IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 3, 2019, accepted April 22, 2019, date of publication April 26, 2019, date of current version May 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2913449

PartNRL: Partial Nodes Representation Learning in Large-Scale Network

JUAN-HUI LI[1,2], LING HUANG[1,2], CHANG-DONG WANG[1,2], DONG HUANG[3], AND JIAN-HUANG LAI[1]

[1]Key Laboratory of Machine Intelligence and Advanced Computing, School of Data and Computer Science, Ministry of Education, Sun Yat-sen University, Guangzhou 510006, China
[2]Guangdong Province Key Laboratory of Computational Science, Guangzhou 510275, China
[3]College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China

Corresponding author: Ling Huang (e-mail: huanglinghl@hotmail.com)

This work was supported in part by the NSFC under Grant 61876193 and Grant 61602189, in part by the Guangdong Natural Science Funds for Distinguished Young Scholar under Grant 2016A030306014, and in part by the Tip-Top Scientific and Technical Innovative Youth Talents of Guangdong Special Support Program under Grant 2016TQ03X542.

ABSTRACT Recently, the low-dimensional embedding of nodes has received a large amount of attention in the field of network analysis. While the existing methods mostly focus on the network embedding of the entire network, there are also some situations, where people may only be interested in some nodes (i.e., partial nodes) rather than all nodes, especially in large-scale networks. Although there are some approaches dealing with the large-scale network, most of them require that all nodes are present during the optimization procedure. The necessity to generate the embedding results for redundant non-interested nodes makes these methods quite inefficient. In this paper, we present a novel node representation framework termed partial nodes representation learning (PartNRL), which is capable of preserving the local similarity of the interested node pairs and generating the embedding results efficiently. Two phases are carefully designed in PartNRL. The first phase is to use local random walk to capture the t-step local similarity for memory-efficiency. By modeling the inherent properties of nodes through the first phase, we design the second phase to learn the node representations by maximizing a likelihood function based on the skip-gram model. To make the optimization procedure more efficient, the negative sampling strategy is applied. The extensive experiments have been conducted on three tasks: node clustering task, single-labeled classification task, and multi-labeled classification task. The experimental results confirm the superior performance of the proposed model over the state-of-the-art network embedding methods.

INDEX TERMS Network embedding, large-scale network, partial nodes, representation learning.

I. INTRODUCTION

Node representation learning aims to learn the low-dimensional node representations that reflect the inherent properties of a network. The basic idea behind the node embedding approach is to use the dimension reduction technique to transform the topological linkage structure into the low-dimensional embedding space, whereby the node-interdependence is implicitly encoded into the node representations. These node embeddings can be fed into the network analysis tasks such as visualization, node classification, link prediction, and entity retrieval [1]–[11]. Moreover, with the node embedding results, both the large-scale distributed computing models (e.g., MapReduce) and off-the-shelf machine learning methods (e.g., node classification) can be directly applied.

In real-world applications on large-scale networks, there exist some situations where only some nodes in a given network are the interested ones, i.e., only partial nodes are being focused on. For example, in a large Facebook network consisting of thousands of millions of users, people may only be interested in the relationship among users coming from the same university. In some academic sites, such as DBLP and Google Scholar, sometimes only the relationships among the authors attending the same conference are being focused on. Despite the progress in the network embedding

The associate editor coordinating the review of this manuscript and approving it for publication was Alba Amato.

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

research, there still exist great challenges when dealing with the interested nodes in a large-scale network by using traditional machine learning algorithms, which usually take the feature vectors as input and are impractical to handle the graph directly. General representations of the interested nodes in the large-scale network is desired for some real-world applications focusing only on interested nodes. Recently there are some efforts to handle issues of interested nodes in network analysis, like local community detection [13]–[18]. It discovers a subgraph that best captures the relationship among the query nodes, i.e., nodes in this subgraph sharing similar properties and they are in the same community. Different from local community detection, the interested nodes we focus on may share heterogeneous properties, hence they may belong to different classes which is intuitively more general in the real-world applications.

How to efficiently generate the low-dimensional embedding results for these interested nodes in a large-scale network remains an open problem. In recent years, many research works are proposed to learn the node representations for a given network. We can just apply the existing node representation learning methods on the large-scale network to generate the embedding results for all nodes, so that the node representations of the interested nodes can be obtained. But they may suffer or even fail to get the node representations due to the time complexity and out-of-memory issues. One remedy is to apply some existing approaches [19], [20] dealing with the large-scale network. However, most of them need to generate the node embedding results for the non-interested nodes which are redundant and also bring memory consuming. To solve this problem, we propose a novel framework to learn the node representations of the interested nodes with efficiency.

Given a large-scale network and some interested nodes, we propose a novel network embedding method termed Partial Nodes Representation Learning (PartNRL) for learning the low-dimensional node representations of the interested nodes, in which the node representation learning of non-interested nodes is avoided. To this end, two phases are designed in PartNRL. 1) The first phase is to model the similarity of the interested node pairs. Due to the good quality for time complexity and memory allocation, local random walk [21] is used to generate the t-step similarity matrix $\mathbf{S}^{LRW}$. By superposing the contribution of each walker, another similarity matrix $\mathbf{S}^{SRW}$ can be generated. 2) The second phase is to learn the node representations. Motivated by some models [22], [23], embedding results are obtained by maximizing a likelihood function preserving the local proximity generated from the first phase. To optimize the objective function, negative sampling is used to make the optimization process more efficient.

The main contributions of this paper are summarized as follows:

- **Partial nodes embedding**: This paper addresses the node representation learning problem for partial interested nodes in large-scale network and avoiding redundant costs of generating node representations for non-interested nodes.
- **Innovative framework**: A novel framework termed Partial Nodes Representation Learning (PartNRL) is designed which preserves high-order proximity for large-scale network with two phases: generating similarity matrix based on local random walk, and learning node representations by maximizing a likelihood function.
- **Efficiency**: With two carefully designed phases and model optimization by negative sampling, our model can generate the embedding result efficiently.
- **Effectiveness**: Extensive experiments show that our model can provide better embedding result compared with some state-of-the-art methods on different tasks.

The rest of this paper is organized as follows. Section II briefly reviews some related work. The carefully designed model with terminology definitions is described in section III. Experimental results on four real-world datasets in several different tasks are reported in section IV. Finally, we conclude this paper and introduce some future works in section V.

II. RELATED WORK

Traditional network embedding methods such as MDS [24], IsoMap [25], LLE [26] and Laplacian Eigenmap [27] typically first construct the affinity graph and learn the node representations from this affinity graph. But most of them need to use the eigendecomposition skill to embed the network, which results in inefficiency for handling the large-scale network embedding. Recent network embedding methods have been developed from different perspectives. Wang *et al.* [28] proposed a M-NMF model combining the microscopic proximity structure and the mesoscopic community structure. But it's expensive to obtain the microscopic proximity matrix, since it needs to compute the proximity of every node pairs in the network. The time complexity is quadratic to the number of nodes which is obviously impractical on large-scale network. Huang *et al.* [29] proposed the LANE framework by incorporating the label information into the attributed network, in which the optimization scheme is turned into solving leading eigenvectors. The necessity to apply the eigendecomposition makes LANE also impractical for embedding large-scale network. Cao *et al.* [30] developed a GraRep model, which aims at capturing different k-step local relational information for different k, and then obtains the global representation via SVD skill. To capture the different k-step information, k times computation of the k-step probability of every node pair is required, which is also quite memory consuming and inefficient. To optimize the objective function, matrix factorization is used which needs the eigendecomposition skill. Therefore, GraRep also doesn't fit for handling the large-scale network embedding.

Some recent efforts have been made that can be used for embedding the large-scale network, such as LINE [20],

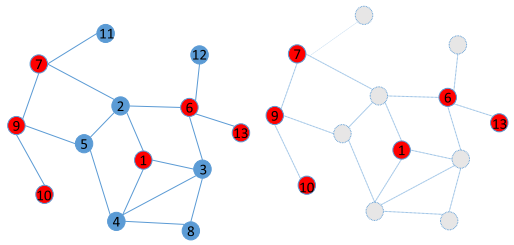


FIGURE 1. (a): A graph with sampled nodes (i.e., nodes in red) we are interested in (b) shaded version of (a).

DeepWalk [22] and Node2vec [31]. By preserving the first-order proximity and the second-order proximity, LINE proposes two objective functions and modifies them into the negative sampling forms considering efficiency. DeepWalk deploys truncated random walks to generate the node sequences, and uses hierarchical softmax function to optimize the likelihood function. Node2vec model defines a flexible notion of a node's neighborhood and generates the node sequence by balancing the breadth-first sampling and depth-first sampling. The embedding results are generated by maximizing the likelihood of preserving network neighborhoods of nodes. Both DeepWalk and Node2ve are based on the language model Skip-gram, which aims to learn continuous feature representations for words in natural language processing. Although they all show effective performance for network embedding on large-scale network, they also have to generate the embeddings for non-interested nodes which are redundant.

III. THE PROPOSED MODEL

Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of n nodes, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Let $\mathcal{V}_s \subset \mathcal{V}$ denote the interested node set from set $\mathcal{V}$. $\mathcal{V}_s = \{v_1, v_2, \ldots, v_{n_s}\}$ consists of n_s nodes. The ratio of the interested nodes number in $\mathcal{V}_s$ to the total number in $\mathcal{V}$ is denoted by $\rho = |\mathcal{V}_s|/|\mathcal{V}|$. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ denote the adjacency matrix, where element $a_{ij} = 1$ when node v_i and node v_j are connected and $a_{ij} = 0$ otherwise. The goal is to find the low-dimensional network embedding result denoted as $\mathbf{U} \in \mathbb{R}^{n_s\times d}$ with d being the dimensionality of each node, whose i-th row u_i is the embedding result of node v_i. Note that, we assume $n_s \ll n$.

A. PAHSE 1: MODELING SIMILARITY IN $\mathcal{V}_s$

Consider a network with some interested nodes like the graph presented in FIGURE 1(a). If the non-interested information is omitted like the graph presented in FIGURE 1(b), it can be seen that the network structure of the interested nodes is not clear. Some interested nodes may be isolated from the others, e.g., node v_1 in FIGURE 1(b) has no connections to the other interested nodes; The original topological network structure being destroyed leads to information missing issues and results in bad embedding result. It's important to capture some essential network structure information for interested nodes.

Most traditional network embedding approaches basically model the low-order information of the network topological structure to generate the embedding result, e.g., two directly connected nodes share some similarities (first-order proximity [20]) or two nodes tend to be similar if they both connect to a same node (second-order proximity [20]). Hence, only 2-step information is captured. However, for a node in $\mathcal{V}_s$, it may take more than 2 steps to reach another node, which indicates that it's impractical to capture the similarity of node pairs using low-order information in $\mathcal{V}_s$. Meanwhile, as we are only interested in the embedding results of nodes in $\mathcal{V}_s$, taking the global information of the whole network into consideration is quite inefficient. Motivated by the process of random walk [21], t-step local random walk is applied here to capture the higher-order proximity of node pairs.

We use $p_{i\to j}(t)$ to denote the probability for node v_i to reach node v_j after t steps. At the initial state, $p_{i\to i}(0) = 1$ and $p_{i\to j}(0) = 0\ (i \neq j)$. We use $\mathcal{V}_w^i(t)$ to denote all nodes that the random walker may visit after step t when it starts from node v_i with $\mathcal{V}_w^i(0) = \{i\}$. During the random walk procedure from node v_i to node v_j, the walker may go far away from both v_i and v_j although they may be close to each other. This may result in inaccurate similarity measuring since in real-world applications, nodes tend to be more similar with the ones that are closer rather than further. We consider the local random walk model with restart: at each step t, there is a probability λ that the random walker will continue, and a probability $1-\lambda$ that it will return to starting node and restart the procedure. Therefore, we have

$$p_{i\to j}(t) = (1-\lambda)p_{i\to i}(0)\delta(i,j) + \lambda \sum_{c\in\mathcal{N}(j)} m_{cj}p_{i\to c}(t-1)h_c \tag{1}$$

where function $\delta(i,j) = 1$ if $i = j$, $\delta(i,j) = 0$ otherwise. And $\mathcal{N}(j)$ is a set consisting of node v_j's immediate neighbors. $m_{cj} = \frac{a_{cj}}{d_c}$ is the transition probability denoting the probability that a random walker walking from node v_c to node v_j at the next step, where d_c is the degree of node v_c. When node $v_c \in \mathcal{V}_w^i(t-1)$, $h_c = 1$, and $h_c = 0$ otherwise.

Take the graphs in FIGURE 2 with $\lambda = 0.85$ as an example. Graphs in FIGURE 2 are all unweighted, thus $a_{ij} = 1$ for each node v_i and node v_j that are connected. We have $m_{12} = \frac{1}{d_1} = \frac{1}{3}$ for node v_1 in FIGURE 2(a).

1) The initial state for the local random walker is shown in FIGURE 2(b).
2) When $t = 1$, random walker may visit node v_2, v_3, v_4, we have $\mathcal{V}_w^1(1) = \{1,2,3,4\}$, $p_{1\to 1}(1) = (1-\lambda) * p_{1\to 1}(0) = 0.15$, $p_{1\to 2}(1) = \lambda * m_{12} * p_{1\to 1}(0) = 0.2833$.
3) When $t = 2$, we have $\mathcal{V}_w^1(2) = \{1,2,3,4,5,6,7,8\}$, $p_{1\to 1}(2) = (1-0.85)*p_{1\to 1}(0)+0.85*(0.25*0.2833+0.25*0.2833+0.25*0.2833) = 0.3306$, $p_{1\to 5}(2) =$

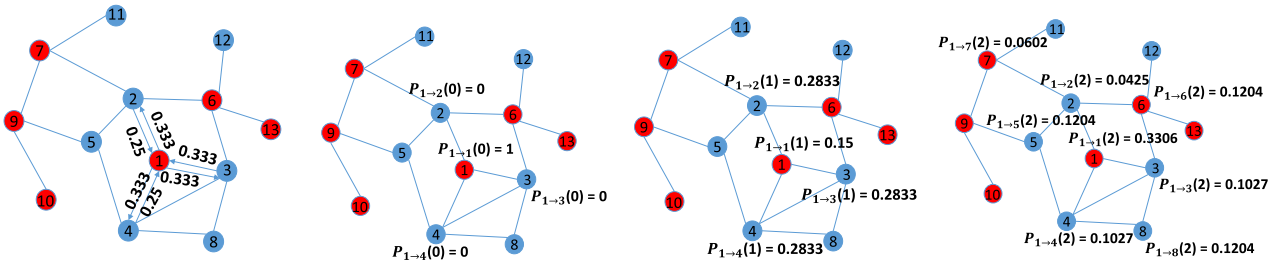


FIGURE 2. (a): Transition probabilities of random walk that starts from node v_1 to its neighbors and vice versa. (b), (c), (d): Probability of random walk that starts from node v_i and reaches at the corresponding nodes after t steps when $\lambda = 0.85$.

$\lambda * (m_{25} * p_{1\to2}(1) + m_{45} * p_{1\to4}(1)) = 0.85 * (0.25 * 0.2833 + 0.25 * 0.2833) = 0.1024$.

We use $\mathbf{S}^{LRW}(t) \in \mathbb{R}^{n_s \times n_s}$ to denote the similarities between nodes in $\mathcal{V}_s$ based on the t-step local random walk. It's intuitive that the similarity matrix $\mathbf{S}$ is symmetric, we have

$$s_{ij}^{LRW}(t) = p_{i\to j}(t) + p_{j\to i}(t) \tag{2}$$

where t is the number of steps (i.e., $t = 1, 2, \ldots, T$) and $s_{ij}^{LRW}(t)$ is the entry in the i-th row and j-th column of $\mathbf{S}^{LRW}(t)$. Each t-step local random walker moves independently. By superposing the contribution of each walker, superposed random walk is obtained:

$$s_{ij}^{SRW}(t) = \sum_{\tau=1}^{t} s_{ij}^{LRW}(\tau) = \sum_{\tau=1}^{t} (p_{ij}(\tau) + p_{ji}(\tau)) \tag{3}$$

where $s_{ij}^{SRW}(t)$ is the entry in the i-th row and j-th column of $\mathbf{S}^{SRW}(t) \in \mathbb{R}^{n_s \times n_s}$.

There are several benefits of local random walk for similarity measuring. It exhibits good quality for satisfying both space and time computation requirements. The space complexity to store the neighbors for each node is $O(|\mathcal{E}|)$. To store matrix $\mathbf{S}^{LRW}$ or $\mathbf{S}^{SRW}$, it needs $O(|\mathcal{V}_s|^2)$ space complexity, where $|\mathcal{V}_s|$ is assumed to be small that will avoid memory issue. To fit for the large-scale network and reduce the memory consuming, block-stripe update algorithm [32] can be used. In each iteration, the cost is $O(|\mathbf{M}|(1+\varepsilon)+(m+1)|\mathbf{r}|)$, where $\mathbf{r}$ is a vector containing the probabilities for a starting node i to the nodes in $\mathcal{V}_w^i(T) \cup \mathcal{V}_s$. Nodes in set $\mathcal{V}_w^i(T)$ can be preprocessed via breadth first search. The updated $\mathbf{r}^{new}$ is broken into m blocks that fit in memory. And matrix $\mathbf{M}$ is broken into stripes with each stripe containing only destination nodes in the corresponding block of $\mathbf{r}^{new}$.

B. PHASE 2: LEARNING GRAPH REPRESENTATIONS

After obtaining the similarity matrix from phase 1, we sample "neighbors" for each node by utilizing the alias table method [33] which can sample a node in constant time. For each node v_i, "neighbors" are sampled from nodes sharing positive similarity values with node v_i. Alias table method is run for l times to sample "neighbors" according to the similarity values of node pairs. We use $\mathcal{N}_i^s$ to denote the set of the "neighbors" of node v_i. We can assume there is an edge between each node with its "neighbours" so that an "edge" set can be formed: $\mathcal{E}_s = \{(v_i, v_j) | v_i, v_j \in \mathcal{V}_s, v_j \in \mathcal{N}_i^s\}$. Motivated by two node representation models [22], [23], for nodes in set $\mathcal{V}_s$, we define the t-step likelihood function assuming conditional independence as follows:

$$l_t = \prod_{v_i \in \mathcal{V}_s} \prod_{v_j \in \mathcal{N}_i^s} p_t(v_j | v_i) \tag{4}$$

where $p_t(v_j|v_i)$ denotes the conditional probability of occurring node v_j given node v_i. By maximizing this likelihood function, we can achieve the goal of preserving the "structural" information obtained from section III-A (i.e., the information in $\mathcal{E}_s$). Take log to both sides of Eq. (4) and switch from product to sum, we have:

$$\log l_t = \sum_{v_i \in \mathcal{V}_s} \sum_{v_j \in \mathcal{N}_i^s} \log p_t(v_j | v_i). \tag{5}$$

Therefore, to maximize l_t, we can maximize the right side of Eq. (5). Let f be a function that estimates the proximity score of nodes v_i and v_j. Intuitively, when nodes v_i and v_j are more similar to each other, i.e., higher proximity score, the probability of occurring node v_j given node v_i is higher. More formally, the t-step conditional probability of node v_j on v_i is defined using the softmax function:

$$p_t(v_j | v_i) = \frac{\exp(f(u_j, u_i))}{\sum_{v_c \in \mathcal{V}_s} \exp(f(u_c, u_i))}. \tag{6}$$

Traditionally in language model [34], [35], we define proximity function f as follows:

$$f(u_j, u_i) = u_j' \cdot u_i \tag{7}$$

where u_j' is the j-th row of $\mathbf{U}' \in \mathbb{R}^{n_s \times d}$ and it's the "context" expression of node v_j. We can also treat u_j' as an abstractive representation of a neighbor v_j for easier understanding [23]. In this case, two nodes tend to be similar if they share similar neighborhoods, which can also be treated in the view of the second-order proximity [20]. our t-step objective function is defined as follows:

$$\arg\max_{\mathbf{U}, \mathbf{U}'} \quad L_t = \sum_{v_i \in \mathcal{V}_s} \sum_{v_j \in \mathcal{N}_i^s} \log p_t(v_j | v_i). \tag{8}$$

By maximizing L_t, the proximity information derived from the t-step local random walk or t-step superposed random walk is preserved, which is intuitively contributory to achieve a better network embedding result.

C. MODEL OPTIMIZATION

To optimize the objective function in Eq. (8), we need to update the context representations of all nodes in $\mathcal{V}_s$ in each iteration, which is time consuming. One straightforward remedy is only updating a sample of them. And negative sampling [34] has been designed to sample some negative edges according to some noise distribution. It not only makes the optimization more efficient, but also is capable of providing good embedding result [36]. The negative sampling is defined by the objective as follows:

$$L_{NS} = \log \sigma(u_j'^T \cdot u_i) + \sum_{k=1}^{K} \mathbb{E}_{v_k \sim P_n(v)}[\log \sigma(-u_k'^T \cdot u_i)] \quad (9)$$

where $\sigma(x) = 1/(1+\exp(-x))$ is the sigmoid function, $P_n(v)$ is the noise distribution and $v_k, k = 1, \ldots, K$ are the sampled nodes. $P_n(v)$ is set as a unigram distribution: $P_n(v) \propto d_v^{3/4}$ where d_v is the degree of node v. Objective in Eq. (9) is used to replace the $\log p_t(v_j|v_i)$ term in Eq. (5). Thus, our objective function can be rewritten as:

$$\arg\max_{\mathbf{U},\mathbf{U}'} \quad L_t = \sum_{v_i \in \mathcal{V}_s, v_j \in \mathcal{N}_i^s} L_{NS} \quad (10)$$

To optimize the objective function in Eq. (10), the asynchronous stochastic gradient algorithm (ASGD) [37] is adopted. In each step, some edges are sampled and the corresponding embedding vectors and context representation vectors are updated. If edge (i, j) is sampled, we can obtain the gradient of the embedding vector of node v_i as follows:

$$\frac{\partial L_t}{\partial u_i} = u_j'(1-\sigma(u_j'^T \cdot u_i)) - \sum_{k=1}^{K} \mathbb{E}_{v_k \sim P_n(v)} u_k' \sigma(u_k'^T \cdot u_i) \quad (11)$$

The gradient of the context representation vector of node v_j is obtained as follows:

$$\frac{\partial L_t}{\partial u_j'} = u_i(1 - \sigma(u_j'^T \cdot u_i)) \quad (12)$$

And the gradient of the context representation vector of node v_k sampled form the negative sampling is obtained as follows:

$$\frac{\partial L_t}{\partial u_k'} = -u_i \sigma(u_k'^T \cdot u_i) \quad (13)$$

With different similarity matrix $\mathbf{S}^{LRW}(t)$ and $\mathbf{S}^{SRW}(t)$, PartNRL yields a node representation matrix $\mathbf{U}_t^{LRW} \in \mathbb{R}^{n_s \times d}$ or $\mathbf{U}_t^{SRW} \in \mathbb{R}^{n_s \times d}$ at each step t. The $\mathbf{U}_t^{LRW}$ $(t = 1, 2, \ldots, T)$ can be concatenated to form a matrix $\mathbf{U}_c^{LRW} \in \mathbb{R}^{n_s \times Td}$ and similar for $\mathbf{U}_t^{SRW}(t = 1, 2, \ldots, T)$. From our experiment, we can know that usually with smaller T, PartNRL using $\mathbf{U}_T^{LRW}$ can generate relatively better result, thus we choose $\mathbf{U}_T^{LRW}$ as output.

For clarity, Algorithm 0 summarizes the main procedure of the proposed PartNPL algorithm.

Algorithm 1 PartNRL

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, steps of local random walk T, dimension d, interested nodes set $\mathcal{V}_s$, alias sample number l, number of negative edges K

Output: $\mathbf{U}_T$

```
Initialize P_{i→j}(0) for node v_i in V_s, and v_j ∈ N(i)
Initialize S^{SRW}(0)
t = 0, count = 0
while t < T do
    for v_i ∈ V_s do
        for v_c ∈ V_w^i(t) do
            p_{i→c}(t) ← Eq. (1)
        end for
    end for
    for v_i, v_j ∈ V_s do
        s_{ij}^{LRW}(t) = p_{i→j}(t) + p_{j→i}(t)
        s_{ij}^{SRW}(t) = s_{ij}^{SRW}(t) + s_{ij}^{LRW}(t)
    end for
end while
for all nodes v_i ∈ V_s do
    for count < l do
        E_s = AliasSample(v_i, S^{LRW}(T))
    end for
end for
U_T = StochasticGradientDescent(d, K, E_s)
```

IV. EXPERIMENTS

In this section, we evaluate the performance of our algorithm on four data sets for several different tasks, and compare it with some baseline algorithms.

A. EXPERIMENTAL SETTINGS

1) DATA SETS

We use four publicly available datasets to evaluate the effectiveness of our algorithm, which are the Cora dataset [38], Citeseer dataset [38], PPI dataset [19] and dblp dataset [39].

1) **Cora:** This dataset is a citation network containing the machine learning papers classified into 7 classes. It consists of 2708 papers and 5429 links. Each node is classified into a single class.
2) **Citeseer:** This dataset is a citation network consisting of 3312 scientific publications and 4732 links. The publications are from distinct research areas which are divided into 6 classes. Each node is classified into a single class.
3) **PPI:** This dataset is a protein-protein interaction network consisting of 20 different subgraphs corresponding to different human tissue. The whole dataset consists of 56944 nodes and 818717 links with 121 classes. Each node is classified into multiple classes.
4) **dblp:** This dataset is a co-authorship network of research papers in computer science. Two authors are connected if they publish at least one paper together. It consists of 317080 nodes and 1049866 links. Authors

TABLE 1. Comparison results of clustering in terms of ACC and purity on Cora.

	ACC					Purity				
	10%	15%	20%	25%	30%	10%	15%	20%	25%	30%
DeepWalk	26.13	23.01	23.93	23.73	23.26	34.27	28.80	31.87	30.69	30.71
LINE	26.94	25.36	23.59	24.66	24.24	37.59	32.70	32.71	32.89	32.86
M-NMF	35.58	29.36	32.01	33.19	31.98	38.42	32.05	34.48	35.06	33.76
GraRep	31.60	31.98	32.98	32.19	33.03	*41.40*	38.88	39.97	38.92	39.99
Node2vec	22.44	20.74	20.16	19.39	18.64	33.96	28.59	31.34	30.23	30.46
SNE	35.08	33.96	33.86	30.45	34.26	38.82	36.55	38.36	35.26	37.07
PartNRL_LRW	*47.66*	**53.19**	**48.81**	**41.22**	**43.12**	**57.13**	**59.50**	**52.71**	*46.63*	**49.85**
PartNRL_LRW_con	**49.01**	*51.68*	*45.28*	*40.62*	*43.02*	*56.89*	*57.73*	*50.84*	**46.72**	*49.51*
PartNRL_SRW	43.53	47.10	42.40	33.24	39.39	52.33	52.99	49.59	46.5	46.5
PartNRL_SRW_con	41.48	44.57	38.88	33.15	37.27	49.43	49.66	46.68	43.57	43.57

are classified into 13477 communities, where authors publishing in the same venue are in the same community. Each node is classified into multiple communities.

2) COMPARED ALGORITHMS

we use the following six methods as the compared algorithms:

1) **DeepWalk** [22]: It uses the truncated random walks to generate the linear sequence as the ''corpus'' for each node, and then uses the Skip-gram model with the hierarchical softmax to generate the embedding result.
2) **LINE** [20]: It defines two distributions preserving the first-order and second-order proximity respectively. By minimizing the KL-divergence of the empirical distributions and the defined distributions, loss functions are obtained. In the optimization procedure, negative sampling is used to make LINE more efficient.
3) **M-NMF** [28]: It integrates the microscopic topological structure information and mesoscopic community structure information to learn the node embedding, where first-order and second-order proximity are utilized to preserve the microscopic topological structure, and modularity is utilized to preserve the community structure. Nonnegative matrix factorization (NMF) [40] is used to update the model.
4) **GraRep** [30]: It captures the global structure properties of the network through defining k-step loss functions which integrate rich local structural information. The loss function is motivated from the Skip-gram model and is solved through the SVD skill.
5) **Node2vec** [31]: It defines a flexible notion of a node's neighborhood and designs a biased random walk to explore various neighborhoods. By maximizing the likelihood function of preserving the neighborhoods of nodes, the low-dimensional representations of nodes are obtained.
6) **SNE** [23]: It combines both the structure and attribute information by utilizing the multi-layer neural network. By modeling a conditional probability, a likelihood function preserving the global structural proximity and attribute proximity is designed.

To evaluate the performance of our algorithm, we conduct experiments on several tasks. For the single-labeled networks (i.e., Cora and Citeseer), we apply K-means for the clustering task and KNN for the single-labeled classification task. For the multi-labeled networks (i.e., PPI and dblp), one-vs-rest logistic regression model implemented by LIBLINEAR [41] for the multi-label classification is applied. All the codes of the above methods are obtained from the authors' websites. The parameters for these six compared algorithms are set in such a way that either the default settings suggested by the authors or they are tuned by trials to find the best settings. And the dimensionality d is set in such a way that either 100 or the default settings suggested by the authors. After obtaining the embedding results of all nodes from the compared algorithms, only the embedding results of the interested nodes in $\mathcal{V}_s$ are fed into the clustering and classification algorithms.

Interested nodes $\mathcal{V}_s$ are randomly chosen from the input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We set the maximum value of t in LRW and SRW as 5 (i.e., $T = 5$), and times l to sample ''neighbors'' for each node as 15. Each $\mathbf{U}_t^{LRW}$, $\mathbf{U}_t^{SRW}$ and the concatenated matrices $\mathbf{U}_c^{LRW}$, $\mathbf{U}_c^{SRW}$ are evaluated in our experiments and the best result of $\mathbf{U}_t^{LRW}$ and $\mathbf{U}_t^{SRW}$ are reported in the following section. We use PartNRL_LRW to represent our model based on local random walk which yields the result of $\mathbf{U}_t^{LRW}$, and similarly we use PartNRL_SRW for $\mathbf{U}_t^{SRW}$, PartNRL_LRW_con for $\mathbf{U}_c^{LRW}$ and PartNRL_SRW_con for $\mathbf{U}_c^{SRW}$.

B. COMPARISON RESULTS WITH THE EXISTING ALGORITHMS

1) SINGLE-LABELED NETWORKS

In the clustering task, K-means is applied on the node representations to generate the clusters on Cora and Citeseer datasets. The ratio of the interested nodes number $|\mathcal{V}_s|$ to the total number $|\mathcal{V}|$ (i.e., ρ) varies from 10% to 30%. Due to the sensitivity of K-means on the initial values, we repeat K-means 20 times, each with randomly initial centroids. ACC [42] and Purity are used as metrics to access the quality of the clustering results, and the average values of ACC and Purity are reported in Table 1 and Table 2. The best result is represented in bold form and the second best result is represented in italic form. First and foremost, PartNRL_LRW achieves the best performance in most cases, which justifies the usefulness of capturing the t-step proximity. Notably,

TABLE 2. Comparison results of clustering in terms of ACC and purity on Citeseer.

	ACC					Purity				
	10%	15%	20%	25%	30%	10%	15%	20%	25%	30%
DeepWalk	23.84	22.58	21.88	22.71	21.95	24.99	24.95	22.75	24.23	22.89
LINE	25.17	25.38	24.63	24.77	24.37	27.32	27.92	26.33	26.58	25.96
M-NMF	23.16	24.57	23.07	24.09	23.43	24.34	26.55	24.32	25.39	24.28
GraRep	29.05	27.69	27.73	28.94	27.67	29.88	29.04	28.75	30.35	28.91
Node2vec	22.79	22.08	20.98	20.78	20.12	25.60	25.45	23.71	23.82	22.65
SNE	27.64	32.39	33.47	**36.22**	**35.79**	29.08	33.95	34.73	37.81	**37.20**
PartNRL_LRW	**39.69**	*39.23*	**35.65**	*35.03*	*32.39*	**43.82**	**43.24**	**39.62**	**41.40**	*36.65*
PartNRL_LRW_con	*36.34*	37.56	*35.21*	34.02	32.04	*39.67*	40.14	*38.26*	*38.99*	35.71
PartNRL_SRW	32.84	**39.27**	31.18	33.16	30.23	35.67	*42.33*	34.68	37.50	33.99
PartNRL_SRW_con	32.18	34.11	31.47	30.52	29.44	34.63	36.98	34.62	34.25	32.87

TABLE 3. Comparison results of single-labeled classification in terms of ACC and F1 score on Cora.

	ACC					F1 score				
	10%	15%	20%	25%	30%	10%	15%	20%	25%	30%
DeepWalk	35.07	33.20	37.19	39.06	40.91	24.03	29.88	35.22	38.67	38.49
LINE	54.26	62.88	57.546	62.91	64.45	62.30	69.04	70.8	72.16	75.31
M-NMF	64.33	60.56	69.27	67.22	68.32	63.16	62.48	68.12	66.19	66.90
GraRep	53.20	50.41	51.03	50.94	53.17	52.14	50.22	50.71	51.22	54.19
Node2vec	31.52	27.78	27.28	25.46	25.96	17.86	15.4	17.19	17.13	18.33
SNE	36.52	35.44	34.02	38.59	35.55	20.38	29.57	30.82	30.85	28.52
PartNRL_LRW	72.11	71.78	**75.94**	72.81	77.36	70.98	**73.43**	74.07	73.50	75.20
PartNRL_LRW_con	*72.41*	**73.02**	*75.45*	*73.28*	*77.52*	**73.61**	*72.33*	*74.10*	**73.96**	**75.65**
PartNRL_SRW	**73.11**	*71.93*	74.50	71.99	**77.87**	71.88	72.28	**74.70**	73.28	*75.50*
PartNRL_SRW_con	71.15	71.22	73.25	**73.97**	76.83	*72.58*	71.55	71.65	*73.61*	75.05

TABLE 4. Comparison results of single-labeled classification in terms of ACC and F1 score on Citeseer.

	ACC					F1 score				
	10%	15%	20%	25%	30%	10%	15%	20%	25%	30%
DeepWalk	30.78	36.56	31.47	39.04	40.44	21.93	32.34	28.22	36.55	40.19
LINE	41.73	51.22	47.226	50.93	56.09	38.30	49.53	49.67	52.16	54.72
M-NMF	39.31	44.10	39.98	42.49	44.20	35.32	43.28	41.07	42.57	43.62
GraRep	37.58	43.10	40.27	41.07	43.02	32.24	43.68	39.49	41.51	43.54
Node2vec	29.97	27.26	26.09	25.58	24.22	15.40	14.74	17.63	14.56	16.05
SNE	31.04	30.50	26.15	30.12	38.03	20.44	18.87	9.97	21.16	34.55
PartNRL_LRW	**45.37**	53.12	49.29	52.81	**56.17**	*43.91*	51.73	47.97	52.44	*55.40*
PartNRL_LRW_con	*43.13*	*53.24*	*50.99*	**53.75**	55.44	**44.15**	**52.93**	*50.42*	**53.75**	**55.49**
PartNRL_SRW	41.76	52.98	49.85	*53.65*	*55.56*	41.12	*52.88*	49.37	52.05	53.66
PartNRL_SRW_con	43.40	**53.56**	**51.94**	52.40	54.93	42.35	52.58	**50.56**	*52.82*	54.48

PartNRL_LRW_con also provides satisfactory results. Our algorithm achieves the best result with small ρ on both Cora and Citeseer, which demonstrates the effectiveness of our algorithm to embed the small number partial nodes. Among the compared algorithms, GraRep which also utilizes the k-step information outperforms other compared algorithms in terms of Purity on Cora. While SNE exhibits competitive performance with GraRep, due to its integration of the network structure information and rich attribute information.

In the single-labeled classification task, the node representations are fed into the KNN classifier with 80% randomly selected nodes as the training data while the remaining 20% as the testing data. we set the number of the nearest neighbors K as 3 here. The cross-validation process is repeated 20 times and the mean values of ACC and F1 score are reported in Table 3 and Table 4. We can see that our algorithm outperforms other compared algorithms in all the cases. The ACC values and Purity values yielded from our four variations are very close, and PartNRL_LRW_con achieves the best result in most cases. The performance trends are consistent with that of the clustering task.

2) MULTI-LABELED NETWORKS

For the Multi-labeled networks (i.e., PPI and dblp), one-vs-all logistic regression for multi-label classification is applied to generate the labels for the testing nodes. We randomly selected 80% of nodes as the training set and the rest as the testing set. In the test phase, each testing node has a ranking of labels indicating the probability of belonging to the corresponding labels, rather than an exact label assignment. We repeat the cross-validation process 20 times and the average Precision score and F1 score are reported in Table 5 and Table 6. From the tables, we can see that due to the large scale of data PPI and dblp, some compared algorithms can't yield node representations except for DeepWalk and LINE. And with some unknown reasons, the source code of Node2vec

TABLE 5. Comparison results of multi-labeled classification in terms of precision and F1 score on PPI.

	Precision					F1 score				
	10%	15%	20%	25%	30%	10%	15%	20%	25%	30%
DeepWalk	59.44	59.96	59.66	60.75	60.80	56.13	55.98	55.65	56.01	55.74
LINE	59.67	60.59	60.64	61.22	61.48	58.48	58.84	58.48	58.97	59.03
M-NMF	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
GraRep	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Node2vec	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SNE	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
PartNRL_LRW	**61.46**	61.19	*61.96*	62.26	62.48	59.03	58.51	59.14	59.19	59.54
PartNRL_LRW_con	*60.88*	60.55	**62.36**	*62.54*	**63.01**	**60.43**	*59.61*	**61.44**	*61.49*	**61.94**
PartNRL_SRW	60.46	**61.98**	61.37	62.28	*62.78*	*59.11*	59.29	58.58	59.23	58.77
PartNRL_SRW_con	59.84	*61.63*	61.22	**62.60**	61.90	57.45	**61.00**	*60.07*	**61.55**	*60.45*

TABLE 6. Comparison results of multi-labeled classification in terms of precision and F1 score on dblp.

	Precision						F1 score					
	5%	6%	7%	8%	9%	10%	5%	6%	7%	8%	9%	10%
DeepWalk	5.51	5.83	5.92	5.88	6.22	6.07	4.83	5.22	5.24	5.20	5.46	5.35
LINE	31.24	32.2	33.10	33.79	34.29	34.81	*30.27*	31.15	32.27	32.70	33.45	33.82
M-NMF	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
GraRep	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Node2vec	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SNE	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
PartNRL_LRW	24.37	25.90	27.04	27.53	28.26	26.45	21.76	22.91	23.99	24.53	25.37	23.33
PartNRL_LRW_con	**33.15**	**35.14**	**36.65**	**37.36**	**38.39**	*36.20*	**32.18**	**33.91**	**35.34**	**36.10**	**37.22**	*34.73*
PartNRL_SRW	22.09	22.98	24.53	24.96	26.71	26.71	18.98	19.40	21.01	20.96	22.91	22.91
PartNRL_SRW_con	*31.20*	*32.95*	*34.56*	*35.43*	*36.43*	**37.01**	30.05	*31.42*	*33.12*	*33.86*	*34.99*	**35.63**

collapses on these two datasets each time we run it, thus no results can be obtained from Node2vec. Notably, among the four variations of our algorithm, PartNRL_LRW_con provides the best performance. Although the PartNRL_LRW and PartNRL_SRW don't outperform LINE, they exhibit much better performance than DeepWalk, which suffers a lot on dblp.

C. VISUALIZATION

In this section, we present the visualization results of the learned representations on a real citation network - Cora. The network embedding results derived from different algorithms are mapped into a 2-D space through the t-SNE package [43]. For all algorithms, the parameter configuration are set the same. FIGURE 3 shows the visualization results from different embedding approaches. We can see that the results generated by DeepWalk, M-NMF, GraRep, Node2vec are not informative since nodes with different colors are mixed together. It's quite tricky to separate the nodes provided by them. For LINE and SNE, the embedding results look better, but nodes with the same color don't form a cluster with clear bound. Nodes belonging to different class are clustered closely in the central area. Most of the compared algorithms fail to provide clearly separable clusters, probably because this is the embedding result for interested nodes in a given graph and they lose much meaningful global information. While PartNRL_LRW and PartNRL_SRW exhibit much better performance than the compared algorithms. Although node in some clusters are not very close, we can see some clear bound among clusters.

D. ANALYSIS OF STEP T

We further explore the influence of t which is the step of LRW and SRW. It controls how many steps of structure information that random walk is going to capture for measuring two nodes' similarity. The clustering task, single-labeled classification task and multi-labeled classification task are set under the same strategy in section IV-B. The ACC values for the clustering task and single-labeled classification task on Cora and Citeseer are plotted in FIGURE 4 and FIGURE 5. The F1 scores for the multi-labeled classification on PPI and dblp are plotted in FIGURE 6.

Generally, we can see that in FIGURE 4 and FIGURE 5, there is a trend that the performance of our algorithm improves when t increases. When $t \geq 3$, the ACC values tend to reach a steady state in most cases. When we set $t = 1$ which just takes the directed link information into account, its corresponding performance is the worst for both clustering task and single-labeled classification. Moreover, we observe a dramatic improvement of performance on Cora for the single-labeled classification, which demonstrate the importance of preserving $t > 1$ step information. Notably, PartNRL still shows superior performance compared with the compared algorithms with small t (e.g., $t = 3$ in FIGURE 4(c) and FIGURE 4(d)). While in FIGURE 6 and FIGURE 7, the F1 score values do not change much and the performances of our algorithm are relatively stable.

E. ANALYSIS OF TIME COMPLEXITY

The time complexity of PartNRL mainly depends on two designed phases. In the first phase, the time complexity of

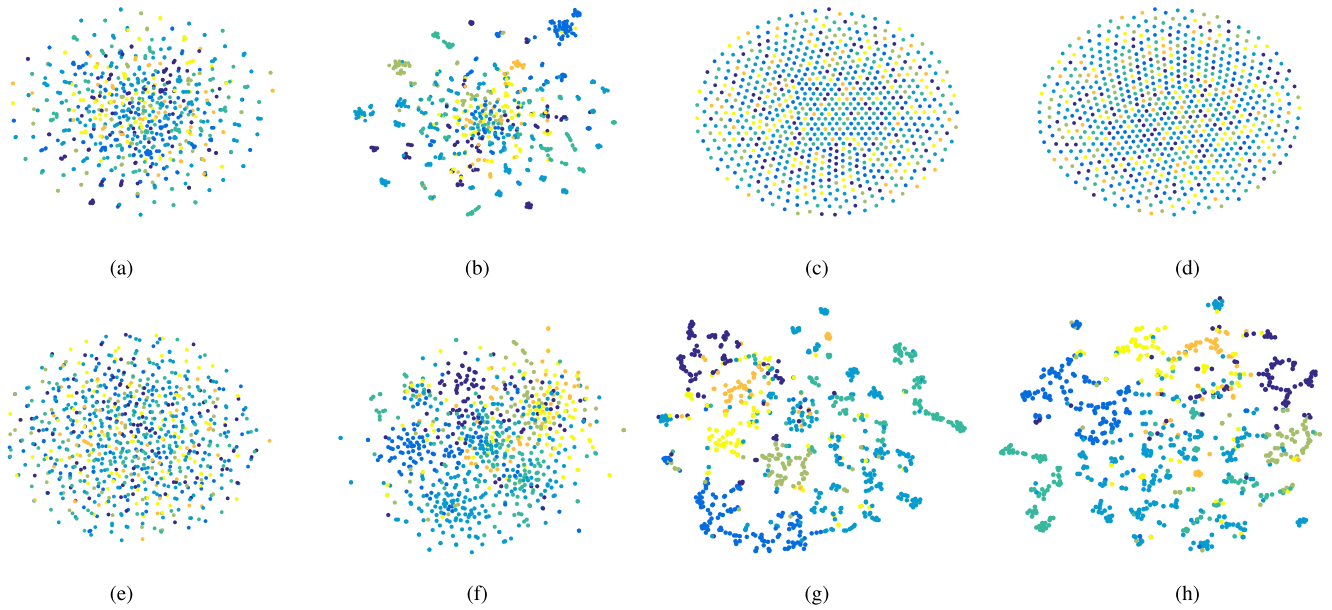


FIGURE 3. Visualization of Cora with $\rho = 30\%$. Each point indicates an author. Color of the point indicates the class label of the author. (a) DeepWalk. (b) LINE. (c) M-NMF. (d) GraRep. (e) Node2vec. (f) SNE. (g) PartNRL_LRW. (h) PartNRL_SRW.

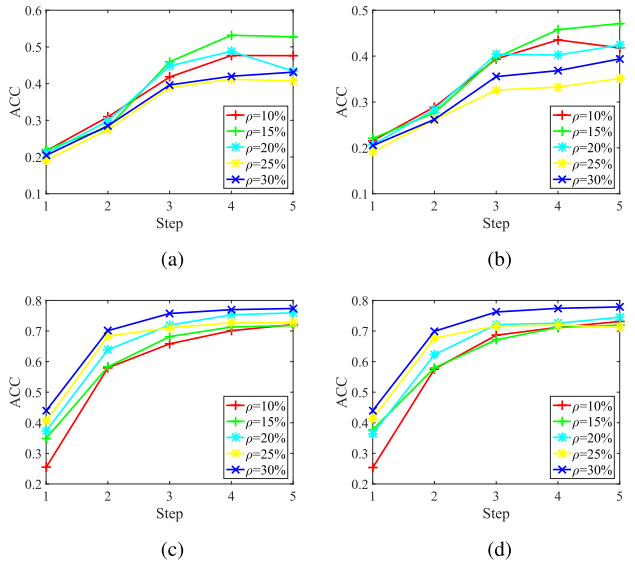


FIGURE 4. (a) and (b) are the ACC results with different step t on Cora for the clustering task. (c) and (d) are the ACC results with different step t on Cora for the single-labeled classification task. (a) PartNRL_LRW. (b) PartNRL_SRW. (c) PartNRL_LRW. (d) PartNRL_SRW.

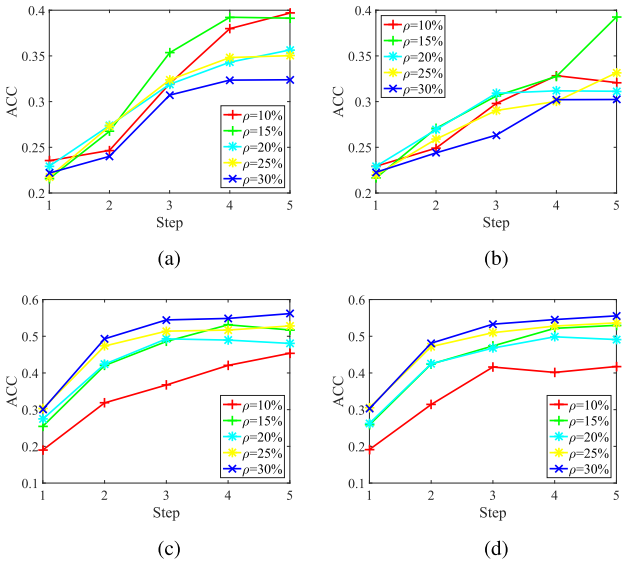


FIGURE 5. (a) and (b) are the ACC results with different step t on Citeseer for the clustering task. (c) and (d) are the ACC results with different step t on Citeseer for the single-labeled classification task. (a) PartNRL_LRW. (b) PartNRL_SRW. (c) PartNRL_LRW. (d) PartNRL_SRW.

the t-step local random walk is approximately $O(\bar{D}^t n_s)$ [21] where $\bar{D}$ is the average degree of the given network. And in the second phase, we apply the alias table method [20] to sample an "edge" which takes constant time. For each node, we iterate the sampling method l times, namely $O(ln_s)$. In the optimization procedure with negative sampling, the time complexity is $O(d(K+1))$ where K is the number of the negative samples. While the iteration number is usually proportional to the number of $|\mathcal{E}_s|$, therefore the time complexity in phase 2 is $O(dK|\mathcal{E}_s|)$. Let $\bar{D}_p$ be the average degree of nodes forming set $\mathcal{E}_s$, the time complexity in phase 2 can be rewritten as $O(dK\bar{D}_p n_s)$. And the time complexity for our whole model is $O(\bar{D}^t n_s + (\gamma dK\bar{D}_p + l)n_s)$ with γ representing the iteration number in the optimization process. In most cases, $\bar{D}$ is very small in real-world network and small t can provide satisfactory result (e.g., 5 in our experiment) which conform the efficiency of our algorithm.

Both LINE [20] and node2vec [31] adopt negative sampling and asynchronous SGD in the optimization process, the time complexity of LINE is $O(\gamma dK|\mathcal{E}|)$ which similarly

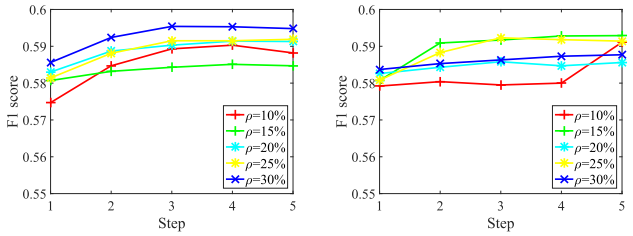


FIGURE 6. (a) is the F1 score results with different step t on PPI for the multi-label classification task. (b) is the F1 score results with different step t on PPI for the multi-label classification task. (a) PartNRL_LRW. (b) PartNRL_SRW.

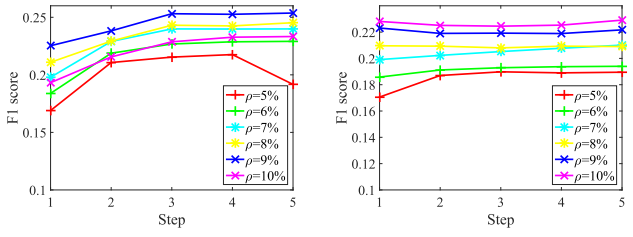


FIGURE 7. (a) is the F1 score results with different step t on dblp for the multi-label classification task. (b) is the F1 score results with different step t on dblp for the multi-label classification task. (a) PartNRL_LRW. (b) PartNRL_SRW.

can be rewritten as $O(\gamma dK\bar{D}n)$ and n is the total number of nodes in the entire network. Before the optimization process, node2vec proposes a flexible random walk method which costs $O(\gamma nb)$ to do the computation, where b is the length of random walk. Thus, the total time complexity of node2vec is $O(\gamma nb+\gamma dK\bar{D}n)$. Furthermore, DeepWalk adopts hierarchy softmax in the update procedure which costs $O(2\gamma wbn\log n)$ to do the computation where w is the window size. Additionally, The random walk procedure takes γ walks with length b start at each vertex, thus the whole time complexity of DeepWalk is $O(\gamma nb+2\gamma wbn\log n)$.

From the above analysis, we can see that the time complexities of three compared algorithms are all based on the total number of nodes n. They are more costly than PartNRL whose time complexity is based on the number of interested nodes n_s especially when n is quite large. Moreover, multiple threads are deployed for asynchronous stochastic gradient and considering the good quality of local random walk to parallel, the efficiency of PartNRL can be confirmed.

V. CONCLUSION AND FUTURE WORK

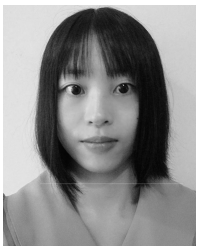
In this paper, we propose a novel network embedding model termed Partial Nodes Representation Learning (PartNRL) for the interested nodes in a large-scale network. It has two carefully designed phases to learn the low-dimensional network embedding results. The first phase is to find the similarities of each node pairs via local random walk. The second phase is to generate the node representations by maximizing a likelihood function with preserving the similarities from local random walk. Practically, we apply the negative sampling skill to update the variables with efficiency. Extensive experiments demonstrate the superior performance of PartNRL over some state-of-art network embedding methods.

New Nodes: Representation learning for new nodes [19], [44], [45] is also an essential issue in online network representation learning and dynamic network representation learning. When considering the highly dynamic nature of real-world networks, one common issue is how to find the representations of newly arrived nodes. Before new nodes arrive, we have the initial network $\mathcal{G}=(\mathcal{V},\mathcal{E})$ consisting of n nodes. The low-dimensional node representations $\mathbf{U}\in\mathbb{R}^{n\times d}$ for $\mathcal{V}$ can be generated by some traditionally network embedding algorithms. As the network evolves over time, n_s new nodes arrive, and $\mathcal{G}$ evolves into a larger network $\mathcal{G}_*$. To obtain the representations of additional nodes, we can treat these new nodes as the interested nodes in our model, in which the new-node representations $\mathbf{U_s}\in\mathbb{R}^{n_s\times d}$ can be generated. Finally we have the evolved network representation $\mathbf{U}_*\in\mathbb{R}^{(n+n_s)\times d}$ by combining $\mathbf{U}$ and $\mathbf{U_s}$. However, the embedding space derived from our algorithm may be different from that from the traditional embedding algorithm, and we leave it as our future work.

REFERENCES

[1] D. Yang, S. Wang, C. Li, X. Zhang, and Z. Li, "From properties to links: Deep network embedding on incomplete graphs," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 367–376.
[2] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1225–1234.
[3] L. Xu, X. Wei, J. Cao, and P. S. Yu, "Embedding identity and interest for social networks," in *Proc. 26th Int. Conf. World Wide Web Companion*, 2017, pp. 859–860.
[4] J. Zhang, C. Xia, C. Zhang, L. Cui, Y. Fu, and P. S. Yu, "BL-MNE: Emerging heterogeneous social network embedding through broad learning with aligned autoencoder," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2017, pp. 605–614.
[5] P. Yang and J. He, "Heterogeneous representation learning with structured sparsity regularization," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2016, pp. 539–548.
[6] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Homophily, structure, and content augmented network representation learning," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2016, pp. 609–618.
[7] B. Alharbi and X. Zhang, "Learning from your network of friends: A trajectory representation learning model based on online social ties," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2016, pp. 781–786.
[8] J. Lu *et al.*, "Structural property-aware multilayer network embedding for latent factor analysis," *Pattern Recognit.*, vol. 76, pp. 228–241, Apr. 2018.
[9] T. Li, J. Zhang, S. Y. Philip, Y. Zhang, and Y. Yan, "Deep dynamic network embedding for link prediction," *IEEE Access*, vol. 6, pp. 29219–29230, 2018.
[10] J. Lin *et al.*, "Multi-path relationship preserved social network embedding," *IEEE Access*, vol. 7, pp. 26507–26518, 2019.
[11] W. Shi, L. Huang, C.-D. Wang, J.-H. Li, Y. Tang, and C. Fu, "Network embedding via community based variational autoencoder," *IEEE Access*, vol. 7, pp. 25323–25333, 2019.
[12] J.-H. Li, C.-D. Wang, L. Huang, D. Huang, J.-H. Lai, and P. Chen, "Attributed network embedding with micro-meso structure," in *Proc. 23rd Int. Conf. Database Syst. Adv. Appl. (DASFAA)*, Gold Coast, QLD, Australia, May 2018, pp. 20–36.
[13] J. Chen, O. Zaïane, and R. Goebel, "Local community identification in social networks," in *Proc. Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Jul. 2009, pp. 237–242.

[14] M. Sozio and A. Gionis, "The community-search problem and how to plan a successful cocktail party," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 939–948.
[15] L. Waltman and N. J. van Eck, "A smart local moving algorithm for large-scale modularity-based community detection," *Eur. Phys. J. B*, vol. 86, no. 11, p. 471, 2013.
[16] C. Faloutsos, K. S. McCurley, and A. Tomkins, "Fast discovery of connection subgraphs," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 118–127.
[17] A. De, S. Bhattacharya, S. Sarkar, N. Ganguly, and S. Chakrabarti, "Discriminative link prediction using local, community, and global signals," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2057–2070, Aug. 2016.
[18] Z. Bu, Z. Wu, J. Cao, and Y. Jiang, "Local community mining on distributed and dynamic networks from a multiagent perspective," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 986–999, Apr. 2016.
[19] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.
[20] 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.
[21] W. Liu and L. Lü, "Link prediction based on local random walk," *Europhys. Lett.*, vol. 89, no. 5, 2010, Art. no. 58007.
[22] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.
[23] L. Liao, X. He, H. Zhang, and T.-S. Chua. (2017). "Attributed social network embedding." [Online]. Available: https://arxiv.org/abs/1705.04969
[24] T. F. Cox and M. A. Cox, *Multidimensional Scaling*. Boca Raton, FL, USA: CRC Press, 2000.
[25] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 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] 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.
[28] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. AAAI*, 2017, pp. 203–209.
[29] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, pp. 731–739.
[30] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 891–900.
[31] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.
[32] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[33] A. Q. Li, A. Ahmed, S. Ravi, and A. J. Smola, "Reducing the sampling complexity of topic models," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 891–900.
[34] 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.*, 2013, pp. 3111–3119.
[35] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2177–2185.
[36] Y. Goldberg and O. Levy. (2014). "word2vec explained: Deriving Mikolov et al.'s negative-sampling word-embedding method." [Online]. Available: https://arxiv.org/abs/1402.3722
[37] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 693–701.
[38] L. Liu, L. Xu, Z. Wangy, and E. Chen, "Community detection based on structure and content: A content propagation perspective," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2015, pp. 271–280.
[39] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 181–213, 2015.
[40] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 556–562.
[41] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.
[42] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1548–1560, Aug. 2011.
[43] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[44] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 387–396.
[45] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Proc. 3rd Int. Workshop Learn. Represent. Big Netw. (WWW BigNet)*, 2018, pp. 969–976.
[46] L. Huang, C.-D. Wang, and H.-Y. Chao, "A harmonic motif modularity approach for multi-layer network community detection," in *Proc. ICDM*, Nov. 2018, pp. 1043–1048.
[47] H. Zhang, C.-D. Wang, J.-H. Lai, and P. S. Yu, "Community detection using multilayer edge mixture model," *Knowl. Inf. Syst.*, pp. 1–23, 2018.
[48] L. Huang, C.-D. Wang, and H.-Y. Chao, "Overlapping community detection in multi-view brain network," in *Proc. BIBM*, Dec. 2018, pp. 655–658.
[49] J.-H. Li, C.-D. Wang, P.-Z. Li, and J.-H. Lai, "Discriminative metric learning for multi-view graph partitioning," *Pattern Recognit.*, vol. 75, pp. 199–213, Mar. 2018.
[50] L. Huang, C.-D. Wang, and H.-Y. Chao, "Higher-order multi-layer community detection," in *Proc. AAAI*, 2019.



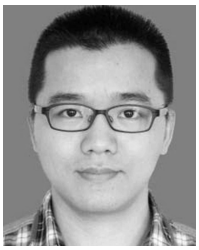
JUAN-HUI LI received the B.S. degree in software engineering from Sun Yat-sen University, Guangzhou, China, where she is currently pursuing the M.Sc. degree. Her research interests include machine learning and data mining.



LING HUANG received the bachelor's degree and master's degree from the South China University of Technology, in 2009 and 2013, respectively. She is currently pursuing the Ph.D. degree with Sun Yat-sen University. She has published several papers in international journals including *Pattern Recognition*, the IEEE Access, *Information Sciences*, *Knowledge-Based Systems*. She has published several papers in conferences, including AAAI, the IEEE ICDM, the IEEE BIBM, and DASFAA. Her research interest is data mining.



CHANG-DONG WANG received the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2013. He was a Visiting Student with the University of Illinois at Chicago, in 2012. He joined the School of Mobile Information Engineering, Sun Yat-sen University, in 2013, as an Assistant Professor, where he is currently an Associate Professor with the School of Data and Computer Science. His current research interests include machine learning and data mining. He has published over 100 scientific papers such as the IEEE TPAMI, the IEEE TKDE, the IEEE TCYB, and the IEEE TSMC-C. He has published over 100 scientific papers in international journals, including *Pattern Recognition*, the IEEE Access, KAIS, AAAI, ICDM, CIKM, SDM, DASFAA, and BIBM. His ICDM 2010 paper has received the Honorable Mention for the Best Research Paper Award. He has received the 2012 Microsoft Research Fellowship Nomination Award. He has the 2015 Chinese Association for Artificial Intelligence (CAAI) Outstanding Dissertation.



DONG HUANG received the B.S. degree in computer science from the South China University of Technology, China, in 2009, and the M.Sc. and Ph.D. degrees in computer science from Sun Yat-sen University, China, in 2011 and 2015, respectively. He joined South China Agricultural University, in 2015, where he is currently an Associate Professor with the College of Mathematics and Informatics. His research interests include data mining and pattern recognition.



JIAN-HUANG LAI received the M.Sc. degree in applied mathematics and the Ph.D. degree in mathematics from Sun Yat-sen University, China, in 1989 and 1999, respectively. He joined Sun Yat-sen University, in 1989, as an Assistant Professor, where he is currently a Professor with the School of Data and Computer Science. His current research interests include the areas of digital image processing, pattern recognition, multimedia communication, and wavelet and its applications. He has published over 200 scientific papers in international journals, such as the IEEE TPAMI, the IEEE TKDE, the IEEE TNN, the IEEE TIP, the IEEE TSMC-B, *Pattern Recognition*. He has published over 200 scientific papers in international conferences on image processing and pattern recognition. He also serves as a Standing Member of the Image and Graphics Association of China and also the Standing Director of the Image and Graphics Association of Guangdong.