIter$ do
4     Generate $\mathbf{H}$ with AGCN;
5     Calculate $\mathcal{L}_{\text{Rec}}$;
6     Feed $\mathbf{H}$ into MLP to predict links in $\mathcal{G}^{\text{P}}$ and $\mathcal{G}^{\text{N}}$;
7     Calculate $\mathcal{L}_{\text{Pre}}$ and $\mathcal{L}$;
8     Back propagation and update learned parameters in AGCN
        and MLP, respectively;
9 return $\mathbf{H}$, learned parameters in AGCN and MLP;
```

Algorithm 1 mainly involves the training of AGCN and MLP, and when the training finishes, we can use Eq. (11) to make the link prediction. As for its time complexity, Line 1 involves constructing a kNN graph using Eq. (1). If we use a minimum heap to store the top k nearest vectors, its time complexity is $\mathcal{O}(n(r + \log k))$. As for the iterative part from Line 4 to Line 8, let L and K respectively denote the numbers of the layers in AGCN and MLP, and the dimension settings of layers in AGCN and MLP are respectively $r, d_1, d_2, \ldots, d_{L-1}, d$ and $2d, s_1, s_2, \ldots, s_{K-1}, 1$. We can compute the time complexity line by line. Line 4 involves the forward propagation of AGCN and its time complexity is $\mathcal{O}(nrd_1d_2...d_{L-1}d)$. By sampling the same number of nonexistent links as $\mathcal{E}^{\text{Training}}$, the time complexity of Line 5 can be reduced to $\mathcal{O}(|\mathcal{E}^{\text{Training}}|d)$. Taking links in $\mathcal{G}^{\text{P}}$ and $\mathcal{G}^{\text{N}}$ as the input, the time complexity of MLP in Line 6 is $\mathcal{O}(|\mathcal{E}^{\text{Training}}|ds_1s_2...s_{K-1})$. Line 7 includes the computations of $\mathcal{L}_{\text{Pre}}$ and $\mathcal{L}$ and its time complexity is obviously $\mathcal{O}(|\mathcal{E}^{\text{Training}}|)$. As we know, many optimization algorithms (e.g., Adam) and strategies (e.g., dropout) often can make the back propagation algorithm obtain the linear time complexity with respect to the number of input samples. Therefore, we deduce that the time complexity of Line 8 is linearly proportional to $n + |\mathcal{E}^{\text{Training}}|$. In total, the approximate time complexity of CPAGCN is $\mathcal{O}(n(r+\log k)+MaxIter(nrd_1d_2...d_{L-1}d+|\mathcal{E}^{\text{Training}}|ds_1s_2...s_{K-1}))$. By controlling the numbers of layers in AGCN and MLP, the time cost of CPAGCN will be dominated by the number of nodes and links in $\mathcal{G}^{\text{Training}}$ and can be expected to perform efficiently.

4. Experiments

In this section, we evaluate our proposed method CPAGCN and its competitors on six real-world attributed networks introduced in Section 2, and conduct related experiments on a PC with 64-bit Windows 10 system, 3.5 GHz Intel Core i9-11900K CPU and 64 GB RAM. All methods are implemented using Python 3.7. Especially, we utilize Deep Graph Library (DGL)[2] to develop graph neural networks.

4.1. Baselines

Lots of works on link prediction in attributed networks have proved that methods that can integrate link information and attribute information often perform better than those using only one type of information. Therefore, in order to avoid repeating existing comparison work as much as possible, here we only select methods which can simultaneously utilize two types of information as baselines. These baselines comprise network embedding-based methods and learning-based methods mentioned in Section 1, and their details are as follows.

The network embedding-based methods include:

- TADW [26]. TADW is based on the matrix factorization framework and can incorporate attribute information into the node representation learning. In theory, it is proved to be equivalent to the text-associated DeepWalk [34].
- HSCA [35]. Similar to TADW, HSCA is also based on matrix factorization framework. To obtain node representations, it proposes to augment homophily, topology structure and node content information sources into one learning objective function.
- ANEM [36]. ANEM is a Nonnegative Matrix Factorization (NMF)-based network embedding method. In addition to attribute and lower-order proximities, it also can preserve community structure by introducing Modularity maximization or BigCLAM community detection model.
- GCN [37]. GCN is a convolutional neural network extended to graph data. It derives node representations by utilizing graph-based spectral convolution and neighborhood aggregation.
- GraphSAGE [38]. GraphSAGE is an inductive network embedding method. It is a GCN variant with sampling strategy and can learn node representations by aggregating features from selected neighbors.
- GAT [39]. GAT is based on graph attention networks. By introducing the attention mechanism, it computes the representation of each node by attending over its neighbors differentially.
- VGAE [28]. VGAE is composed of a GCN encoder combining with variational Bayesian inference model, and a simple inner product decoder. It learns the node representations via minimizing the graph reconstruction error.
- GIN [40]. GIN is based on graph isomorphism networks. At every layer, it aggregates features of neighbor nodes by summation, and uses MLP to learn the final node representations.
- AGE [41]. AGE adopts the graph autoencoder framework. It introduces the Laplacian smoothing filter into its encoder, and applies a novel adaptive learning strategy to train node representations.

The learning-based methods include:

- SEAL [20]. SEAL converts the link prediction task to a graph classification problem, and solves it using an end-to-end graph neural networks architecture.
- LGLP [21]. Unlike SEAL, LGLP treats the link prediction task as a node classification problem in line graphs, and addresses it using line graph neural networks.
- EdgeConvNorm (ECN) [42]. ECN also treats the link prediction task as a node classification problem, and utilizes GNNs with stacking edge convolution manipulations to solve it.
- CPGCN. CPGCN is a simplified version of CPAGCN. It does not apply AGCN to fuse link information and attribute information, but directly utilize MGCN to learn node representations from $\mathcal{G}^{\text{Training}}$ by treating node attribute vectors as the initial node representations.

Note that for all network embedding-based baselines here, we select their widely used dot product as the subsequent link prediction model. Besides, we use the prediction loss shown in Eq. (12) to train GCN, GraphSAGE, GAT and GIN, which can help them to obtain better performance in the context of link prediction.

4.2. Experimental setup

4.2.1. Parameter settings

For fair comparison, the key parameters of every method are set to their optimal values. Specifically, for TADW, we set the maximum order of the transition matrix to be 2. For ANEM, we set the number of communities to the ground-truth number, and select BigCLAM as its community detection model for better performance. For GraphSAGE, we set the search depth to 2 and use the Mean aggregator. For GAT, the number of attention heads is 3. For GIN, ϵ is set to 0. For SEAL, we extract 2-hop enclosing subgraphs and the ratio of the sort pooling layer is set to 0.6. For LGLP, we also extract 2-hop enclosing subgraphs. For our method CPAGCN, we respectively set the layer configurations of AGCN and MLP to n-256-d and 2d-d-1, $k = 10$ and $\alpha = 10$.

Note that for GCN, GraphSAGE, GAT, VGAE, GIN, AGE, SEAL, LGLP amd ECN, their layer configurations of graph neural networks are set to the same: n-256-64. For every method, the experiments are repeated for ten times using its optimal settings and the average results are reported here.

4.2.2. Dataset divisions and evaluation metrics

To test the performance of every method, we respectively randomly select 10%, 20% and 80% links in $\mathcal{E}$ as positive training samples, and the rest (90%, 80% and 20%) are treated as positive test samples. Besides, the same number of non-observed links in $\mathcal{U} - \mathcal{E}$ are randomly selected as negative samples for both training and testing.

In terms of evaluation metrics, we employ two widely used measures: area under the curve (AUC) and average precision (AP) [3,4], where AUC $\in [0, 1]$ and AP $\in [0, 1]$. Larger AUC and AP values indicate better performance.

[2] https://www.dgl.ai.

Table 3
AUC comparison with different training set sizes (TSS%). Bold numbers denote the best results.

Dataset	TSS%	TADW	HSCA	ANEM	GCN	GraphSAGE	GAT	VGAE	GIN	AGE	SEAL	LGLP	ECN	CPGCN	CPAGCN
Cora	10%	0.535	0.541	0.574	0.621	0.629	0.704	0.676	0.703	0.535	0.619	0.697	0.661	0.728	**0.765**
	20%	0.552	0.562	0.578	0.766	0.717	0.765	0.762	0.769	0.699	0.641	0.761	0.683	0.777	**0.803**
	80%	0.556	0.567	0.759	0.930	0.887	0.954	0.918	0.894	0.857	0.884	0.943	0.788	0.971	**0.972**
PubMed	10%	0.577	0.583	0.623	0.863	0.696	0.854	0.857	0.856	0.861	0.779	0.828	0.836	0.869	**0 892**
	20%	0.617	0.626	0.668	0.867	0.732	0.884	0.891	0.864	0.885	0.827	0.870	0.833	0.929	**0.937**
	80%	0.661	0.677	0.678	0.951	0.873	0.916	**0.964**	0.872	0.914	0.949	0.955	0.837	0.958	0.962
Fb1684	10%	0.565	0.576	0.869	0.860	0.767	0.732	0.865	0.863	0.834	0.809	0.881	0.839	0.891	**0.893**
	20%	0.581	0.589	0.914	0.916	0.809	0.862	0.905	0.880	0.857	0.910	0.908	0.844	0.926	**0.928**
	80%	0.692	0.711	0.950	0.937	0.845	0.943	0.946	0.890	0.873	**0.951**	0.940	0.896	0.948	0.950
Fb1912	10%	0.540	0.551	0.942	0.945	0.887	0.944	0.951	0.942	0.835	0.944	0.947	0.908	0.960	**0.961**
	20%	0.593	0.607	0.952	0.951	0.889	0.952	0.955	0.951	0.864	0.946	0.951	0.918	0.968	**0.971**
	80%	0.698	0.712	0.964	0.964	0.918	0.963	0.973	0.957	0.896	0.972	0.962	0.931	0.976	**0.977**
CS	10%	0.502	0.518	0.885	0.835	0.742	0.790	0.756	0.777	0.751	0.801	0.823	0.831	0.893	**0.904**
	20%	0.570	0.583	0.892	0.915	0.892	0.904	0.908	0.897	0.901	0.889	0.905	0.858	0.913	**0.932**
	80%	0.582	0.597	0.901	0.963	0.946	0.958	0.963	0.945	0.933	0.958	0.963	0.885	0.972	**0.974**
Eng	10%	0.541	0.552	0.845	0.862	0.850	0.835	0.882	0.839	0.819	0.649	0.837	0.836	0.903	**0.922**
	20%	0.559	0.566	0.898	0.908	0.894	0.903	0.910	0.905	0.902	0.691	0.910	0.871	0.914	**0.940**
	80%	0.562	0.575	0.913	0.958	0.935	0.955	0.961	0.963	0.930	0.954	0.933	0.898	0.967	**0.969**

Table 4
AP comparison with different training set sizes (TSS%). Bold numbers denote the best results.

Dataset	TSS%	TADW	HSCA	ANEM	GCN	GraphSAGE	GAT	VGAE	GIN	AGE	SEAL	LGLP	ECN	CPGCN	CPAGCN
Cora	10%	0.545	0.550	0.579	0.612	0.613	0.734	0.700	0.732	0.532	0.633	0.726	0.638	0.754	**0.784**
	20%	0.578	0.587	0.586	0.747	0.708	0.802	0.770	0.779	0.675	0.669	0.802	0.678	0.811	**0.864**
	80%	0.586	0.599	0.779	0.942	0.870	0.959	0.901	0.895	0.845	0.899	0.957	0.745	0.968	**0.969**
PubMed	10%	0.591	0.607	0.632	0.862	0.687	0.861	0.854	0.851	0.860	0.762	0.832	0.808	0.868	**0.892**
	20%	0.610	0.622	0.676	0.888	0.728	0.897	0.897	0.865	0.876	0.820	0.877	0.827	0.926	**0.929**
	80%	0.634	0.641	0.712	0.958	0.851	0.908	**0.967**	0.869	0.905	0.953	0.963	0.887	0.962	0.964
Fb1684	10%	0.547	0.553	0.832	0.831	0.727	0.703	0.863	0.825	0.816	0.836	0.874	0.817	0.891	**0.895**
	20%	0.557	0.572	**0.908**	0.904	0.771	0.836	0.888	0.847	0.836	0.901	0.897	0.832	0.903	0.904
	80%	0.634	0.646	0.928	0.927	0.798	0.921	0.920	0.858	0.859	0.922	0.921	0.869	0.932	**0.934**
Fb1912	10%	0.553	0.566	0.935	0.925	0.849	0.936	0.928	0.923	0.861	0.934	0.928	0.899	0.952	**0.953**
	20%	0.582	0.595	0.929	0.931	0.856	0.931	0.938	0.929	0.882	0.943	0.939	0.906	0.959	**0.961**
	80%	0.670	0.688	0.962	0.949	0.886	0.950	0.966	0.937	0.906	**0.972**	0.949	0.909	0.970	0.971
CS	10%	0.519	0.531	0.855	0.834	0.827	0.832	0.858	0.839	0.831	0.696	0.823	0.831	0.892	**0.905**
	20%	0.547	0.559	0.902	0.906	0.880	0.908	0.896	0.908	0.902	0.770	0.895	0.866	0.917	**0.935**
	80%	0.590	0.610	0.921	0.966	0.926	0.960	0.958	0.951	0.922	**0.970**	0.975	0.892	0.964	0.966
Eng	10%	0.563	0.576	0.876	0.869	0.834	0.861	0.855	0.871	0.869	0.670	0.853	0.868	0.901	**0.903**
	20%	0.571	0.585	0.893	0.906	0.884	0.901	0.900	0.909	0.901	0.744	0.902	0.880	0.916	**0.926**
	80%	0.578	0.591	0.921	0.932	0.926	0.939	0.927	0.943	0.909	0.937	0.926	0.913	0.955	**0.956**

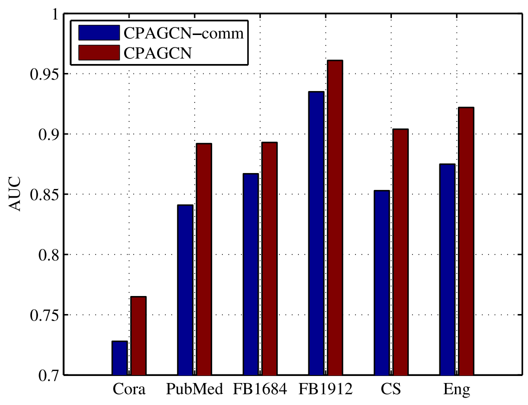
4.3. Experimental results and analysis

4.3.1. Comparison results with baselines

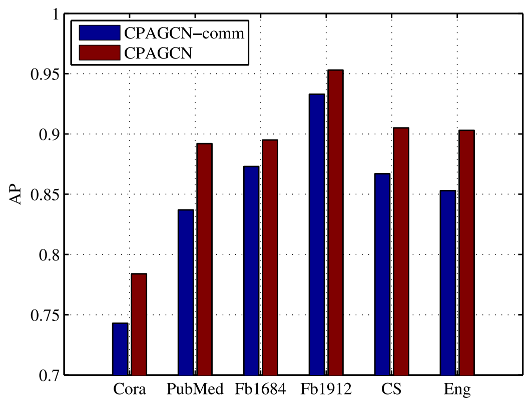
We first compare CPAGCN with baseline methods on every dataset. The results in terms of AUC and AP are respectively shown in Tables 3 and 4, from which we have the following observations:

- On each network, CPAGCN and its variant CPGCN show superior performance compared to other existing methods in most cases, especially when the training set size is small. For instance, on Cora with a training set size of 10%, CPAGCN and CPGCN demonstrate a notable increase in AUC and AP of at least 8.66% and 6.81%, and 3.41% and 2.72%, respectively, compared to other methods. Similar results can be observed on other networks. Moreover, CPAGCN and CPGCN perform remarkably well on networks with overlapping communities (i.e., Fb1684, Fb1912, CS, and Eng), particularly when the number of training samples is limited. We notice that ANEM, which also takes community structure into account, shows the same characteristic of achieving satisfactory performance with fewer training samples on networks with overlapping communities. This may be due to the fact that nodes in these networks are more likely to share communities, which increases their likelihood of creating links.
- CPAGCN performs better than CPGCN, especially on sparser networks with fewer training samples. For example, on Cora with 10% training set size, the AUC and AP scores of CPAGCN respectively increase by 5.08% and 3.98% compared to CPGCN. There are similar results on PubMed, CS and Eng networks. Considering that CPGCN does not use AGCN, we have reason to believe that AGCN used in CPAGCN is useful: it can fully fuse link information and attribute information to further improve the performance of link prediction, especially when the available information is not sufficient.

Overall, the comparison results above demonstrate the advantages of CPAGCN, i.e., confirming the effectiveness of incorporating community structure and introducing the mechanism of adaptive information fusion to boost the performance of link prediction in attributed networks. In the following Sections 4.3.2 and 4.3.3, we will further confirm these through the ablation study and robust test.

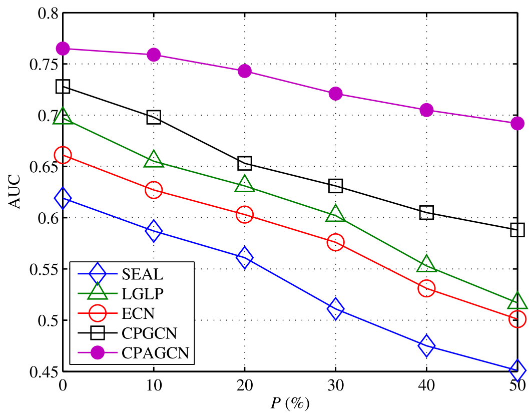


(a) AUC

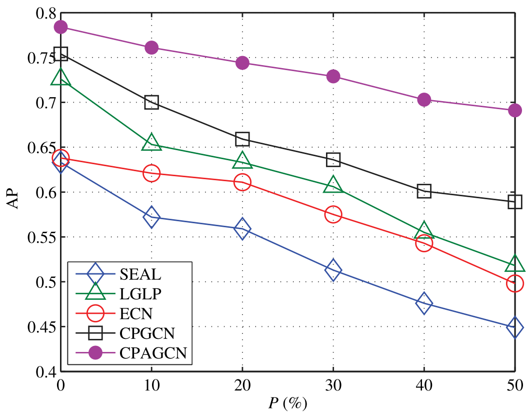


(b) AP

Fig. 4. Performance comparison between CPAGCN-comm and CPAGCN.



(a) AUC



(b) AP

Fig. 5. Performance comparison on Cora with different ratios of exchanged attribute vectors.

4.3.2. *Ablation study*

In this study, we investigate the impact of incorporating community structure on the performance of link prediction in attributed networks. To this end, we introduce a variant: CPAGCN-comm, which does not consider community structure by setting the hyperparameter α of $\mathcal{L}$ to 0 in Eq. (13). We then compare CPAGCN-comm with our proposed method CPAGCN on six real attributed networks, all of which have 10% training samples. Our results indicate that CPAGCN outperform CPAGCN-comm significantly, particularly on the sparser networks. For instance, on Cora, the AUC and AP scores of CPAGCN respectively improve by 5.29% and 5.52% compared to CPAGCN-comm. Similar observations are made on PubMed, CS, and Eng networks (see Fig. 4). Overall, these findings further demonstrate the positive influence of community structure on link prediction in attributed networks and the efficacy of CPAGCN.

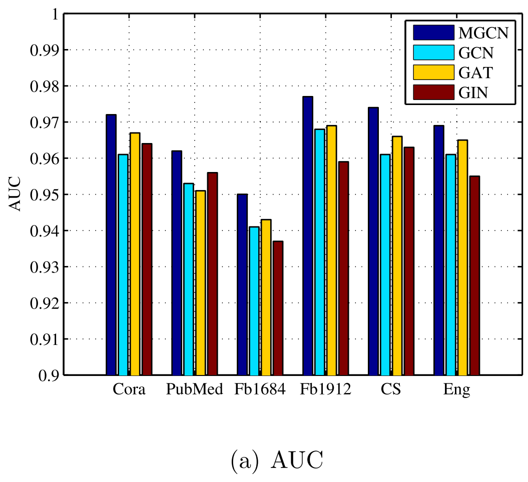
4.3.3. *Robust test*

CPAGCN has the ability to fuse link information and attribute information adaptively, which makes it be robust against noises. To test this, we first randomly select P% nodes to exchange attribute vectors with each other. This makes any two connected nodes not necessarily have similar attributes, i.e., artificially adding noises into the graph. Then, we vary P from 0 to 50 with a step size of 10, and compare CPAGCN with other four learning-based baselines: SEAL, LGLP, ECN and CPGCN. We select Cora as the test dataset and the results are shown in Fig. 5. As we can see, when P increases, the performance of all methods tend to decrease. However, CPAGCN still performs better than other methods for any value of P. Besides, CPAGCN does not deteriorate severely like other methods: when P varies from 0 to 50, the AUC and AP scores of CPAGCN respectively decrease by 9.54% and 11.2%, but those of other methods at least respectively decrease by 19.2% and 21.8%. These results show that CPAGCN is more robust against noises than other methods. The ability of adaptive information fusion can help CPAGCN to exploit useful information as much as possible. Instead, SEAL, LGLP, ECN and CPGCN all directly take attribute vectors as the initial input of GNNs used in them, and hence noisy attribute vectors is easy to disturb their learning process, resulting in poorer performance.

4.3.4. *Comparison results with its variants*

In the framework of CPAGCN, MGCN, BigCLAM and MLP models can be seamlessly replaced with other similar models. This feature allows CPAGCN to have multiple potential variants. Here, we compare CPAGCN with the following three types of variants on every dataset with 80% training samples:

(1) Variants using other GNNs. Here, we respectively select GCN, GAT and GIN to replace MGCN in CPAGCN and use the same parameter settings suggested in Section 4.2.1. The comparison results in term of AUC and AP are shown in Fig. 6. We find that CPAGCN using MGCN performs a little better than using GCN,



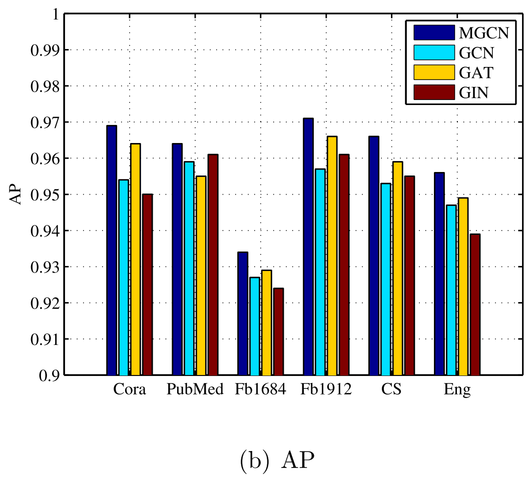
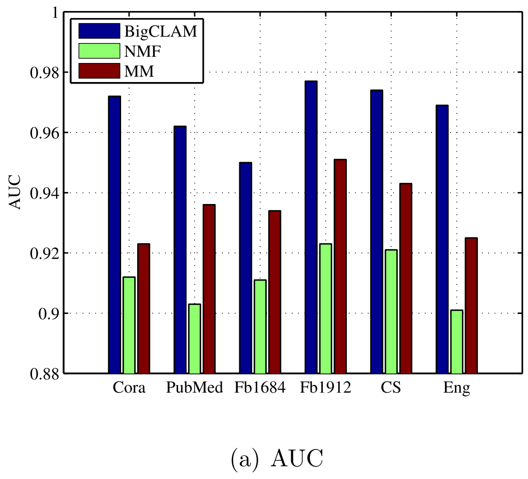


Fig. 6. Comparison with different GNNs.



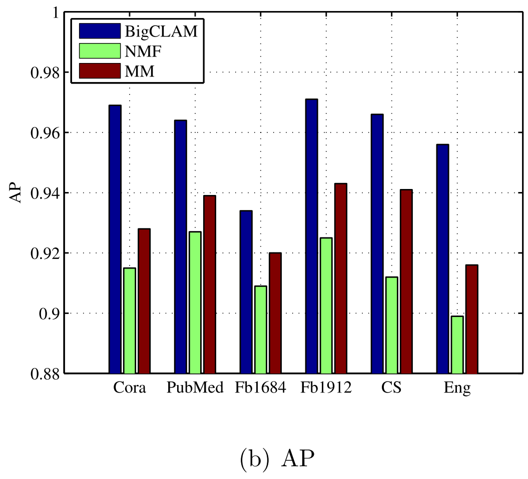


Fig. 7. Comparison with different community detection models.

GAT and GIN. More importantly, MGCN can help CPAGCN become more efficient, especially on larger networks with more training samples, which will be specially analyzed in Section 4.3.7.

(2) Variants using other community detection models. Recently, both NMF and Modularity Maximization (MM) are widely applied in community detection, and hence we try to apply them to replace BigCLAM. To this end, we respectively redefine $\mathcal{L}_{\text{Rec}}$ as the following objective functions of symmetric NMF and MM models:

$$\mathcal{L}_{\text{Rec}} = \mathcal{L}_{\text{NMF}} = \|\mathbf{A}^{\text{Training}} - \mathbf{H}\mathbf{H}^T\|_F^2, \tag{14}$$

$$\mathcal{L}_{\text{Rec}} = \mathcal{L}_{\text{MM}} = -tr(\mathbf{H}^T\mathbf{M}\mathbf{H}), \tag{15}$$

where $\mathbf{M} = [\mathbf{M}_{ij}]^{n\times n}$ is the modularity matrix, whose element $\mathbf{M}_{ij} = \mathbf{A}_{ij}^{\text{Training}} - \frac{d_i d_j}{2|\mathcal{E}^{\text{Training}}|}$ and d_i (d_j) is the degree of v_i (v_j). The comparison results are presented in Fig. 7. As we can see, CPAGCN using BigCLAM performs much better than using NMF and MM. This shows that BigCLAM has better community detection ability, which is more beneficial to the link prediction. Moreover, both NMF and MM have quadratic time complexity $\mathcal{O}(n^2)$ due to more dense matrix multiplication operations.

(3) Variants using other prediction score computation models. Essentially, the prediction score computation here is the similarity computation between node representations. In view of this, we respectively use two widely used models, dot product (DP) and cosine similarity (Cosine), to replace MLP in CPAGCN, and their comparison results are shown in Fig. 8. It can be observed that CPAGCN using MLP obviously performs much better. This may be because MLP can obtain the prediction scores more accurately through the joint learning with MGCN, which DP and Cosine cannot achieve.

The comparison results above demonstrate that the current models used in CPAGCN are more effective. Meanwhile, this also means that CPAGCN provides an open framework that allows to explore more powerful models to replace MGCN, BigCLAM and MLP, helping to further improve its performance.

4.3.5. Joint optimization analysis

CPAGCN trains AGCN and MLP jointly, instead of separately. This can help AGCN to learn node representations that are more beneficial to the link prediction. To demonstrate the advantages of this strategy, on every dataset with 80% training samples we conduct the comparative analysis by respectively training AGCN and MLP jointly and separately. Note that when adopting the separate training strategy, AGCN and MLP are respectively trained using $\mathcal{L}_{\text{Rec}}$ and $\mathcal{L}_{\text{Pre}}$. The results are shown in Fig. 9, from which we can find that the joint training strategy is significantly superior to the separate training strategy. The AUC and AP scores improve by 0.12 and 0.11 on average, respectively. The major reason is that the joint training can help AGCN produce more targeted node representations $\mathbf{H}$.

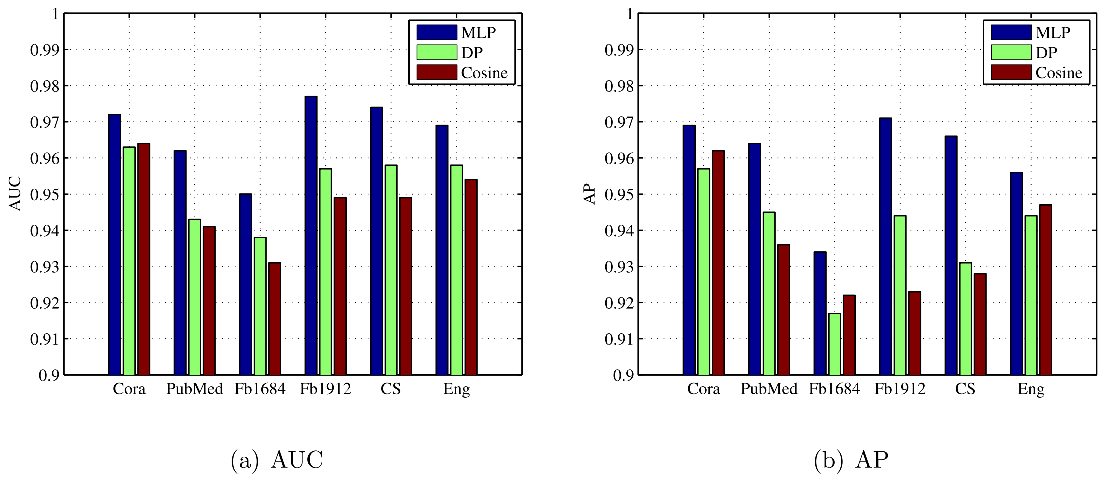


Fig. 8. Comparison with different prediction score computation models.

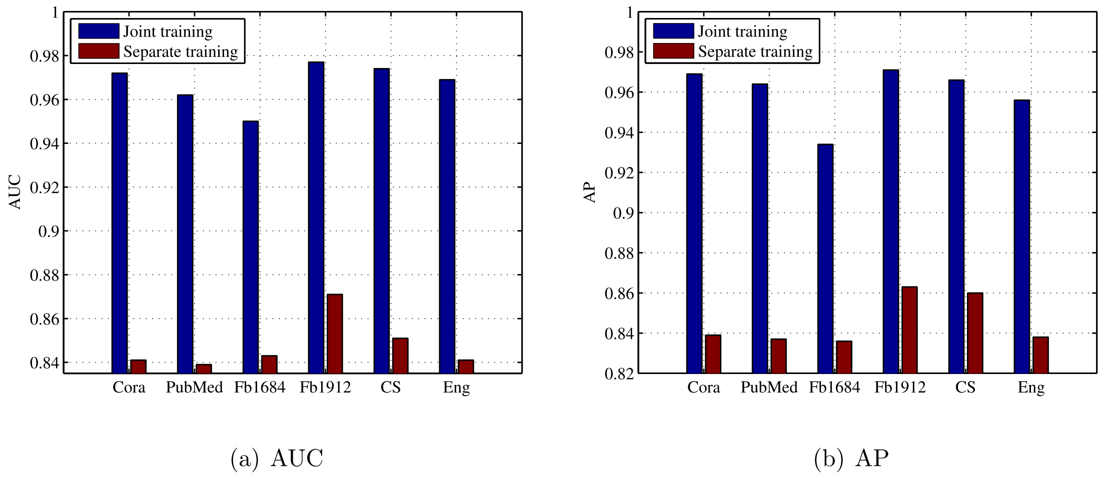


Fig. 9. Joint training vs. separate training.

4.3.6. Parameter sensitivity analysis

There are two hyperparameters in CPAGCN: k used in constructing the nearest neighbor graph based on attribute information and α used to balance the contribution of the graph reconstruction loss $\mathcal{L}_{\text{Rec}}$ in Eq. (13). To study their effects on the performance of CPAGCN, we respectively conduct the sensitivity analysis.

Firstly, we set α to 10 and vary k from 5 to 20 with a step size of 5, and test CPAGCN on all networks with 80% training samples. The results are shown in Fig. 10, from which we can find that on every network CPAGCN obtains the best performance when $k = 10$. The larger the k, the more likely it will introduce noises into $\mathcal{G}_{\mathbf{F}}$, resulting in the performance degradation. To obtain the stable and satisfactory performance, we set k to 10 on every network.

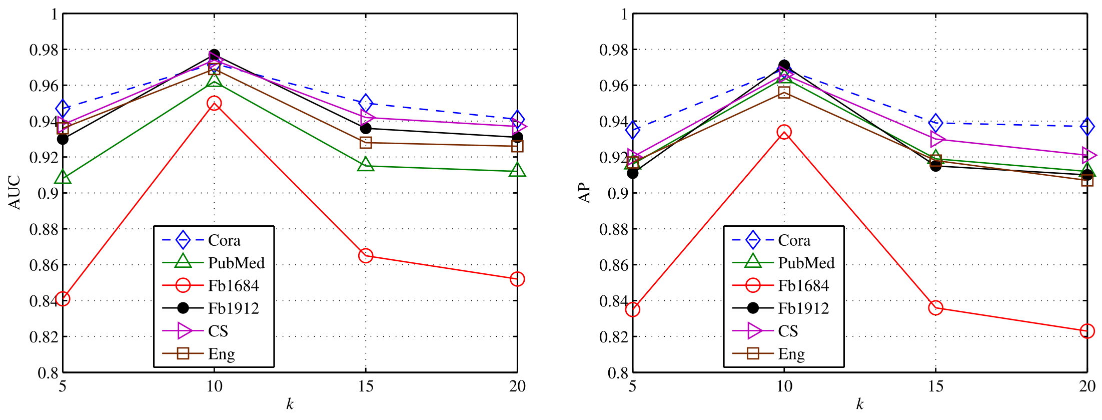
Secondly, we set k to 10 and vary α in the range of $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$, and test CPAGCN on all networks with 80% training samples. The results are presented in Fig. 11 and it can be seen that: on every network the AUC and AP scores of CPAGCN all begin to rise up with the increase of α, but begin to drop when α is over 10. Therefore, to consistently achieve the best performance, we set α to 10 in all experiments.

4.3.7. Running efficiency analysis

To test the efficiency of CPAGCN, we run CPAGCN on two relatively large-scale networks: CS and Eng, and the training set size is uniformly set to 80%. We compare CPAGCN against SEAL, LGLP, and ECN, which are also capable of simultaneously conducting network embedding and link prediction. We analyze the number of iterative convergences required by each method, and the results are presented in Fig. 12. It is evident that ECN requires the most iterations to converge, while CPAGCN requires almost the same number of iterations as SEAL and LGLP, ranging from 15 to 20 iterations.

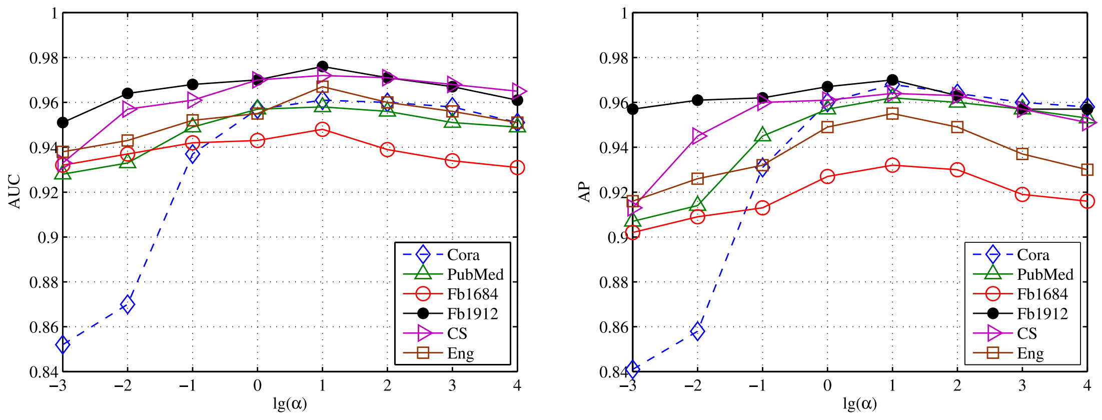
We further compare the running time before convergence and the results are presented in Fig. 13(a). We can find that CPAGCN run much faster than ECN, and is slightly inferior to SEAL and LGLP due to considering the community structure and the adaptive information fusion additionally. As analyzed in Section 3.7, the time complexity of CPAGCN is linearly proportional to n and m. In other words, for a given network, the time cost is linearly related to the training set size. To verify this, on CS and Eng we vary the training set size from 10% to 90% with the fixed step length of 10% and calculate the running time of 500 iterations of CPAGCN. As can be seen, the results shown in Fig. 13(b) well support the previous conclusion.

Overall, although CPAGCN needs to incorporate community structure, and fuse link information and attribute information, its AGCN and BigCLAM models are quite efficient, which in turn helps CPAGCN perform efficiently.



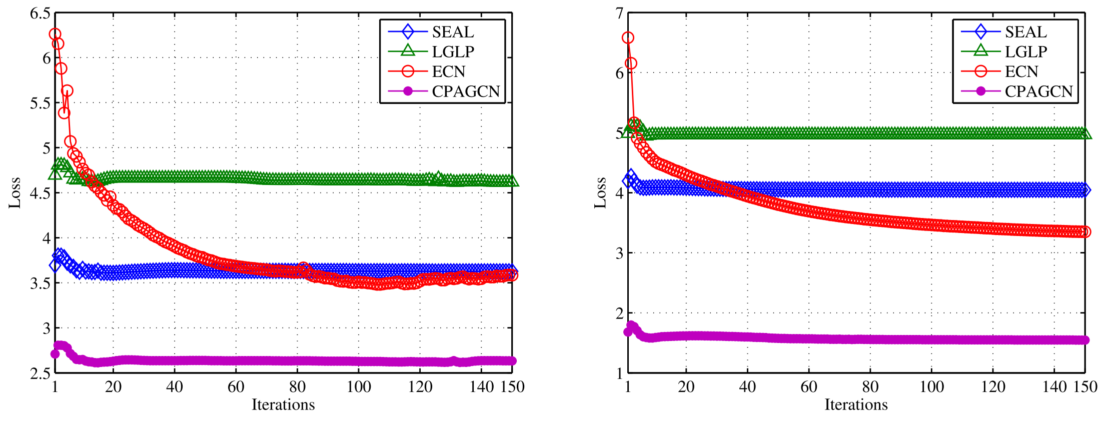
(a) AUC (b) AP

Fig. 10. Performance with different k.



(a) AUC (b) AP

Fig. 11. Performance with different α.

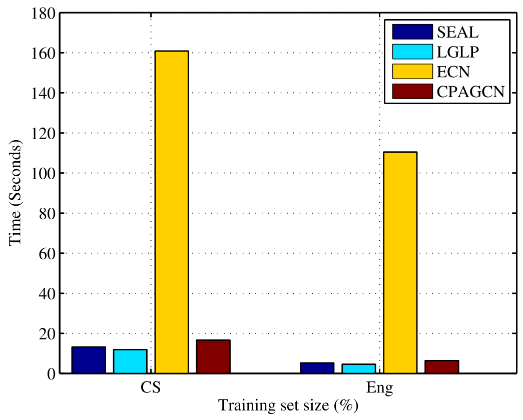


(a) CS (b) Eng

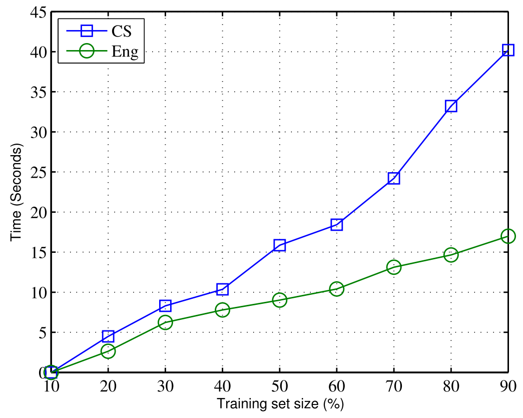
Fig. 12. The number of iterative convergence.

5. Related work

As a long standing network analysis task, link prediction has attracted a lot of attention. In the past, many similarity-based methods are widely used, such as Common Neighbors (CN), Katz index and SimRank [5,6]. These methods manually design similarity metrics based on assumptions on when links may exist, but these assumptions are not always correct for all real-world networks, thus limiting their scope of application [20]. More recently, network embedding-based methods become more



(a) Convergence time



(b) Running time with different training set sizes. For ease of presentation, the running time here is the incremental part relative to that at 10%.

Fig. 13. Running time analysis.

popular. They can automatically learn low-dimensional node representations by using specific models and these representations can well preserve low-order and even high-order proximities, which are very beneficial to predicting links. Various network embedding-based methods have been proved to be effective, including matrix factorization-based methods HSCA [35], FSSDNMF [43], NetSMF [44], M-NMF [45], ICP [46], MLjFE [47] and TADW [26], random walk-based methods DeepWalk [34], node2vec [48] and PMIPJ [49], and deep learning-based methods VGAE [28], GraphSAGE [38], GCN [37], GAT [39], GIN [40] and AGE [41].

Network embedding-based methods undoubtedly can provide more promising solutions to link prediction. However, most existing network embedding-based methods do not simultaneously consider integrating link information, attribute information and community structure. For example, NetSMF, DeepWalk and node2vec cannot deal with attribute information. TADW, GraphSAGE, VGAE, GCN, GAT, GIN, AGE and MTSN [50] can combine link information and attribute information, but they ignore community structure. As the first network embedding method preserving community structure, M-NMF is only applicable to networks without attribute information. Although ANEM [36] and NEC [51] overcome the problems faced by these methods, ANEM is based on the NMF framework and needs to set the hyperparameters to balance the contributions of link information and attribute information, which requires extra tuning cost, and NEC is not an end-to-end method for link prediction.

Due to more powerful ability of representation learning, GNNs has become increasingly popular for link prediction in attributed networks, such as SEAL [20] using GCN with graph pool layers, LGLP [21] based on line GNN, EdgeConvNorm [42] with edge convolution and normalization operations and NANs [52] using generative adversarial network. These methods treat the link prediction task as a graph classification problem and use end-to-end GNNs, resulting in good performance. However, they all fail to take community structure into account. Furthermore, they fuse link information and attribute information by simply using attribute vectors as the initial input of GNNs. This approach does not adaptively learn deep correlation information between link and attribute information and may even weaken the GNN's capabilities in graph data mining tasks [53]. This is also well confirmed by robust test experiments in Section 4.3.3.

In summary, our method CPAGCN not only considers how to fuse link information and attribute information adaptively, but also specially incorporates community structure. These features make it different from existing link prediction methods for attributed networks. In practice, extensive comparative experiments in Section 4 also comprehensively demonstrate its superiority.

6. Conclusion and future work

In this work, we focus on the problem of link prediction in attributed networks and devise a novel method named CPAGCN. In addition to fusing attribute information and link information adaptively, CPAGCN specially incorporates community structure into the network embedding framework for link prediction. By means of the joint training mechanism, CPAGCN can learn node representations preserving community structure, which are more beneficial to predicting links. Extensive experiments on several real-world attributed networks demonstrate the superiority over state-of-the-art methods. In the future, we will study how to extend CPAGCN to more complicated attributed networks, such as dynamic attributed networks, multi-layer attributed networks and heterogeneous attributed networks. Besides, as analyzed in Section 4.3.4, CPAGCN provides an open framework that incorporating community structure to boost the performance of link prediction in attributed networks, so exploring more powerful models to replace MGCN, BigCLAM and MLP in CPAGCN is also of great research value.

CRediT authorship contribution statement

Chaobo He: Writing – original draft, Conceptualization, Methodology. **Junwei Cheng:** Software, Visualization. **Xiang Fei:** Writing – review & editing. **Yu Weng:** Software, Visualization. **Yulong Zheng:** Software, Visualization. **Yong Tang:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 62077045.

References

[1] Y.L. Pei, T.J. Huang, V.I. Werner, M. Pechenizkiy, ResGCN: attention-based deep residual modeling for anomaly detection on attributed networks, Mach. Learn. 111 (2022) 519–541, http://dx.doi.org/10.1007/s10994-021-06044-0.
[2] J.J. Pfeiffer, S. Moreno, T.L. Fond, J. Neville, B. Gallagher, Attributed graph models: modeling network structure with correlated attributes, in: Proceedings of the 23rd International Conference on World Wide Web, 2014, pp. 831–842, http://dx.doi.org/10.1145/2566486.2567993.
[3] L.Y. Lü, T. Zhou, Link prediction in complex networks: a survey, Phys. A 390 (2011) 1150–1170, http://dx.doi.org/10.1145/3012704.
[4] V. Martinez, F. Berzal, J.C. Cubero, A survey of link prediction in complex networks, ACM Comput. Surv. 49 (2017) 1–33, http://dx.doi.org/10.1145/3012704.
[5] A. Kummar, S.S. Singh, K. Singh, B. Biswas, Link prediction techniques, applications, and performance: a survey, Phys. A 533 (2020) 124289.
[6] S. Haghani, M.R. Keyvanpour, A systemic analysis of link prediction in social network, Artif. Intell. Rev. 52 (2019) 1961–1995, http://dx.doi.org/10.1007/s10462-017-9590-2.
[7] F.F. Gou, J. Wu, Triad link prediction method based on the evolutionary analysis with IoT in opportunistic social networks, Comput. Commun. 181 (2022) 143–155, http://dx.doi.org/10.1016/j.comcom.2021.10.009.
[8] H.Y. Chen, J. Li, Exploiting structural and temporal evolution in dynamic link prediction, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 427–436, http://dx.doi.org/10.1145/3269206.3271740.
[9] F.P. Santos, Y. Lelkes, S.A. Levin, Link recommendation algorithms and dynamics of polarization in online social networks, Proc. Natl. Acad. Sci. 118 (2021) http://dx.doi.org/10.1073/pnas.2102141118.
[10] Z.P. Li, X. Fang, O.R.L. Sheng, A survey of link recommendation for social networks: methods, theoretical foundations, and future research directions, ACM Trans. Manag. Inf. Syst. 9 (2018) 1–26, http://dx.doi.org/10.1145/3131782.
[11] N.N. Daud, S.H.A.B. Hamid, M. Saadoon, F. Sahran, N.B. Anuar, Applications of link prediction in social networks: a review, J. Netw. Comput. Appl. 166 (2020) http://dx.doi.org/10.1016/j.jnca.2020.102716.
[12] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, P. Merialdo, Knowledge graph embedding for link prediction: a comparative analysis, ACM Trans. Knowl. Discov. Data 15 (2021) 1–49, http://dx.doi.org/10.1145/3424672.
[13] D.Q. Nguyen, V. Tong, D. Phung, Node co-occurrence based graph neural networks for knowledge graph link prediction, in: Proceedings of the 15th ACM International Conference on Web Search and Data Mining, 2022, pp. 1589–1592, http://dx.doi.org/10.1145/3488560.3502183.
[14] C. Fokoue, O. Hassanzadeh, M. Sadoghi, P. Zhang, Predicting drug-drug interactions through similarity-based link prediction over web data, in: Proceedings of the 25th International Conference Companion on World Wide Web, 2016, pp. 175–178, http://dx.doi.org/10.1145/2872518.2890532.
[15] Y.H. Long, M. Wu, Y. Liu, Y. Fang, C.K. Kwoh, J.M. Chen, J.W. Luo, X.L. Li, Pre-training graph neural networks for link prediction in biomedical networks, Bioinformatics 38 (8) (2022) 2254–2262, http://dx.doi.org/10.1093/bioinformatics/btac100.
[16] A. Pecli, M.C. Cavalcanti, R. Goldschmidt, Automatic feature selection for supervised learning in link prediction applications: a comparative study, Knowl. Inf. Syst. 56 (2018) 85–121, http://dx.doi.org/10.1007/s10115-017-1121-6.
[17] N. Shan, L.J. Li, Y.K. Zhang, S.S. Bai, X.Y. Chen, Supervised link prediction in multiplex networks, Knowl.-Based Syst. 203 (2020) http://dx.doi.org/10.1016/j.knosys.2020.106168.
[18] C. Wang, V. Satuluri, S. Parthasarathy, Local probabilistic models for link prediction, in: Proceedings of the 7th IEEE International Conference on Data Mining, 2007, pp. 322–331, http://dx.doi.org/10.1109/ICDM.2007.108.
[19] X. Fu, E. Seo, J. Clarke, R.A. Hutchinson, Link prediction under imperfect detection: collaborative filtering for ecological networks, IEEE Trans. Knowl. Data 33 (2021) 3117–3128, http://dx.doi.org/10.1109/TKDE.2019.2962031.
[20] M.H. Zhang, Y.X. Chen, Link prediction based on graph neural networks, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 5171–5181.
[21] L. Cai, J.D. Li, J. Wang, S.W. Ji, Line graph neural networks for link prediction, IEEE Trans. Pattern Anal. Mach. Intell. 44 (2022) 5103–5113, http://dx.doi.org/10.1109/TPAMI.2021.3080635.
[22] P. Cui, X. Wang, J. Pei, W.W. Zhu, A survey on network embedding, IEEE Trans. Knowl. Data Eng. 31 (2019) 833–852, http://dx.doi.org/10.1109/TKDE.2018.2849727.
[23] X. Wang, D.Y. Bo, C. Shi, S.H. Fan, Y.F. Ye, P.S. Yu, A survey on heterogeneous graph embedding: methods, techniques, applications and sources, IEEE Trans. Big Data (2022) http://dx.doi.org/10.1109/TBDATA.2022.3177455.
[24] H.Y. Cai, V.W. Zheng, K.C.C. Chang, A comprehensive survey of graph embedding: problems, techniques, and applications, IEEE Trans. Knowl. Data Eng. 30 (2018) 1616–1637, http://dx.doi.org/10.1109/TKDE.2018.2807452.
[25] D.K. Zhang, J. Yin, X.Q. Zhu, C.Q. Zhang, Network representation learning: a survey, IEEE Trans. Big Data 6 (2020) 3–28, http://dx.doi.org/10.1109/TBDATA.2018.2850013.
[26] C. Yang, Z.Y. Liu, D.L. Zhao, M.S. Sun, E.Y. Chang, Network representation learning with rich text information, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence, 2015, pp. 2111–2117.
[27] L.Z. Liao, X.N. He, H.W. Zhang, T.S. Chua, Attributed social network embedding, IEEE Trans. Knowl. Data Eng. 30 (2018) 2257–2270, http://dx.doi.org/10.1109/TKDE.2018.2819980.
[28] T.N. Kipf, M. Welling, Variational graph auto-encoders, in: Proceedings of NIPS Workshop on Bayesian Deep Learning, 2016, pp. 1–3.
[29] C.B. He, X. Fei, Q.W. Cheng, H.C. Li, Z. Hu, Y. Tang, A survey of community detection in complex networks using nonnegative matrix factorization, IEEE Trans. Comput. Soc. Syst. 9 (2) (2022) 440–457, http://dx.doi.org/10.1109/TCSS.2021.3114419.
[30] D. Jin, Z.Z. Yu, P.F. Jiao, S.R. Pan, D.X. He, J. Wu, P. Yu, W.X. Zhang, A survey of community detection approaches: from statistical modeling to deep learning, IEEE Trans. Knowl. Data Eng. 35 (2) (2023) 1149–1170, http://dx.doi.org/10.1109/TKDE.2021.3104155.
[31] O. Shchur, S. Günnemann, Overlapping community detection with graph neural networks, in: Proceedings of 1st International Workshop on Deep Learning for Graphs, 2019, pp. 1–7.
[32] V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. Theory Exp. 2008 (2008) http://dx.doi.org/10.1088/1742-5468/2008/10/P10008.
[33] J. Yang, J. Leskovec, Overlapping community detection at scale: a nonnegative matrix factorization approach, in: Proceedings of the 6th ACM International Conference on Web Search and Data Mining, 2013, pp. 587–596, http://dx.doi.org/10.1145/2433396.2433471.
[34] B. Perozzi, R. Al-Rfou, S. Skiena, DeepWalk: online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710, http://dx.doi.org/10.1145/2623330.2623732.
[35] D.K. Zhang, J. Yin, X.Q. Zhu, C.Q. Zhang, Homophily, structure, and content augmented network representation learning, in: Proceedings of the 16th International Conference on Data Mining, 2016, pp. 609–618, http://dx.doi.org/10.1109/ICDM.2016.0072.
[36] J.H. Li, L. Huang, C.D. Wang, D. Huang, J.H. Lai, P. Chen, Attributed network embedding with micro-meso structure, ACM Trans. Knowl. Discov. Data 15 (2021) 1–26, http://dx.doi.org/10.1145/3441486.
[37] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the 5th International Conference on Learning Representations, 2017, pp. 1–14.
[38] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 1025–1035.
[39] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: Proceedings of the 6th International Conference on Learning Representations, 2018, pp. 1–12.
[40] K.Y.L. Xu, W.H. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks? in: Proceedings of 7th International Conference on Learning Representations, 2019, pp. 1–17, http://dx.doi.org/10.48550/arXiv.1810.00826.
[41] G.Q. Cui, J. Zhou, C. Yang, Z.Y. Liu, Adaptive graph encoder for attributed graph embedding, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2020, pp. 976–985, http://dx.doi.org/10.1145/3394486.3403140.
[42] Z.W. Zhang, L. Cui, J. Wu, Exploring an edge convolution and normalization based approach for link prediction in complex networks, J. Netw. Comput. Appl. 189 (2021) http://dx.doi.org/10.1016/j.jnca.2021.103113.
[43] G.F. Chen, H.B. Wang, Y.L. Fang, L. Jiang, Link prediction by deep nonnegative matrix factorization, Expert Syst. Appl. 188 (2022) 115991, http://dx.doi.org/10.1016/j.eswa.2021.115991.
[44] J.Z. Qiu, Y.X. Dong, H. Ma, J. Li, C. Wang, K.S. Wang, J. Tang, NetSMF: large-scale network embedding as sparse matrix factorization, in: Proceedings of the World Wide Web Conference, 2019, pp. 1509–1520, http://dx.doi.org/10.1145/3308558.3313446.

[45] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, S. Yang, Community preserving network embedding, in: Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017, pp. 203–209.

[46] Z.L. Zhao, Z.Y. Gou, Y.H. Du, J. Ma, T.F. Li, R.S. Zhang, A novel link prediction algorithm based on inductive matrix completion, Expert Syst. Appl. 188 (2022) http://dx.doi.org/10.1016/j.eswa.2021.116033.

[47] X.K. Ma, S.Y. Tan, X.H. Xie, X.X. Zhong, J.J. Deng, Joint multi-label learning and feature extraction for temporal link prediction, Pattern Recognit. 121 (2022) http://dx.doi.org/10.1016/j.patcog.2021.108216.

[48] A. Grover, J. Leskovec, Node2vec: scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864, http://dx.doi.org/10.1145/2939672.2939754.

[49] A. Agibetov, Neural graph embeddings as explicit low-rank matrix factorization for link prediction, Pattern Recognit. 133 (2023) 108977, http://dx.doi.org/10.1016/j.patcog.2022.108977.

[50] Z.J. Liu, C. Huang, Y.W. Yu, J.Y. Dong, Motif-preserving dynamic attributed network embedding, in: Proceedings of the 2021 Web Conference, 2021, pp. 1629–1638, http://dx.doi.org/10.1145/3442381.3449821.

[51] H.L. Sun, F. He, J.B. Huang, Y.Z. Sun, Y. Li, C.Y. Wang, L. He, Z.B. Sun, X.L. Jia, Network embedding for community detection in attributed networks, ACM Trans. Knowl. Discov. Data 14 (3) (2020) 36, http://dx.doi.org/10.1145/3385415.

[52] Z. Wang, Y. Lei, W. Li, Neighborhood attention networks with adversarial learning for link prediction, IEEE Trans. Neural Netw. Learn. Syst. 32 (2021) 3653–3663, http://dx.doi.org/10.1109/TNNLS.2020.3015896.

[53] X. Wang, M.Q. Zhu, D.Y. Bo, P. Cui, C. Shi, J. Pei, AM-GCN: adaptive multi-channel graph convolutional networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2020, pp. 1243–1253, http://dx.doi.org/10.1145/3394486.3403177.