This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

PSO-ANE: Adaptive Network Embedding With Particle Swarm Optimization

Jinyin Chen, Yangyang Wu, Xuanheng Xu, Haibin Zheng, Zhongyuan Ruan, and Qi Xuan, *Member, IEEE*

***Abstract*— Network embedding plays an important role in various network applications, such as node classification and link prediction. Lots of structure-based network embedding methods have been proposed. Yet, they suffer from unsteady embedding performances due to the parameter sensitivity. How to extract valuable attribute information in networks with less parameter influence is still a challenge. In this paper, we propose a novel algorithm, named adaptive network embedding with particle swarm optimization (PSO-ANE), which is based on the second-order dynamic random walk and PSO method, for learning network representations. A second-order dynamic random walk is designed to search a suitable strategy for each node based on the structure-based transition probability, the centrality-based transition probability, and the static link weights. To reduce the parameter dependence, PSO is adopted for key-parameter optimization to get global steady network embedding. The experiments validate that the proposed method outperforms the existing state-of-the-art techniques on multilabel classification, multiclass classification, and link prediction tasks.**

***Index Terms*— Centrality-based transition probability, network embedding, particle swarm optimization (PSO), second-order dynamic random walk.**

I. Introduction

NETWORK embedding [1], [2], as an efficient way to represent high-dimensional and sparse networks, has been applied to a wide range of areas, such as link prediction [3], [4], node classification [5], [6], clustering [7], [8], and classification [9]. By learning low-dimensional representations for nodes in a network, it overcomes the insuperable difficulties of traditional network representation method but still leaves many challenges due to the complexity of real-world networks [10], [11].

In natural language processing (NLP) [12], [13], Skip-Gram [14] is an efficient technique to learn embeddings for text data. DeepWalk [3] was the first model to adopt a neural language model (Skip-Gram [14]) for network embedding. Specifically, it uses the random walk to sample a sequence of nodes as the context nodes for each node and then treats these sequences of nodes as sentences by the Skip-Gram method [14]. It is supposed that close nodes may have similar contexts and, thus, have similar embeddings. Since then, a series of network embedding methods have been proposed. For instance, LINE [15] can be considered as a special case of DeepWalk, with the window size of contexts being 1. Node2vec [16] can be considered as an extension of DeepWalk, which introduces a biased second-order random walk model to provide more flexibility for generating the context nodes. Struc2vec [17] is also a flexible framework to learn the representations that capture the structural identity of nodes in a network. Graph2Gauss [18] is the first unsupervised approach that represents nodes in attributed networks as Gaussian distributions. In addition, GraphGAN [19], inspired by GAN [20], is a novel framework that unifies generative and discriminative models for graph representation learning.

Though these learning embedding methods are effective in classification tasks, they may fail to capture some valuable information in the network and be vulnerable when the link sparsity problem occurs. For some methods, the parameters setting may have a significant impact on the quality of network embedding. Taking node2vec as an example, the returning parameter R and the in–out parameter q balance the search strategy between breadth-first-search (BFS) and depth-first-search (DFS), i.e., they adjust the walk toward different network exploration strategies. The optimal values of R and q vary dramatically when node2vec is applied to different tasks, and it is always difficult to set in advance. Moreover, to the best of our knowledge, for all of these embedding methods, each node in a network adopts the same search strategy without considering its own structural properties. However, real-world networks are always heterogeneous [21], [22]; in this case, different nodes may need different searching strategies to improve the embedding effect.

In this paper, we propose an adaptive network embedding (ANE) method, in which different searching strategies are adopted for different nodes based on a second-order dynamic random walk. The obtained feature representations of ANE can maximize the likelihood of preserving network neighborhoods of nodes in a d-dimensional feature space. In particular, for each node, the search strategy is determined by its structure-based transition probability, the centrality-based transition probability, and the static link weights, based on which the context for each node is generated. Particle swarm optimization (PSO) [23], [24] is then applied to optimize the parameters. The main contributions of this paper are as follows.

Manuscript received August 27, 2018; revised January 8, 2019 and April 15, 2019; accepted April 30, 2019. This work is partially supported by Zhejiang Natural Science Foundation (LY19F020025), Integration and Application of Human-computer Fusion Intelligent Image Recognition Algorithm (2018B10063), National Natural Science Foundation of China (61502423, 61572439), Engineering Research Center of Cognitive Healthcare of Zhejiang Province, Zhejiang Science and Technology Plan Project (LGF18F030009, 2017C33149), and Zhejiang Outstanding Youth Fund (LR19F030001), Key technologies, system and application of Cyberspace Big Search, Major project of Zhejiang Lab (2019DH0ZX01). *(Corresponding authors: Jinyin Chen; Zhongyuan Ruan.)*

The authors are with the College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China (e-mail: chenjinyin@zjut.edu.cn; 2111603080@zjut.edu.cn; 2111603112@zjut.edu.cn; 201303080231@zjut.edu.cn; zyruan@zjut.edu.cn; xuanqi@zjut.edu.cn).

Digital Object Identifier 10.1109/TCSS.2019.2914935

2329-924X © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

1) We propose a new network embedding method, namely, ANE, based on a second-order dynamic random walk. In ANE, node centrality is used to guide the search process. Thus, each node has its unique search strategy according to the node centrality distribution of its first-order neighbors, which can reflect the global and local features of each node in the network simultaneously.
2) We adopt the PSO algorithm to determine the optimal parameters, where a novel Fitness function is proposed based on multiple indicators. This can make the whole process more automatic and robust for different real-world applications.
3) The experiments show that the PSO-ANE method is better than those state-of-the-art network embedding methods in the tasks, such as multilabel classification, multiclass classification, and link prediction on real-world network data sets. Furthermore, based on PSO-ANE, we also propose two special variants, called PSO-ANE-DW and PSO-node2vec, which outperform the competitors on some data sets.

The rest of this paper is organized as follows. In Section II, we survey the related work in network embedding and search strategies with network centrality. We then present the details of the ANE algorithm in Section III and empirically compare it with other embedding algorithms on prediction tasks over nodes and links in various networks in Section IV. Finally, we conclude this paper and highlight some promising research directions for future work in Section V.

II. Related Works

In this section, we briefly introduce the recent studies on network embedding and search strategies, which are related to our current work.

A. Network Embedding

Network embedding has received much attention over the past decades. Some earlier works, such as multidimensional scaling [25], local linear embedding [26], IsoMAP [27], and the Laplacian eigenmap [28], tried to transform the network into a similarity graph and then embed this similarity graph by solving the eigenvectors of the similarity matrix.

Quite recently, more and more researchers have focused on the methods of transforming each node into a low-dimensional vector based on its local structure in the network. For instance, DeepWalk [3] uses the random walk to sample context nodes for each node and tries to maximize the log-likelihood of observing context nodes for the source node. LINE [15] learns half features by breadth-first sampling over first-order near-neighbors of nodes and also learns the other half features by sampling the first-order near-neighbors of nodes. Perozzi [3] used random walks to generate sequences of nodes from the network. The sequences of nodes are treated as text in language models, based on which one can learn the embeddings by the Skip-Gram model. The idea, closer nodes in the network space tend to have more similar contexts and have embedding nearer one another, was later introduced to node2vec [16] as a biased second-order random walk model. Node2vec provides more flexibility when generating the context of a node. The link weights driving the biased random walks can be designed in an attempt to capture both node homophily and structural equivalence. GraRep [29], as a matrix factorization-based embedding method, employs positive pointwise mutual information matrix as the preprocessing based on a proof of the equivalence between a k-step random walk in DeepWalk and a k-step probability transition matrix. In graph2vec, Narayanan *et al.* [30] proposed to view an entire graph as a text and the subgraphs around each node as words and extend the Skip-Gram model to learn representations. The struc2vec method [17] is a general framework for learning latent representations for the structural identity. It constructs a multilayer graph to encode structural similarities and generate a structural context for nodes. Defferrard *et al.* [31] and Kipf and Welling [32] proposed a multilayer graph convolutional network, which uses an efficient layerwise propagation rule based on a first-order approximation of spectral convolutions on graphs. GraphGAN [19] is another novel graph representation learning framework unifying generative and discriminative models by a minimax game.

However, these algorithms fail to capture some valuable attribute information in networks, resulting in less informative embeddings. In addition, these methods get affected easily when the link sparsity problem occurs. Considering these problems, we propose a second-order dynamic search algorithm that measures the values of nodes based on their centrality in the network to guide the search process.

B. Search Strategies With Network Centrality

In network embedding, probabilistic embedding methods (including DeepWalk [3] and node2vec [16]) use the Skip-Gram model [14] to learn network embedding. Due to the nonlinearity of the network [33], search strategies are used to define the concept of the near-neighbors and extract the local structural features for each node in the network. The present search strategies in network embedding use a single-search strategy to sample the first-order neighbors for each node that reduces the reliability of the search strategy. Moreover, these embedding methods ignore the impact of some valuable information [34] in the network on the search strategy. Thus, it is meaningful to develop a more suitable search strategy for network sampling.

In network analysis [35], various centrality measures [36]–[38] were introduced in order to understand both the global dynamics of the network and the roles of the individual nodes. De Domenico *et al.* [39] proposed a mathematical framework to calculate the centrality in a network and found the nodes that play the most central roles in the cohesion of the whole structure. In order to extend the eigenvector-based centrality measures to time-dependent networks, Taylor *et al.* [40] introduced a principled generalization of network centrality measures, which is valid for any eigenvector-based centrality. Li *et al.* [41] proposed a novel method to identify essential proteins by integrating the protein complexes with the centrality features of protein–protein interaction networks.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Here, we propose a novel search strategy, called second-order dynamic random walk. In this algorithm, each node has its unique search strategy according to its node centrality distribution of the first-order neighbors, which, thus, can reflect the global and local features of each node in the network simultaneously.

III. ANE With PSO

Similar to node2vec [16], we extend the Skip-Gram method to network and consider the feature learning in a network as a kind of maximum likelihood optimization problem.

A. Basic Definitions

Our analysis could be applied to any (un)directed, (un)weighted networks. Given a network $G = (V, E, W)$, $f : V \to R^d$ is the mapping function from nodes to d-dimensional feature vectors. For a node $v \in V$, its near-neighbor set $N_s(v)$ is generated according to the search strategy s. Our objective function is designed to maximize the log-probability of observing a network near-neighbor set $N_s(v)$ for the source node v, given the mapping feature $f(v)$

$$\max_f \sum_{v \in V} log(Pr(N_s(v)|f(v))). \tag{1}$$

Based on the network analysis, we can see that the likelihood of observing a near-neighbor is independent of observing any other and the definition of nearest-neighbor is symmetric, i.e., [A is a nearest-neighbor of B] $\Leftrightarrow$ [B is a near-neighbor of A]. Therefore, we factorize the likelihood of observing near-neighbors and model the likelihood of each near-neighbor pair as a softmax unit that is parametrized by a dot product of their mapping features. Then, the probability of observing the near-neighbors of node v can be represented as

$$\Pr(N_s(v)|f(v)) = \prod_{n_i \in N_s(v)} \frac{e^{f(n_i)\cdot f(v)}}{\sum_{\theta \in V} e^{f(\theta)\cdot f(v)}} \tag{2}$$

where $n_i \in N_s(v)$ is the ith near-neighbor of node v. Substituting (2) into (1), we have

$$\max_f \sum_{v \in V} \left[-|N_s(v)| \cdot \log(Z_v) + \sum_{n_i \in N_s(v)} f(n_i) \cdot f(v) \right] \tag{3}$$

with $Z_v = \sum_{\theta \in V} e^{f(\theta)\cdot f(v)}$. Considering the complexity of this function, we use negative sampling strategy [42] to approximate it, and the objective can be efficiently optimized using the stochastic gradient descent (SGD). Due to the nonlinear nature of real-world networks, we define a novel search strategy for nearest-neighbor sampling in the following.

B. ANE Method

Our ANE method is based on the second-order dynamic random walk. For each node in the network, a unique search strategy is adopted according to its structure-based transition probability, the centrality-based transition probability, and the static link weights.

First, we perform a random walk of fixed length for a given source node v. If n_i is the ith node in the walk, then node n_{i+1} is generated based on the following probability:

$$P_{cx} = P(n_{i+1} = x | n_i = c) = \frac{w_{cx}}{\sum_{k \in N_s(c)} w_{ck}} \tag{4}$$

where w_{cx} is the unnormalized transition probability between nodes c and x.

The simplest strategy for searching the next node could be based on its link weight to the source node. However, this strategy may get affected easily when the links are relatively sparse in the network. The Node2vec method develops a second-order random walk that can explore neighborhoods in a mixed fashion of BFS and DFS. However, this strategy ignores some important local structural properties, such as node centrality, thus may lead to incomplete network representation.

Typically, the first-order near-neighbors of each node can be divided into three categories: inward nodes, outward nodes, and return node. The return node represents the last selected node, and the inward (or outward) nodes are those linked (or not linked) with the return node. These different kinds of nodes may play different roles in the network and, thus, present a different amount of information. Taking their roles into consideration, for each node's next walk, the feature distribution of all its first-order near-neighbors will be considered. Moreover, the importance of a node in the network depends on the position of that node. We assume that more central nodes are more important. Thus, the search direction of the source node is more inclined to the neighbors of more importance, which may result in more informative embeddings.

With these assumptions, we propose a second-order dynamic random walk. Suppose a random walk resides at node c at present. We need to calculate the transition probabilities of its first-order near-neighbors. In the second-order dynamic random walk, we propose a double-layer transition probability that is composed of the static link weights, the structure-based transition probability, and the centrality-based transition probability between the source node and its first-order near-neighbors.

1) Structure-Based Transition Probability: The same as node2vec, we define the structure-based transition probability P^1 between the source node c and its first-order near-neighbors x with the in–out parameter q and the return parameter R, which is calculated as follows:

$$P^1_{cx} = \begin{cases} 1/q, & d_{tx} = 2 \\ 1, & d_{tx} = 1 \\ 1/R, & d_{tx} = 0 \end{cases} \tag{5}$$

where d_{tx} denotes the shortest path distance between nodes t and x and $d_{tx} \in \{0, 1, 2\}$.

That is to say, the performances of the node2vec method and the ANE method are particularly dependent on the selection of the in–out parameter q and the return parameter R.

2) Centrality-Based Transition Probability: In order to introduce more information into the embedding algorithm, we propose a centrality-based transition probability based on the node centrality distribution of the first-order nodes. In networks, the most direct measure of node centrality is

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

degree centrality—-the greater the degree of a node, the more important the node is.

According to the node centrality distribution of first-order nodes of node c, we define the dynamic in–out parameter q_c

$$q_c = \frac{\left|\left\{x \in O_c | D_x > \sum_{j \in L_c} \frac{D_j}{|L_c|}\right\}\right|}{\left|\left\{x \in O_c | D_x < \sum_{j \in L_c} \frac{D_j}{|L_c|}\right\}\right|}, \quad 0 \tag{6}$$

where L_c and O_c are the sets of inward and outward nodes for node c, respectively. In a network with N nodes, the normalized degree centrality of node c of degree d_c is defined as

$$D_c = \frac{d_c}{N-1}. \tag{7}$$

Finally, according to the dynamic in–out parameter q_c for a given source node c, we define the centrality-based transition probability for all kinds of near-neighbors as follows:

$$P_{cx}^2 = \begin{cases} 1/q_c, & d_{tx} = 2 \\ 1, & d_{tx} = 1 \\ 1, & d_{tx} = 0. \end{cases} \tag{8}$$

3) Double-Layer Transition Probability: We propose the double-layer transition probability P^3 (see Fig. 1), which simultaneously considers the structure-based transition probability, the centrality-based transition probability, and the static link weights for each node.

For each node c, we set the double-layer in–out parameter $q_c^3 = \frac{q_c - q}{\theta} + q$, where θ is the convergence rate used to control the distribution of the ratio. It is worth noticing that the initial in–out parameter value q determines searching directions and the dynamic in–out parameter value q_c guides the search procedure toward the best near-neighbor.

According to the above-defined parameters, we then define the double-layer transition probability P^3 as

$$P_{cx}^3 = \begin{cases} P_{cx}/q_c^3, & d_{tx} = 2 \\ P_{cx}, & d_{tx} = 1 \\ P_{cx}/R, & d_{tx} = 0 \end{cases} \tag{9}$$

where P_{cx} is the static link weight between nodes c and x.

Based on the double-layer transition probability and the random walk algorithm, we propose the second-order dynamic random walk that determines a suitable search strategy for each node based on its feature distribution of all first-order near-neighbors. In the double-layer transition probability, the in–out and return parameters in the structure-based transition probability directly affect the direction of the node's walk, and the centrality-based transition probability may adjust it according to the centrality distribution of all first-order near-neighbors. Return parameter controls the possibility of revisiting the return node in the walk for the source node. However, due to the fact that the outward nodes are much more than the return nodes in general cases, the influence of the parameter R on the embedding vector is relatively small. The in–out parameter q determines the search direction of the node's walk. If $q > 1$, the search directions of most nodes bias toward the inward ones, which is similar to the BFS behavior. If $q < 1$, the walk is more inclined to visit outward nodes.

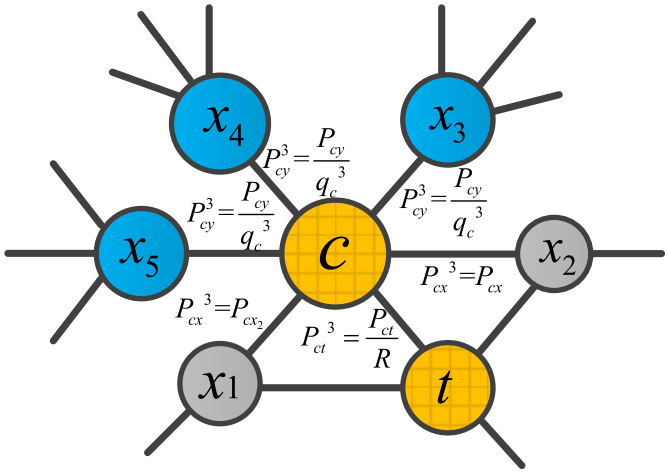


Fig. 1. Illustration of calculating the double-layer transition probability. For a source node c, node t represents the return node, nodes x_1 and x_2 are inward nodes that are first-order near-neighbors of c and have links with t, where $d_{tx_1} = d_{tx_2} = 1$, and x_3, x_4, and x_5 are outward nodes that are first-order near-neighbors of c and have no links with t, where $d_{tx_3} = d_{tx_4} = d_{tx_5} = 2$. The size of the node is proportional to its degree centrality.

Such behavior is reflective of DFS, which encourages outward exploration. The convergence rate θ is used to control the distribution of the ratio q_c^3. The greater the θ value, the smaller the effect of node centrality on the transition probability.

The pseudocode for ANE is given in Algorithm 1. We simulate r random walks of fixed length l starting from each node, and the sampling is done based on the double-layer transition probability P^3.

C. PSO-ANE Method

In the ANE method, there are three critical keyword parameters, namely, the in–out parameter q, the return parameter R, and the convergence rate θ, which could directly affect

Algorithm 1 ANE Method

Input: $G = (V, E, W)$, return R, in–out q, convergence rate θ, dimensions d, each node walks r, the length of walk l, the size of context window k.
Output: $f(R, q, \theta)$.

```
P^3 = ModifiedWeights(G, q, R, θ);
G' = (V, E, P^3);
for iteration = 1 to r do
    for all nodes c ∈ V do
        for walk-iteration = 1 to l do
            node = walk[−1];
            V_node = GetNeighbors(node, G');
            s = AliasSample(V_node, P^3);
            Append s to walk;
        end
        Append walk to walks;
    end
end
f(R, q, θ) = SGD(k, d, walks);
return f(R, q, θ).
```

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

the quality of embedding vector. The optimal parameters for different tasks are quite different. How to choose the optimal parameters will play a decisive role in network embedding. We consider the parameter selection problem as an optimization problem and adopt PSO to find out the optimum or suboptimum for related downstream algorithms. In our recent work, we proposed a method, namely, PSO-ANE, which is introduced PSO for three critical keyword parameters' selection on ANE.

1) Swarm Initialization: In our PSO, each particle represents three critical keyword parameters. For the ith particle in the swarm, its initial position λ_i^0 is randomly sampled from a uniform distribution, bounded by the parameter lower and upper limits. Similarly, the velocity v_i^0 of this particle is randomly drawn from a uniform distribution. Afterward, the particles undergo the evolution.

2) Swarm Evaluation: After swarm initialization, the second step of PSO is evolution. In the $(t+1)$th evolution, the velocity and position of the jth dimension in the ith particle is updated according to the following equations:

$$v_{ij}^{t+1} = \omega v_{ij}^t + \phi_l r_l \left(L_i^t - \lambda_{ij}^t\right) + \phi_m r_m \left(M^t - \lambda_{ij}^t\right) \quad (10)$$

$$\lambda_{ij}^{t+1} = \lambda_{ij}^t + v_{ij}^t \quad (11)$$

where r_l and r_m are applied to the velocity updates to diversify the search, ω is an inertia weight scaling factor, and ϕ_l and ϕ_m are the acceleration coefficients. After this, in the $(t+1)$th evolution, the personal best position L_i^{t+1} for the ith particle and the global best position M^{t+1} in the swarm are updated according to the Fitness value of all particles. The evolution proceeds until the termination condition is reached. Finally, the network representation with optimal set of parameters is returned.

3) Fitness Evaluation: Fitness function is the guidance to the convergence of particles in PSO. In PSO-ANE, we need to design an evaluation indicator for the embedding vector. Most of network embedding methods employ final results of the downstream algorithm to evaluate the quality of embedding. Therefore, a Fitness function based on the downstream algorithm (a supervised learning algorithm) is adopted.

We mainly consider the node classification and the link prediction tasks as supervised learning in this paper. Taking node classification as an example, we design the multiple evaluation indicators for classification results as the final Fitness function. It also works out for multilabel classification tasks (multilabel classification is a generalization of multiclass classification). In different application scenarios, the algorithm has different emphases on the F1-macro and F1-micro values [43]. In order to consider the relative influence of these evaluation indicators, we define a Fitness function based on the weighted harmonic mean of the F1-macro and F1-micro values

$$\text{Fitness} = \frac{(1+\beta^2)\text{F1-macro} \times \text{F1-micro}}{\beta^2(\text{F1-macro} + \text{F1-micro})} \quad (12)$$

where $\beta > 0$ is a weight parameter to balance the importance of the F1-macro and F1-micro values. If $\beta > 1$ ($\beta < 1$), the F1-macro (F1-micro) value is relatively more important than the F1-micro (F1-macro) value in evaluating the quality of particles. If $\beta = 1$, the two values are equally important.

Algorithm 2 PSO-ANE Method

Input: $G = (V, E, W)$, the maximum iteration T.
Output: the network representation $f(M^T)$ with optimal set of parameters.

```
Randomly initialize the initial position λ^0 and velocity v^0 of each particle;
Initialize the initial person best position L^0 and global best position M^0 with initial position of each particle;
for t=1 to T do
    Update the position and velocity of each particle λ^t according to Eq. 10 and Eq. 11;
    Calculate the network representation of each particle f(λ^t) = ANE(G, λ^t), and its Fitness value;
    Update L_i^t of ith particle and M^t based on the Fitness value of all particles;
end
return the network representation f(M^T) with optimal set of parameters.
```

Each iteration of the particle population will be evaluated by the Fitness function, and the global best particle will be kept when the iteration ends. According to the PSO, the optimal pair of parameters are selected according to the optimal Fitness value.

The pseudocode for PSO-ANE is given in Algorithm 2.

IV. Experiments

In order to testify the efficiency of the PSO-ANE, comprehensive experiments are carried out and compared with other classic embedding methods. Our method is implemented in python, and the experimental results are analyzed with explanations.

A. Experiment Setup

For both node classification and link prediction tasks, we compare our model with the state-of-the-art embedding methods, including DeepWalk, LINE, struc2vec, Graph2Gauss, GraphGAN, PSO-ANE-DW, and PSO-node2vec. The descriptions of these methods are as follows.

1) *Spectral Clustering [44]:* This method generates a representation from the d-smallest eigenvectors, the normalized Laplacian matrix of the network.
2) *DeepWalk [3]:* DeepWalk [3] first transforms the network into node sequences by random walk and then uses it as an input to the Skip-Gram model to learn representations.
3) *LINE [15]:* LINE can preserve both first- and second-order proximities for the undirected network through modeling node co-occurrence probability and node conditional probability.
4) *Struc2vec [17]:* Struc2vec is a novel method to learn representations based on capturing the structural identity of nodes in a network.
5) *Graph2Gauss [18]:* Graph2Gauss is the first unsupervised approach that represents nodes in attributed networks as the Gaussian distributions. Since most of

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

data sets have no attributes, we compare Graph2Gauss without considering attributes to other network embedding methods.

6) *GraphGAN [19]:* GraphGAN unifies two schools of graph representation learning methodologies via adversarial training in a minimax game.
7) *PSO-ANE-DW:* DeepWalk is the first model to adopt a neural language model for network embedding and can be seen as a special case of node2vec. Thus, we also propose a special case of PSO-ANE method, namely, PSO-ANE-DW, which sets the return parameter $R = 1$ and the in–out parameter $q = 1$.
8) *PSO-node2vec:* node2vec [16] develops a second-order biased random walk procedure to explore the neighborhood of a node, which can strike a balance between local properties and global properties of a network. The PSO-node2vec is a method that selects the return parameter and the in–out parameter using the PSO method.

In the node classification task, each node is assigned one or more labels. In the training phase, we utilize a certain percentage of labeled nodes. The node classification task is to predict the remaining node labels. We utilize the following multiple multilabel data sets and multiclass data sets in our experiments, and the basic statistics of the data sets are summarized in Table I.

Multilabel Data Sets: Multilabel data sets are described as following.

1) *BlogCatalog [45]:* This is a network that shows the social relationships of the bloggers in the BlogCatalog website. This network has 10 312 bloggers, 333 983 links, and 39 different labels.
2) *Protein–Protein Interactions [46]:* This is a subgraph of the protein–protein interactions network for *Homo Sapiens*. The related subgraph has 3890 nodes, 76 584 links, and 50 different labels.
3) *Wikipedia [47]:* Wikipedia is a co-occurrence network of words appearing in the first million bytes of the Wikipedia dump. It has 4777 nodes, 184 812 links, and 40 different labels.

Multiclass Data Sets: Multilabel data sets are described as following.

1) *Cora [48]:* The Cora data set contains a number of machine-learning papers divided into one of seven classes. The final corpus has 2708 documents, 1433 distinct words in the vocabulary, and 5429 links in the case of Cora.
2) *Citeseer [48]:* Citeseer is a paper citation network. It consists of 3312 scientific publications classified into one of six classes and 4732 links.
3) *Wiki [49]:* The Wiki data set is provided by the LBC project. This data set consists of 2405 nodes, 17 981 links, and 17 labels. This is a network with nodes as web pages and links as the hyperlinks between web pages.

1) Parameter Setting: For the node classification task, the parameter settings used for the ANE and PSO-ANE methods are in line with the typical values used for LINE and DeepWalk. Specifically, we set the dimensions $d = 128$, the number of walks per node $r = 10$, the length of walk $l = 80$, and the size of context window $k = 10$. According to the analysis in Section III-C, we prefer to set $\beta = 1$. For the experiments of each network embedding method, we repeat it five times and report the average performance in terms of both F1-micro and F1-macro.

TABLE I
BASIC STATISTICS FOR THE FOUR NETWORK DATA SETS

Dataset		#Nodes	#Links	#Classes
Multi-label	BlogCatalog	10,312	333,983	39
	PPI	3,890	76,584	50
	Wikipedia	4,777	184,812	40
Multi-class	Cora	2,708	5,429	7
	Citeseer	3,312	4,732	6
	Wiki	2,405	17,981	17

B. Node Classification

In node classification, we observe a certain percentage of labeled nodes and aim to predict labels for the remaining nodes. Therefore, the performance of node classification can reveal the distinguishing ability of nodes under different representation learning methods.

We evaluate the quality of the obtained features on the node classification task with multiple representation learning methods. In the sampling phase, the parameters for all comparison methods are set such that they generate an equal number of samples at runtime, and the node feature representations are input to a logistic regression classifier to perform node classification with 9:1 train–test ratio. For PSO-ANE, PSO-node2vec, and PSO-ANE-DW methods, the validation sets $|V|$ are extracted from the training set $|T|$, $|T| = 9|V|$. The training set T is used for the classifier training, while V is exploited to find the Fitness during the PSO optimization. We use the F1-micro and F1-macro to compare the performance of multiple methods in Tables II and III.

Tables II and III show the experimental results of multiple representation learning methods for multilabel data sets and multiclass data sets, respectively. From them, we can see that the PSO-ANE method outperforms all the baselines in node classification in all the cases, in terms of higher F1-micro and F1-macro, while the spectral clustering (SC) method performs worst. For the BlogCatalog data set, the PSO-ANE method improves F1-micro by 0.07%–83.81% and improves F1-macro by 0.55%–310.96%. For the PPI data set, the PSO-ANE method improves F1-micro by 0.33%–24.42% and improves F1-macro by 1.36%–79.52%. For the Wikipedia data set, the PSO-ANE method improves F1-micro by 0.07%–33.52% and improves F1-macro by 3.51%–170.52%. For the Cora data set, the PSO-ANE method improves F1-micro by 0.12%–27.44% and improves F1-macro by 0.35%–29.76%. For the Citeseer data set, the PSO-ANE method improves F1-micro by 0.49%–22.48% and improves F1-macro by 0.67%–22.49%. For the Wiki data set, the PSO-ANE method improves F1-micro by 0.61%–69.40% and improves F1-macro by 0.18%–77.26%.

Comparing the classification results among DeepWalk, PSO-ANE-DW, PSO-node2vec, and PSO-ANE, we can

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE II
CLASSIFICATION RESULTS ON THE TWO MULTILABEL DATA SETS

Model	Multi-label Data Sets					
	BlogCatalog		PPI		Wikipedia	
	F1-micro	F1-macro	F1-micro	F1-macro	F1-micro	F1-macro
SC	0.2242	0.0584	0.1978	0.1123	0.4135	0.0502
LINE	0.2354	0.0913	0.2027	0.1586	0.5061	0.1165
DeepWalk	0.3988	0.2263	0.2244	0.1875	0.5223	0.1202
Struc2vec	0.4005	0.2268	0.2378	0.1912	0.5345	0.1223
Graph2Gauss	0.4050	0.2206	0.2187	0.1892	0.5405	0.1231
GraphGAN	0.4089	0.2310	0.2453	0.1989	0.5414	0.1234
PSO-ANE-DW	0.4023	0.2284	0.2330	0.1894	0.5386	0.1254
PSO-node2vec	0.4118	0.2387	0.2382	0.1975	0.5517	0.1312
PSO-ANE	**0.4121**	**0.2400**	**0.2461**	**0.2016**	**0.5521**	**0.1358**

TABLE III
CLASSIFICATION RESULTS ON THE TWO MULTICLASS DATA SETS

Model	Multi-class Data Sets					
	Cora		Citeseer		Wiki	
	F1-micro	F1-macro	F1-micro	F1-macro	F1-micro	F1-macro
SC	0.6523	0.6243	0.5045	0.4883	0.4066	0.3465
LINE	0.7912	0.7698	0.5646	0.5496	0.5768	0.5090
DeepWalk	0.8123	0.7863	0.5834	0.5693	0.6472	0.5511
Struc2vec	0.8172	0.7864	0.5964	0.5701	0.6693	0.5902
Graph2Gauss	0.8164	0.7904	0.5950	0.5743	0.6703	0.5926
GraphGAN	0.8254	0.7944	0.5968	0.5771	0.6846	0.6095
PSO-ANE-DW	0.8210	0.7898	0.6024	0.5840	0.6735	0.6021
PSO-node2vec	0.8303	0.8073	0.6150	0.5941	0.6843	0.6131
PSO-ANE	**0.8313**	**0.8101**	**0.6180**	**0.5981**	**0.6888**	**0.6142**

TABLE IV
PERFORMANCE EVALUATION OF NETWORK EMBEDDING METHODS ON A DIFFERENT AMOUNT OF TRAINING (LABELED) DATA ON THE PPI DATA SETS

Model	30%		60%		90%	
	F1-micro	F1-macro	F1-micro	F1-macro	F1-micro	F1-macro
SC	0.1150	0.0562	0.1686	0.1002	0.1978	0.1123
LINE	0.1703	0.1268	0.1989	0.1502	0.2027	0.1586
DeepWalk	0.1798	0.1560	0.2140	0.1750	0.2244	0.1875
Struc2vec	0.1804	0.1582	0.2200	0.1748	0.2378	0.1912
Graph2Gauss	0.1754	0.1356	0.2020	0.1739	0.2187	0.1892
GraphGAN	0.1857	0.1602	0.2246	0.1804	0.2453	0.1989
PSO-ANE-DW	0.1810	0.1578	0.2188	0.1765	0.2330	0.1894
PSO-node2vec	0.1835	0.1592	0.2209	0.1800	0.2382	0.1975
PSO-ANE	**0.1873**	**0.1611**	**0.2262**	**0.1812**	**0.2461**	**0.2016**

find that the method with double-layer transition probability performs better than that with the structure-based transition probability. We first compare the classification results between the DeepWalk method and the PSO-ANE-DW method in multiple data sets. For all data sets, the PSO-ANE-DW method outperforms the DeepWalk method. Specifically, the PSO-ANE-DW improves F1-micro by 0.88%–5.71% and improves F1-macro by 0.45%–9.25%. Next, we compare the classification results between the PSO-node2vec method and the PSO-ANE method in these data sets. For all data sets, the PSO-ANE method outperforms the PSO-node2vec method. Specifically, the PSO-ANE improves F1-micro by 0.07%–3.32% and improves F1-macro by 0.13%–3.51%.

Then, for more fine-grained analysis, we show the performance evaluation of different network embedding methods on a different amount of training (labeled) data (30%, 60%, and 90%) on the PPI data sets. All results are shown in Table IV, where we can see that PSO-ANE consistently outperforms baselines in terms of higher F1-micro and F1-macro. In addition, all methods significantly outperform SC, and PSO-ANE-DW achieves large improvement over DeepWalk. For example, PSO-ANE-DW achieves the biggest improvement over DeepWalk of 1.27% at 90% labeled data, and PSO-ANE achieves the biggest improvement over GraphGAN of 0.39% at 90% labeled data.

1) Searching the Optimal Parameters Using PSO: There are some key parameters in the ANE and node2vec methods. The choice of parameters may directly affect the quality of the vector generated by related algorithms. We use the PSO method to select the optimal return parameter R, in–out parameter q, and convergence rate θ for the ANE method and also to select the optimal return parameter R and in–out parameter q for the node2vec method. In this paper, we set the value of return parameter R in the range $[0.1, 10]$, the value of in–out parameter q in the range $[0.1, 10]$, and the

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

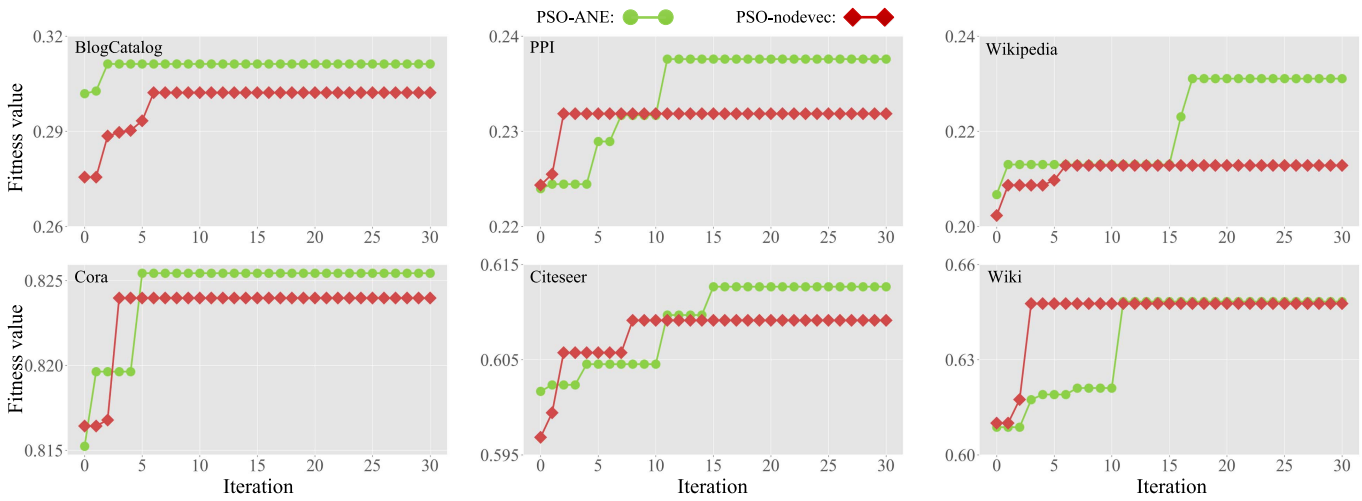


Fig. 2. Convergence curves of the globally optimal particle's Fitness for multiple data sets.

TABLE V
OPTIMAL PARAMETERS FOR PSO-ANE AND PSO-NODE2VEC METHODS

Data sets	PSO-ANE			PSO-node2vec	
	R	q	θ	R	q
BlogCatalog	0.14	0.15	0.32	7.04	0.23
PPI	6.66	0.12	1.22	5.66	2.67
Wikipedia	9.94	0.16	0.28	4.65	0.87
Cora	0.23	4.20	0.74	5.57	2.62
Citeseer	3.71	3.65	0.25	2.63	4.78
Wiki	6.62	9.89	9.54	0.25	5.06

value of convergence rate θ in the range $[0.1, 5]$. For the PSO algorithm, the value of Fitness function, corresponding to the global optimal particle in the iterative process, can best reflect the convergence effect of the algorithm.

We show the convergence curves of the globally optimal fitness value for multiple data sets with PSO-node2vec and PSO-ANE methods in Fig. 2. The data sets for this experiment include BlogCatalog, PPI, Wikipedia, Cora, Citeseer, and Wiki. In Fig. 2, the blue and red lines show the convergence trend of the PSO-ANE and PSO-node2vec methods, respectively. Comparing the two, we can find that for all the six data sets, the performance of the PSO-ANE method outperforms the PSO-node2vec method.

The optimal parameters for the PSO-ANE and PSO-node2vec methods are listed in Table V, where we can find that different data sets have different optimal parameters. Moreover, the values of these key parameters are continuous and, thus, can be searched effectively by the PSO method.

2) Parameter Sensitivity: There are many parameters in the ANE method, such as the in–out parameter q, the return parameter R, the convergence rate θ, the number of features d, the number of walks r, and the length of each walk l. We examine how different values of these parameters may affect the performance of the ANE method on the PPI data set with 9:1 labeled–unlabeled data ratio. The other parameters are set default: the in–out parameter $q = 1$, the return parameter $R = 1$, the convergence rate $\theta 1 = 1$, the number of features $d = 128$, the number of walks $r = 10$, and the walk length $l = 80$. We use the F1-micro score and the F1-macro score to measure the performance of the ANE with different parameter values.

From Fig. 3, we can find that for the PPI data set, the choice of the parameters will directly affect the quality of the vector generated by the ANE method. In ANE method, the in–out parameter, the return parameter, and the convergence rate can affect the search strategy. From Fig. 3(a)–(c), we can find that different search strategies have different effects on the performance, which indicates that an optimal search strategy is needed. The change of return parameter has less influence on the performance of the ANE method due to the fact that the number of outward nodes is much larger than that of the return nodes in general case. In general, all of these parameters have a relatively high impact on the performance of the method, especially those related to the search strategy.

C. Link Prediction

In link prediction, our goal is to predict whether there exists a link between two given nodes. In the experiments, we are given a network with a certain fraction of links removed, and we try to predict these missing links. We randomly hide 10% of links in the original network as the ground truth and use the remaining to train all representation learning models. After training, we obtain the representation vectors for all nodes and use the logistic regression method to predict the probability of link existence for a given node pair. Our test set consists of the hidden 10% links in the original network as the positive samples and the randomly selected disconnected node pairs as negative samples with equal number. We compare the performance of many methods for link prediction task on the data sets, as follows, and then report the results of F1-micro and F1-macro in Table VI.

1) *arXiv-AstroPh [50]:* This is a network from the e-print arXiv, and it covers scientific collaborations between authors with papers submitted to the Astrophysics category. The nodes represent authors, and the link indicates the coauthor relationship. This network has 18 772 nodes and 198 110 links.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

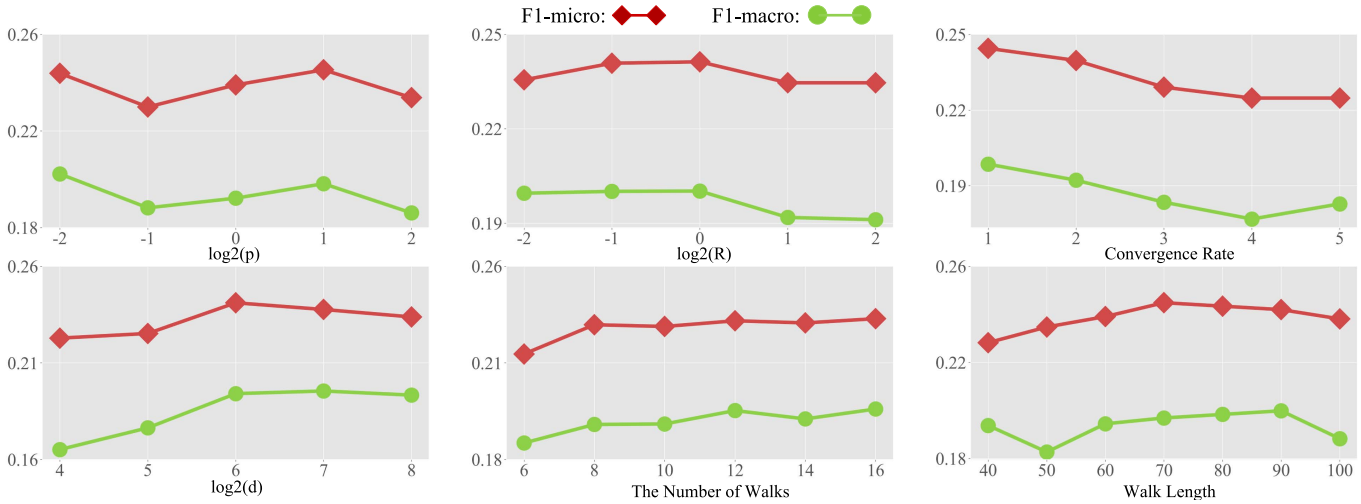


Fig. 3. Parameter sensitivity analysis on the PPI data set.

TABLE VI
LINK PREDICTION RESULTS ON THE TWO DATA SETS: ARXIV-ASTROPH AND ARXIV-GRQC

Model	arXiv-AstroPh		arXiv-GrQc	
	F1-micro	F1-macro	F1-micro	F1-macro
SC	0.7322	0.7132	0.6689	0.6548
LINE	0.8245	0.8142	0.7653	0.7621
DeepWalk	0.8423	0.8325	0.8045	0.8013
Struc2vec	0.8214	0.8102	0.7832	0.7765
Graph2Gauss	0.8406	0.8212	0.8032	0.8005
GraphGAN	0.8592	0.8442	0.8284	0.8215
PSO-ANE-DW	0.8468	0.8375	0.8123	0.8113
PSO-node2vec	0.8568	0.8425	0.8225	0.8186
PSO-ANE	**0.8610**	**0.8482**	**0.8302**	**0.8247**

2) *arXiv-GrQc [50]:* This is also a network from arXiv and covers scientific collaborations between authors with papers submitted to the General Relativity and Quantum Cosmology categories. This network has 5242 nodes and 14 496 links.

The link prediction results are shown in Table VI, where we can see that these results are similar to those in node classification. From Table VI, we can find that the performances of SC, LINE, and struc2vec are relatively poor in link prediction, as they cannot well capture the pattern of link existence in the network. PSO-ANE-DW performs better than the DeepWalk method because the centrality-based transition probability can improve the performance of search strategy. The PSO-node2vec method performs better than the above-mentioned methods. This is mainly because the added flexibility in exploring neighborhoods allows node2vec to outperform the other representation learning methods, and in PSO-node2vec, the best parameters can be obtained for the proposed second-order random walk. PSO-ANE outperforms all the baselines in link prediction because the centrality-based transition probability can improve the performance of search strategy and PSO can help to find the best parameters for the proposed second-order dynamic random walk. In particular, our method improves F1-micro on arXiv-AstroPh and arXiv-GrQc by 0.21%–17.59% and 0.22%–24.11% and improves F1-macro on arXiv-AstroPh and arXiv-GrQc by 0.47%–18.93% and 0.39%–25.95%, respectively.

V. CONCLUSION

Network embedding has wide applications in network data mining. Its performance affects the efficiency of downstream algorithms, such as node classification and link prediction. In this paper, we propose the PSO-ANE model for learning network representations based on the second-order dynamic random walk and the PSO method. The second-order dynamic random walk uses the structure-based transition probability and the centrality-based transition probability to determine a suitable search strategy for each node, while PSO is used to find the optimal parameters. We also propose the PSO-ANE-DW that is a special case of the ANE method with return parameter $R = 1$ and in–out parameter $q = 1$ and the PSO-node2vec that determines the parameters using PSO for node2vec method. Experiments on a variety of data sets illustrate the effectiveness of our model on challenging multilabel classification tasks, multiclass classification tasks, and link prediction tasks, compared with several state-of-the-art embedding methods.

Here, we simply utilize the node centrality to guide the search, resulting in a better network embedding method. In the future, we will consider more kinds of structural information in the network and integrate them into embedding methods to further improve their effectiveness. Furthermore, we want to make the discussion on random walk properties that are interesting for different networks, even optimal search strategies for each node in the network.

REFERENCES

[1] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.
[2] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2724–2743, Dec. 2017.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[3] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.

[4] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, "Signed network embedding in social media," in *Proc. SDM*, 2017, pp. 327–335.

[5] J. Tang, M. Qu, and Q. Mei, "PTE: Predictive text embedding through large-scale heterogeneous text networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 1165–1174.

[6] S. Wang, J. Tang, C. Aggarwal, and H. Liu, "Linked document embedding for classification," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, 2016, pp. 115–124.

[7] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1293–1299.

[8] K. Allab, L. Labiod, and M. Nadif, "A semi-NMF-PCA unified framework for data clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 2–16, Jan. 2017.

[9] L. Liu, W. K. Cheung, X. Li, and L. Liao, "Aligning users across social networks using network embedding," in *Proc. IJCAI*, 2016, pp. 1774–1780.

[10] X. Zhang, C. Wang, Y. Su, L. Pan, and H. F. Zhang, "A fast overlapping community detection algorithm based on weak cliques for large-scale networks," *IEEE Trans. Comput. Social Syst.*, vol. 4, no. 4, pp. 218–230, Dec. 2017.

[11] S. Muhuri, S. Chakraborty, and S. N. Chakraborty, "Extracting social network and character categorization from bengali literature," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 2, pp. 371–381, Jun. 2018.

[12] R. Al-Rfou, B. Perozzi, and S. Skiena. (2013). "Polyglot: Distributed word representations for multilingual NLP." [Online]. Available: https://arxiv.org/abs/1307.1662

[13] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 160–167.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean. (2013). "Efficient estimation of word representations in vector space." [Online]. Available: https://arxiv.org/abs/1301.3781

[15] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.

[16] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. SIGKDD*, 2016, pp. 855–864.

[17] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 385–394.

[18] A. Bojchevski and S. Günnemann. (2018). "Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking." [Online]. Available: https://arxiv.org/abs/1707.03815

[19] H. Wang *et al.* (2017). "GraphGAN: Graph representation learning with generative adversarial nets." [Online]. Available: https://arxiv.org/abs/1711.08267

[20] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[21] Z. Ruan, J. Wang, Q. Xuan, C. Fu, and G. Chen, "Information filtering by smart nodes in random networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 98, no. 2, 2018, Art. no. 022308.

[22] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 64, p. 026118, Jul. 2001.

[23] W.-N. Chen *et al.*, "Particle swarm optimization with an aging leader and challengers," *IEEE Trans. Evol. Comput.*, vol. 17, no. 2, pp. 241–258, Apr. 2013.

[24] J. Liu, Y. Mei, and X. Li, "An analysis of the inertia weight parameter for binary particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 666–681, Oct. 2016.

[25] T. F. Cox and M. A. Cox, *Multidimensional Scaling*. Boca Raton, FL, USA: CRC Press, 2000.

[26] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.

[27] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, p. 2319, 2000.

[28] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 585–591.

[29] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 891–900.

[30] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. (2017). "graph2vec: Learning distributed representations of graphs." [Online]. Available: https://arxiv.org/abs/1707.05005

[31] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[32] T. N. Kipf and M. Welling. (2016). "Semi-supervised classification with graph convolutional networks." [Online]. Available: https://arxiv.org/abs/1609.02907

[33] T. Wei, C. Wang, Y. Rui, and C. W. Chen, "Network morphism," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 564–572.

[34] R. Boorstyn and H. Frank, "Large-scale network topological optimization," *IEEE Trans. Commun.*, vol. 25, no. 1, pp. 29–47, Jan. 1977.

[35] Q. Xuan *et al.*, "Modern food foraging patterns: Geography and cuisine choices of restaurant patrons on yelp," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 2, pp. 508–517, Jun. 2018.

[36] P. Crucitti, V. Latora, and S. Porta, "Centrality in networks of urban streets," *Chaos*, vol. 16, no. 1, 2006, Art. no. 015113.

[37] H. Jeong, S. P. Mason, A.-L. Barabási, and Z. N. Oltvai, "Lethality and centrality in protein networks," *Nature*, vol. 411, no. 6833, pp. 41–42, 2001.

[38] M. K. Tarkowski, P. Szczepański, T. Rahwan, T. P. Michalak, and M. Wooldridge, "Closeness centrality for networks with overlapping community structure," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 1–8.

[39] M. De Domenico, A. Solé-Ribalta, E. Omodei, S. Gómez, and A. Arenas, "Ranking in interconnected multilayer networks reveals versatile nodes," *Nature Commun.*, vol. 6, Apr. 2015, Art. no. 6868.

[40] D. Taylor, S. A. Myers, A. Clauset, M. A. Porter, and P. J. Mucha, "Eigenvector-based centrality measures for temporal networks," *Multiscale Model. Simul.*, vol. 15, no. 1, pp. 537–574, 2017.

[41] M. Li, Y. Lu, Z. Niu, and F.-X. Wu, "United complex centrality for identification of essential proteins from PPI networks," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 14, no. 2, pp. 370–380, Mar./Apr. 2017.

[42] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 3111–3119.

[43] Y. Yang, "An evaluation of statistical approaches to text categorization," *Inf. Retr.*, vol. 1, nos. 1–2, pp. 69–90, 1999.

[44] L. Tang and H. Liu, "Leveraging social media networks for classification," in *Data Mining and Knowledge Discovery*. Norwell, MA, USA: Kluwer, 2011.

[45] R. Zafarani and H. Liu, "Social computing data repository at ASU," School Comput., Inform. Decis. Syst. Eng., Arizona State Univ., Tempe, AZ, USA, 2009. [Online]. Available: http://socialcomputing.asu.edu

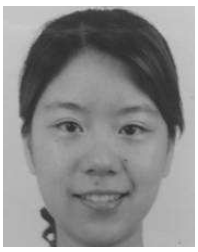
[46] M. S. Livstone *et al.*, "The biogrid interaction database: 2011 update," *Nucleic Acids Res.*, vol. 39, no. 1, pp. D698–D704, 2011.

[47] M. Mahoney. (2011). *Large Text Compression Benchmark*. [Online]. Available: http://www.mattmahoney.net/text/text.html

[48] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of Internet portals with machine learning," *Inf. Retr.*, vol. 3, no. 2, pp. 127–163, 2000.

[49] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, p. 93, 2008.

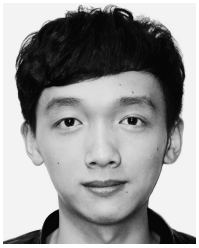
[50] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discovery Data*, vol. 1, no. 1, p. 2, 2007.



Jinyin Chen received the B.S. and Ph.D. degrees from the Zhejiang University of Technology, Hangzhou, China, in 2004 and 2009, respectively.

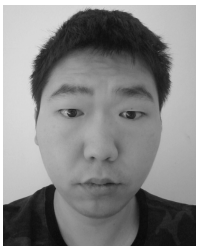
She is currently an Associate Professor with the College of Information Engineering, Zhejiang University of Technology. Her current research interests include deep learning, intelligent computing, complex networks, and algorithm security.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



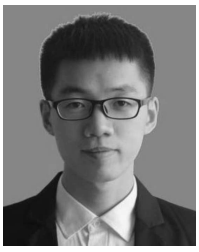
Yangyang Wu received the bachelor's degree from the Ningbo Institute of Technology, Zhejiang University, Hangzhou, China, in 2016. He is currently pursuing the master's degree with the Institute of Information Engineering, Zhejiang University of Technology, Hangzhou.

His current research interests include data mining and applications, complex networks, and clustering analysis.



Xuanheng Xu is currently pursuing the master's degree with the Institute of Information Engineering, Zhejiang University of Technology, Hangzhou, China.

His current research interests include data mining and applications, social network data mining, and intelligent computing.



Haibin Zheng received the bachelor's degree from the Zhejiang University of Technology, Hangzhou, China, in 2017, where he is currently pursuing the master's degree with the Institute of Information Engineering.

His current research interests include data mining and applications, social network data mining, and optimization.

Zhongyuan Ruan received the B.Sc. degree in physics from Guizhou University, Guiyang, China, in 2008, and the Ph.D. degree in physics from East China Normal University, Shanghai, China, in 2013.

He is currently a Lecturer with the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China. His current research interests include complex systems and complex networks.

Qi Xuan (M'18) received the B.S. and Ph.D. degrees in control theory and engineering from Zhejiang University, Hangzhou, China, in 2003 and 2008, respectively.

He was a Post-Doctoral Researcher with the Department of Information Science and Electronic Engineering, Zhejiang University, from 2008 to 2010, and a Research Assistant with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, in 2010. From 2012 to 2014, he was a Post-Doctoral Fellow with the Department of Computer Science, University of California at Davis, Davis, CA, USA. He is currently a Professor with the College of Information Engineering, Zhejiang University of Technology, Hangzhou. His current research interests include network-based algorithm design, social network data mining, social synchronization and consensus, reaction-diffusion network dynamics, machine learning, and computer vision.