Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.040818

ARTICLE

LC-NPLA: Label and Community Information-Based Network Presentation Learning Algorithm

Shihu Liu, Chunsheng Yang* and Yingjie Liu

School of Mathematics and Computer Science, Yunnan Minzu University, Kunming, 650504, China

*Corresponding Author: Chunsheng Yang. Email: ycs__246@163.com

Received: 31 March 2023 Accepted: 21 June 2023 Published: 27 February 2024

ABSTRACT

Many network presentation learning algorithms (NPLA) have originated from the process of the random walk between nodes in recent years. Despite these algorithms can obtain great embedding results, there may be also some limitations. For instance, only the structural information of nodes is considered when these kinds of algorithms are constructed. Aiming at this issue, a label and community information-based network presentation learning algorithm (LC-NPLA) is proposed in this paper. First of all, by using the community information and the label information of nodes, the first-order neighbors of nodes are reconstructed. In the next, the random walk strategy is improved by integrating the degree information and label information of nodes. Then, the node sequence obtained from random walk sampling is transformed into the node representation vector by the Skip-Gram model. At last, the experimental results on ten real-world networks demonstrate that the proposed algorithm has great advantages in the label classification, network reconstruction and link prediction tasks, compared with three benchmark algorithms.

KEYWORDS

Label information; community information; network representation learning algorithm; random walk

1 Introduction

In the real world, various complex networks are abstracted from social scenarios, such as social networks [1], traffic networks [2], citation networks [3] and biological networks [4], and so on [5]. Complex networks describe not only the structure information of nodes, but also the attribute information of nodes. Therefore, due to their quantities of valuable information [6], complex networks have been given special attention in many fields. Especially in recent years, NPLA has attracted the attention of many scholars. This information obtained from network representation learning can be used for downstream tasks, such as node classification [7], link prediction [8], network clustering [9] and network community detection [10], and so on [11–13].

However, with the development of society, the scale of complex networks has changed dramatically, and the link relationships between nodes have become more and more complicated. The more space is needed to store networks. As a result, these traditional complex network analysis methods

[14–21] are unable to deal with the impact of increasing network scale and specific machine learning tasks [22].

To solve this problem, many scholars gradually put forward NPLA based on the random walk strategy, which simulates natural language processing model to achieve network embedding, so as to map irregular dimensional complex networks into regular low-dimensional vector spaces. In this process, several models have been proposed. For example, the Deep Walk [23] algorithm is one of the most representative algorithms. And construction is similar to that of Word2vec [24]. The main idea is to use the random walk strategy to sample nodes in the network and learn the vector representation of nodes by utilizing the co-occurrence relationship between nodes in the network. The Line [25] algorithm describes the first-order similarity and second-order similarity of nodes with two different functions from the topological level of the network, and the final vector representation of nodes is obtained by integrating the above two levels. The Sdne [26] algorithm is the first model that applies deep learning to network representation. It considers that Deep Walk, the Line and other algorithms can't learn high-order nonlinear network structure at the level of the algorithm, so the idea of the Sdne algorithm is to describe the nonlinear network structure by using the nonlinear characteristics of the neural network. At the same time, the model still maintains the first-order similarity and second-order similarity mentioned in the Line algorithm. The Struc2vec [27] algorithm holds that the above models are all based on the hypothesis of nearest neighbor similarity, that is, the closer any two nodes are, the more similar they will be. However, in some networks, two nodes may not be close to each other in two-dimension space, but they may be more similar in three-dimensional space.

Unfortunately, these random walks-based NPLA algorithms can only capture the structural information of networks. That is to say, the community information of nodes fails to not be considered in the construction of NPLA. Aiming at this problem, some scholars have also therewith proposed the community information-based random walk NPLA. For instance, based on the local structure information and global structure information of network nodes, CARE [28] algorithm learns the representation vector of nodes. A novel semi-supervised Deep Walk method (SSDW [29]) is proposed for network representation learning, which successfully preserves the community structure in network embedding space. Specifically, a semi-supervised random walk sampling method with effective integration of paired constraints is proposed. Although random walk NPLA based on community information can effectively solve the impact of network size, a lot of information is lost when network is embedded. This is because the networks are always multi-information.

In addition, the link between the label information of nodes is also a kind of very important information in the network. For example, the bipartite graph [30] is a special kind of network (i.e., the vertex set in the graph is divided into two disjoint subsets, so that each edge connects the vertices of the two sets separately). To solve the problem of information loss during network embedding, thus, GraphRNA [31] based on the node attribute of the bipartite graph is proposed. The algorithm can take the direct neighbor of the node as the sampling target, and take the attribute of the node as the sampling target. In other words, attributes are also abstracted into nodes for a random walk in this algorithm, and so alleviates the problem of sampling deviation to central nodes.

Based on the above analysis and discussion, in this paper we propose a label and community information-based network presentation learning algorithm (LC-NPLA) to capture more information. In LC-NPLA, the first-order neighbors of nodes are reconstructed by using the community information and label information of nodes. Moreover, the degree information and label information of nodes are fused to construct a random walk strategy in LC-NPLA. The experimental results on

many network data demonstrated that the proposed algorithm achieves a great advantage in the label classification, network reconstruction and link prediction tasks.

The contribution of this paper is as follows:

- The first-order neighbors of nodes are reconstructed by utilizing the community information and label information of nodes.
- A random walk strategy is constructed by integrating the degree information and label information of nodes.
- The LC-NPLA algorithm proposed in this paper can achieve better performance, compared to other benchmark algorithms.

The rest of this article is organized as follows. Section 2 proposes the concepts of network embedding, random walk sequence and pairwise constraint matrix. Section 3 gives the detailed construction procedure of the proposed LC-NPLA. Section 4 gives the experimental materials. Section 5 gives the results and analysis. Section 6 gives the conclusion of this paper.

2 Preliminaries

In this section, some necessary knowledge is introduced, which are network embedding, random walk sequence and pairwise constraint matrix. For more detailed knowledge, one can refer to the references [23,29].

2.1 Network Embedding

In general, an undirected and unweighted network can be represented as a tuple $G = (V, E)$, where $V = \{v_i \,|i = 1, 2, \cdots, N\}$ is the set of nodes and $E = \left\{e_{ij} \,|i, j = 1, 2, \cdots, N\right\}$ is the set of edges. The purpose of network embedding is to find a mapping function $f: V \rightarrow U_i \in R^d$, where U_i is the representation vector of the corresponding v_i in low dimensional space.

2.2 Random Walk Sequence

The random walk traverses a network starting at a node. At any node, it goes to the neighbor of this node with probability a. A random walk sequence with v_i as the root node is denoted as $RW(i)$. It consists of the random variable $W_1(v_i), W_2(v_i), \cdots, W_k(v_i)$, where $W_{k+1}(v_i)$ is the next walk node that randomly selected from the neighborhood of v_k. As a result, a random walk sequence represents a path in each network.

2.3 Pairwise Constraint Matrix

Matrix C_{ml} is defined as the must-link constraint, if $C_{ml}(i, j) = 1$, then v_i and v_j belong to the same community; similarly, matrix C_{ul} is defined as the unlinked constraint, if $C_{ul}(i, j) = 1$, then v_i and v_j belong to different communities.

In order to facilitate understanding of this article, some of the relevant concepts are summarized in Table 1.

3 The Proposed LC-NPLA Algorithm

In this section, we propose LC-NPLA and discuss the main components of our algorithm. The specific algorithm flow is divided into the following two parts: Preprocessing process of nodes, Label and Degree strategy (**LD strategy**). And we also introduce the framework of LC-NPLA.

Table 1: Notations and descriptions

Notations	Descriptions
$LI\ (i)$	The label information vector of v_i
wl	Weight of label
pp	Predetermined proportion
μ	Number of random walks per node
$d\ (i)$	The degree of v_i
L	The set of labels
d	The dimension of the node embedding vector
$\Phi_{N\times d}$	Node embedding vector matrix
α	The learning rate of stochastic gradient descent algorithm

3.1 Preprocessing Process of Nodes

In Fig. 1, $v_0 - v_6$ and $v_7 - v_{10}$ represent the node and label value, respectively. There is no direct connection between v_0 and v_1, but the label value can make the direct connection between v_0 and v_1. Fig. 2 is a network with four communities. v_{13} and v_{22} have no direct connection in green community. Therefore, to enhance the node's ability that collects information, we consider integrating the label information and the community information of nodes to reconstruct its first-order neighbors.

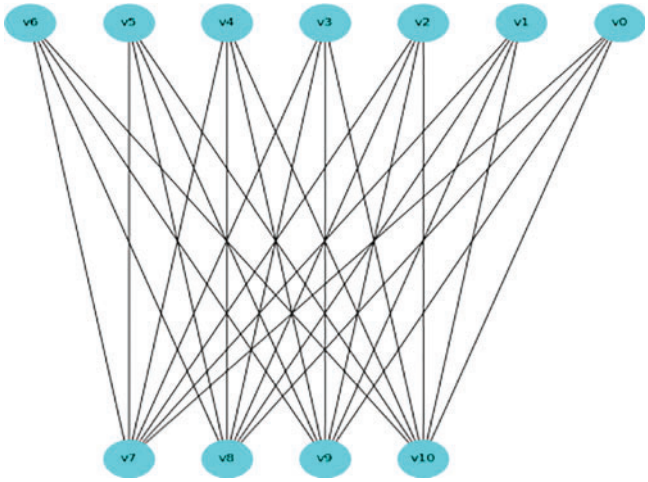

Figure 1: Bipartite graph

From the above discussion, we intend to use the Louvain [32] algorithm to calculate the community distribution of networks for ease of calculation. In this process, due to the limitation of experimental equipment, 10% and 20% of the community information is used to reconstruct the node's first-order neighbors, respectively. When the community information is 10% and 20%, the specific form of LC-NPLA is **LC-NPLA_P10** and **LC-NPLA_P20**. Therefore, the details of the preprocessing process are given as follows: in the newly first-order neighbors of defined nodes, nodes that belong to different communities but have the same label will be classified into the first-order neighbors of the known nodes. Nodes that belong to different communities but have different labels are not included

in the first-order neighbors of the nodes. Therefore, the new first-order neighbors of the nodes will contain more information.

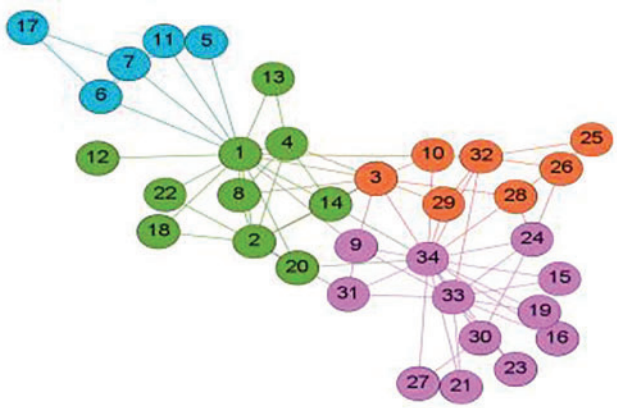

Figure 2: Network with community distribution

3.2 LD Strategy

During the preprocessing process of nodes, the new first-order neighbors of nodes are generated. In order to sample these, we design a **LD strategy**. Specifically, the **LD strategy** constructs node sequence from the structural level and attribute level of the network.

At the structural level, due to the complexity of node information, the similarity between node and node is asymmetrical even for undirected network. Therefore, KL divergence is selected to measure the structural information of networks based on computational complexity and other factors. So, the KL divergence is used to calculate the difference of the degree distribution of any two nodes as the structural information of the network. The calculation formula of KL divergence is defined as

$$KL_{ij} = \frac{1}{2}\left(\frac{d(i)}{N} log\left(\frac{d(i)}{d(j)}\right) + \frac{d(j)}{N} log\left(\frac{d(j)}{d(i)}\right)\right). \tag{1}$$

At the attribute level, because we can't directly represent the label information of nodes, we first utilize one-hot coding to express the label information of nodes. Then, to represent the attribute information of the network, the Hadamard product is used to calculate the correlation of label information of any two nodes.

In this paper, the weight of label information refers to the proportion of the total number of labels corresponding to each type of label information on all nodes. The specific calculation method is as follows:

$$wl_i = \frac{l_i}{L} \tag{2}$$

where l_i represents the number of labels i, and L represents the number of labels corresponding to all nodes. Parts of the modifications are listed below for your convenience.

The calculation formula is

$$AS_{ij} = \left|\sum_i wl_i \left(LI_i + LI_j\right)_i\right|. \tag{3}$$

where the calculation formula of the **LD strategy** is defined as

$$LDGS = pp * AS + (1 - pp) * KL, \tag{4}$$

where $LDGS$ indicates the global similarity matrix of nodes.

The pseudo-code of the **LD strategy** can be shown as following algorithm 1:

Algorithm 1: Label and degree strategy (LD strategy)

Input: V, pp, LI,wl, $d(i)$.
Output: $LDGS^{LD}_{n\times n}$.

```
1:  begin:
2:  LDGS^LD_{n×n} ← 0                    % Initialization phase
3:  for v_i, v_j in V do
4:      KL ← KL_ij                       % The KL_ij is calculated by Eq. (1)
5:      AS ← AS_ij                       % The AS_ij is calculated by Eq. (3)
6:      LDGS^LD_ij ← LDGS                % The LDGS is calculated by Eq. (4)
7:  end for
8:  end
9:  return LDGS^LD_{n×n}                 % Return the global similarity matrix of nodes
```

Algorithm 1 description: The Step 4 of algorithm is used to calculate the KL divergence of degree distribution of nodes. The Step 5 of algorithm is used to calculate the one-hot coded Hadamard product of the label information. The Step 6 of algorithm is used to calculate the global similarity matrix.

3.3 LC-NPLA Framework

The Skip-Gram model is a language model. Given a predefined window$w$, its main idea is to maximize the co-occurrence probability of words in a statement. The basic formula of Skip-Gram model is

$$\max_{f} \sum_{v\in V} \log P_r\left(N_s(v) \mid f(v)\right), \tag{5}$$

where for $v \in V$, $N_S(v) \subset V$ is defined as the neighborhood of v in the network. It is generated by v through the neighborhood sampling strategy S.

Under the Conditional independence hypothesis and Symmetry in feature space hypothesis, Eq. (5) is optimized into the form of probabilistic conjunction. However, in order to reduce the time complexity and facilitate the calculation, we use the log function to transform the form of probabilistic conjunction into the form of continued addition.

To simplify the optimization problem, we made two standard assumptions:

- Conditional independence: by assuming that the likelihood of observing a neighborhood node is independent of observing any other neighborhood node, we factorize the likelihood. The mathematical expression of the vector representation of the source is

$$P_r\left(N_s(v) \mid f(v)\right) = \prod_{u\in N_S(v)} P_r\left(u \mid f(v)\right). \tag{6}$$

- Symmetry in feature space: a source node and its neighborhood node have a symmetric effect in feature space. Accordingly, we model the conditional likelihood of every source-neighborhood node pair as a SoftMax unit parametrized by a dot product of their features. The mathematical expression is

$$P_r\left(u\left|f\left(v\right)\right.\right)=\frac{exp\left(f\left(u\right)f\left(v\right)\right)}{\sum_{m\in V}exp\left(f\left(m\right)f\left(v\right)\right)}. \tag{7}$$

With above assumptions, the objective in Eq. (5) simplifies to

$$\max_{f}\sum_{v\in V}\left[-\log Z_v+\sum_{u\in N_S(v)}f\left(u\right)f\left(v\right)\right], \tag{8}$$

where $Z_v=\sum_{m\in V}exp\left(f\left(v\right)f\left(m\right)\right)$ is expensive to compute for large networks.

As we can see from the second term of Eq. (8), the complexity is very high because we need to traverse the entire thesaurus, so we need to simplify the calculation of this step and reduce the complexity of the operation. Therefore, in order to reduce the complexity of Eq. (8), we convert Eq. (5) into

$$\max\prod_{\left(v_i,v_j\right)\in D}p\left(D=1\left|v_i,v_j\right.\right)\prod_{\left(v_i,v_j\right)\in\tilde{D}}p\left(D=0\left|v_i,v_j\right.\right), \tag{9}$$

where $\prod_{\left(v_i,v_j\right)\in D}p\left(D=1\left|v_i,v_j\right.\right)$ represents the probability that both node v_i and node v_j serve as context nodes at the same time, $\prod_{\left(v_i,v_j\right)\in\tilde{D}}p\left(D=0\left|v_i,v_j\right.\right)$ represents the probability that both node v_i and node v_j do not serve as context nodes at the same time, represents the set of context nodes and represents the set of non-context nodes. For a central node, we randomly select several groups from the negative samples corresponding to the central node to do gradient descent, which greatly reduces the computational complexity. The real example of the negative sample is shown as Fig. 3:

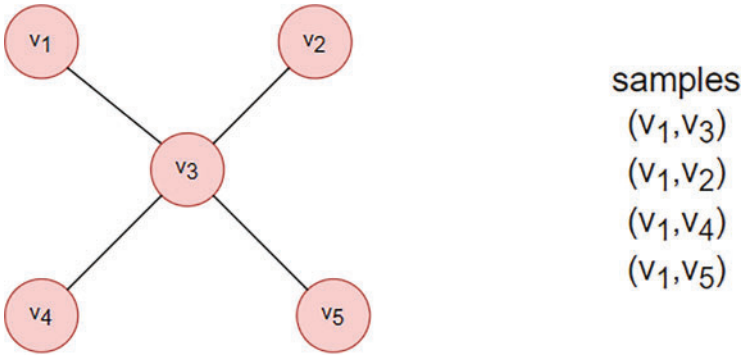

Figure 3: Network G

With v_1 as the central node, the positive sample is $\{(v_1, v_3)\}$, the negative sample is $\{(v_1, v_2), (v_1, v_4), (v_1, v_5)\}$. So, we approximate it using negative sampling [33].

The set of node sequences is entered into the Skip-Gram model. After training, we obtained the node embedding vector. The pseudo-code of the LC-NPLA can be shown as following Algorithm 2:

Algorithm 2: Label information and Community information-based network presentation learning algorithm (LC-NPLA)

Input: $G(V,E)$, wl, d, μ, pp.
Output: $\Phi_{N\times d}$.

1: begin:
2: $\Phi_{N\times d} \leftarrow 0$ % Initialization phase
3: for $j = 1, 2, \cdots, \mu$ do
4: $\Gamma \leftarrow Shuffle(V)$ % Shuffling the order of the list elements
5: for v_i in Γdo
6: $RW(i) \leftarrow \text{LD } strategy_i$ % Random walk sequence is generated
7: $\Phi_{N\times d} \leftarrow Skip-Gram(\Phi_{N\times d}, \text{RW (i)}, w)$ % Node vector representation is generated
8: end for
9: end for
10: end
11: return $\Phi_{N\times d}$ % Return node embedding vector matrix

Algorithm 2 description: Given a real network, the Step 6 of algorithm updates the first-order neighbors of nodes through the preprocessing process of nodes, then generates a random walk $RW(i)$ for each vertex. Finally, the Step 7 of algorithm uses Skip-Gram model to update the embedding vector of nodes with Eq. (5).

Algorithm 3: Skip-Gram

Input: v_j, $RW(i)$, α.
Output:$\Phi_{N\times d}$.

1: begin:
2: $\Phi_{N\times d}$ % Initialization phase
3: for v_j in $RW(i)$ do
4: for u_k in $RW(i)[j-w, j+w]$do
5: $J(f) \leftarrow -\log \Pr\left(u_k \left| f\left(v_j\right)\right.\right)$
6: $\Phi_{N\times d} \leftarrow \Phi_{N\times d} - \alpha \dfrac{\partial J}{\partial f}$ % Stochastic gradient descent
7: end for
8: end for
9: end
10: return $\Phi_{N\times d}$ % Return the global similarity matrix of nodes

Algorithm 3 description: This paper uses Skip-Gram model to update the embedded vector matrix of nodes according to Eq. (5). Line 2: Initializes a node-embedding vector matrix. The details of the algorithm are described in the following formula. Lines 3–8: The outer for loop represents traversing every node in the random walk sequence of a given node v. The inner for loop represents that each node in the random walk sequence is taken as the target, and w nodes before and after the target node are combined with the target node. The specific operation mode is to update the initial vector matrix of the node by using the stochastic gradient descent algorithm.

4 Experimental Materials

In this section, we introduce some experimental materials, which are the experiment datasets, evaluation criteria and benchmark algorithms. In this paper, the experimental environment is listed in Table 2.

Table 2: Experimental environment

Parameter	Parameter value
RAM	62 GB
Programming	Python
CPU	13th Gen Intel(R) Core (TM) i9-13900K
System	Ubuntu 20.04

4.1 Description

To verify the effectiveness of LC-NPLA, experiments are conducted on the following 10 real network data sets, including Polbooks, Adjoun, Football, Europe, USA, Polblogs, Wiki, Cora, Citeseer and PPI. The information of these data sets is listed in Table 3.

4.2 Evaluation Criteria

Here, the evaluation criteria of Micro-F1, Macro-F1 and Weighted-F1 are used for the label classification task; the MAP evaluation criterion is used for the network reconstruction task; the AUC evaluation criterion is used for the link prediction task.

4.2.1 Label Classification Task

- **Micro-F1:** this criterion is determined globally by counting total true positives (TP), false negatives (FN), and false positives (FP). The mathematical expression of it is

$$F_I^1 = \frac{2PR}{P+R} \tag{10}$$

where

$$P = \frac{\sum_{A\in L} TP(A)}{\sum_{A\in L} (TP(A) + FP(A))}$$

is the accuracy rate and

$$R = \frac{\sum_{A\in L} TP(A)}{\sum_{A\in L} (TP(A) + FN(A))}$$

is the recall rate for all labels.

- **Macro-F1:** This criterion is the average of F1 scores for all labels with following formula:

$$F_A^1 = \frac{\sum_{A\in L} F1(A)}{|L|}, \tag{11}$$

where

$$F1\,(A) = \frac{2P\,(A)\,R\,(A)}{P\,(A) + R\,(A)}$$

is $F1$ scores for the label A, P(A) and R(A) is the accuracy rate and recall rate of label A, and L is the set of labels.

Table 3: Experimental datasets[1]

Dataset	N	$\|E\|$	$<d>$	$<c>$	$<l>$	$<task>$
Polbooks	105	441	8	0.4875	4	Label classification
USA	1190	13599	23	0.6090	4	
Adjoun	112	425	8	0.2000	2	Network reconstruction
Europe	399	5995	30	0.5670	4	
Football	115	613	6	0.3708	12	Link prediction task
Polblogs	1224	16718	27	0.3610	2	
Wiki	2405	16523	11	0.4800	17	
Cora	2708	5429	4	0.2461	7	Label classification and network reconstruction
Citeseer	3312	4732	3	0.2590	6	
PPI	3890	76584	19	0.1660	50	

Notes:

- **Polbooks**[2] https://www.cc.gatech.edu/dimacs10/archive/clustering.shtml: The network is made up of books about US politics published in 2004.
- **Adjoun**[3] https://networkrepository.com/adjnoun.php: The network is made up of common adjective and nouns in "David Copperfield".
- **Football**[4] http://vlado.fmf.uni-lj.si/pub/networks/data/sport/football.htm: It is the network of US football games between division IA colleges.
- **Europe**[5] http://ec.europa.eu/: It is European air-traffic network that collected from the Statistical Office of the European Union from January to November, 2016.
- **USA**[6] https://transtats.bts.gov/: It is the American air-traffic network that collected from the Bureau of Transportation Statistics from January to October, 2016.
- **Polblogs**[7] http://moreno.ss.uci.edu/data.html#blogs: The network is the maximum connected set of blog pages network about US politics.
- **Wiki**[8] https://gitee.com/yang-doctor/GraphEmbedding/tree/master/data: This is a web of web links and contains the category of each web page.
- **Cora**[9] https://linqs-data.soe.ucsc.edu/public/lbc/cora.tgz: It is the scientific publications citation network.
- **Citeseer**[10] https://linqs-data.soe.ucsc.edu/public/lbc/citeseer.tgz: It is the scientific publications citation network.
- **PPI**[11] http://snap.stanford.edu/graphsage/ppi.zip: It is a subgraph of the PPI network for Homo Sapiens.

- **Weighted-F1:** This criterion is the weighted average of Macro-F1 with the following for:

$$F_W^1 = \frac{\sum_{A\in L, w_i\in W} w_i F_A^1}{|L|}, \tag{12}$$

where $W = \{w_1, w_2, \cdots, w_{|L|}\}$ is the weighting set with the condition $\sum_{i=1}^{|L|} w_i = 1$.

[1] $<d>$ is the average degree of network, $<c>$ is the average clustering coefficient of network, $<l>$ is the number of nodes labels, $<task>$ indicates that different tasks are performed in different data sets.

4.2.2 Network Reconstruction Task

- **MAP:** This criterion can be applied to estimate the average precision of nodes, and the following mathematical expression is:

$$MAP = \frac{\sum_{v_i \in V} AP(v_i)}{N}, \tag{13}$$

where

$$AP(v_i) = \frac{\sum_{j=1}^{k} predict_{v_i}(v_j) \cdot \Delta_{v_i}(v_j)}{\left|\Delta_{v_i}(v_j) = 1\right|}$$

is the accuracy of v_i, $\Delta_{v_i}(v_j) = 1$ indicates that there is an edge between v_i and v_j on the original network, and $predict_{v_i}(v_j)$ represents the predicted value of this edge.

4.2.3 Link Prediction Task

- **AUC:** This criterion can be interpreted as the probability that an edge randomly selected in the test set would be assigned a higher similarity than that of an edge randomly selected in the unknown edge set. It has the following mathematical expression:

$$AUC = \frac{2n_1 + n_2}{2n}, \tag{14}$$

where n is the times of independent comparisons, n_1 is the times that the similarity of the test edge is greater than that of the unknown edge and n_2 is the times that they have the same similarity.

4.3 Benchmark Algorithms

Here, we introduce three network presentation learning algorithms that can be applied to make a comparison with our proposed algorithm. Detailed descriptions of them can be found in references [25–27].

4.3.1 Line Algorithm

The main mechanism of line algorithm can be divided into to two stages to learn the d-dimensional feature representation of nodes. In the first step, it learns the $d/2$-dimensional representation of nodes by simulating the BFS mode on adjacent nodes. In the second step, it learns the remaining $d/2$ -dimensional representation of the node by sampling nodes that are 2-hops away from the source node.

4.3.2 Sdne Algorithm

The main mechanism of the Sdne algorithm can be divided into to two stages to learn the feature representation of nodes. In the first step, it learns the local structure of network by depth automatic encoder. In the second step, it learns the global structure of network by Laplace mapping.

4.3.3 Struc2vec Algorithm

The basic idea of the Struc2vec algorithm is that if two nodes own more similar structures in a network, they should have a higher similarity. The mechanism of it can be divided into four steps. In the first step, structural similarity between each pair of nodes is learned. In the second step, a weighted multilayer network is constructed. In the third step, the multi-layer network is used to generate a

sequence of nodes for each node. In the fourth step, feature representation of nodes is learned by Skip-Gram model.

5 Results and Analysis

In this section, we focus on testing the effectiveness of the proposed LC-NPLA for label classification, network reconstruction, and link prediction task. Detailed descriptions of benchmark algorithms can be found in references [25–27].

5.1 Results and Analysis of Label Classification Task

We use the one-vs-rest logistic regression that applied in Log-linear [34] algorithm to do the label for nodes. Then we select an arbitrary percentage of labeled nodes as the training data, and the remaining percentage of nodes are regarded as the testing data. In this experiment, the training ratio on Polbooks, USA, Cora, Citeseer and PPI in was set to 10%, 30%, 50%, 70% and 90%, respectively. For LC-NPLA, the parameters are separately made to $\mu = 190$, $w = 5$, $d = 128$ and $l = 15$. What is more, the dimension of node representation vector of the benchmark algorithms is set to 128 dimensions. And the experiment is executed 10 times for the purpose of avoiding the chance of experimental bias. The obtained results for the label classification task summarized in Tables 4–8.

Table 4: Label classification results in Polbooks

	Labeled nodes%	10%	30%	50%	70%	90%
$F_I^1\%$	LC-NPLA_P20	**77.89**	71.62	**81.13**	**78.13**	**100.00**
	LC-NPLA_P10	76.84	**75.68**	75.47	**78.13**	81.82
	Line	49.47	39.19	49.06	53.13	18.18
	Sdne	69.47	66.22	66.04	68.75	81.82
	Struc2vec	46.81	51.35	61.54	75.00	50.00
$F_A^1\%$	LC-NPLA_P20	**60.54**	**57.98**	**65.05**	56.53	**66.67**
	LC-NPLA_P10	59.32	54.98	61.25	**66.40**	54.44
	Line	28.48	27.81	42.52	52.13	11.11
	Sdne	48.94	47.60	48.20	49.47	54.44
	Struc2vec	46.81	51.35	61.54	75.00	50.00
$F_W^1\%$	LC-NPLA_P20	**74.71**	**72.28**	**81.12**	76.81	**100.00**
	LC-NPLA_P10	73.30	**72.28**	76.13	**78.23**	81.52
	Line	38.48	36.51	48.28	52.94	15.15
	Sdne	64.27	62.53	65.43	67.29	81.52
	Struc2vec	37.97	48.41	58.04	70.71	51.11

Table 5: Label classification results in USA

Labeled nodes%	10%	30%	50%	70%	90%
LC-NPLA_P20	24.37	26.77	26.89	**27.17**	**32.77**
LC-NPLA_P10	25.77	25.57	25.21	26.33	30.25

(Continued)

Table 5 (continued)

	Labeled nodes%	10%	30%	50%	70%	90%
$F_I^1\%$	Line	25.21	27.85	27.23	25.77	26.05
	Sdne	**26.70**	24.97	24.20	26.05	29.41
	Struc2vec	24.63	**29.74**	**30.54**	25.14	31.67
$F_A^1\%$	LC-NPLA_P20	24.27	26.29	26.57	**26.65**	**32.56**
	LC-NPLA_P10	**25.38**	25.03	24.83	26.18	29.99
	Line	24.93	27.84	27.19	25.78	26.36
	Sdne	20.73	21.25	20.22	22.00	28.08
	Struc2vec	24.48	**29.16**	**30.42**	25.05	31.75
$F_W^1\%$	LC-NPLA_P20	24.26	26.32	26.50	**26.52**	**32.10**
	LC-NPLA_P10	**25.40**	25.01	24.64	26.07	28.18
	Line	24.90	27.84	27.13	25.66	26.05
	Sdne	20.60	21.14	20.05	21.68	27.71
	Struc2vec	24.50	**29.22**	**30.60**	24.93	31.64

Table 6: Label classification results in Cora

	Labeled nodes%	10%	30%	50%	70%	90%
$F_I^1\%$	LC-NPLA_P20	49.30	**54.32**	54.43	**56.21**	**53.87**
	LC-NPLA_P10	**50.12**	54.06	**55.17**	55.10	53.14
	Line	22.19	22.84	24.67	23.49	26.20
	Sdne	17.14	17.46	16.63	17.59	14.94
	Struc2vec	30.11	33.12	35.30	37.10	44.12
$F_A^1\%$	LC-NPLA_P20	**45.82**	**51.10**	**50.94**	**52.51**	**49.30**
	LC-NPLA_P10	45.27	50.85	50.92	51.14	48.09
	Line	22.19	22.84	24.67	23.49	26.20
	Sdne	3.37	3.58	3.88	4.34	3.84
	Struc2vec	18.24	23.97	26.03	26.99	30.92
$F_W^1\%$	LC-NPLA_P20	48.30	**53.21**	53.57	**55.04**	**52.51**
	LC-NPLA_P10	**48.59**	53.13	**54.36**	53.90	52.09
	Line	19.19	20.11	21.87	19.65	21.88
	Sdne	8.86	9.27	9.35	9.98	8.44
	Struc2vec	25.60	30.05	32.22	33.94	40.68

Table 7: Label classification results in Citeseer

	Labeled nodes%	10%	30%	50%	70%	90%
	LC-NPLA_P20	**34.79**	**38.68**	39.92	**42.86**	41.27
	LC-NPLA_P10	34.32	37.73	**40.04**	41.85	**44.28**
$F_I^1\%$	Line	20.16	19.84	18.78	20.22	24.40
	Sdne	26.94	28.29	30.37	31.29	30.12
	Struc2vec	26.63	26.64	29.47	31.99	35.54
	LC-NPLA_P20	**31.52**	**34.75**	**36.46**	**39.65**	35.28
	LC-NPLA_P10	31.47	34.40	36.17	38.01	**39.93**
$F_A^1\%$	Line	17.77	15.96	15.25	15.97	18.81
	Sdne	17.76	18.81	21.50	22.22	20.41
	Struc2vec	22.85	23.59	25.47	27.08	29.29
	LC-NPLA_P20	**33.94**	**37.72**	**39.19**	**42.18**	39.25
	LC-NPLA_P10	33.89	36.93	39.18	40.81	**43.12**
$F_W^1\%$	Line	19.50	18.35	17.50	18.59	21.98
	Sdne	20.93	21.87	24.79	25.81	23.65
	Struc2vec	25.26	26.34	29.13	31.21	35.44

Table 8: Label classification results in PPI

	Labeled nodes%	10%	30%	50%	70%	90%
	LC-NPLA_P20	7.48	8.54	8.47	8.71	11.83
	LC-NPLA_P10	7.31	8.48	9.48	9.94	10.25
$F_I^1\%$	Line	7.54	7.80	7.99	7.98	8.80
	Sdne	**11.31**	**13.73**	**14.14**	**13.66**	**14.86**
	Struc2vec	7.41	8.01	8.44	8.61	9.24
	LC-NPLA_P20	5.70	6.09	6.20	6.35	7.76
	LC-NPLA_P10	5.51	6.08	6.92	7.02	6.82
$F_A^1\%$	Line	6.09	5.98	6.10	6.14	6.76
	Sdne	**8.35**	**10.97**	**11.63**	**11.62**	**13.90**
	Struc2vec	5.34	5.96	6.23	6.16	6.89
	LC-NPLA_P20	6.64	7.60	7.60	8.11	10.21
	LC-NPLA_P10	6.57	7.57	8.51	8.65	8.36
$F_W^1\%$	Line	7.16	7.25	7.53	7.74	7.81
	Sdne	**10.16**	**13.15**	**14.05**	**13.56**	**15.47**
	Struc2vec	6.77	7.52	8.02	8.11	8.86

5.1.1 Results on Polbooks Data Set

The experimental results on Polbooks are shown in Table 4. From Table 4, we find that LC-NPLA_P10 and LC-NPLA_P20 are superior to the other benchmark algorithms. This reason may be that the embedding dimension is close to the number of nodes. In reality, LC-NPLA_P20 only needs to mark 20% of the nodes to achieve the great performance of Sdne algorithm with 90% data. This strong performance when only small fractions of the network are labeled is a core strength of our approach.

However, as the ratio of community information increases, noise may affect the accuracy of the algorithm. As a consequence, the Micro-F1, Macro-F1, and Weighted-F1 scores of the LC-NPLA_P20 may be lower than the LC-NPLA_P10. Based on the above analysis, compared with the experimental results on USA, Cora and Citeseer, we find that LC-NPLA can also obtain better results on small data sets.

5.1.2 Results on USA Data Sets

The experimental results on USA are shown in Table 5. From Table 5, when the training ratio is 10%, 30% and 50%, the Micro-F1, Macro-F1 and Weighted-F1 scores of the best LC-NPLA are all lower than those of benchmark algorithms. However, when the training ratio reaches 70%, the three scores of the best LC-NPLA are higher than those of three benchmark algorithms. This reason may be that the ability of the best LC-NPLA to get information is affected by the training ratio.

Specially, although the USA has the largest number of edges for Polbooks, USA, Cora and Citeseer, LC-NPLA can still enhance the ability of label classification when the training ratio is 70%. And the Micro-F1, Macro-F1 and Weighted F1 scores of the best LC-NPLA are 1.12%, 0.87% and 0.86% higher than that of the best benchmark algorithm. When the training ratio is 90%, the Micro-F1, Macro-F1 and Weighted F1 scores of the best LC-NPLA are 1.10%, 0.81% and 0.46% higher than that of the best benchmark algorithm, respectively.

Based on the above analysis, although the effect of LC-NPLA is affected by the training ratio, the advantages of LC-NPLA are gradually obvious with the increase of training ratio. When the training ratio increases, it is found that LC-NPLA has obvious advantages in a dataset with a large number of edges.

5.1.3 Results on Cora and Citeseer Data Sets

The experimental results on Cora and Citeseer are shown in Tables 6 and 7. For Cora and Citeseer, three scores of the best LC-NPLA are higher than that of the Line algorithm and Struc2vec algorithm. This reason may be that label information and the average clustering coefficient of Cora and Citeseer affect the label classification task. In a nutshell, combining experimental results from Polbooks, USA, Cora and Citeseer, we found that LC-NPLA had better generalization ability in citation network datasets.

5.1.4 Results on PPI Data Set

The experimental result on PPI is shown in Table 8. For PPI, it is evident that the Micro-F1, Macro-F1 and Weighted-F1 scores for LC-NPLA_P20 and LC-NPLA_P10 are lower than those of the SDNE model, indicating the SDNE model's superior local and global learning abilities compared to the LC-NPLA model. Apart from special cases, the LC-NPLA model outperforms the Line and Struc2vec models in terms of Micro-F1, Macro-F1, and Weighted-F1 scores. Although the LC-NPLA

model shows some improvement over the Line and Struc2vec models in these metrics on the PPI network dataset, as presented in Table 8, the enhancement effect remains minimal.

5.2 Results and Analysis of Network Reconstruction Task

We rebuild the proximity of the nodes and sort the nodes by proximity and calculate the proportion of true links in the top k predictions as the reconstruction accuracy. And for a network with a large number of nodes, the number of possible node pairs ($|V|(|V|-1)$) may be very large, so 50% of the number of nodes is randomly selected for evaluation. The experimental results are shown in Fig. 4.

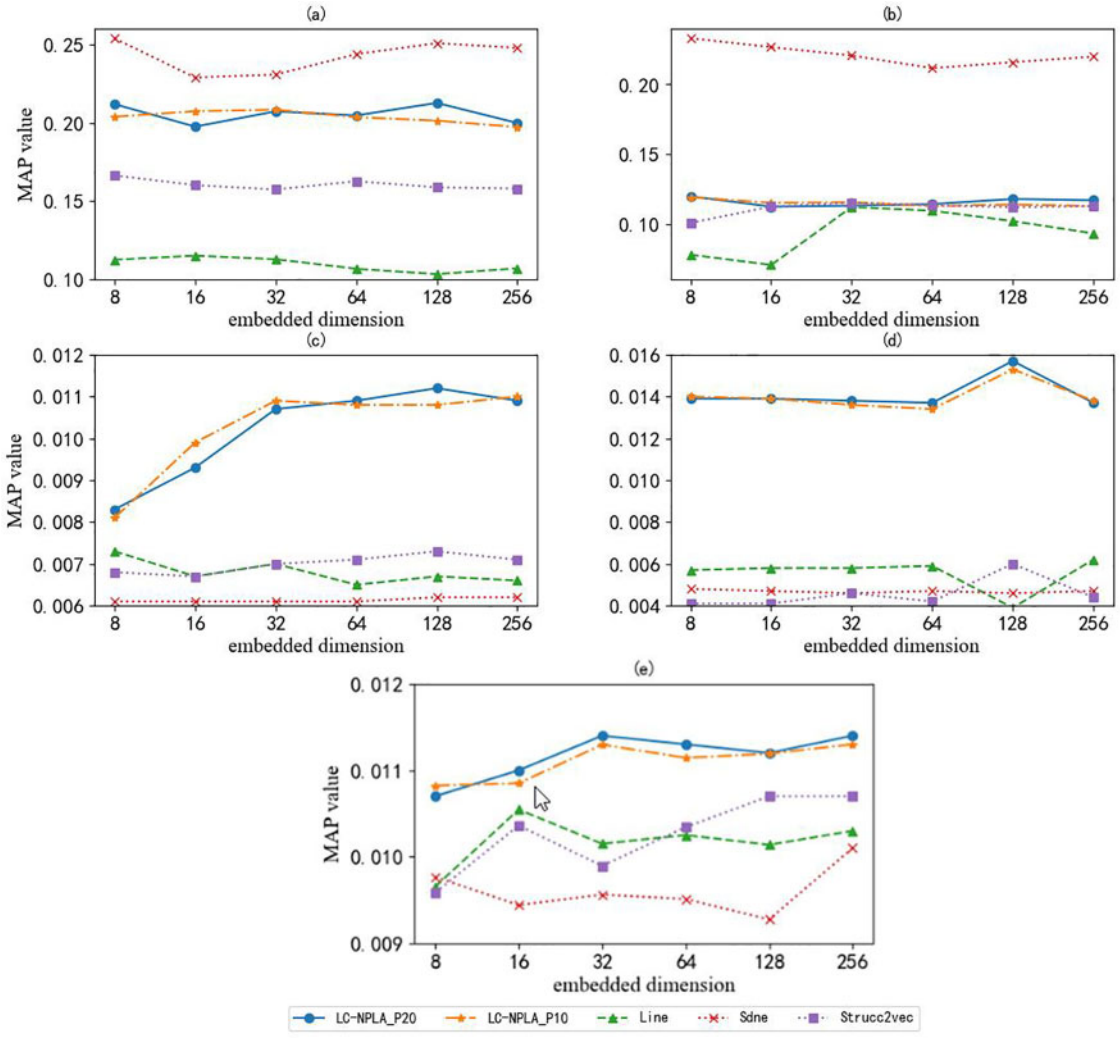

Figure 4: The MAP value of different data sets

5.2.1 Results on Adjoun and Europe Data Sets

Here we can see that the performance of LC-NPLA is highly data set dependent. It achieves good performance on Polblogs but performs poorly on other data sets. As a result, the reconstruction performance of the Sdne algorithm is better than the best LC-NPLA algorithm. The reason may be

that the Sdne algorithm collects both local structure information and global structure information. In the Fig. 4a, as the dimension of network embedding increases, we find that the MAP value of LC-NPLA increases weakly. The reason may be that superposition of information leads to overfitting of the model. In the Fig. 4b, we find that the LC-NPLA, Line algorithm, the Sdne algorithm and Struc2vec algorithm are not sensitive to the Europe data set. But in the network reconstruction task, the LC-NPLA is still superior to the Struc2vec algorithm and Line algorithm. Based on the above analysis, we find that LC-NPLA is still effective for the network reconstruction task.

5.2.2 Results on Cora, Citeseer and PPI Data Sets

In the Figs. 4c–4e, here we observe better rebuilding performance for citation network data sets like Cora and Citeseer. The reason may be that label information affects reconstruction performance. These experimental results show the influence of size on reconstruction error. With a few exceptions, MAP values increase as the number of dimensions increases. This makes sense, because more dimensions can store more information.

5.3 Results and Analysis of Link Prediction Task

To ensure the accuracy of the LC-NPLA, we select the hidden ratio of 15%, 30% and 45% to divide the train set, verification set and test set. Specially, the degree of AUC greater than 0.50 measures how well the algorithm is better than the algorithm selected randomly. And comparison with embedding based methods bootstrapped using binary operators: Hadamard, Weighted-L1 and Weighted-L2 (See Table 9 for definitions). We summarize our results for link prediction in Table 10.

Table 9: Choice of binary operators for learning edge features

Operator	Symbol	Definition
Hadamard	$\boxdot$	$\left[f\left(v_i\right) \boxdot f\left(v_j\right)\right]_i = f_i\left(v_i\right) f_i\left(v_j\right)$
Weighted-L1	$\|x\|_1$	$\left\|f\left(v_i\right) \cdot f\left(v_j\right)\right\|_{1i} = \left\|f_i\left(v_i\right) - f_i\left(v_j\right)\right\|$
Weighted-L2	$\|x\|_2$	$\left\|f\left(v_i\right) \cdot f\left(v_j\right)\right\|_{2i} = \left\|f_i\left(v_i\right) - f_i\left(v_j\right)\right\|^2$

Table 10: Area Under Curve (AUC) scores for link prediction

Operator	Algorithm	Football	Polblogs	Wiki
Hadamard	LC-NPLA_P20	0.5837	**0.7248**	0.5736
	LC-NPLA_P10	0.5746	0.7202	0.5805
	Line	0.4725	0.5843	0.5539
	Sdne	**0.7014**	0.6336	**0.6311**
	Struc2vec	0.4389	0.6304	0.5528
Weighted-L1	LC-NPLA_P20	0.5938	0.7116	0.6066
	LC-NPLA_P10	0.6061	**0.7133**	0.5980
	Line	0.5751	0.6513	0.5813
	Sdne	**0.6067**	0.6488	**0.6504**

(Continued)

Table 10 (continued)

Operator	Algorithm	Football	Polblogs	Wiki
	Struc2vec	0.5860	0.6283	0.4308
Weighted-L2	LC-NPLA_P20	0.5893	0.7050	0.6286
	LC-NPLA_P10	0.6026	**0.7088**	0.6383
	Line	0.5719	0.6436	0.5609
	Sdne	**0.6230**	0.6773	**0.6840**
	Struc2vec	0.5736	0.6179	0.4426

5.3.1 Results on Polblogs Data Set

From Table 10, we find that LC-NPLA has an AUC value greater than 0.50 in each data set. This indicates that LC-NPLA has the predictive ability for Polblogs, Football and Wiki. For Polblogs, LC-NPLA performs better on Polblogs than on Football and Wiki. Therefore, the combination of the three operators with LC-NPLA is stable and gives the best performance on average. This reason may be that the higher average degree increases the predictive ability of LC-NPLA.

5.3.2 Results on Football Data Set

For Football, the link prediction ability of the Sdne algorithm is better than that of LC-NPLA, this reason may be that it can capture the higher-order structure in the network. For the Hadamard operator, neither the Line nor the Struc2vec has predictive ability. Therefore, in the Hadamard operator, the best LC-NPLA is not comparable with the Line algorithm and the Struc2vec algorithm. For the Weighted-L1 operator, compared with the Line algorithm and the Struc2vec algorithm, the best LC-NPLA gains 3.10% and 2.01%, respectively. For the Weighted-L2 operator, gains of 3.07% and 2.90% are obtained, respectively.

5.3.3 Results on Wiki Data Set

For Wiki, the best LC-NPLA gains 2.66% and 2.77% for the Hadamard operator, respectively. For the Weighted-L1 operator, compared to the Line and the Struc2vec, the best LC-NPLA gains 2.53%. For the Weighted-L2 operator, 7.74% is obtained. Experimental results show that LC-NPLA has better link prediction ability on Polblogs than Wiki. This reason may be that LC-NPLA has a better prediction advantage for network data sets with fewer nodes, when the number of edges is similar.

All in all, combining experimental results from Polblogs, Football and Wiki, it is found that the link prediction ability of LC-NPLA is not obvious in Table 10, but the LC-NPLA still has predictive ability on Football, Polblogs and Wiki.

5.4 Time Complexity of LC_NPLA

The Fig. 5 illustrates the runtime of the LC_NPLA algorithm in comparison to other algorithms. As shown in the figure, Struc2vec algorithm requires the most time among all the remaining nine datasets except for Europe dataset, while SDNE algorithm is the fastest. It can be observed that LC_NPLA algorithm not only exhibits an improved embedding effect, but also demonstrates a favorable time complexity compared to Struc2vec algorithm.

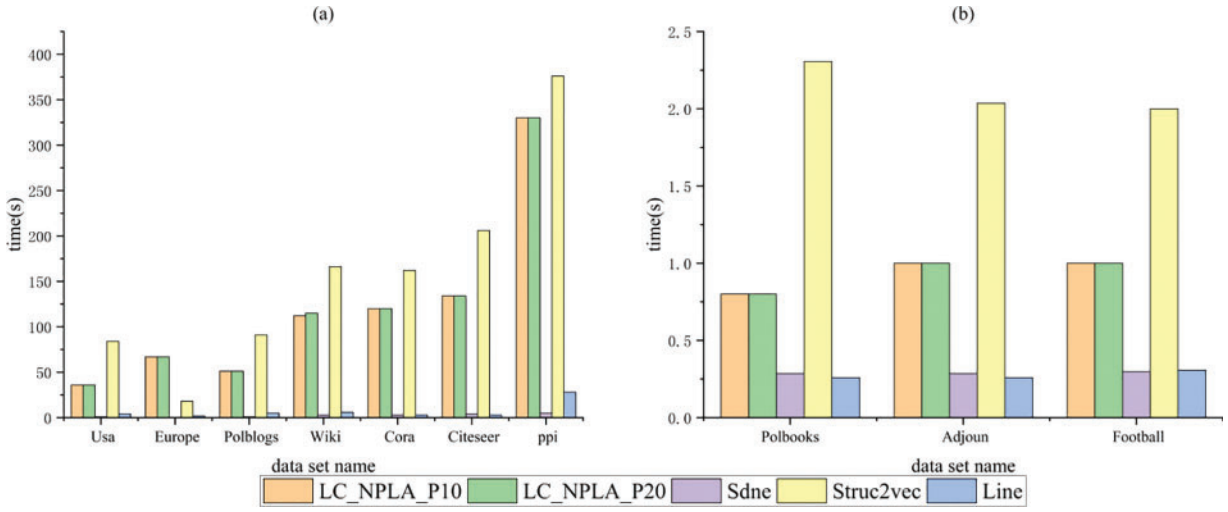

Figure 5: The running time of the algorithm on different network

6 Conclusions

To further improve the application of node representation vector in downstream tasks, this paper has designed a label and community information-based network presentation learning algorithm. By integrating the label information of nodes and the community information in the network, the LD strategy is defined to expand the first-order neighbor nodes of nodes. In this process, the sequence of nodes is generated by the LD strategy, then the node representation vector is generated by the Skip-Gram model. In 9 real networks, the performance comparison between the proposed algorithm and the other benchmark algorithms was displayed with the help of 5 evaluation metrics. A large number of theoretical derivation and experimental analyses demonstrated that the proposed LC-NPLA was more advantageous in the label classification task, network reconstruction task and link prediction task.

Acknowledgement: We are hugely grateful to the possible anonymous reviewers for their constructive comments with respect to the original manuscript.

Funding Statement: We are hugely grateful to the possible anonymous reviewers for their constructive comments with respect to the original manuscript. What is more, we thank the National Natural Science Foundation of China (Nos. 61966039, 62241604) and the Scientific Research Fund Project of the Education Department of Yunnan Province (No. 2023Y0565). Also, this work was supported in part by the Xingdian Talent Support Program for Young Talents (No. XDYC-QNRC-2022-0518).

Author Contributions: S.L.: Responsible for proposing algorithm ideas, analyzing experimental data as well as writing the paper. C.Y.: Responsible for proposing guidance and revising the final version of the paper. Y.L.: Responsible for collecting network data.

Availability of Data and Materials: All data generated or analyzed during this study are included in this published article.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Y. F. Zhang, S. Q. Gao, J. Pei and H. Huang, "Improving social network embedding via new second-order continuous graph neural networks," in *Proc. of the 28th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, Washington DC, USA, pp. 2515–2523, 2022.

[2] M. Aslani, M. S. Mesgari and M. Wiering, "Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events," *Transportation Research Part C: Emerging Technologies*, vol. 85, no. 85, pp. 732–752, 2017.

[3] C. Pornprasit, X. Liu, P. Kiattipadungkul, N. Kertkeidkachorn, K. Kim *et al.,* "Enhancing citation recommendation using citation network embedding," *Scientometrics*, vol. 127, no. 1, pp. 233–264, 2022.

[4] W. Nelson, M. Zitnik, B. Wang, J. Leskovec and A. Goldenberg, "To embed or not: Network embedding as a paradigm in computational biology," *Frontiers in Genetics*, vol. 10, pp. 381, 2019.

[5] I. Tiddi and S. Schlobach, "Knowledge graphs as tools for explainable machine learning: A survey," *Artificial Intelligence*, vol. 302, no. 302, pp. 103627, 2022.

[6] L. Deng, S. H. Liu and G. Duan, "Random walk and shared neighbors-based similarity for patterns in graph data," in *Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery*, vol. 88. Cham: Springer International Publishing, pp. 1297–1306, 2021.

[7] J. Z. Li, J. Zhu and B. Zhang, "Discriminative deep random walk for network classification," in *Proc. of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, pp. 1004–1013, 2016.

[8] Y. J. Liu, S. H. Liu, F. S. Yu and X. Y. Yang, "Link prediction algorithm based on the initial information contribution of nodes," *Information Sciences*, vol. 608, no. 608, pp. 1591–1616, 2022.

[9] C. Wang, S. Pan, R. Q. Hu, G. D. Long, J. Jiang *et al.,* "Attributed graph clustering: A deep attentional embedding approach," in *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence*, Macao, China, pp. 3670–3676, 2019.

[10] S. Cavallari, V. W. Zheng, H. Cai, K. C. Chang and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proc. of the 2017 ACM on Conf. on Information and Knowledge Management*, New York, NY, USA, pp. 377–386, 2017.

[11] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, no. 151, pp. 78–94, 2018.

[12] X. Du and F. Yu, "A fast algorithm for mining temporal association rules in a multi-attributed graph sequence," *Expert Systems with Applications*, vol. 192, no. 192, pp. 116390, 2022.

[13] S. H. Liu, Y. J. Liu, C. S. Yang and L. Deng, "Relative entropy of distance distribution based similarity measure of nodes in weighted graph data," *Entropy*, vol. 24, no. 8, pp. 1154, 2022.

[14] L. Tang and H. Liu, "Leveraging social media networks for classification," *Data Mining and Knowledge Discovery*, vol. 23, no. 3, pp. 447–478, 2011.

[15] B. B. Sang, H. M. Chen, L. Yang, T. R. Li and W. H. Xu, "Incremental feature selection using a conditional entropy based on fuzzy dominance neighborhood rough sets," *IEEE Transactions on Fuzzy Systems*, vol. 30, no. 6, pp. 1683–1697, 2022.

[16] B. B. Sang, H. Chen, L. Yang, J. Wan, T. Li *et al.,* "Feature selection considering multiple correlations based on soft fuzzy dominance rough sets for monotonic classification," *IEEE Transactions on Fuzzy Systems*, vol. 30, no. 12, pp. 5181–5195, 2022.

[17] B. B. Sang, H. Chen, L. Yang, J. Wan, T. Li *et al.,* "Incremental feature selection using a conditional entropy based on fuzzy dominance neighborhood rough sets," *IEEE Transactions on Fuzzy Systems*, vol. 30, no. 6, pp. 1683–1697, 2021.

[18] B. B. Sang, H. Chen, L. Yang, J. Wan, T. Li *et al.,* "Self-adaptive weighted interaction feature selection based on robust fuzzy dominance rough sets for monotonic classification," *Knowledge-Based Systems*, vol. 253, no. C, pp. 109523, 2022.

[19] D. Luo, F. Nie, H. Huang and C. H. Ding, "Cauchy graph embedding," in *Proc. of the 28th Int. Conf. on Machine Learning (ICML-11)*, Madison, M, USA, pp. 553–560, 2011.

[20] M. Ou, P. Cui, J. Pei, Z. Zhang and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, New York, NY, USA, pp. 1105–1114, 2016.
[21] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems*, Vancouver, Canada, pp. 585–591, 2001.
[22] P. Cui, X. Wang, J. Pei and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.
[23] B. Perozzi, R. Al-Rfou and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. of the 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, New York, NY, USA, pp. 701–710, 2014.
[24] T. Mikolov, K. Chen and G. Corrado, "Efficient estimation of word representations in vector space," in *Proc. of the Int. Conf. on Learning Representations (ICLR) Workshop*, Copenhagen, Denmark, pp. 1–12, 2013.
[25] J. Tang, M. Qu, M. Z. Wang, M. Wang, J. Yan *et al.,* "Line: Large-scale information network embedding," in *Proc. of the 24th Int. Conf. on World Wide Web*, New York, NY, USA, pp. 1067–1077, 2015.
[26] D. Wang, P. Cui and W. Zhu, "Structural deep network embedding," in *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, New York, NY, USA, pp. 1225–1234, 2016.
[27] L. F. R. Ribeiro, P. H. P. Saverese and D. R. Figueiredo, "Struc2vec: Learning node representations from structural identity," in *Proc. of the 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, New York, NY, USA, pp. 385–394, 2017.
[28] M. M. Keikha, M. Rahgozar and M. Asadpour, "Community aware random walk for network embedding," *Knowledge-Based Systems*, vol. 148, no. 148, pp. 47–54, 2018. https://doi.org/10.1016/j.knosys.2018.02.028
[29] D. Liu, Q. Li, Y. Ru and J. Zhang, "The network representation learning algorithm based on semi-supervised random walk," *IEEE Access*, vol. 8, pp. 222956–222965, 2020.
[30] W. Dou, W. Zhang and Z. Weng, "An attributed network representation learning method based on biased random walk," *Procedia Computer Science*, vol. 174, no. 174, pp. 291–298, 2020.
[31] X. Huang, Q. Q. Song, Y. E. Li and X. Xu, "Graph recurrent networks with attributed random walks," in *Proc. of the 25th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, New York, NY, USA, pp. 732–740, 2019.
[32] V. D. Blondel, J. L. Guillaume, R. Lambiotte and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, pp. P10008, 2008.
[33] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, Lake Tahoe, LT, USA, pp. 3111–3119, 2013.
[34] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang and C. Lin, "LIBLINEAR: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, no. 61, pp. 1871–1874, 2008.