MODEL: Motif-based Deep Feature Learning for Link Prediction

Lei Wang, Jing Ren, Bo Xu, Jianxin Li, Wei Luo, and Feng Xia, *Senior Member, IEEE*

***Abstract*—Link prediction plays an important role in network analysis and applications. Recently, approaches for link prediction have evolved from traditional similarity-based algorithms into embedding-based algorithms. However, most existing approaches fail to exploit the fact that real-world networks are different from random networks. In particular, real-world networks are known to contain motifs, natural network building blocks reflecting the underlying network-generating processes. In this paper, we propose a novel embedding algorithm that incorporates network motifs to capture higher-order structures in the network. To evaluate its effectiveness for link prediction, experiments were conducted on three types of networks: social networks, biological networks, and academic networks. The results demonstrate that our algorithm outperforms both the traditional similarity-based algorithms (by 20%) and the state-of-the-art embedding-based algorithms (by 19%).**

***Index Terms*—Network Motif, Link Prediction, Network Embedding, Deep Learning, Autoencoder.**

I. Introduction

COMPLEX networks occur in many natural and social settings, examples including academic networks [1], biological networks [2], social networks [3], [4], and vehicular networks [5]. A network contains both vertices representing entities and links representing relationships between entities. It is a well-known problem that many networks generated from relational databases suffer from missing links. *Link prediction* aims to identify such missing links in a network [6]–[9]. It is an active research area that has many useful applications: Through link prediction, we can recommend a scientific paper for scholars in academic networks [1], build social recommender systems [10], [11], and discover currently unknown protein-protein interactions in biological networks [12].

In link prediction, a score is assigned to each pair of unconnected vertices based on their features. A higher score implies a higher probability of a missing link connecting the two vertices. Traditional similarity-based algorithms for link prediction rely on hand-crafted features, such as Common Neighbors and Jaccard's Coefficient [6]. However, when creating hand-crafted features, we need to select informative features and consider specific domain knowledge. To solve this problem, the idea of automatically learning features, known as network embedding [13], [14], has emerged. The main idea of network embedding is to map each vertex in a network to a vector in a d-dimensional space, with the correlation of vectors reflecting the correlation of the corresponding vertices. For example, the cosine of the vectors of two vertices can indicate the strength of their similarity or correlation.

Although existing algorithms for link prediction achieve good performance, they face the following challenges:

(1) Higher-order structures: Real-world networks usually exhibit higher-order structures. This is, the networks contain a large number of connected subgraphs that occur more frequently than in random networks. The well-known Science paper [15] has demonstrated that such subgraphs, known as network motifs, are basic network "building blocks" reflecting specific types of networks. A lot of studies in network analysis [16], [17] have confirmed that leveraging subgraphs can improve experimental performance. In particular, frequent subgraphs have been successfully applied to improve whole-graph embedding [18]. However, to our knowledge, network motifs have not been used to improve link prediction.

(2) Weak ties: The number of common neighbors between two vertices is informative of measuring their similarity. However, some vertices exhibit a weak tie, which means they share a few or no common neighbors [19]. We can find this phenomenon where two vertices reside in two different communities [20]. If we only concentrate on common neighbors, it will be difficult to discover the potential edge between two vertices that exhibit a weak tie.

(3) Complex network structures: Many networks have complex and highly non-linear structures, as demonstrated in [21]. Existing embedding-based algorithms rely on linear and shallow models, such as matrix factorization, which are unable to capture the complex structure of networks [8].

To sum up, the majority of existing algorithms fail to capture higher-order structures and potential weak ties since they rely on only pair-wise similarity and immediate neighbors. This problem can be addressed using special subgraphs (such as motifs) that preserve higher-order connectivity patterns [15]. Such higher-order connectivity patterns are often crucial for link prediction. In addition, existing algorithms, for example HONE [9], often learn features leveraging only shallow models. Often a complex model is necessary to capture complex network structures to achieve optimal link prediction [21].

Manuscript received August 31, 2019; revised November 18, 2019; accepted December 21, 2019. This work is partially supported by National Natural Science Foundation of China under Grant No. 61872054 and the Fundamental Research Funds for the Central Universities (DUT19LAB23). This research also is partially supported by Australian Research Council (ARC) Linkage Project with Grant Number- ARC LP180100750. *(Corresponding author: Bo Xu; e-mail: boxu@dlut.edu.cn)*

L. Wang, J. Ren, and B. Xu are with Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian University of Technology, Dalian 116620, China.

J. Li and W. Luo are with School of Information Technology, Deakin University, Burwood, Melbourne, VIC 3125, Australia.

F. Xia is with School of Science, Engineering and Information Technology, Federation University Australia, Ballarat, VIC 3353, Australia.

In this paper, we propose a **MO**tif-based **DE**ep feature learning algorithm for **L**ink prediction named MODEL. The proposed algorithm uses motifs to automatically learn vectorized features. When learning vector representations, we hope that learned vectors of these vertices composing a motif close to each other, and ensure that overlapping motifs are together. To achieve this goal, we redefine the first-order proximity and second-order proximity, both of which are different from the definition in [8], [22]. The redefined first-order proximity transforms the connectivities in a network from two vertices to more vertices, strengthening intra-motif relationships as connectivity exists between any two vertices in a motif. The redefined second-order proximity captures the relationship between the two vertices which do not reside in a motif but share many redefined neighbors. The two redefined proximities also address the problem that two potentially connected vertices exhibit a weak tie, since we add more neighbors for each vertex by redefining neighbors.

Aiming to address the complexity and non-linearity of network structures, we adopt a deep learning model called autoencoders which enjoy high non-linearity. The proposed algorithm first compresses the second-order proximity of vertices to low-dimensional vectors. From the perspective of the first-order proximity, we adjust vectors obtained from the second-order proximity to keep the first-order proximity in the vector space. At the same time, a method of negative sampling based on motifs is proposed to optimize the proposed algorithm.

To illustrate the effectiveness of our algorithm, we extensively conduct experiments on multiple types of networks, including social, biological and academic networks. Experimental results show that our algorithm outperforms state-of-the-art embedding-based algorithms by 19% and traditional similarity-based algorithms by 20% for link prediction. In addition, these experiments also show that our algorithm performs better in predicting potential weak ties than the baseline algorithms. At last, we investigate the influence of motif types on experiments, which is not discussed before but provides insight for selecting the most appropriate motif type for a network.

Contributions: The contributions of this paper are outlined as follows:

1) We propose a novel motif-based embedding algorithm for link prediction. This method seamlessly incorporates motifs and deep learning model into link prediction.
2) We redefine the first-order proximity and the second-order proximity of two vertices by motifs. The two redefined proximities are beneficial for preserving higher-order structures, and can be applied to different kinds of network analysis tasks.
3) Experiments on six networks empirically demonstrate that our algorithm achieves 19% gain over embedding-based algorithms and 20% gain over baseline algorithms for link prediction.

Organization: The rest of this paper is organized as follows: In Section II, we give an overview of related work on network embedding and link prediction. Section III introduces some important preliminaries, including the redefined first-order proximity and the second-order proximity. In Section IV, we present the overall framework of our proposed algorithm and the loss function in detail. In Section V, we evaluate the effectiveness of the algorithm over three distinct types of networks. Finally, Section VI concludes the paper.

II. Related Work

In this section, we review key applications of motifs, and algorithms for network embedding and link prediction. Besides, we state the relationship and difference between our study and some similar studies.

A. Motif

Previously, the concept of motif is used to apply largely to biological networks. Uri Alon [23] discusses two types of transcription network: sensory networks and developmental networks where motifs can be found. Grundy et al. [24] propose motif-based HMMs to solve the problem that HMMs require a relatively large training set. Recently researchers apply motifs to multiple networks and a variety of scenes. Benson et al. [15] propose a motif-cut approach to improve network clustering accuracy, which is successfully applied to social networks, academic networks, and so on. Arenas et al. [16] have used motifs to define which communities vertices belong to. Xia et al. [25] surveyed measures for network motifs.

For detecting motifs in networks, Milo et al. [26] propose an approach to find the striking appearance of motifs in networks. The approach only selects patterns appearing in real networks with the numbers being significantly higher than those in the randomized networks. S. Wernicke proposes an algorithm [27] that can enumerate and sample subgraphs by selecting a random edge and randomly extending the subgraph. Based on the algorithm [27], Wernicke et al. [28] devise a tool, namely FANMOD, for network motifs detection.

B. Network Embedding

1) Pair-wise Connectivity: Inspired by the success of word vectors, DeepWalk [29] considers a vertex sequence sampled by random walk as a sentence. These sentences can be used to learn a vector representation for each vertex by the power of SkipGram [30]. LINE [22] uses the first-order and the second-order proximity to learn vectors. Meanwhile, edge sampling and negative sampling [31] are used to optimize the model. GraRep [32] illustrates the importance of using global information when learning vector representations. Based on the discussion in the study [33] where negative sampling is equal to matrix factorization, GraRep can obtain a low-dimensional vector representation for each vertex using SVD at each step, and concatenate all vectors in K step as the last vector representations. SDNE [8] uses a deep learning model that uses the second-order proximity as input to mine highly non-linear relations between vertices and attaches the first-order proximity as a supervised component to the model. The algorithm is successfully applied to link prediction. Since

SDNE and LINE only consider pair-wise similarity, the first-order and second-order proximities defined by the similarity can not preserve higher-order structures.

Besides homophily, some networks exhibit structural equivalence or structural similarity. Node2vec [29] defines a second-order random walk using return parameter p and in-out parameter q, and also shows effectiveness on link prediction. Different from Node2vec that considers both homophily and structural equivalence of networks, Struc2Vec [34] exclusively focuses on structural identity by vertex sequences ordered by degree of vertices. This approach shows great superiority over Node2vec on node classification.

2) Higher-order Structure: The approaches listed above ignore an important characteristic of networks, i.e., higher-order structures. Recently, some researchers have begun to leverage subgraphs, graphlets or motifs to improve the performance of vector representations used to perform network tasks. SNS [17] defines the structural similarity of two vertices based on graphlet and orbit, which can be combined with original methods of learning word2vec, such as CBOW. HONE [9] defines the problem of higher-order network representation learning based on network motifs. This approach defines a new adjacency matrix for each motif from two-node to four-node and demonstrates its effectiveness on link prediction. However, this approach does not indicate why motifs are informative of link prediction, and involves a number of matrix multiplications and linear matrix factorizations. As a result, HONE has high computational complexity and, as our experiment results show, may not able to fully capture complex network structures and relationships between higher-order structures.

C. Link Prediction

1) Tradititonal Similarity-based Algorithms: Traditional similarity-based algorithms [6] for link prediction mainly focus on neighbors of two potentially connected vertices to measure their similarity. Common Neighbors directly simply counts common neighbors. Jaccard's Coefficient divides the number of common neighbors by the number of all neighbors of the two vertices. Adamic-Adar also counts common neighbors, but assigns the less-connected neighbors more weight. Preferential attachment assumes that two vertices of higher degrees are more likely to link to each other.

2) Embedding-based Algorithms: Embedding-based algorithms measure the similarity of two vertices by their learned vectorized features. SDNE [8] uses the cosine as an indicator of similarity. Node2vec [29] designs four different operators over vertices' features to generate edges' features, including Average, Hadamard, Weighted-L1 and Weighted-L2. HONE [9] uses Average operation to construct edges' features. DeepGL [35] uses the four operators to construct edges' features for inductive network representation. The paper [29] illustrates that embedding-based algorithms can outperform traditional similarity-based algorithms on link prediction. Wang et al. [36] propose a predictive network representation learning named PNRL to solve the structural link prediction problem. PNRL jointly optimizes two objectives for observed links and assumed hidden links. Tu et al. [37] propose a Community-enhanced Network Representation Learning named CNRL. The algorithm shows its superiority on link prediction. Liao et al. [38] propose a network embedding for attributed social networks, and show that the method achieves substantial gains on the task of link prediction. Zou et al. [39] propose an encoder-decoder model for graph generation. The encoder is a Gaussianized graph scattering transform and the decoder can be adapted to link prediction. RGCN [40] uses graph convolutional networks to learn embedding vectors and uses a tensor factorization model to predict links.

D. Similar Work

There are some studies similar to our. Both E-LSTM-D [41] and MODEL aim at predicting links in networks and use auto-encoder. However, E-LSTM-D focuses on dynamic networks and thus needs multiple graphs. MODEL focuses on static networks and thus needs only one graph. S-LSTM-D combines auto-encoder and a stacked long short-term memory (LSTM), a deep learning model, and does not consider the higher-order structure. MODEL combines auto-encoder and motifs, higher-order structures that naturally exist in networks. Qi et al. propose a concept of subgraph network (SGN) that can be applied to network models [42]. Both SGN and MODEL consider subgraphs, but SGN uses various types of subgraphs. MODEL uses only one type of subgraphs, i.e., motif. SGN constructs structural features without considering the nonlinear network structure, while MODEL learns features by auto-encoder that addresses the nonlinear network structure. The goal of the paper [42] seams to address network classification; our goal is to predict links of networks. Both the study [43] and MODEL leverage network property. But the study uses the centrality features; MODEL uses motifs. The study combines network property with line graphs to predict link weights, i.e., to determine the weight of links, while MODEL combines network property with auto-encoder to predict links, i.e., to judge whether there is a link between two vertices. Both RUM [44] and MODEL leverage motifs to capture high-order network structure. However, RUM needs to be combined with existing embedding algorithms, such as DeepWalk and LINE, and does not consider the nonlinear network structure. On the contrary, MODEL is an individual algorithm and uses auto-encoder to capture the nonlinear network structure. Motif2Vec [45] also leverages motifs, but its goal is to design a new random-walk method, which is unsatisfactory to capture the nonlinearity of the network structure. Furthermore, Motif2Vec focuses on heterogeneous networks, while MODEL focuses on homogeneous networks. The study [46] defines topological features by motifs from three-node to five-node, without considering the nonlinearity of the network structure. MODEL learns features only by one type of motifs and considers the nonlinearity. The study [46] sees link prediction as a binary classification problem, while MODEL sees it as a probability problem.

III. Preliminaries

In this section, we list some important preliminaries and notations for this paper. It is worthwhile to note that the

definitions of the first-order proximity and the second-order proximity in this section are redefined by motifs, different from the definitions from the papers [8], [22]. Some key notations are shown in Table I.

TABLE I: Key notations in this paper

Notations	Meaning
G	A graph
V	The vertex set
$\|V\|$	The number of vertices
v_i	The vertex i
E	The edge set
$\|E\|$	The number of edges
M	A motif/A motif type
$\{M\}$	The motif set
M_{xy}	A motif of order x and number y
d	The dimention of vector representations
y_i	The vector representation of vertex i
N_i	The first-order proximity of vertex i and other vertices
$N(i)$	The set of neighbors of vertex i
f,g	Mapping functions
x	An input vector of the model
x^j	The j_{th} element of x
y^k	An output vector of k_{th} hidden layer in encoders/decoders
x'	An ouput vector of decoders
z	The penalty vector
W_k	The weights of k_{th} hidden layer in encoders
B_k	The biases of k_{th} hidden layer in encoders
W'_k	The weights of k_{th} hidden layer in decoders
B'_k	The biases of k_{th} hidden layer in decoders
K	The number of hidden layers in encoders/decoders
θ	The set of parameters to be learned
$\odot$	The Hadamard product of matrices
$\|\|\ \|\|_F$	Frobenius norm

A. Graph and Motif

We denote a graph as $G = (V, E)$, where V denotes the set of all vertices and E denotes the set of all edges in G, i.e., $(v_i, v_j) \in E$ if v_i and v_j are connected. A motif is defined as a connected subgraph with a small number of vertices. Formally, we denote a motif as $M = (V_M, E_M)$. If a motif is a connected subgraph from graph G, $(v_i, v_j) \in E$ for any $(v_i, v_j) \in E_M$. A motif is not merely any subgraph. It exists in a type of networks significantly more frequently than a random subgraph, reflecting an underlying process specific to the type of networks. For example, feed-forward loops are known to be prevalent and has functional significance in Protein–protein interaction networks.

In general, network motif search is NP-hard. Higher-order motifs require more time to find and more space to store. Aiming at reducing the time complexity and space complexity, we only consider all three-node and four-node motifs in this paper. Besides, our proposed approach can also be extended to arbitrary-node motifs. In Fig. 1, we draw all types of three-node and four-node motifs. Note that a motif M_{xy} represents the type of motif with order x and number y. Efficient algorithms are available for finding motifs in a network [28], [47].

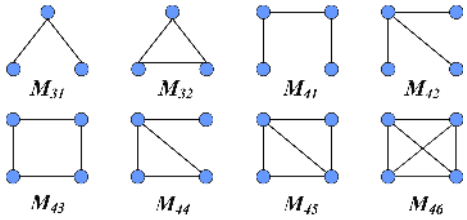


Fig. 1: All three-node and four-node motifs. Circles represent vertices, and lines represent edges.

B. First-order Proximity and Second-order Proximity

First-order proximity: The first-order proximity is directly defined on pair-wise vertices. If v_i and v_j reside at the same motif, there exists the first-order proximity between v_i and v_j. Otherwise, the proximity is 0. The more motifs both v_i and v_j reside at are, the larger the first-order proximity between v_i and v_j is.

The way to define the first-order proximity is different from the definition in LINE [22] and SDNE [8]. LINE and SDNE define the first-order proximity on two connected vertices, while we define it on any two vertices in a motif, despite whether they are connected. Taking M_{43} in Fig. 1 as an example, we attach the first-order proximity on any two diagonal vertices and any two connected vertices as they all reside at M_{43}. However, LINE and SDNE do not attach the first-order proximity on two diagonal vertices since they are not connected. It is worth noting that our way defining the proximity covers the definition in LINE and SDNE. If we only consider two-node motifs, the two definitions are equivalent. Since the first-order proximity exists between any two vertices in a motif, the relationship of intra-motif can be strengthened.

In this paper, let $N_i = \{w_{i1}, w_{i2}, \ldots, w_{in}\}$ denote the first-order proximity of v_i and other vertices. $w_{ij} \in R\ (R >= 0)$ denotes the first-order proximity of v_i and v_j, which is defined as the number of motifs where v_i and v_j occur simultaneously.
Second-order proximity: The second-order proximity is defined on neighbors of two vertices. It is founded on the cognition that the more common neighbors two vertices have, the more similar the two vertices are [8], [22]. In this paper, we redefine the neighbors of every two nodes via a specific motif.

According to the first-order proximity we defined, the second-order proximity of v_i and v_j can be calculated by the similarity of N_i and N_j. If v_i and v_j share a great number of same neighbors, the two nodes and their neighbors will be close in the vector space, which leads to the two motifs of v_i and v_j become close.

C. Autoencoder

Autoencoder, introduced first in [48] by Hinton et al., is a non-linear way of reducing the data dimension by multi-layer neural network.

A universal framework of autoencoder is shown in Fig. 2. An autoencoder consists of two parts, one part compressing high-dimensional data x to low-dimensional data y and the other part decompressing y to high-dimensional data x' seen

as reconstructed x. Each part is actually a multi-layer neural network with a large number of non-linear activation functions. We use f_θ and $g_{\theta'}$ to denote the encoding and decoding function respectively, thus the process of encoding and decoding is equal to address the function $\arg\min_{\theta,\theta'} \ell(x, g_{\theta'}(f_\theta(x)))$, wher $\ell(x, x')$ is the function measuring the difference of x and x', e.g., Euclidean distance.

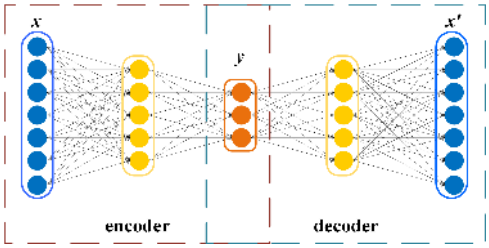


Fig. 2: A universal framework of autoencoder. The left part is an encoder and the right part is a decoder. In the autoencoder, x is the input vector, y is the output vector, and x' is the reconstructed vector of x.

D. Problem Definition

Motif-based Embedding: Given all motifs with a specific type in a network, we need to learn a mapping function $f : V \to y \in R^d$, where $d \ll n$. We hope that learned mapping function can preserve well higher-order structures in original networks, i.e., the similarity of intra-motif and relationships between motifs.

Link prediction: Link prediction aims to predict missing edges or edges that will appear in the future. A graph $G = (V, E)$ can be regarded as $G = \{(v_i, v_j) \in E\} \subseteq V \times V$, a subset of all possible edges. In practice, we may not be able to model every edge, particular those considered weak ties.

The objective of *link prediction* is to find a scoring function $S : V \times V \mapsto [0, 1]$ such that $S(v_i, v_j)$ is high if $(v_i, v_j) \in E$ and $S(v_i, v_j)$ is low if $(v_i, v_j) \notin E$. That is, we aim to maximize

$$\sum_{(v_i,v_j)\in E} \log S(v_i, v_j) + \sum_{(v_i,v_j)\notin E} \log(1 - S(v_i, v_j)).$$

Since we use a d-dimensional vector to represent each vertex in this paper, we can use the cosine similarity of $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ to indicate the score that there is an underlying edge between v_i and v_j. That is

$$S(v_i, v_j) = \cos(\boldsymbol{y}_i, \boldsymbol{y}_j).$$

IV. Design of Model

In this section, we first give the overall framework of proposed algorithm MODEL. Subsequently, We introduce how to preserve higher-order structures, i.e., the second-order proximity and the first-order proximity, and how to optimize them by sampling negative nodes. Last, we conclude the complexity of training the model.

A. Framework

To learn a mapping that preserves higher-order structures in networks, we perform the next two steps: First, we construct a vector representation for each vertex by their neighbors. A simple way is that we directly use N_i as a vector to represent v_i. However, due to its high sparsity, discrete values and the large dimension, the vector is inappropriate for downstream network tasks. In practice, we can compress N_i to a low-dimensional vector with entries being real values to address the problem. In the process of compression, we employ an autoencoder that enjoys non-linearity and flexibility [8], [49], [50]. Second, considering the first-order proximity, we need to attach a loss term on vector representations of all vertices in a motif. An effective and widely used way is to minimize the Euclidean distances of vectors of any two vertices in a motif. For taking better advantage of the power of the first-order proximity, neighbors of all vertices in a motif are inputted to encoder simultaneously, so that vectors of the vertices can be obtained at the same time.

We show the framework of the proposed approach in Fig. 3. The model consists of four autoencoders which share the same weights, biases and activation functions. The part below is encoder and the one above is decoder. Given a motif composed of v_i, v_j and v_k, the model first input neighbor vectors x_i, x_j and x_k, corresponding to v_i, v_j and v_k respectively. By encoding and decoding, y_i, y_j and y_k are obtained as vector representations, which can reflect the second-order proximity between these vertices. Due to the first-order proximity in a motif, a loss term is attached to regularise vector representations. To optimize the model, an autoencoder for a negative vertex v_l is added. We hope the model can distinguish y_i, y_j and y_k from y_l in the vector space.

The Model in Fig. 3 is drawn based on three-node motifs, so that there are only four auto-encoders, in which v_i, v_j, and v_k compose a three-node motif and v_l is a negative vertex. Even so, the model can be transformed easily to be of four-node motifs and even higher-order motifs. For example, we can add another autoencoder sharing the same parameters, so that the new model can be applied to four-node motifs. All next introductions about the proposed algorithm are based on three-node motifs. Meanwhile, we list some key symbols of the model in Table I. Note that a vector with subscript i represents the specific vector to v_i, e.g., x_i is the input vector of v_i.

Next, we detail how our model learns a vector representation for each vertex to preserve the second-order proximity and the first-order proximity, and how to add noise to optimize the model.

B. Preserving Second-order Proximity

For preserving the second-order proximity, we first need to obtain neighbors of each vertex according to Section III. These neighbors are stored in $N(i)$ and seen as the input vector x_i of v_i in encoders. Through Equation 1 and 2, where σ denotes the activation function, such as *sigmoid*, we can obtain output vectors of encoders as vector representations. The more similar two input vectors are, the more similar two output vectors are.

$$y_i^1 = \sigma(W_1 x_i + B_i) \quad (1)$$

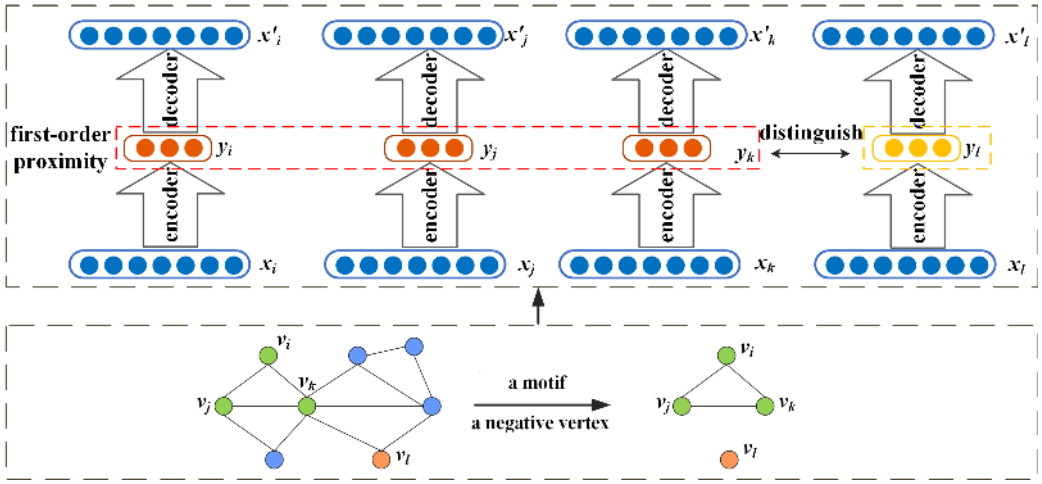


Fig. 3: The overall framework of MODEL. The model first selects a motif composed of v_i, v_j and v_k, and a negative vertex v_l. The vectorized neighbors of the four vertices are inputted simultaneously into four autoencoders sharing the same parameters. The outputs of autoencoders, y_i, y_j, y_k and y_l, serve as learned features of their corresponding vertices. Due to the first-order proximity, a loss term is added for y_i, y_j and y_k. Meanwhile, a separate loss term is used to distinguish a motif from a negative vertex in the vector space.

$$y_i^k = \sigma(W_k y_i^{k-1} + B_k) \quad k = 2, \ldots, K \tag{2}$$

After compression, decoders accept output vectors of encoders as input vectors and then reconstruct input vectors of encoders. Through Equation 3 and 4, we can obtain reconstructed vector x_i' of vector x_i. In Equation 3, y_i^0 is the input vector of decoders and also the output vector of encoders.

$$y_i^k = \sigma(W_k' y_i^{k-1} + B_k') \quad k = 1, \ldots, K-1 \tag{3}$$

$$x_i' = y_i^K = \sigma(W_K' y_i^{K-1} + B_K') \tag{4}$$

In the process of encoding and decoding, we minimize the distances of original vectors and reconstructed vectors, i.e., $\|x_i - x_i'\|_2^2$ for input vector x_i. Given all motifs denoted by $\{M\}$ in a network, the loss function is Equation 5, where $\| \ \|_2$ denotes the Euclidean norm.

$$\ell_{2nd} = \sum_{M \in \{M\}} \sum_{i \in M} \|x_i - x_i'\|_2^2 \tag{5}$$

Due to the high sparsity in a lot of networks, a large number of elements in x are zero, so that autoencoders are prone to reconstruct these elements with value zero [8]. To address this problem, we add a penalty vector z_i for input vector x_i in Equation 5. If $x_i^j > 0$, $z_i^j = \beta\,(\beta > 1)$, else $z_i^j = 1$. Through Equation 6 that gives more weight to non-zero elements than zero elements, autoencoders are prone to reconstruct non-zero elements of input vector x.

$$\ell_{2nd} = \sum_{M \in \{M\}} \sum_{i \in M} \|(x_i - x_i') \odot z_i\|_2^2 \tag{6}$$

In the above equation, $\odot$ denotes the Hadamard product which represents element-wise multiplications for matrices.

C. Preserving First-order Proximity

Besides the second-order proximity, we need to capture the intra-motif similarities, i.e., the first-order proximity between vertices. Given a motif consisting of v_i, v_j, and v_k, we can obtain vector representations y_i, y_j, and y_k by autoencoders. Because of the existence of the first-order proximity between any two vertices in v_i, v_j, and v_k, we need to minimize the Euclidean distances of vector representations of any two vertices. The equation is:

$$\ell = \|y_i - y_j\|_2^2 + \|y_i - y_k\|_2^2 + \|y_j - y_k\|_2^2 \tag{7}$$

Through the above equation, vector representations of all vertices residing in the same motif are close to each other. Note that for three-node motifs, there are three Euclidean distances in Equation 7. If we consider four-node motifs, there will be six Euclidean distances. Given all motifs denoted by $\{M\}$ in a network, the loss function is Equation 8.

$$\ell_{1st} = \sum_{M \in \{M\}} \sum_{i,j \in M, i \neq j} \|y_i - y_j\|_2^2 \tag{8}$$

From the above equation, we can find that the more the number of motifs both v_i and v_j reside in is, the closer y_i and y_j are.

D. Negative Sampling

In Section 4.3, we only strengthen intra-motif relationships. As a result, the algorithm may converge to the solution that all motif-wise relationships are similar and cannot distinguish a motif from other motifs, which harms experimental performance. In the study of [31], an effective approach called negative sampling is proposed. The main idea of this approach is to distinguish right from noises. Inspired by this approach, we hope to distinguish what vertices can compose a motif from a noise. The noise can be sampled randomly or based on the

number of motifs each vertex resides in. The distinguishing process can be formulated as follows:

$$\ell_{dis} = \max(\lambda + \sum_{i,j \in M, i \neq j} \|y_i - y_j\|_2^2 - \mu \sum_{i \in M} \|y_i - y_l\|_2^2, 0) \quad (9)$$

where we denote a noise by v_l. A balance factor μ is added in Equation 9, since the number of Euclidean distances in the first integral sum may be greater than that in the second integral sum. For three-node motifs, μ is one while it is 3/2 for four-node motifs. λ is a slack parameter that represents the margin by which the second sum should exceed the first sum.

Combined with Equation 9, the Equation 8 is transformed to Equation 10:

$$\ell_{1st} = \sum_{M \in \{M\}} \ell_{dis} \quad (10)$$

E. Optimization

From Section 4.2 and 4.3, we obtain two loss functions to preserve the first-order proximity and the second-order proximity respectively. Meanwhile, to prevent overfitting, a loss function of the regularization on weights of autoencoders needs to be added [8]. The loss function is Equation 11:

$$\ell_{reg} = \sum_{k=1}^{K} (\|W_k\|_F^2 + \|W'_k\|_F^2) \quad (11)$$

where $\| \ \|_F$ denotes the Frobenius norm.

Combining Equation 5, 10 and 11, we can obtain a joint loss function as follows:

$$\ell_{loss} = \ell_{2nd} + \alpha \ell_{1st} + \gamma \ell_{reg} \quad (12)$$

where α and γ are the weights of the first-order proximity and the regularization respectively.

To minimize Equation 12 and make the algorithm converge fast, we use Adam algorithm [51] and back-propagation [52] to optimize our model. We summarize the process of our approach in Algorithm 1.

In Algorithm 1, we first need to look for all motifs $\{M\}$ with type *M* and then obtain neighbors for each vertex according to Section III. At step 3, to prevent the model suffering from local optimality, we use DBN [53] to pre-train parameters θ at first.

F. Complexity Analysis

When looking for all motifs, we resort to FANMOD [28] which is easy to implement and can quickly enumerate all motifs.

In the process of training our model, the data used is a small batch of motifs. Therefore, the time complexity is $O(c\Delta dTK)$ where *c* is the order of a motif. Since only the three-node and four-node motifs are considered, *c* is either 3 or 4. *T* is the maximal iteration number, and is set as 200 in all experiments. d is the dimension of vector representations, and K is the number of layers in encoders or decoders. At last, it is easy to find that the time complexity is linear to the number of motifs. During experiments, only a small portion of all motifs in a network is needed, which avoids the problem that some networks contain tens of thousands of motifs.

Algorithm 1: MODEL

Input: a network $G = (V, E)$, motif type *M*, parameters α, γ, β, λ, batch size Δ, learning rate δ
output: vector representations y of all vertices
1: Look for all motifs $\{M\}$ with type *M* in *G*
2: Obtain redefined neighbors for each vertex in *G*
3: Initiate parameters $\theta = \{W_k, B_k, W'_k, B'_k\}_{k=1}^{K}$
4: **Repeat:**
5: Randomly select a subset $\{M\}_\Delta$ with size Δ from $\{M\}$
6: Calculate ℓ_{loss} according to Equation 12
7: Update parameters θ by back-propagation and Adam with the learning rate δ
8: **Until convergence**

V. Experiments and Analysis

A. Datasets

In this paper, we select six networks with three distinct types to comprehensively evaluate the effectiveness of our proposed algorithm. The six networks are introduced as follows.

Youtube [54] and LiveJournal [54] are social networks where vertices denote users and each edge indicates the friendship between corresponding two vertices. In the two networks, we randomly select 5,346 vertices in Youtube and 2,456 vertices in LiveJournal as our experimental datasets and ensure that there are not isolated vertices in the two datasets.

Bio-sc-cc [55] and Bio-sc-ht [55] are biological networks where vertices denote genes and each edge indicates the interaction between corresponding two genes. Bio-sc-cc has 2,223 vertices and 34,879 edges, and Bio-sc-ht has 2,084 vertices and 63,027 edges.

DBLP [54] and Ca-GrQc [56] are academic networks where vertices denote scholars and each edge indicates the collaboration between the corresponding two scholars. We randomly select 4,424 vertices from the original DBLP network as our experimental dataset and ensure that there are not isolated vertices in the dataset. Ca-GrQc has 4,158 vertices and 13,422 edges.

Some detailed statistics about the six network datasets are summarized in Table II.

TABLE II: Statistics of the six datasets

datasets	$\|V\|$	$\|E\|$	Avg. Degree	Network Type
Youtube	5,246	24,121	9.02	Social
LiveJournal	2,456	188,490	153.49	Social
Bio-sc-cc	2,236	34,879	31.2	Biological
Bio-sc-ht	2,084	63,027	60.49	Biological
DBLP	4,424	12,169	5.5	Academic
Ca-GrQc	4,158	13,422	6.46	Academic

B. Baseline Algorithms

Although there are a large number of algorithms on network embedding, only small of them illuminate their effectiveness for link prediction. In this paper, we use the following algorithms as baselines, including three traditional methods that can achieve good performance in link prediction [6].

DeepWalk [29]: It randomly samples many vertex sequences via random walk and generates vector representations of vertices using Skip-Gram on these vertex sequences.

Node2vec [7]: It can be seen as the extended DeepWalk and randomly samples vertex sequences via 2^{nd} order random walk with two parameters p and q.

SDNE [8]: It employs an autoencoder as a deep model to generate vector representations of vertices. The autoencoder inputs the adjacent vector of each vertex and adds the first-order proximity as supervised information.

HONE [9]: It considers all types of motifs from two-node to four-node, and generates a local k-step embedding for each motif. At last, it concatenates all local embeddings for all motifs in k steps, and obtains low-dimensional vector representations via matrix factorization.

SCAT [39]: It uses a Gaussianized graph scattering transform as an encoder and a fully connected network as a decoder. The decoder is adapted to link prediction.

RGCN [40]: It produces latent feature representations by an encoder and predicts label edges by a decoder that is a tensor factorization model exploiting these representations.

Traditional methods [6]: Common Neighbors (CN) defines the similarity of vertex i and j as $|N(i)\bigcap N(j)|$. Jaccard's Coefficient (JC) defines them as $\frac{|N(i)\bigcap N(j)|}{|N(i)\bigcup N(j)|}$. And Adamic-Adar (AA) Score defines them as $\sum_{t\in N(i)\bigcap N(j)}\frac{1}{\log|N(t)|}$.

C. Evaluation Metrics

For link prediction, precisionK, AUC, and Avg. Rank are three widely used evaluation metrics [6]. In our experiments, we use these metrics to evaluate the performance of our proposed algorithm and other algorithms. The detailed introduction about the three metrics is as follows: Let D^+ denote the set of positive examples, D^- denote the set of negative examples, and $P(e)(e\in D^+\cup D^-, D^+\cap D^-=\varnothing)$ denote the probability that $e\in D^+$. Based on P, a decreasing ordered list, denoted by RANK, can be obtained. $RANK(e)$, from 1 to $|D^+|+|D^-|$, represents the rank of e. We hope elements in D^+ can be ranked in front of these elements in D^-.

PrecisionK is the proportion of elements from D^+ in the top-K elements. It is calculated as follows:

$$\text{PrecisionK}=\frac{|\{e|RANK(e)\leqslant \text{K}, e\in D^+\}|}{\text{K}} \tag{13}$$

AUC is related to the ranking quality of all examples. It is calculated as follows:

$$\text{AUC}=1-\frac{1}{|D^+||D^-|}\sum_{e^+\in D^+}\sum_{e^-\in D^-}(\dagger(RANK(e^+)< RANK(e^-))+0.5\,\dagger\,(RANK(e^+)=RANK(e^-))) \tag{14}$$

where $\dagger(x)$ is 1 if x is true, else it equals to 0. From the equation 14, we can know that AUC always exceeds 0.5 or equals to 0.5.

Avg. Rank is the average rank of elements in D^+. It is calculated as follows:

$$Avg.Rank=\frac{\sum_{e\in D^+}RANK(e)}{|D^+|} \tag{15}$$

D. Experimental Settings

Before doing experiments, we first randomly hide some edges truly existing in original networks as D^+ and ensure that the degree of each vertex is greater than zero. In the meantime, we randomly select some edges not existing as D^- and ensure that $|D^+|=|D^-|$. During the process of training, hidden edges are not used. For sparse networks, 20% edges are hidden in DBLP and 30% edges are hidden in Youtube and Ca-GrQc. For dense networks, 60% edges are hidden in Bio-sc-cc, 80% edges are hidden in Bio-sc-ht and 90% edges are hidden in LiveJournal. In experiments, we need to predict which edges are from D^+ in $D^+\cup D^-$.

We compare our proposed algorithm with those baseline algorithms introduced in Section 5.2. For embedding based algorithms, we set all embedding dimension to 128. For DeepWalk and Node2vec, we set *walks per node r*=10, *walk length l*=80, and *context size k*=10. The optimal p and q are selected from $\{0.25, 0.5, 1, 2, 4\}$ for Node2vec. For HONE, we also consider all motifs with 2-4 vertices and select step K from $\{1, 2, 3, 4\}$. According to [8] and [39], we use two-layers encoders for SDNE and SCAT. The dimensions of hidden layers are set to 1000 and 512, respectively. For RGCN, we use two-layers graph convolutional networks with 512 and 128 hidden units, and regard existent and nonexistent edges as two types of edges. In the studies [9] and [39], there are four variants of HONE and two variants of SCAT. In experiments, we only show the best performance in these variants for the comparison.

For our proposed algorithm, different types of motif have different performance. We conduct experiments over all motif types from M_{31} to M_{46} and show the best performance in them. In section 5.6, we show the influence of motif types on experiments. We select a negative vertex based on the number of motifs it resides in. The hyper-parameters α, β, λ and γ are set 20, 30, 30 and 0.0001 respectively. The learning rate is set 0.001 and the size of a batch is set 500. All encoders are single-layer neural networks. We select $tanh$ as the activation function, because we hope that the final vector representations contain both positive and negative values. All experiments run 10 times and these results are averaged.

After obtaining the vector representation of each vertex, the cosine value of vectors of two vertices represents the similarity of them. A good algorithm for link prediction can rank elements in D^+ in front of elements in D^- based on the similarity.

E. Experimental Results

In this section, we use AUC and precisionK to compare our proposed algorithm with nine baseline algorithms over six networks. The experimental results for AUC and precisionK are shown in Table III and Fig. 4, respectively. In Table III, the best performance is highlighted in bold.

Table III shows that our algorithm, MODEL, outperforms other related algorithms in most cases. Furthermore, the superiority of MODEL for percisionK is shown in Fig. 4. On the two social networks Youtube and LiveJournal, MODEL outperforms these embedding-based algorithms which do not

TABLE III: Comparisons of ten algorithms on AUC

Algorithm	Youtube	LiveJournal	Bio-sc-cc	Bio-sc-ht	DBLP	Ca-GrQc
MODEL	**0.843**	0.919	**0.793**	**0.841**	**0.944**	0.820
DeepWalk	0.684	0.724	0.677	0.746	0.901	0.761
Node2vec	0.699	0.736	0.685	0.747	0.922	0.781
SDNE	0.662	0.747	0.692	0.717	0.913	0.807
HONE	0.624	0.535	0.709	0.609	0.937	**0.841**
SCAT	0.812	**0.930**	0.718	0.826	0.931	0.815
RGCN	0.808	0.848	0.721	0.786	0.883	0.805
Common Neighbors (CN)	0.631	0.818	0.711	0.679	0.881	0.732
Jaccard's Coefficient (JC)	0.624	0.816	0.702	0.678	0.883	0.733
Adamic-Adar (AA)	0.637	0.819	0.715	0.68	0.884	0.725

TABLE IV: Influences of motif types on AUC

Motif type	Youtube	LiveJournal	Bio-sc-cc	Bio-sc-ht	DBLP	Ca-GrQc
M_{31}	0.826	0.911	0.781	0.840	0.937	0.813
M_{32}	0.832	0.895	0.786	0.819	0.913	0.793
M_{41}	0.838	0.912	0.786	**0.841**	0.833	0.819
M_{42}	0.819	**0.919**	0.777	**0.841**	0.942	**0.820**
M_{43}	**0.843**	0.902	**0.793**	0.832	0.931	0.802
M_{44}	0.833	0.913	0.787	0.840	**0.944**	0.807
M_{45}	0.839	0.899	0.787	0.824	0.909	0.781
M_{46}	0.824	0.891	0.755	0.821	0.899	0.787

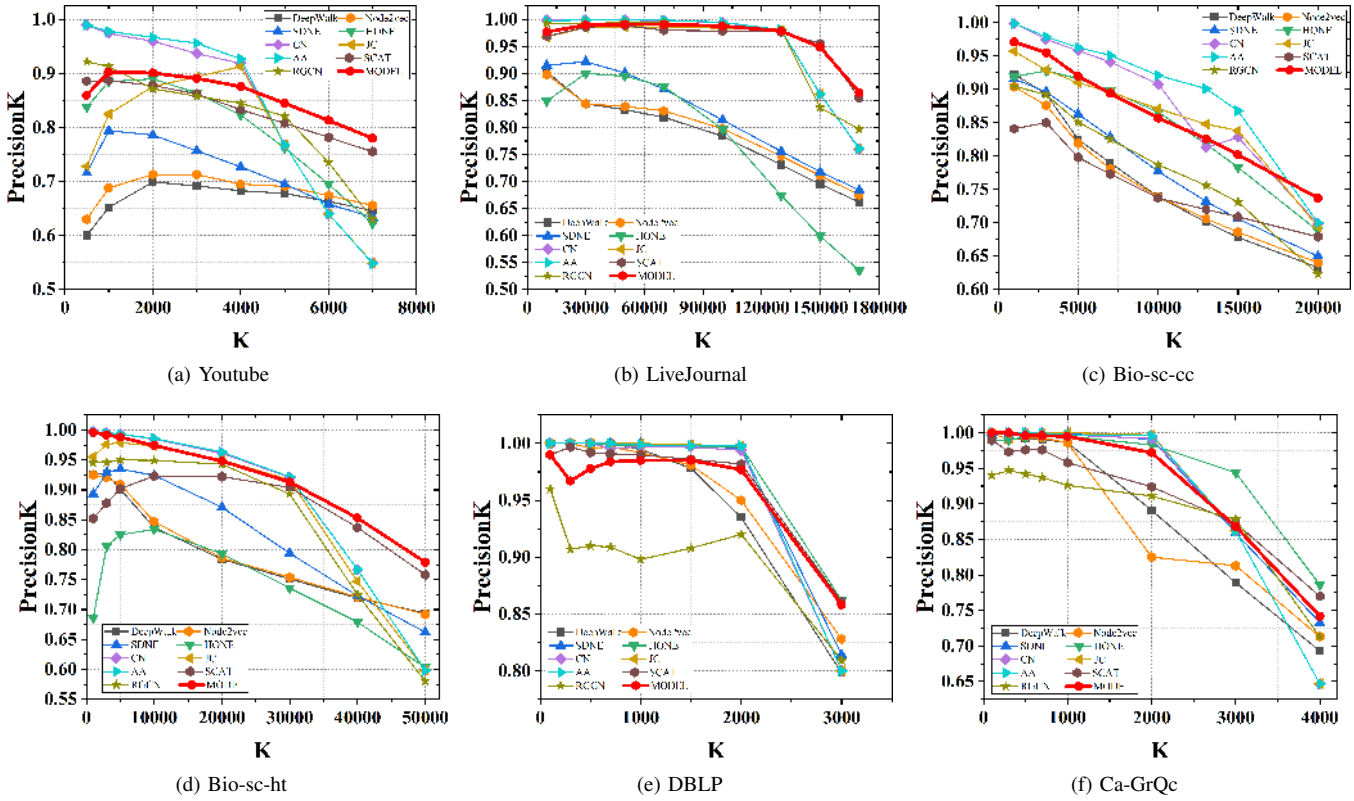
TABLE V: Average number of motifs each vertex resides in

Motif type	Youtube	LiveJournal	Bio-sc-cc	Bio-sc-ht	DBLP	Ca-GrQc
M_{31}	198.23	357.78	323.545	403.01	29.58	36.56
M_{32}	11.83	45.78	50.12	82.33	13.15	22.14
M_{41}	6733.23	18058.09	15118.55	28366.22	519.67	1073.97
M_{42}	55,003.23	25,824.65	43,435.38	50,762.38	946.32	1,550.54
M_{43}	101.78	665.07	359.82	738.77	34.97	66.08
M_{44}	4,768.39	9,777.80	13,718.34	29,609.21	1,063.26	3,002.26
M_{45}	39.11	155.38	179.61	308.36	43.99	82.43
M_{46}	14.81	16.64	163.02	204.42	47.13	174.10

consider motifs by up to 18.1% and 19.5% for AUC, respectively. Moreover, as K increases, MODEL consistently exceeds these algorithms to a large percentage for percisionK. On the two social networks, though considering motifs, HONE has the worst performance in the eight algorithms, particularly on LiveJournal. From Fig. 4, we can see that when K is relatively small, HONE has a good performance, particularly on Youtube. With K increasing, the precisionK of HONE drastically drops. On the two biological networks Bio-sc-cc and Bio-sc-ht, MODEL obtains a maximal gain of 11.6% and 12.4% over these embedding-based algorithms which do not consider motifs, respectively. Besides, MODEL still consistently outperforms the three algorithms for precisionK, which is evidenced by (b) and (c) in Fig. 4. We also notice that HONE outperforms DeepWalk, Node2vec and SDNE, but is inferior to MODEL. In the two academic networks DBLP and Ca-GrQc, the maximal gain over embedding-based algorithms is 6.1% and 5.9% respectively, which is lower than that of social networks and biological networks. On the two networks, MODEL is not that superior to HONE.

Table III and Fig. 4 also show that embedding-based algorithms are comparable to traditional similarity-algorithms for link prediction. Traditional algorithms achieve better performance on LiveJournal and Bio-sc-cc, compared with embedding-based algorithms. It is worth noting that MODEL outperforms the three traditional algorithms over the six networks.

One noticeable finding is that traditional algorithms outperform embedding-based algorithms when K is not very big. Traditional algorithms measure the similarity between two vertices by their neighbors. According to homophily [57], the similarity from neighbors greatly reflects whether two vertices are potentially connected or not. As a result, traditional algorithms give two vertices that share some common neighbors a high similarity score, whereby they can predict accurately some links. However, when vertices share few or no common neighbors, traditional algorithms cannot work. For example, common neighbors give two vertices that are connected but have no common neighbors a similarity score of zero. Thus, traditional algorithms only can predict a portion of links, which explains the phenomenon that their performance abruptly becomes poor when K exceeds a threshold. Examples include Youtube, where their PrecisionKs drop drastically when K is over 4000. Embedding-based algorithms use overall network structure, keeping better balance between vertices that have some and others that have no common neighbors. Although embedding-based algorithms may not give a higher score to two vertices that have more common neighbors, they



(a) Youtube (b) LiveJournal (c) Bio-sc-cc

(d) Bio-sc-ht (e) DBLP (f) Ca-GrQc

Fig. 4: Comparisons of ten algorithms on PrecisionK.

give a reasonable score to two vertices that have no common neighbors. Thus, Embedding-based algorithms outperform traditional algorithms in most cases when K is the maximum value.

F. Influence of Motif Types

In the above section, we show the superiority of our proposed algorithm. In this section, we investigate the influence of motif types on the six networks. The experimental results for AUC are shown in Table IV and the best performance is highlighted in bold.

Table IV shows that the most appropriate motif type is one among M_{42}, M_{43}, and M_{44}. For the six networks, M_{32}, M_{45}, and M_{46} always achieve bad performance. The reasons may be as follows: 1) From Fig. 1, we notice that M_{32}, M_{45}, and M_{46} are close-knit cliques, and even there is not another edge in M_{32} and M_{46}. As a result, leveraging them cannot better capture missing edges than leveraging sparser cliques. 2) The average number of motifs that each vertex resides in can be related to experimental results. To validate the assumption 2, we conduct an experiment to investigate the average number of motifs each vertex resides in. The result is shown in Table V. From Table V, we can see that the three smallest number is M_{32}, M_{45}, and M_{46} in social networks and biological networks. In academic networks, the number of these three motif types is more than M_{43}, but less than M_{42} and M_{44}. From the above two findings, we can conclude that the influence of motifs on experiment is related to their sparsity and number.

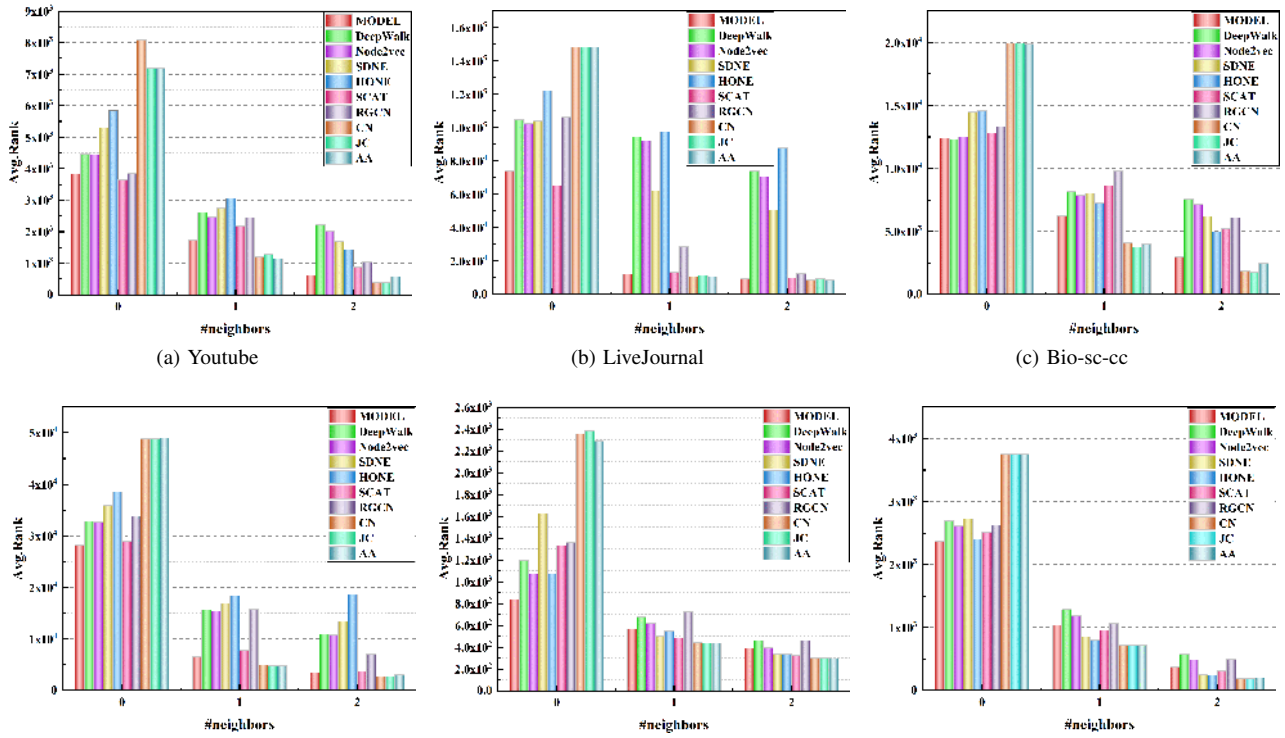
From the above experiments, we know that the number and sparsity of a motif type have an impact on performance. Therefore, we can use a small number of vertices to estimate the number of a sparse motif, such as M_{42}, thereby finding the most appropriate motif type.

G. Weak Ties

For link prediction, a serious problem is how to effectively predict potential weak ties as their two endpoints share a small number of common neighbors. However, the problem is always ignored by existing algorithms. Defining the strength of an edge is beyond the scope of this study, but we can use the number of common neighbors to define the strength of an edge according to [19].

To evaluate the effectiveness in predicting potential weak ties, we first select all hidden edges whose two endpoints share a few common neighbors as positive examples. In this paper, a hidden egde is selected if the number of neighbors shared by its two endpoints is less than three. The new test examples consist of the selected positive examples and all negative examples used for experiments in Section *Experimental Results*. We sort the scores of elements in the new test examples by decreasing, and report the average rank of selected positive examples. A smaller average rank implies a better prediction of weak ties. We show the results in Fig. 5.

From Fig. 5, we can see that when the number of common neighbors is zero, embedding-based algorithms achieve the best performance, and three traditional similarity-based



(a) Youtube (b) LiveJournal (c) Bio-sc-cc

(d) Bio-sc-ht (e) DBLP (f) Ca-GrQc

Fig. 5: Avg. Rank of potential weak ties.

algorithms, Common Neighbors (CN), Jaccard's Coefficient (JC) and Adamic-Adar (AA), achieve always the worst performance. Moveover, MODEL always ranks at the top and achieve the best performance in Bio-sc-ht, DBLP and Ca-GrQc. However, when the number is one or two, traditional algorithms all achieve the best performance. Even so, MODEL still achieves great performance, no big gap compared to traditional algorithms. The above phenomenon may be due to that traditional algorithms only focus on direct neighbors to derive the similarity of two vertices. Thus, when the number of common neighbors is zero, the derived similarity of potentially connected vertices is zero, which is the same as the similarity of most unconnected vertices. MODEL addresses the problem of no common neighbors by redefining vertices' neighbors, thus it significantly outperforms traditional algorithms when no common neighbors exist. If two vertices have common neighbors, the number of common neighbors can be a good indicator for predicting edges. Fig. 5 also shows that MODEL outperforms embedding-based algorithms regardless of the number of common neighbors.

H. Parameter Sensitivity

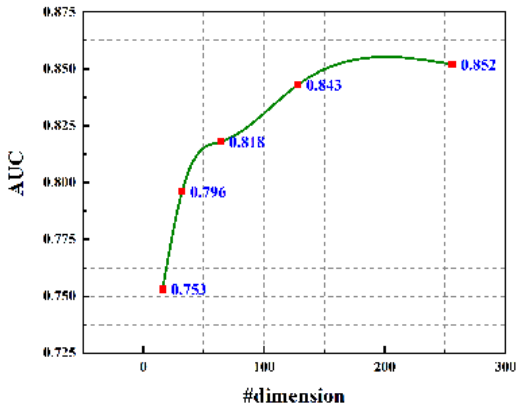
In this section, we investigate the parameter sensitivities of α and the number of dimensions, and report how the two hyper-parameters affect experimental results. we select a network Youtube and motif type M_{43} to conduct experiments. The experimental results are shown in Fig. 6.

The number of embedding dimensions: In experiments, it is necessary to choose an appropriate number of embedding dimensions. A vector space with higher dimensions is more capable of encoding more useful information in original networks. However, it is more difficult to train models to learn higher-dimensional vector representations since time complexity is higher. In this experiment, we choose five different number of dimensions (16, 32, 64, 128, 256). From Fig. 6a, we can see that the value of AUC increases rapidly as the dimension goes from 16 to 128. However, when the dimension increase from 128 to 256, the value increases gently, and is insensitive to the change of dimension. Overall, we can achieve a good performance in a low-dimensional space.

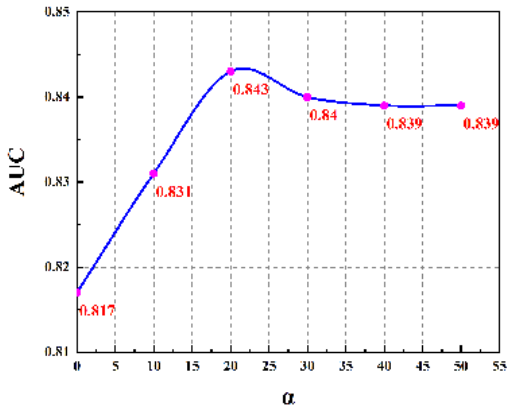
Balance between the first-order proximity and the second-order proximity: In the proposed algorithm, α is used to control the balance between the first-order proximity and the second-order proximity. From Fig. 6b, we can see that the performance of $\alpha > 0$ is better than that of $\alpha = 0$. This is due to the reason that we only preserve the second-order proximity when $\alpha = 0$. With α increasing, the weight of the first-order proximity becomes large gradually, and the performance becomes better. We also see that the performance declines slightly and keeps stable at last when $\alpha > 20$, which demonstrates that we must make a trade-off between the two proximities.

I. Time Complexity

To compare the time complexity of baselines, we run experiments of the running time on a large network Youtube and a small network Bio-sc-cc. Note that we compare the running time over baselines that need to train corresponding

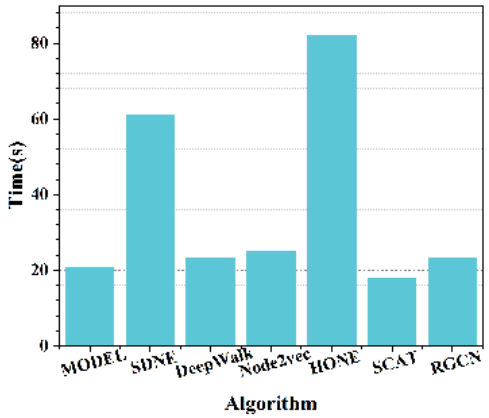


(a) #dimension

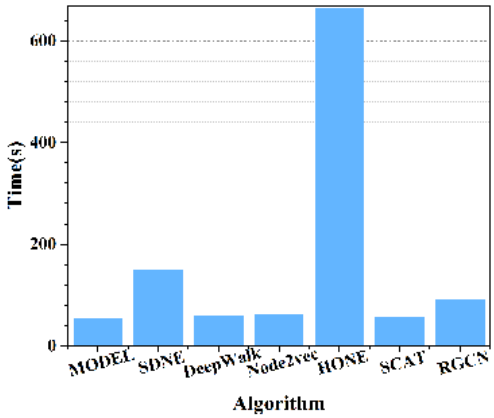


(b) α

Fig. 6: Sensitivity w.r.t. dimension and α



(a) Bio-sc-cc



(b) Youtube

Fig. 7: The running time of seven algorithms

models. Traditional algorithms are omitted here because they directly use the network structure and need no time to train a model. The results are shown in Fig. 7.

From Fig. 7, it can be seen that MODEL has less running time, compared with these baselines. In Bio-sc-cc, the running time of MODEL is only more than that of SCAT, and the time of MODEL is lowest in larger network Youtube. These results suggest that MODEL achieve satisfactory performance in time complexity. We can notice that the running time of HONE is highest in the two networks. Furthermore, in Youtube it significantly exceeds the running time of SDNE, the second most time. This reason is that HONE has a great number of matrix computations and uses all two-node and four-node motifs.

VI. Conclusion

We proposed a motif-based algorithm to learn node embedding for link prediction. The algorithm uses motifs, through defining two levels of proximity in autoencoders, to capture the higher-order structure and alleviate the problem of predicting weak ties. Experiments on six real-world networks, covering social, biological and academic networks, demonstrate that the proposed algorithm significantly outperforms the state-of-the-art baseline algorithms. The maximal gain on AUC over embedding-based algorithms can be up to 19%, and up to 20% over traditional similarity-based algorithms. In addition, we have also investigated the influence of motif types on the link prediction performance, which hasn't studied before. From the experimental results, we obtained insights on how to select the most appropriate motif type for a specific task in networks.

In the future, we will consider sampling importance motifs so as to apply this method to large networks.

References

[1] F. Xia, W. Wang, T. M. Bekele, and H. Liu, "Big scholarly data: A survey," *IEEE Transactions on Big Data*, vol. 3, no. 1, pp. 18–35, 2017.

[2] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[3] X. Liu, D. He, and C. Liu, "Information diffusion nonlinear dynamics modeling and evolution analysis in online social network based on emergency events," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 1, pp. 8–19, 2019.

[4] L. M. Aiello, A. Barrat, R. Schifanella, C. Cattuto, B. Markines, and F. Menczer, "Friendship prediction and homophily in social media," *ACM Transactions on the Web*, vol. 6, no. 2, p. 9, 2012.

[5] X. Kong, F. Xia, Z. Ning, A. Rahim, Y. Cai, Z. Gao, and J. Ma, "Mobility dataset generation for vehicular social networks based on floating car data," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 3874–3886, 2018.

[6] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[7] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[8] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1225–1234.

[9] R. A. Rossi, N. K. Ahmed, and E. Koh, "Higher-order network representation learning," in *Companion Proceedings of the The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 2018, pp. 3–4.

[10] J. Chen, Y. Wu, L. Fan, X. Lin, H. Zheng, S. Yu, and Q. Xuan, "N2vscdnnr: A local recommender system based on node2vec and rich information network," *IEEE Transactions on Computational Social Systems*, 2019.

[11] X. Qian, H. Feng, G. Zhao, and T. Mei, "Personalized recommendation combining user interest and social circle," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 7, pp. 1763–1777, 2013.

[12] B. Xu, Y. Liu, S. Yu, L. Wang, L. Liu, H. Lin, Z. Yang, J. Wang, and F. Xia, "Multipath2vec: Predicting pathogenic genes via heterogeneous network embedding," in *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2018, pp. 951–956.

[13] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[14] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[15] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, no. 6295, pp. 163–166, 2016.

[16] A. Arenas, A. Fernandez, S. Fortunato, and S. Gomez, "Motif-based communities in complex networks," *Journal of Physics A: Mathematical and Theoretical*, vol. 41, no. 22, p. 224001, 2008.

[17] T. Lyu, Y. Zhang, and Y. Zhang, "Enhancing the network embedding quality with structural similarity," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 147–156.

[18] D. Nguyen, W. Luo, T. D. Nguyen, S. Venkatesh, and D. Phung, *Learning Graph Representation via Frequent Subgraphs*, pp. 306–314. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611975321.35

[19] J.-P. Onnela, J. Saramäki, J. vönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, and A.-L. Barabási, "Structure and tie strengths in mobile communication networks," *Proceedings of the national academy of sciences*, vol. 104, no. 18, pp. 7332–7336, 2007.

[20] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "On facebook, most ties are weak," *Communications of the ACM*, vol. 57, no. 11, pp. 78–84, 2014.

[21] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2257–2270, 2018.

[22] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[23] U. Alon, "Network motifs: theory and experimental approaches," *Nature Reviews Genetics*, vol. 8, no. 6, p. 450, 2007.

[24] W. N. Grundy, T. L. Bailey, C. P. Elkan, and M. E. Baker, "Meta-meme: motif-based hidden markov models of protein families," *Bioinformatics*, vol. 13, no. 4, pp. 397–406, 1997.

[25] F. Xia, H. Wei, S. Yu, D. Zhang, and B. Xu, "A survey of measures for network motifs," *IEEE Access*, vol. 7, pp. 106 576–106 587, 2019.

[26] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[27] S. Wernicke, "A faster algorithm for detecting network motifs," in *International Workshop on Algorithms in Bioinformatics*. Springer, 2005, pp. 165–177.

[28] S. Wernicke and F. Rasche, "Fanmod: a tool for fast network motif detection," *Bioinformatics*, vol. 22, no. 9, pp. 1152–1153, 2006.

[29] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[30] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *International Conference on Learning Representations (ICLR)*, 2013.

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[32] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 891–900.

[33] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in neural information processing systems*, 2014, pp. 2177–2185.

[34] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 385–394.

[35] R. A. Rossi, R. Zhou, and N. K. Ahmed, "Deep inductive network representation learning," in *Companion of the The Web Conference 2018 on The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 2018, pp. 953–960.

[36] Z. Wang, C. Chen, and W. Li, "Predictive network representation learning for link prediction," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 969–972.

[37] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, and L. Lin, "A unified framework for community detection and network representation learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 6, pp. 1051–1065, 2018.

[38] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2257–2270, 2018.

[39] D. Zou and G. Lerman, "Encoding robust representation for graph generation." in *International Joint Conference on Neural Networks (IJCNN)*, 2019.

[40] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *15th International Conference on Extended Semantic Web Conference, ESWC 2018*, 2018, pp. 593–607.

[41] J. Chen, J. Zhang, X. Xu, C. Fu, D. Zhang, Q. Zhang, and Q. Xuan, "E-lstm-d: A deep learning framework for dynamic network link prediction," *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 1–14, 2019.

[42] Q. Xuan, J. Wang, M. Zhao, J. Yuan, C. Fu, Z. Ruan, and G. Chen, "Subgraph networks with application to structural feature space expansion." *arXiv preprint arXiv:1903.09022*, 2019.

[43] C. Fu, M. Zhao, L. Fan, X. Chen, J. Chen, Z. Wu, Y. Xia, and Q. Xuan, "Link weight prediction using supervised learning methods and its application to yelp layered network," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 8, pp. 1507–1518, 2018.

[44] Y. Yu, Z. Lu, J. Liu, G. Zhao, and J. rong Wen, "Rum: Network representation learning using motifs," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, pp. 1382–1393.

[45] M. R. Dareddy, M. Das, and H. Yang, "motif2vec: Motif aware node representation learning for heterogeneous networks." *arXiv preprint arXiv:1908.08227*, 2019.

[46] G. AbuOda, G. D. F. Morales, and A. Aboulnaga, "Link prediction via higher-order motif features." *arXiv preprint arXiv:1902.06679*, 2019.

[47] E. Wong, B. Baur, S. Quader, and C.-H. Huang, "Biological network motif detection: principles and practice," *Briefings in bioinformatics*, vol. 13, no. 2, pp. 202–215, 2011.

[48] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[49] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[50] Z. Wang, Y. Zhang, H. Chen, Z. Li, and F. Xia, "Deep user modeling for content-based event recommendation in event-based social networks," in *IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2018, pp. 1304–1312.

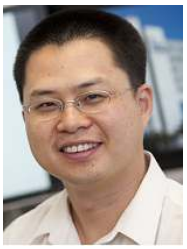
[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.
[52] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.
[53] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
[54] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.
[55] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
[56] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, p. 2, 2007.
[57] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001.

Lei Wang received the BSc degree in software engineering from Dalian University of Technology, China, in 2018. He is currently working toward the master's degree in the School of Software, Dalian University of Technology, China. His research interests include data mining, analysis of complex networks, and machine learning.

Jing Ren received the BSc degree in software engineering from Huaqiao University, Xiamen, China. She is currently pursuing the master's degree in software engineering in Dalian University of Technology, China. Her research interests include big scholarly data, network science, and computational social science.

Bo Xu received the BSc and PhD degrees from the Dalian University of Technology, China, in 2007 and 2014, respectively. She is currently an Associate Professor in School of Software at the Dalian University of Technology. Her current research interests include data science, network analysis and natural language processing.

Jianxin Li received the PhD degree in computer science from the Swinburne University of Technology, Australia, in 2009. He is an Associate Professor in the School of Info Technology, Deakin University. His research interests include database query processing & optimization, social network analytics, and traffic network data processing.



Wei Luo is a Senior Lecturer in Data Science at Deakin University. Wei's recent research focuses on machine learning and its application in health, sports, and cybersecurity. Wei has tackled a number of key information challenges in healthcare delivery and has published more than 50 papers in peer-reviewed journals and conferences. Wei holds a PhD in computer science from Simon Fraser University, where he received training in statistics, machine learning, computational logic, and modern software development.

Feng Xia Feng Xia (M'07-SM'12) received the BSc and PhD degrees from Zhejiang University, Hangzhou, China. He is currently an Associate Professor and Discipline Leader in School of Science, Engineering and Information Technology, Federation University Australia, and on leave from School of Software, Dalian University of Technology, China, where he is a Full Professor. Dr. Xia has published 2 books and over 300 scientific papers in international journals and conferences. His research interests include data science, knowledge management, social computing, and systems engineering. He is a Senior Member of IEEE and ACM.