Network representation learning guided by partial community structure

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Network Representation Learning Guided by Partial Community Structure

HANLIN SUN[1,2], WEI JIE[3], ZHONGMIN WANG[1,2], HAI WANG[4], and SUGANG MA[1,2]

[1]School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an, China (e-mail: sunhanlin@xupt.edu.cn, zmwang@xupt.edu.cn, msg@xupt.edu.cn, xingwei@xupt.edu.cn)

[2]Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing, Xi'an University of Posts and Telecommunications, Xi'an, China

[3]School of Computing and Engineering, University of West London, London, UK (email: wei.jie@uwl.ac.uk)

[4]School of Engineering and Applied Science, Aston University, Birmingham, UK (email: h.wang10@aston.ac.uk)

Corresponding author: HANLIN SUN (e-mail: sunhanlin@xupt.edu.cn).

This work was partially supported by the National Natural Science Foundation of China [No. 61702414]; the International Science and Technology Cooperation Project of Shaanxi Province, China [No.2019KW-008]; the Key Industry Innovation Chain Project of Shaanxi Province, China [No.2019ZDLGY07-08].

ABSTRACT Network Representation Learning (NRL) is an effective way to analyse large scale networks (graphs). In general, it maps network nodes, edges, subgraphs, etc. onto independent vectors in a low dimension space, thus facilitating network analysis tasks. As community structure is one of the most prominent mesoscopic structure properties of real networks, it is necessary to preserve community structure of networks during NRL. In this paper, the concept of *k-step partial community structure* is defined and two *Partial Community structure Guided Network Embedding* (PCGNE) methods, based on two popular NRL algorithms (DeepWalk and node2vec respectively), for node representation learning are proposed. The idea behind this is that it is easier and more cost-effective to find a higher quality 1-step partial community structure than a higher quality whole community structure for networks; the extracted partial community information is then used to guide random walks in DeepWalk or node2vec. As a result, the learned node representations could preserve community structure property of networks more effectively. The two proposed algorithms and six state-of-the-art NRL algorithms were examined through *multi-label classification* and *(inner community) link prediction* on eight synthesized networks: one where community structure property could be controlled, and one real world network. The results suggested that the two PCGNE methods could improve the performance of their own based algorithm significantly and were competitive for node representation learning. Especially, comparing against used baseline algorithms, PCGNE methods could capture overlapping community structure much better, and thus could achieve better performance for multi-label classification on networks that have more overlapping nodes and/or larger overlapping memberships.

INDEX TERMS network embedding, network representation learning, partial community structure, community structure, multi-label classification, link prediction.

I. INTRODUCTION

Network (graph) is a direct and natural way for data organization. Information network data is ubiquitous nowadays. Many real world systems, such as the Internet, Webs, online social networks, traffic networks and so forth, can be modeled as information networks first and then be analyzed. Traditionally, an information network (simply referred as network in following) is represented as a matrix, e.g. adjacent matrix, Laplacian matrix, similarity matrix, and so on. This way of representation has drawbacks of being high-dimension and sparse for large-scale networks. Moreover, network analyzing methods usually need iterative computing and thus are computation intensive, since network nodes are strongly correlated. These strong correlations among nodes also render troubles for the design of parallel algorithms. All in all, analysis of large-scale networks that are common today faces great challenges due to the use of matrices for network representation.

Network Representation Learning (NRL) - also referred as Network Embedding (NE) - provides a reasonable and promising way for large-scale network analyzing. The idea of NRL is to project nodes, edges, subgraphs, or even a

whole graph onto representations of some type in a low-dimension space with consideration of network structure and additional information related nodes and edges. The most widely used representation type is a densely and continuous vector, of which the dimension is much smaller than the corresponding representation in matrix. In addition, the resulted representations in low-dimension space by NRL are independent and thus ready for use as inputs of a large number of off-the-shelf machine learning algorithms when they are proper for analyzing tasks. Plenty of NRL algorithms have been proposed [1]–[6]. Among these algorithms, node representation learning that maps each node into a low-dimension vector is studied the most. From the idea how to capture network structure property, most of these methods could be categorized into five types, including *matrix factorizing model*, *probability based model*, *similarity based model*, *neural network model* and *generative adversary network model*. In the matrix factorizing model, eigen vectors of a network matrix (like a Laplacian matrix) are taken as the low-dimension representations of nodes [7]–[9]. In the probability model, random walks on a network are collected to capture co-occurance probability of node pairs within a designated context window, then node representations are learned from these walks by the Skip-Gram model, that is widely used for natural language processing [10]–[14]. LINE [15] is a typical similarity model algorithm. It directly computes the first and the second order similarities between node pairs, and uses optimizing method, like asynchronous stochastic gradient descending, to learn node representations that could preserve such similarity relationships. Qiu *et al*. [16] investigated the relationships between maxtrix factorization and several NRL algorithms, including DeepWalk [10], node2vec [11] and LINE, and showed that these algorithms could be unified into the matrix factorization framework. In the neural network model, a neural network of some type (such as the autoencoder and recurrent neural network), that could capture nonlinear relationships among nodes, is trained to learn low-dimension node representations [17], [18]. ANE [19], NetRA [20] and GraphGAN [21] are three examples of Generative Adversary Network (GAN) [22] model. They design a game-theoretical minmax game to combine the generative and discriminative thinking to learn node representations. We will also focus on node representation learning in this paper.

It is straightforward that the performance of a network analysis based on learned representations is highly dependent on whether the learned representations in low-dimension space are able to preserve structure features of the original network well. Most NRL methods focus only on local or micro topology properties, such as neighbors, two-step neighbors, and so forth. Recently, works in [23]–[28] started to explicitly consider preserving community structure in network representation learning in realizing that community is a prominent mesoscopic structure of networks and has an important effect on network analysis. Simply, a community is a group of nodes that have more connections among them, but have relative less connections with the rest of the network; thus in community semantics, members of a community are more similar. Therefore, in embedded low-dimension space, representations of nodes belonging to a community should be closer.

In this paper, we proposed two NRL algorithms that could preserve network community structure well in learned node embeddings. The basic idea is to first extract information on community structure using community detection methods, and then use the obtained information to enhance node representation learning. The idea of our algorithms is different from most current works that unify community model and node representation learning model together (more details of these works will be discussed in section II). In summary, the main contributions of this work are as follows:

(1) We defined the concepts of *k-step partial community* and *k-step partial community structure*, and then proposed two *Partial Community structure Guiding Node Embedding* (PCGNE) methods, PCGNE-DW and PCGNE-N2V that base on DeepWalk and node2vec, respectively. The two methods extract the information of a 1-*step partial community structure* for a network firstly, and then use the partial communities to guide random walks in DeepWalk or node2vec for node representation learning. Specifically, by giving next walk a prior probability to 1-step neighbors sharing at least one partial community, the random walks could be prone to being trapped within communities; therefore the community structure could be implicitly preserved in collected walks and thus in representations learned from these walks.

(2) We quantitatively showed the impact of community structure on node representations using examples of multi-label classification, thus proving the necessity of explicitly preserving network community structure in network representation learning.

(3) We conducted extensive tests for the two proposed methods and six other state-of-the-art network representation learning algorithms on synthesized and real networks. The results of experiments showed that our two algorithms could preserve the property of network community structure, especially overlapping community structure, well.

(4) We found that the use of these real world networks, including BlogCatalog, Flickr, Protein-Protein Interactions (PPI), and so on for NRL algorithm verification through multi-label classification should be cautioned, since their node labels did not properly encode their network topology, namely node labels are not consistent with their connection relationships. Such networks were widely used in previous NRL works, that learn node representations purely from network topology, for performance verification.

The rest of this paper is arranged as follows: section II introduces some node representation learning methods related to ours or considering community structure preserving. Section III quantitatively shows the impact of community structure on node representations using examples of multi-label classification. Section IV describes details of the proposed PCGNE-DW and PCGNE-N2V, whileas section V presents the results of extensive experiments on synthesized

and real networks. Finally, section VI concludes the paper.

II. RELATED WORKS

In this section, we briefly introduce DeepWalk and node2vec which we have based our methods on, and NRL algorithms that explicitly consider to preserve community structure features.

Perozzi *et al*. proposed DeepWalk [10], the first algorithm that can handle node representation learning of large scale networks. It builds upon the observation that the distribution of node pair appearance in random walks collected from a network within a fixed window is power-law, and such a distribution is considerably similar to the distribution of word co-occurances in the natural language corpus. Therefore, DeepWalk imitates word representation learning to learn node representations: treats a node as a word and a short random walk as a special sentence and then solves node representations using the Skip-Gram model. In fact, DeepWalk tries to keep neighborhood properties of nodes.

Grover *et al*. presented node2vec [11] which also learns node representations by maximizing the likelihood of preserving node neighborhoods. It designs a biased random walk by introducing two controlling parameters, *returning* p and *in-out* q that control how fast the next walk explores or leaves the neighborhood of a starting node, respectively. By setting both parameters as 1.0, where the next walk from current node is to a randomly selected neighbor, node2vec becomes DeepWalk.

Recently, a few studies have considered to preserve community features during node representation learning after noting the importance of community structure on network analysis. Wang *et al*. combined two nonnegative matrix factorizing (NMF) models that are for node representation learning and community structure detecting, and proposed M-NMF (Modularized NMF) [23]. It optimizes both models at the same time, therefore being able to maintain network community properties in final node representations. However, M-NMF adopts a modularity matrix to encode network community structure and assumes that a node can be assigned to only one community (i.e. a disjoint community structure), which is generally not true for real networks. Moreover, M-NMF needed to designate the number of communities, which is usually not known in practice and hard to estimate.

Based on the thought that communities regularize communication pathways for information propagation on networks, Zhang *et al*. proposed COSINE (COmmunity-preserving Social Network Embedding from Information diffusion cascades) [24]. Using the *Gaussian Mixture Model* (GMM) to model communities in a mapped low-dimension space, COSINE faces the same problems as M-NMF. The authors claimed that by replacing GMM with the *hierarchical mixture models with Dirichlet priors*, COSINE can overcome both the problems.

Cavallari *et al*. introduced the ComE (Community Embedding) framework [25]. From ComE's perspective, the three tasks of community detection, community embedding, i.e. attempting to learn a low-dimension representation for each community, and node embedding are closely related and should form a closed loop procedure. ComE first employs DeepWalk to create initial node representations and then updates node representations, community representations and community assignments of nodes iteratively. It takes *Multivariate Gaussian Distribution* (MGD) as the model for community representations, and supposes that node representations are generated from such community distributions. MGD representation has the strength of clearly showing distribution features of community members in low-dimension space. Though ComE supports overlapping community structure, i.e. a node can join in multiple communities, it requires the number of communities as input as well.

Tu *et al*. held a similar view and proposed a unified framework named CNRL (Community-enhanced Network Representation Learning) [26]. CNRL extends the idea of DeepWalk by modeling a community as a topic in natural language. It uses a vector in the same size as node representations as the representation for a community, and hires *Gibbs Sampling of Latent Dirichlet Allocation* to find community assignments for nodes. They developed two community enhanced node representation learning methods, CNRL-DW and CNRL-N2V that base on DeepWalk and node2vec, respectively. The problem faced by CNRL is the same as for ComE.

Jia *et al*. proposed CommunityGAN [27] to learn node representations and detect overlapping communities simultaneously. It uses the theory of GAN as well. However, a node representation by CommunityGAN indicates the membership strength of the node to communities; therefore, it requires the dimension of learned node representation must be same with the number of communities. It is better to view CommunityGAN as a community structure detecting method rather than a general node representation learning one.

Different from the way of aforementioned methods - jointly modeling community structure and node representations in a unified framework - CARE (Community Aware Random walk for network Embedding) [28], which is based on DeepWalk as well, sets up a new way for integrating community features into node representations. It firstly detects a community structure for a network using Louvain, a popular community detection method, and then uses the obtained communities to guide DeepWalk random walks. Specifically, it makes use of community information by increasing co-occurances of node pairs belonging to a same community, i.e. with a prior probability, it randomly chooses a node from communities to which the current node belongs as the next walk. However, the benefit of using such an aggressive way of integrating highly depends on whether a proper community structure could be found. Unfortunately, finding a high quality community structure for large scale networks is not easy.

In short, most preliminary researches that try to take community features into account as learning node representations assume some type model for community structure, and solve

community representations (and community detection) and node representations jointly. However, a priorly set model may not capture features of real communities well. Besides, they need the number of communities, that can not be determined easily, as input. Our methods follow the way of CARE in considering community structure, but extract community information in a less cost approach and integrate community structure to random walks implicitly.

III. IMPACT OF COMMUNITY STRUCTURE ON NRL

We quantitatively test the impact of community structure on node representation learning using CARE on networks having ground truth community structure. Specifically, we take three synthesized networks created by LFR [29] as examples. They are denoted as $\mu = 0.3$, $on = 30\%$, and $om = 6$, respectively, by their key parameter of LFR. The network $\mu = 0.3$ has no overlapping nodes. The network $on = 30\%$ has 30% of nodes as overlapping nodes with community membership 3, whileas the $om = 6$ has 20% overlapping nodes with community membership 6. Each network contains 10,000 nodes. More details about these synthesized networks can be found in section V-C1.

We adopt the *multi-label classification* (MLC) from node representations task to show the impact of explicitly considering community structure in NRL. First, node representations of the three networks are learned using CARE aided by their true community structures (denoted as CARE-RCOM). Then 70% of nodes and their labels are randomly selected as training data, and labels of the rest 30% nodes are predicted using the libsvm [30]. We assigned the community identifiers of each node as its labels. Note that overlapping nodes have multiple labels. Such a label assignment is reasonable since the labels could correctly encode the structure of the network, namely nodes with a same label have more connections among them, which is exactly what we would like to preserve in NRL. $Micro\text{-}F1$ and $Macro\text{-}F1$ are used as evaluation metrics. Refer to section V-A for more information about experiment settings and metrics. We compare the two metrics obtained by CARE with those by DeepWalk to show to what extent the improvement could be. Table 1 shows the results. As can be seen from the first two rows, if the true community structure could be known in some way, both metrics, especially of network $on = 30\%$ and $om = 6$, can be improved significantly.

The way that CARE integrates community information is aggressive, however. It adds in node representations the community semantics that nodes within a same community have more similarity, even if they are not directly connected by an edge. As a result, if the community information used is correct, this way could greatly increase the similarities of node pairs within a community; but if the information is wrong, such a way would cause huge negative impact. For example, if we replace the true community structures with those detected by OSLOM [31] in CARE (denoted as CARE-DCOM), the results do not show obvious improvements any more than those of DeepWalk, except network $om = 6$ (Table 1). The reason is that the detected community structure contains nodes wrongly assigned to some communities, and such nodes introduce too much wrong community semantics. Here we adopt OSLOM since it has been proved to be an effective algorithm for overlapping community detection and is better than Louvain used in original CARE [32]. We ran OSLOM 10 times and selected the best community structure according to the overlapping modularity score [33]. Thereby, the key problem for CARE is how to find an accurate community structure as much as possible for networks.

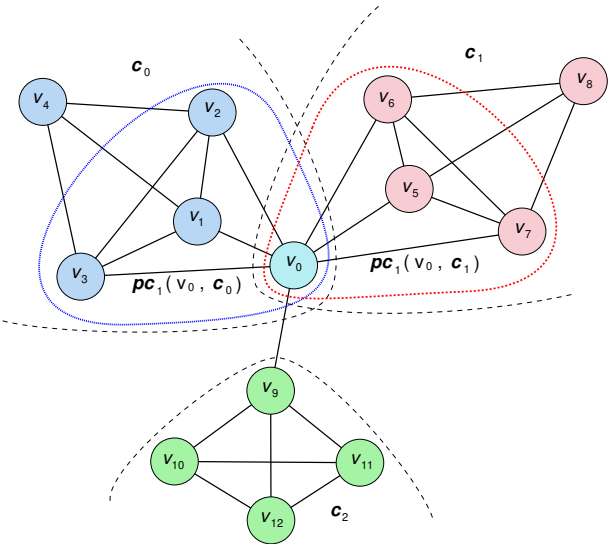


FIGURE 1. Examples of 1-step partial community. This figure shows the partial 2-step neighborhood of node v_0 in a network. Different node colors (except v_0) mean different community assignments. Node v_0 belongs to two communities, $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$. Therefore, v_0 has two 1-step partial communities, $\boldsymbol{pc}_1(v_0, \boldsymbol{c}_0)$ and $\boldsymbol{pc}_1(v_0, \boldsymbol{c}_1)$ which are encircled by blue and red dot lines and consist of v_0 and its 1-step neighbors in the two corresponding communities, i.e. $\{v_0, v_1, v_2, v_3\}$ and $\{v_0, v_5, v_6, v_7\}$, respectively.

Detecting a high quality community structure for networks, especially for large scale networks, is a challenging task. However, finding community boundaries, namely finding that if a node stays in a same community with its neighbors based on the neighborhood topology structure of the node, is a relatively easier and less cost task. Owning to this premise, we introduce the concept of k-step partial community structure.

Definition 1: Given a network $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ where $\boldsymbol{V}$ is the node set and $\boldsymbol{E}$ is the edge set; suppose $\boldsymbol{C}$ is a community structure of $\boldsymbol{G}$. For a node $v \in \boldsymbol{V}$, denote the community (communities) that v belongs to as $\boldsymbol{coms}_j(v) \subseteq \boldsymbol{C}$. A *$k$-step partial community* of v that relates to a community $\boldsymbol{c} \in \boldsymbol{coms}_j(v)$, denoted as $\boldsymbol{pc}_k(v, \boldsymbol{c})$, is a node group that contains v itself and its less than or equal to k-step neighbors that also stay in $\boldsymbol{c}$, i.e.

$$\boldsymbol{pc}_k(v, \boldsymbol{c}) = \{v \cup u | u \in \boldsymbol{ng}_k(v) \wedge u \in \boldsymbol{c}\} \qquad (1)$$

where $\boldsymbol{ng}_k(v)$ is the less than or equal to k-step neighbors of v.

TABLE 1. Impacts of Community Structure on MLC

	Micro-F1			Macro-F1		
Alg.	$\mu = 0.3$	$on = 30\%$	$om = 6$	$\mu = 0.3$	$on = 30\%$	$om = 6$
DeepWalk	0.9988	0.8014	0.6628	0.9992	0.8714	0.8072
CARE-RCOM	1.0000	0.9995	0.9770	1.0000	0.9995	0.9689
CARE-DCOM	1.0000	0.8035	0.6898	1.0000	0.8724	0.8100
PCGNE-RCOM	1.0000	0.8888	0.7379	1.0000	0.9311	0.8460

Fig. 1 demonstrates examples of the simplest 1-step partial community.

Definition 2: A *k-step partial community structure* of a network $\boldsymbol{G}$, denoted as $\boldsymbol{pcs}_k(\boldsymbol{G})$, is the collection of all k-step partial communities of its member nodes, i.e.

$$\boldsymbol{pcs}_k(\boldsymbol{G}) = \cup_{v \in \boldsymbol{V}} \cup_{\boldsymbol{c} \in \boldsymbol{coms}_j(v)} \{\boldsymbol{pc}_k(v, \boldsymbol{c})\}. \quad (2)$$

It should be noted that although a community structure of a given network is involved in the partial community definition, it is not necessary to find a complete community structure first and then to extract the related partial communities. The involved community structure here is used just to clarify which nodes in neighborhood are included in a partial community of the node. We could design an algorithm to find a partial community structure for a network directly.

A partial community structure of a network could be used as an approach for its local topology property description. In particular, a 1-step partial community structure provides community boundaries from viewpoints of nodes. Based on the concept of 1-step partial community, we propose *Partial Community Guided Network Embedding* (PCGNE) for node representation learning. It captures community structure features of a network by random walks defined as follows:

$$next\_node = \begin{cases} usual\_walk & \text{if } r < \beta \\ partial\_community\_guided\_walk & \text{else} \end{cases} \quad (3)$$

where r is a random number uniformly drawn from range $[0, 1]$ before each walk and β is a designated threshold. The $usual\_walk$ stands for taking next walk as DeepWalk or node2vec doing, whileas $partial\_community\_guided\_walk$ means randomly selecting a neighbor that shares at least one 1-step partial community with the current node as the next walk. In the following text, partial community has the same meaning as 1-step partial community, except for clear specification. By giving neighbors sharing communities a priority, that is adjusted by β, the generated walks are likely trapped within communities; therefore, a community structure of the network could be implicitly preserved. Table 1 also lists the results of PCGNE that takes DeepWalk as usual walk and uses the true community structures for partial community guided walks (denoted as PCGNE-RCOM). It can be seen that both metrics are improved greatly comparing against those of DeepWalk, though not as much as the improvements gained by CARE-RCOM.

In practice, we do not need to find an exact partial community structure for a network in PCGNE, but just group neighbors of each node into two classes: those sharing at least one partial community with the given node and those sharing none. Such a grouping reduces the cost of PCGNE further. In the following text, finding a partial community structure means grouping of neighbors for each node of a network.

TABLE 2. Notations of PCGNE

notation	explanation
$\boldsymbol{G}$	network
$\boldsymbol{V}$	nodes of $\boldsymbol{G}$
$\boldsymbol{E}$	edges of $\boldsymbol{G}$
$\boldsymbol{C}$	a community structure of $\boldsymbol{G}$
N	node number of $\boldsymbol{G}$
v	a node
$deg(v)$	degree of v
$\boldsymbol{ng}_k(v)$	k-step neighbors of v
$\boldsymbol{coms}_j(v)$	communities v joins in
$\boldsymbol{c}$	a community
$cn(v, \boldsymbol{c})$	connection number of v to $\boldsymbol{c}$
$cs(v, \boldsymbol{c})$	connection strength of v to $\boldsymbol{c}$
$cc(v, \boldsymbol{c})$	clustering coefficient of v's neighbors in $\boldsymbol{c}$
$cn_{max}(v)$	max connection number of v to joining communities
$cs_{max}(v)$	max connection strength of v to joining communities
$\boldsymbol{C}_i$	an initialized community structure of $\boldsymbol{G}$
$\boldsymbol{pc}_k(v, \boldsymbol{c})$	a k-step partial community of v related to $\boldsymbol{c}$
$\boldsymbol{pcs}_k(\boldsymbol{G})$	a k-step partial community structure of $\boldsymbol{G}$
$\boldsymbol{coms}_n(v)$	distinct joining communities of v's neighbors
$\boldsymbol{ng}_{sc}(v)$	neighbors sharing at least one 1-step partial community with v
α	community joining threshold
β	walking within community threshold
num_E	number of evolving iteration
num_P	number of post processing iteration
num_W	number of walks per node
num_N	number of negative sampling
dim	embedding dimension
len	walk length
$size$	context window size
p	return parameter for node2vec walking
q	in-out parameter for node2vec walking
$\boldsymbol{embs}$	node embeddings of $\boldsymbol{G}$

IV. PARTIAL COMMUNITY STRUCTURE GUIDED NRL

In this section, we detail the two proposed PCGNE algorithms. Roughly, they consist of two stages: 1) finding a partial community structure for a network; and 2) guiding random walks using the found partial community structure and then learning node representations from the collected walks. Notations used in PCGNE are listed in Table 2.

A. CONNECTION STRENGTH OF A NODE TO A COMMUNITY

The key of PCGNE is also to find an accurate partial community structure as much as possible. The first question arose is how we could determine whether a node belongs to a community or not? We use the concept of *connection strength*

that quantitates how strong a node v belongs to a community $\boldsymbol{c}$ as in our previous work [34]. It is defined as:

$$cs(v, \boldsymbol{c}) = \left[\frac{cn(v, \boldsymbol{c})}{deg(v)}\right]^{[1-cc(v,\boldsymbol{c})]} \tag{4}$$

where $cn(v, \boldsymbol{c})$ is the connection number that v has with community $\boldsymbol{c}$, $deg(v)$ indicates the degree of v, and $cc(v, \boldsymbol{c})$ stands for the *clustering coefficient* of v's neighbors assigned to $\boldsymbol{c}$. Note that a connection strength could be computed only for a node having at least three connections to a community, since a meaningful clustering coefficient exists under such a restriction.

B. PARTIAL COMMUNITY STRUCTURE DETECTION

The procedure of detecting a partial community structure for a network is described in Algorithm 1. Its outline is similar to our previous work of overlapping community detection [34], but is much simpler since here we only need to differentiate whether a neighbor of a given node is sharing at least one partial community with the node. Thereby, we do not need to consider mergence of communities during evolving. First, it calls the algorithm "InitCom" (Algorithm 2) to find an initial community for each node of the processed network. Then for each node, the algorithm iteratively evolves its partial communities in order to join in communities of its neighbors. The joining criteria are that: 1) if the ratio between the connection strength of the node to a candidate neighbor community and the maximum connection strength of the node exceeds a given threshold α, the node joins in this neighbor community; 2) if the connection number of the node to a candidate neighbor community is 2 and the maximum connection number of the node is not greater than 3, the node joins in this neighbor community. At the end of each iteration, the algorithm "PostProcCom" (Algorithm 3) is called to rectify wrong community joining due to the sequential order of node processing. Finally, the grouped neighbors of each node could be easily derived from the found node community assignments, according to whether they share at least one partial community with the node.

Algorithm 2 details the procedure of partial community structure initializing. It finds an initial community for each node as follows: 1) collecting joining communities of the node's neighbors and trying to find such a neighbor community that if the node joins in, it is still a k-clique; 2) if there is no such community, trying to find if the node has two uninitialized neighbors with which the node could form a 3-clique; 3) if the first two do not meet, then the node is initialized as a singleton community, i.e. a community by itself. A k-clique is a completely connected graph of k nodes; thereby, all of its members are surely belonging to one community.

As a result of the sequential node processing order in each evolving iteration, there are some nodes whose community assignments may be incorrect. To alleviate the effects of such nodes, we execute a post processing procedure, that is presented in Algorithm 3, after each iteration to rectify these wrong assignments. Its criteria are almost the opposites of community joining in Algorithm 1. Additionally, a node does not join in a community with which it has only one connection.

Algorithm 1 : DetectPartCom

Input: network $\boldsymbol{G}$, community joining threshold α, evolving iteration num_E, post-processing iteration num_P
Output: 1-step partial community structure $\boldsymbol{pcs_1}(\boldsymbol{G})$

$\boldsymbol{pcs_1}(\boldsymbol{G})$ = InitCom($\boldsymbol{G}$); /* call Algorithm 2 */
for $itr = 1$ **to** num_E **do**
  permute nodes of $\boldsymbol{G}$;
  for each v in permuted nodes **do**
    collect $\boldsymbol{coms}_n(v)$ from $\boldsymbol{pcs_1}(\boldsymbol{G})$;
    for each $\boldsymbol{c}$ in $\boldsymbol{coms}_n(v)$ **do**
      compute $cn(v, \boldsymbol{c})$;
      if $cn(v, \boldsymbol{c}) \geq 3$ **then**
        compute $cs(v, \boldsymbol{c})$;
      end if
    end for
    find $cn_{max}(v)$ and $cs_{max}(v)$
    for each $\boldsymbol{c}$ in $\boldsymbol{coms}_n(v)$ **do**
      if ($cn(v, \boldsymbol{c}) \geq 3$ **and** $cs(v, \boldsymbol{c}) \,/\, cs_{max}(v) \geq \alpha$) **or** ($cn(v, \boldsymbol{c}) == 2$ **and** $cn_{max}(v) \leq 3$) **then**
        update $\boldsymbol{pcs_1}(\boldsymbol{G})$ by adding v to $\boldsymbol{c}$;
      else
        update $\boldsymbol{pcs_1}(\boldsymbol{G})$ by removing v from $\boldsymbol{c}$ if existing;
      end if
    end for
  end for
  $\boldsymbol{pcs_1}(\boldsymbol{G})$ = PostProcCom($\boldsymbol{G}$, $\boldsymbol{pcs_1}(\boldsymbol{G})$, α, num_P); /* call Algorithm 3 */
  if $\boldsymbol{pcs_1}(\boldsymbol{G})$ does not change **then**
    break;
  end if
end for
return $\boldsymbol{pcs_1}(\boldsymbol{G})$

Due to the random node processing order in community initializing and evolving, this partial community structure detecting algorithm is unstable, namely rerunnings could result in different communities even on a same network. In PCGNE, we rerun "DetectPartCom" several times and combine all results to get a final partial community structure. As a result, if a neighbor of a node is found sharing a partial community with the node in any run, they are regarded as sharing a community.

C. PCGNE

As we get a partial community structure of a network, we could use it to guide random walks as in (3) and then learn node representations on collected walks by Skip-Gram. Algorithm 4 outlines PCGNE. We use both the random walk manners in DeepWalk and node2vec as the usual walk, and denote the two as PCGNE-DW and PCGNE-N2V, re-

Algorithm 2 : InitCom

Input: network $\boldsymbol{G}$
Output: an initialized community structure $\boldsymbol{C}_i$

1: permute nodes of $\boldsymbol{G}$
2: **for** each v in permuted nodes **do**
3: **if** v has been initialized **then**
4: continue;
5: **end if**
6: collect $\boldsymbol{coms}_n(v)$ from initialized neighbors;
7: **for** each $\boldsymbol{c}$ in $\boldsymbol{coms}_n(v)$ **do**
8: **if** $\boldsymbol{c}$ is a k-clique after v joins in **then**
9: update $\boldsymbol{C}_i$ by adding v to $\boldsymbol{c}$;
10: tag v as initialized;
11: break;
12: **end if**
13: **end for**
14: **if** v has NOT been initialized **and** v forms a 3-clique with two uninitialized neighbors **then**
15: create a new community $\boldsymbol{c}$ containing the three nodes;
16: add $\boldsymbol{c}$ to $\boldsymbol{C}_i$;
17: tag the three nodes as initialized;
18: **end if**
19: **if** v has NOT been initialized **then**
20: create a new community $\boldsymbol{c}$ containing only v;
21: add $\boldsymbol{c}$ to $\boldsymbol{C}_i$;
22: tag v as initialized;
23: **end if**
24: **end for**
25: **return** $\boldsymbol{C}_i$

Algorithm 3 : PostProcCom

Input: network $\boldsymbol{G}$, partial community structure $\boldsymbol{pcs_1}(\boldsymbol{G})$, community joining threshold α, post processing iteration num_P
Output: rectified partial community structure

1: **for** $itr = 1$ **to** num_P **do**
2: permute nodes in $\boldsymbol{G}$;
3: **for** each v in permuted nodes **do**
4: collect $\boldsymbol{coms}_j(v)$ from $\boldsymbol{pcs_1}(\boldsymbol{G})$;
5: **for** each $\boldsymbol{c}$ in $\boldsymbol{coms}_j(v)$ **do**
6: compute $cn(v, \boldsymbol{c})$;
7: **if** $cn(v, \boldsymbol{c}) \geq 3$ **then**
8: compute $cs(v, \boldsymbol{c})$;
9: **end if**
10: **end for**
11: find $cn_{max}(v)$ and $cs_{max}(v)$;
12: **for** each $\boldsymbol{c}$ in $\boldsymbol{coms}_j(v)$ **do**
13: **if** $(cn(v, \boldsymbol{c}) \geq 3$ **and** $cs(v, \boldsymbol{c})$ / $cs_{max}(v) < \alpha)$ **or** $(cn(v, \boldsymbol{c}) == 2$ **and** $cn_{max}(v) > 3)$ **or** $cn(v, \boldsymbol{c}) \leq 1$ **then**
14: update $\boldsymbol{pcs_1}(\boldsymbol{G})$ by removing v from $\boldsymbol{c}$;
15: **end if**
16: **end for**
17: **end for**
18: **if** $\boldsymbol{pcs_1}(\boldsymbol{G})$ does not change **then**
19: break;
20: **end if**
21: **end for**
22: **return** $\boldsymbol{pcs_1}(\boldsymbol{G})$

spectively. Specifically, the codes from line 2 to 23 collect partial community aware random walks. At each walk step, a next node is firstly selected according to the DeepWalk or node2vec manner (line 9); and then, it may be replaced by a random neighbor that shares at least one partial community with the current node with a prior probability $1 - \beta$ (line 11-18). The code in line 25 achieves node embedding learning.

D. COMPLEXITY ANALYSIS

After obtaining a partial community structure, the random walk collecting and then representation learning of PCGNE are similar as in DeepWalk or node2vec. The complexity of random walk has only very slightly increase, due to the partial community aware next walk reselection. Here, we focus on the complexity of partial community structure detecting.

Denote the node number of the processed network as N. The first step of detection is to find an initial community for each node. According to the finding criteria in Algorithm 2, the complexity of initializing step is very close to $O(N)$ since the numbers of neighbors and joining communities of each node are generally dramatically less than N. The second step is partial community structure evolving and post-processing. The evolving operations include, for each node, collecting joining partial communities of its 1-step neighbors, computing connection strength to each of such a partial community, and deciding joining in each one or not. Given that the number of neighbors and joining communities of each node are usually greatly less than N, the complexity of one iteration evolution is also close to $O(N)$. The operations of post-processing are much similar as evolving, and thus its complexity. As a result, considering that the evolving and post-processing iterations are usually small, the total complexity of partial community structure detecting is approximate to $O(N)$.

V. EVALUATION

In this section, we examine the performance of the two PCGNE methods and compare them against six state-of-the-art network representation learning algorithms, including DeepWalk [10], node2vec [11], LINE [15], GraRep [7], ComE [25] and CNRL [26]. Both ComE and CNRL explicitly consider to maintain community structure properties of networks. For CNRL, we do not adopt the "Statistic-based assignment" strategy, that achieves node community assignments using the Gibbs sampling method of Latent Dirichlet Allocation, due to its heavy computation. Instead, we use the "Embedding-based assignment" strategy, which hires embeddings of nodes and communities to estimate node community assignments, owning to its computing efficiency.

Algorithm 4 : PCGNE

Input: network $\boldsymbol{G}$, partial community structure $\boldsymbol{pcs_1}(\boldsymbol{G})$, threshold of walking within communities β, embedding dimension dim, walk length len, walks per node num_W, context window size $size$, negative sampling number num_N [, return p, in-out q (*for node2vec walk*)]

Output: node embeddings $\boldsymbol{embs}$

/* collect walks*/
$\boldsymbol{walks} = [\,]$;
for $itr = 1$ **to** num_W **do**
  permute nodes of $\boldsymbol{G}$;
  for each v in permuted nodes **do**
    $\boldsymbol{node\_walk} = [v]$;
    for $step = 1$ **to** len **do**
      set cur_node as the last node of $\boldsymbol{node\_walk}$;
      set $\boldsymbol{next\_node}$ as DeepWalk (or node2vec) doing;
      collect $\boldsymbol{ng}_{sc}(cur\_node)$ from $\boldsymbol{pcs_1}(\boldsymbol{G})$;
      if $\boldsymbol{ng}_{sc}(cur\_node)$ is not none **then**
        randomly extract a number r from range $[0, 1]$;
        /* partial community guided walk */
        if $r \geq \beta$ **then**
          randomly select a neighbor from $\boldsymbol{ng}_{sc}(cur\_node)$;
          update $next\_node$ as the selected neighbor;
        end if
      end if
      append $next\_node$ to $\boldsymbol{node\_walk}$;
    end for
    append $\boldsymbol{node\_walk}$ to $\boldsymbol{walks}$;
  end for
end for
/* learn embedding */
$\boldsymbol{embs}$ = SkipGram($\boldsymbol{walks}$, dim, $size$, num_N);
return $\boldsymbol{embs}$

We run all involved algorithms on LFR synthesized networks and a real network in order to get their low dimension node representations, and then conduct *multi-label classification* and *link prediction* based on obtained representations.

A. METRICS FOR MULTI-LABEL CLASSIFICATION

In multi-label classification tests, we split a processed network into two parts, training part and test part. We first train a classifier according to the training nodes and their labels, and then use the classifier to predict labels for the test nodes. The classifier hired here is libsvm [30], in which the linear kernel function is used and other parameters are set as defaults.

The metrics for evaluating multi-label classification are $Micro\text{-}F1$ and $Macro\text{-}F1$. $Micro\text{-}F1$ is computed from each label prediction instance of each node, whileas $Macro\text{-}F1$ is the averaged $F1$ scores of each label prediction. Specifically, they are defined as:

$$Micro\text{-}F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{5}$$

where

$$Precision = \frac{\sum_{l \in \boldsymbol{L}} TruePositive(l)}{\sum_{l \in \boldsymbol{L}} [TruePositive(l) + FalsePositive(l)]} \tag{6}$$

and

$$Recall = \frac{\sum_{l \in \boldsymbol{L}} TruePositive(l)}{\sum_{l \in \boldsymbol{L}} [TruePositive(l) + FalseNegative(l)]}; \tag{7}$$

$$Macro\text{-}F1 = \frac{\sum_{l \in \boldsymbol{L}} Micro\text{-}F1(l)}{|\boldsymbol{L}|}. \tag{8}$$

In above equations, $TruePositive(l)$, $FalsePositive(l)$ and $FalseNegative(l)$ are the number of true positives, false positives and false negatives of the instances predicted as label l, respectively. $\boldsymbol{L}$ is the overall label set. $Micro\text{-}F1(l)$ is the $Micro\text{-}F1$ measurement for the label l.

B. METRICS OF LINK PREDICTION

Link prediction is to estimate if a node pair should form an edge between them. We will use the words link and edge interchangeably in following. Intuitively, if two nodes have stronger relationship, e.g. having more common neighbors, they are more likely to form a new edge. In embedding space, the smaller the distance between two node vectors, the more likely the two corresponding nodes will form an edge. Here, by taking community semantics into consideration, we categorize the relationships of node pairs into four classes: 1) node pair having an edge between them and sharing at least one community; 2) node pair having an edge but belonging to different communities; 3) node pair having no edge but sharing at least one community; and 4) node pair without an edge and locating in different communities. We denote the four relationships as *e-incom*, *e-crcom*, *ne-incom*, and *ne-crcom*, respectively. Fig. 2 shows the four type relationships on a toy network. Generally, *e-incom* node pairs have the strongest relationship due to both edge connection and community semantics, whileas the *ne-crcom* ones have the weakest. The relationship strength of *e-crcom* and *ne-incom* node pairs depend on the semantics of edge connection and community.

From the view of community semantics, the likelihood of forming an edge within a community is higher than that of crossing communities. Therefore we will randomly remove a small portion of *e-incom* edges, and evaluate the prediction of these edges in link prediction tests. The community structure of networks should not be changed after such removal to guarantee that these test links are still *e-incom* edges. Thus, we remove only 5% edges within communities from tested networks, and further ensure that no more than one edge leading from a node will be removed. In experiments, the actual number of removed edges depend on the community structure of the processed network, and may be less than 5% of total edges.

The most used metric for link prediction is AUC (Area Under the receiver operating characteristic Curve). We follow

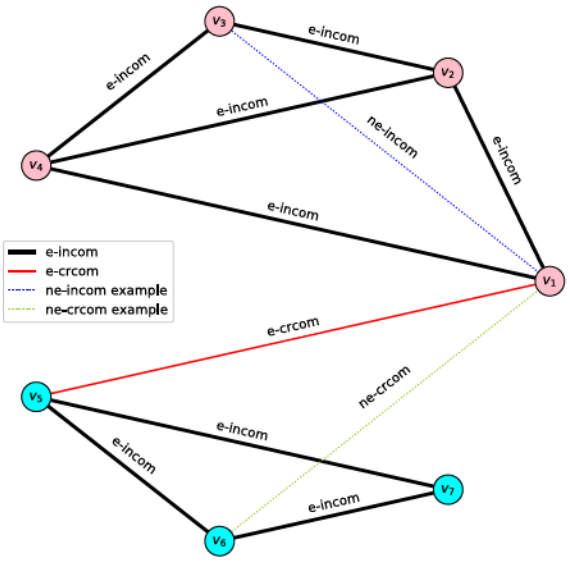


FIGURE 2. Examples of node pair relationships. The toy network has 7 nodes that form 2 communities, i.e. $\{v_1, v_2, v_3, v_4\}$ and $\{v_5, v_6, v_7\}$, denoted by different node colors, and 9 edges that are in bold line . One example of both *ne-incom* and *ne-crcom* edge that do NOT exist in the network are shown in dotted line.

the AUC definition for discrete data in [35]; thereby, a mechanism for ranking continuous node pair distances should be designed. Moreover, in order to fairly compare the ranks of distances obtained by different NRL algorithms - which have their own scales - distances should be normalized at first. The designed distance ranking mechanism is as follows:

(1) sort distances of both *e-incom* and *e-crcom* node pairs, and take the first 95% as *effective distances*;

(2) use the largest effective distance as the normalizer to normalize all involved distances, including distances of *e-incom* and *e-crcom* node pairs and distances of sampled negative node pairs (*ne-incom* and *ne-crcom* node pairs);

(3) divide the normalized effective distance range, from 0 to 1, to several equal parts (ten in this paper) and add an extra part at right end for those normalized distance that are greater than 1;

(4) assign each normalized involved distance a rank according to the range part it falls into.

In such a mechanism, a part represents the possibility that two nodes will form an edge if the distance between them falls into this part. The left most part has the most likelihood (highest rank), whileas the right most part has the least likelihood (lowest rank). To eliminate effects of extreme distances, we select the normalizer as the largest one of the first 95% effective distances. Extreme distances take only a small portion but distribute widely; therefore, they may affect the distribution of effective distances that we reference to evaluate edge forming likelihood. Then AUC can be computed as:

$$AUC = \frac{num_1 + 0.5 \times num_2}{num} \tag{9}$$

TABLE 3. LFR parameter settings of synthesized networks

Parameter	Description	Experiment Setting
N	number of nodes	10,000
k	averaged node degree	15
$maxk$	maximum node degree	50
$minc$	minimum community size	20
$maxc$	maximum community size	1000
$t1$	minus exponent for degree distribution	2
$t2$	minus exponent for community size distribution	1
μ	mixing ratio	0.3, 0.4, or 0.5 with on=0 and om=0
on	number of overlapping nodes	10%, 20%, or 30% of N with μ=0.3 and om=3
om	number of community memberships of overlapping nodes	3, 6, or 9 with μ=0.3 and on=20% of N

where num stands for the number of total observations, num_1 is the times that a predicted edge has a higher rank than a random chosen none existing edge, and num_2 is the times they have a same rank. The value of AUC ranges from 0 to 1. Usually, it falls into $[0.5, 1.0]$. The more it exceeds 0.5, the better the prediction. We follow the fast computing method of AUC in [35].

C. EVALUATION ON SYNTHESIZED NETWORKS

We generate synthesized networks using LFR model [29], which is widely used for community detection evaluation. The community structure properties of generated networks could be controlled by model parameters, therefore it allows us to systematically investigate effect of community structure on learning of node representations.

1) LFR settings

We vary three LFR model parameters that are mixing ratio μ, overlapping density on, and overlapping diversity om, to control community structure properties. The mixing ratio μ controls the average ratio of external degree of a community to its total degree. The smaller the μ, the better the quality of community structure. The overlapping density on is the number of overlapping nodes, whileas overlapping diversity om specifies the number of community memberships of overlapping nodes. The parameter settings in our experiments are shown in Table 3. We vary μ, on and om to generate networks with simple, complex, or none community structure. Note that the parameters of network $on = 20\%$ and $om = 3$ are same, so we simply use a same network for both settings in later experiments.

However, as we check the community structures of these synthesized networks using *connection number*, we find that for network with large on or om, there are a small portion of nodes breaking the property of a strong community, that have more connections within a community but relatively less connection with outside nodes. Therefore, we rectify community structures of these networks as follows:

(1) a node leaves a joining community with which it has

zero or one connection. We note that some overlapping nodes have just one connection to all their joining communities. In such a situation, it is reasonable to assign the node to each community or to none. We take the latter because one connection is a trivial structure.

(2) a node joins in a nonjoining community with which the number of connections of the node is equal to or larger than the minimum connection number of the node with its already joining communities. Such a situation occurs mainly as the connection number of new joining is 2.

Both of the leaving and joining actions are executed iteratively until no node changes its community assignments, or up to a given number of times. Leaving actions are carried out first. The original community memberships of overlapping nodes are specified by om and are same for all nodes, whereas they could be different after rectifying. The distributions of amended community memberships except one are shown in Fig. 3. As can be seen, community memberships could be spread in a wide range, thus making community structures more complicated. The blue bars stand for the om specified community membership. It should also be kept in mind that after rectification, some nodes may form singleton communities.

2) Results of Multi-Label Classification

We label each node of synthesized networks according to its community identifier(s); therefore, nodes with a same label will have more connections among them, namely node labels are consistent with network topology. Similar to previous works, we randomly sample 50% to 90% nodes and their labels as training data to train a *libsvm* [30] classifier, and then use the classifier to predict labels of the rest nodes. Additionally, for networks having overlapping nodes, we ensure that the same ratio of overlapping nodes are sampled as training nodes. We further make sure that none singleton community node is sampled. They are left as test nodes, but their labels cannot be predicted correctly.

Following parameter settings of previous works, the dimension of node representation is set as 128 for all algorithms; for those using random walks to capture network structure features, the length of walk is 40 and the walk number starting from each node is 80; and both the context window size and the negative sample number of Skip-Gram are 5. For ComE and the two CNRL algorithms, the required community number is set as the actual number of communities, excluding singleton communities. Both of the ComE trade-off parameters α and β are set as 0.1 according to the analysis in that paper. The max transition probability order of GraRep is 4. As finding a partial community structure in PCGNE, the rerunning number of "DetectPartCom" is 5. For the parameter returning p and in-out q of node2vec and algorithms based on it, as well as the community joining threshold α and walking within communities threshold β of PCGNE methods, we run the algoroithm with each candidate parameter combination three times and select the one that results the maximum $Micro\text{-}F1$.

We execute each algorithm on each network 10 times and compute average $Micro\text{-}F1$ and $Macro\text{-}F1$ scores. Fig. 4 and 5 present the average scores for networks with mixing parameter μ varying. LINE-1 denotes using only the first order similarity in LINE, whileas LINE-c means using both the first and second order similarities. Keep in mind that there are no overlapping nodes in μ networks. We first examine $Micro\text{-}F1$ and then $Macro\text{-}F1$, as can be seen:

(1) when μ is 0.3, where the community structure is clear, all algorithms are able to make good label predictions. $Micro\text{-}F1$ scores are very close to 1.0.

(2) as μ increases to 0.4, where the community structure becomes blur, all scores deteriorate a little bit. In general, GraRep is mildly better than ComE, which is slightly better than others. DeepWalk, node2vec and the four based on them are a little superior to the two LINE algorithms.

(3) as μ grows to 0.5, where no community structure is supposed to exist, GraRep and ComE are the top two again, and are appreciably better than others. The six based on random walks perform slightly better than the two LINE under this situation as well.

(4) from $Macro\text{-}F1$ scores, we can observe similar phenomenons except GraRep becomes worse as $\mu = 0.3$ and 0.4.

Fig. 6 and 7 show the average $Micro\text{-}F1$ and $Macro\text{-}F1$ scores of networks having overlapping node number on changing, respectively. It could be found that:

(1) as $on = 10\%$, where the community structure becomes more complicated but still is clear, GraRep performs best again and our two PCGNE algorithms become the second best. Both of them bring obvious improvements to their basement, DeepWalk or node2vec. ComE, that models a community as a multivariate Gaussian distribution, is the third best. However, the two CNRL methods, that are based on DeepWalk and node2vec as well and model a community as a topic in natural language processing, become the worst. They are even worse than their own based algorithm.

(2) as on becomes 20%, the ranking order of these algorithms almost remains the same, except node2vec exceeds DeepWalk and CNRL-DW precedes LINE-c.

(3) as on grows further to 30%, where there are more overlapping nodes, PCGNE-DW and PCGNE-N2V become the best two, but GraRep degrades greatly to the second worst. The ranking order of the rests remains similar.

(4) clearly LINE-1 that takes into account only the first order similarity performs better than LINE-c that considers both the first and second order similarities.

(5) from $Macro\text{-}F1$, it can be seen that our PCGNE-DW and PCGNE-N2V are the two best. Additionally, LINE-1 surpasses ComE in general, that is inverse to the result by $Micro\text{-}F1$. Other details are omitted for simplicity.

Fig. 8 and 9 display the average $Micro\text{-}F1$ and $Macro\text{-}F1$ scores for networks with overlapping membership om varying, respectively. Remember that the network $om = 3$ is the same one as $on = 20\%$. We replot the figure here for easy comparison as om changing. We can see that:

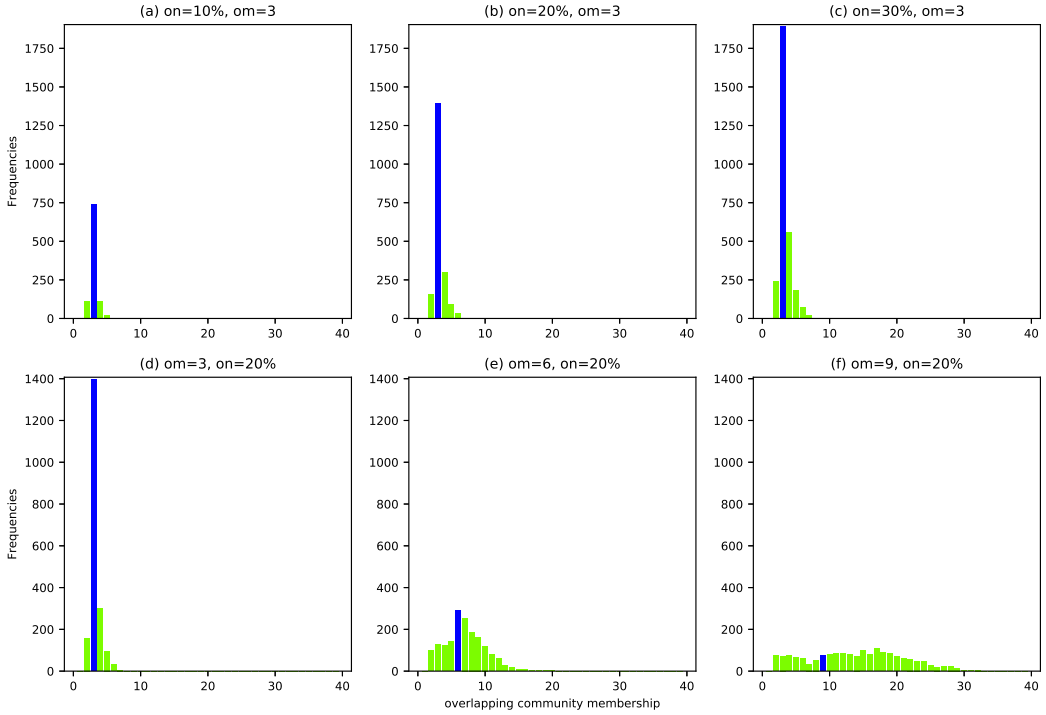


FIGURE 3. Distributions of amended overlapping community memberships of synthesized networks.

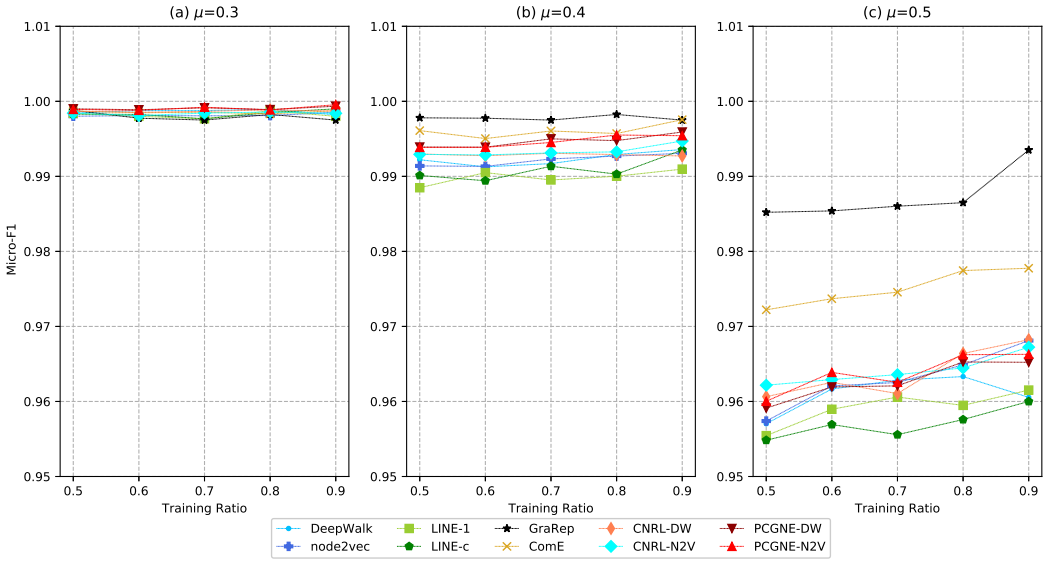


FIGURE 4. $Micro{-}F1$ scores of networks with μ changing, none overlapping structure.

(1) as om is 6, where the overlapping memberships of overlapping nodes become high, all scores deteriorate. Our PCGNE-DW and PCGNE-N2V are the best two, and node2vec becomes the second best in general. Once again, the two CNRL are the two worst in all examined algorithms.

(2) as om increases to 9, scores further deteriorate whileas the ranking order trend remains similar.

(3) for $Macro$-$F1$ scores, PCGNE-DW and PCGNE-N2V are the best two once more. In addition, as om increases, DeepWalk and node2vec perform better, whileas GraRep degrades.

We also compare the results of PCGNE-DW and PCGNE-N2V with those of CARE that uses OSLOM detected community structures as walk guidance (CARE-DCOM) for net-

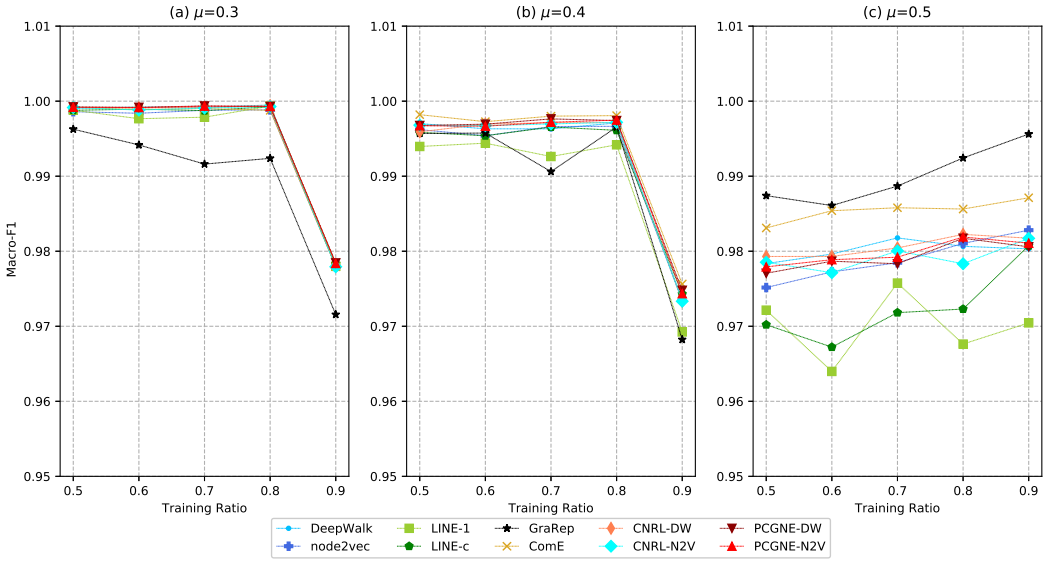


FIGURE 5. $Macro{-}F1$ scores of networks with μ changing, none overlapping structure.

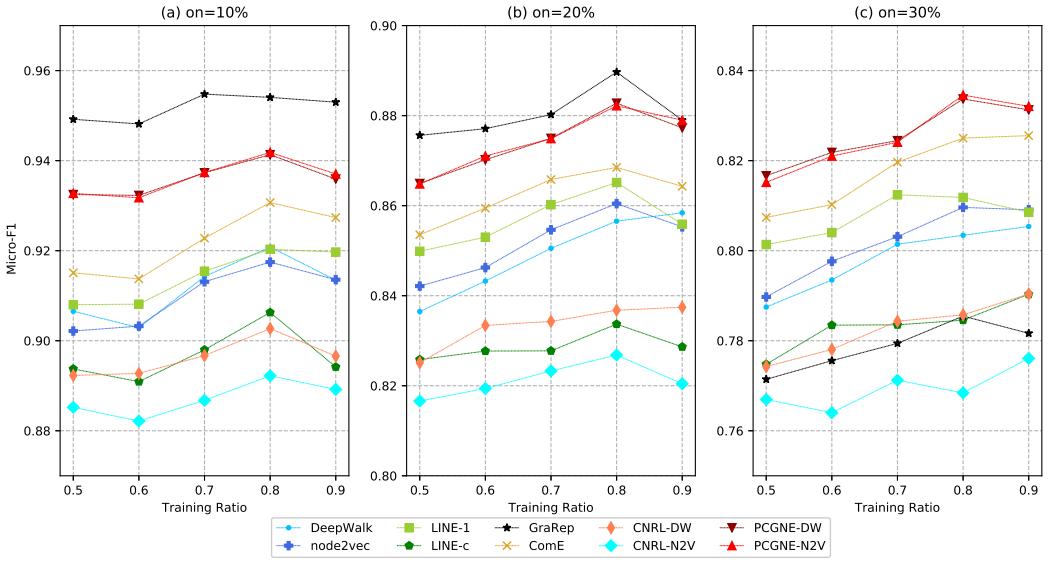


FIGURE 6. $Micro{-}F1$ scores of networks with on changing, $\mu = 0.3$, $om = 3$.

work $on = 30\%$ and $om = 6$. Table 4 shows the average $Micro$-$F1$ and $Macro$-$F1$ scores. As can be seen, our two partial community structure aided algorithms perform better. As having been explained, the reason lies in that CARE integrates community information aggressively, and thus wrong community assignments could induce heavy negative effects on NRL results.

From the aforementioned experiments, we can draw the conclusions that:

(1) both PCGNE-DW and PCGNE-N2V can greatly improve their based algorithm, DeepWalk and node2vec, respectively.

TABLE 4. $F1$ Scores of Detected Community Information Aided Algorithms

	$Micro$-$F1$		$Macro$-$F1$	
Alg.	$on = 30\%$	$om = 6$	$on = 30\%$	$om = 6$
CARE-DCOM	0.8035	0.6898	0.8724	0.8100
PCGNE-DW	0.8244	0.7043	0.9084	0.8338
PCGNE-N2V	0.8241	0.7070	0.9098	0.8331

(2) when the community structure of a network is simple and clear, even there is no community structure, GraRep and ComE could produce better embedding results for multi-label classification. However, the computing cost of GraRep is heavy due to its matrix multiplication, therefore it is not a

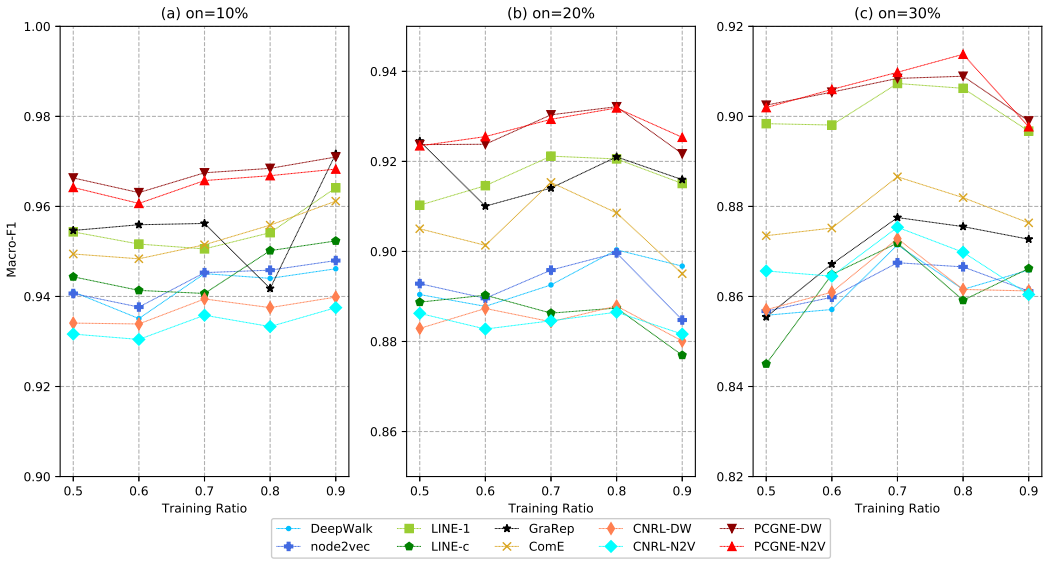


FIGURE 7. $Macro{-}F1$ scores of networks with on changing, $\mu = 0.3$, $om = 3$.

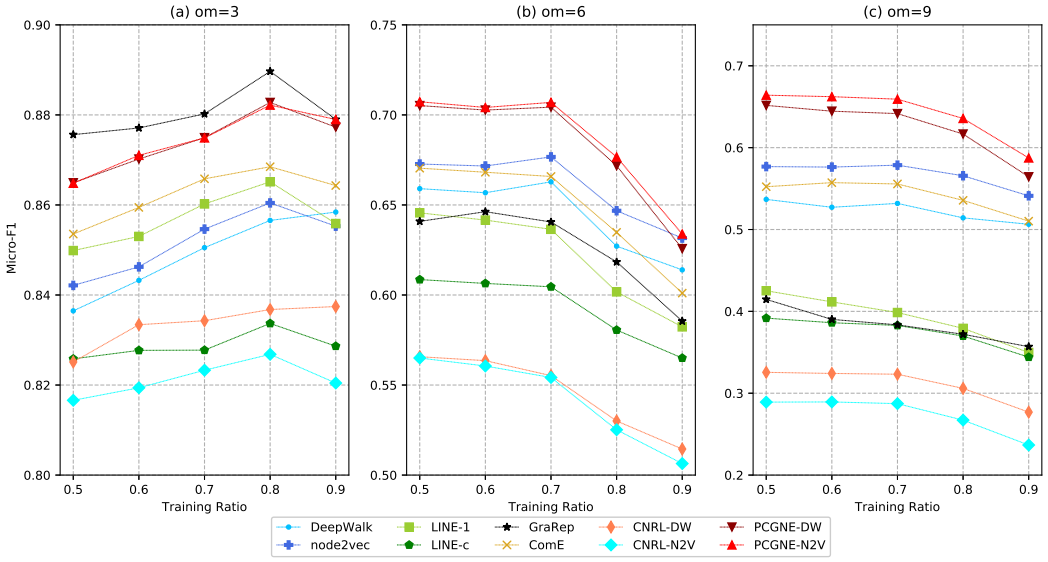


FIGURE 8. $Micro{-}F1$ scores of networks with om changing, $\mu = 0.3$, $on = 20\%$.

good choice for large scale network embedding. ComE needs the number of communities as input, but it is not easy to estimate in practice. Our two PCGNE algorithms perform quite well under such situations.

(3) as community structure becomes complicated, namely more overlapping nodes and higher overlapping memberships, our two partial community structure aided algorithms have more chance to result better embeddings that maintain community structure properties.

3) Results of Link Prediction

The manner of parameter settings of involved algorithms for link prediction are same as in multi-label classification task. We run each algorithm on each *edge removed network* 10 times and compute average AUC scores for inner community link prediction from learned node representations.

Fig. 10, 11, and 12 display these average AUC scores. At a first glance, it can be noticed that GraRep has the best scores except as $\mu = 0.3$, of which the score is still comparable. We believe this is mainly because GraRep takes 4 order node pair relationships into consideration, thus making similarities of node pairs more accurate. The cost, however, is time

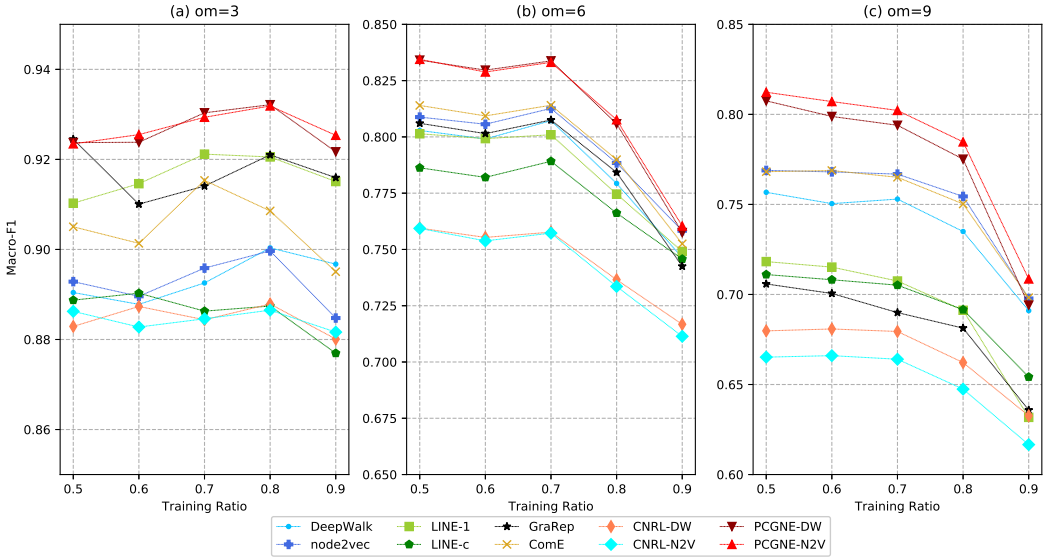


FIGURE 9. $Macro{-}F1$ scores of networks with om changing, $\mu = 0.3$, $on = 20\%$.

consuming matrix computation. For similar reasons, LINE-c that considers both first and second order similarities is superior to LINE-1 that only takes the first order similarity.

It is also easy to find that CNRL-DW and PCGNE-DW improve DeepWalk, and that CNRL-N2V and PCGNE-N2V improve node2vec significantly - even as $\mu = 0.5$ where none community structure is supposed to exist. As we check the community structure of network $\mu = 0.5$, we find that node pairs of *e-incom* edges have more common neighbors than those of *e-crcom* edges, i.e. to some extent, nodes of a community are still more densely connected comparing with nodes locating in different communities. Therefore the four DeepWalk and node2vec based algorithms that take community structure into consideration could achieve better performance.

Though both two CNRL algorithms are not good at multi-label classification, they perform slightly better than the corresponding PCGNE method in a lot of cases at link prediction, including network $on = 10\%, 20\%, 30\%$ and $om = 3, 6$ for DeepWalk based and network $\mu = 0.3, 0.5$, $on = 10\%, 20\%, 30\%$, and $om = 3, 6$ for node2vec based. This could be explained by the property of PCGNE algorithms, since they are able to capture overlapping structure well, thus making communities sharing overlapping nodes become closer in mapped space. As a result, distances of some *ne-crcom* node pairs decrease, and thus the negative samples used in AUC computation will become worse. However, our PCGNE algorithms, especially PCGNE-DW, are comparable to CNRL ones on most networks for inner community link prediction. Particularly, PCGNE-DW is the best (except GraRep) on μ networks where there are no overlapping communities.

The impacts of overlapping nodes on PCGNE methods for link prediction could be eliminated if a mirror node for each overlapping node is added into each community that the node belongs to, and connections between the (mirror) node and its neighbors in the same community are kept. As a result, with no overlapping node existing any more, our PCGNE methods will perform quite well. Under such a setting, multiple embeddings for an overlapping node, as many as the number of its community memberships, will be learned, and a proper one could be selected for link prediction. Such an improvement could benefit multi-label classification task, too. We leave this as a future task.

It can also be noted that node2vec becomes the worst one here. Theoretically, if both p and q are set as 1.0, random walks of node2vec and DeepWalk are equivalent in statistics; therefore, node2vec should not be inferior to DeepWalk if the representation learning procedure from walks are same. In experiments, we use the codes of DeepWalk and node2vec by their authors. The difference between them is the former employs the '*Hierarchical Softmax*' strategy in node representation learning, whileas the later hires the '*Negative Sampling*'. We believe the reason for node2vec being worse than DeepWalk is the adoption of different learning strategies.

As shown, although ComE is relatively good at multi-label classification, it is not at inner community link prediction.

D. EVALUATION ON A REAL NETWORK

In previous NRL works, the evaluation of algorithms by multi-label classification is mainly on real world networks of which nodes are labeled, including Citeseer [36], Protein-Protein Interactions (PPI) [1], BlogCatalog [37], Flickr [37], and so on. Taking the labels of each node as its community identifiers, we check the community structures of these networks using the index *connection number*. We find that

[1]http://snap.stanford.edu/node2vec/POS.mat

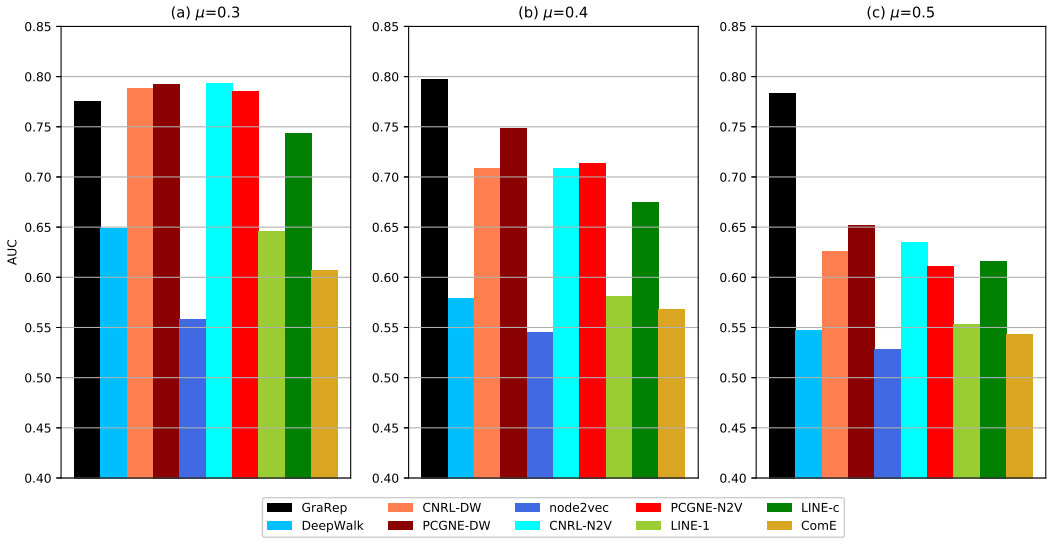


FIGURE 10. AUC scores of networks with μ changing, none overlapping structure.

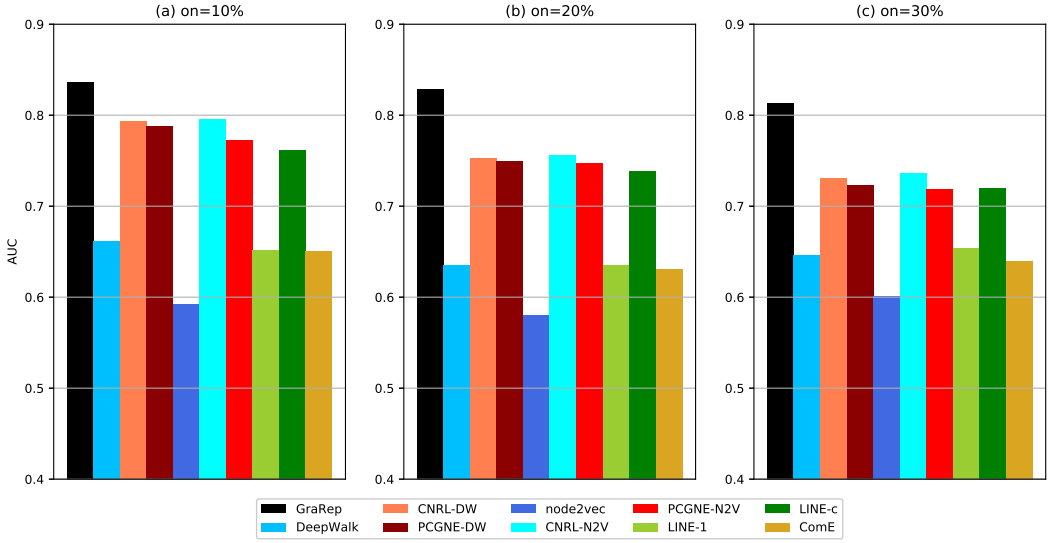


FIGURE 11. AUC scores of networks with on changing, $\mu = 0.3$ and $om = 3$.

node labels of these networks are not consistent with their topology structures. Specifically, there exists a large number of nodes that have zero connection with some of their joining communities (referred as *wrong assignments*), and an even larger number of nodes that have large enough connections to some labels, but do not join in the corresponding communities (referred as *missing assignments*). These phenomenons are reasonable because labels of such a network are assigned according to strategies that cannot assure community property. Take BlogCatelog as an example, of which each node is a user of the *BlogCatelog service* and each edge indicates friendship between two users. A label of a node stands for one topic that the user is interested in. Though two users who are friends are likely to share a common topic of interest, they do not have to. Similarly, two users may have a same interest topic even they are not friends yet. Therefore, we believe it is not appropriate to evaluate node representation learning algorithms, that solve representations merely from network topology, by multi-label classification task on such networks.

Fortunately, there is one exception, Cora[2], which is a social network of research paper citation relationships. It contains 2,708 nodes (papers) and 5,278 edges (citations). Each node has just one label that represents the class of the paper. There are 7 classes in total. Although labels are assigned to papers

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

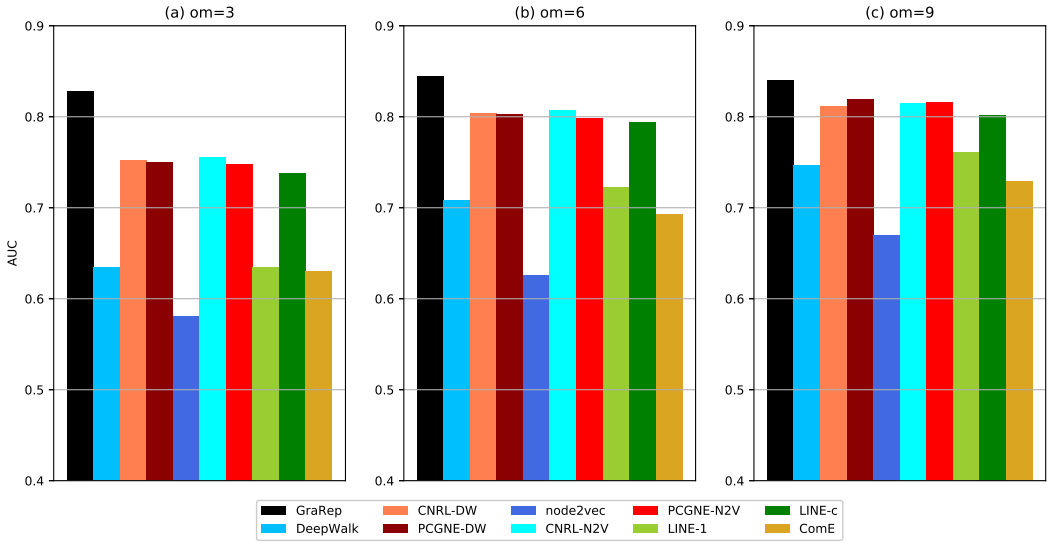


FIGURE 12. AUC scores of networks with om changing, $\mu = 0.3$ and $on = 20\%$.

based on feature words they contain, the network connection structure is quite consistent with label assignments. Taking all label assignments as the total number, the wrong assignments and missing assignments take up only 6% and 9%, respectively. Therefore, we use Cora for evaluating involved algorithms by multi-label classification and inner community link prediction in this paper.

Fig. 13 illustrates the average $Micro\text{-}F1$ and $Macro\text{-}F1$ scores of multi-label classification test. As can be seen, generally ComE is the best, whileas our two PCGNE algorithms are the second best. However, GraRep, which is good at node representation learning on synthesized networks with simple community structure, does not get better results here. The reason may lie in that the wrong and missing label assignments of Cora cause great negative effects on the predicted labels, though GraRep could obtain more accurate node pair relationships from node connection structure.

Fig. 14 displays the average AUC scores of inner community link prediction. We can find that PCGNE-DW, node2vec, PCGNE-N2V, LINE-c and ComE give out comparable best results. Surprisingly, both of node2vec and ComE that perform badly on synthesized networks, achieve quite good results on Cora. We check the connection structure of synthesized networks and Cora and find that the ratio of test edge node pairs of Cora that have at least one common neighbor is at least 20% higher than these ratios of synthesized networks, of which the highest ratio is 42%. Such common neighbors can greatly benefit algorithms based on random walks. We believe here the performance of node2vec is dominated by such many common neighbors of test edge node pairs, instead of learning strategy of Skip-Gram as on synthesized networks. In contrast, GraRep, that is good at inner community link prediction on synthesized networks, becomes the worst on Cora. This may also be due to the wrong and missing label assignments of Cora since they affect the samplings of test edges and thus the predicting results.

From all conducted experiments, we could conclude that our two PCGNE algorithms are able to capture overlapping community structure property of networks well, and generally can improve their basement, DeepWalk or node2vec, greatly for multi-label classification and inner-community link prediction tasks. At least, by setting proper parameters, they will not degrade the performance of their basement.

VI. CONCLUSION

In this paper, we first quantitatively showed the potential effect of community structure on node representation learning, then we defined the concept of *k-step partial community structure* for a network. Based on the 1-*step partial community structure*, two node representation learning algorithms, PCGNE-DW and PCGNE-N2V that are based on DeepWalk and node2vec, respectively, were proposed. The two algorithms use a found 1-step partial community structure of a network to guide random walks that could implicitly capture community structure features of the network. Therefore, node representations learned from such walks could preserve network topology properties better. Since it is easier to find a high quality partial community structure for a network than to find a high quality complete community structure, our two methods do not import too much extra computation to the based methods. Extensive experiments on eight synthesized networks and one real network were conducted. The results suggested that the two partial community structure aided algorithms could improve their based algorithm significantly, especially on networks with complicated overlapping community structure, and are competitive for node representation learning comparing with six other state-of-the-art network

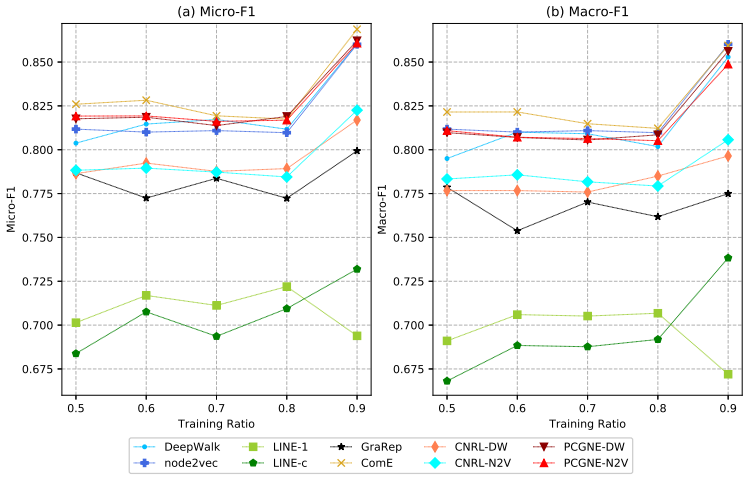


FIGURE 13. $Micro-F1$ and $Macro-F1$ scores of Cora.

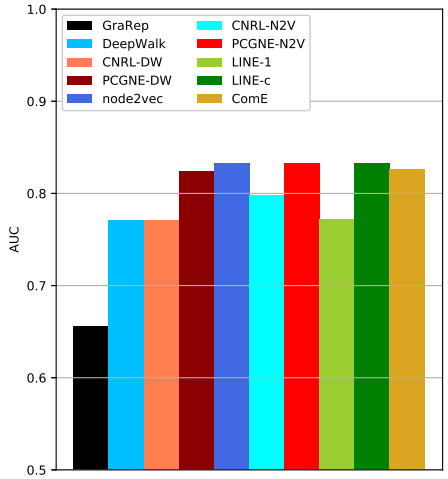


FIGURE 14. AUC scores of Cora.

embedding algorithms.

In future, we will improve our PCGNE algorithms according to the overlapping node removing strategy by adding mirrors for overlapping nodes, as described in section V-C3. We also would like to develop parallel PCGNE algorithms to further speed up large scale network representation learning, since partial community structure detection, random walk collection and representation learning by Skip-Gram could be easily parallelized.

REFERENCES

[1] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: a survey (early access)," IEEE Trans. on Big Data, Jan. 2018.

[2] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: a survey," Knowl.-Based Syst., vol. 151, pp. 78–94, Mar. 2018.

[3] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: methods and applications," IEEE Data Eng. Bull., Apr. 2018. [Online]. Available: http://arxiv.org/abs/1709.05584

[4] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: problems, techniques, and applications," IEEE Trans. Knowl. Data Eng., vol. 30, no. 9, pp. 1616–1637, Feb. 2018.

[5] J. Qi, X. Liang, Z. Li, Y. Chen, and Y. Xu, "Representation learning of large-scale complex information network: concepts, methods and challenges (in chinese)," Chinese J. Comput., vol. 41, no. 10, pp. 2394–2420, Oct. 2018.

[6] C. Tu, C. Yang, Z. Liu, and M. Sun, "Network representation learning: an overview (in chinese)," Scientia Sinica Inf., vol. 47, no. 8, pp. 980–996, Aug. 2017.

[7] S. Cao, W. Lu, and Q. Xu, "Grarep: learning graph representations with global structural information," in the 24th ACM Int'l Conf. on Information and Knowledge Management, Melbourne, Australia, Oct.19-23 2015, pp. 891–900.

[8] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, and W. Zhu, "Arbitrary-order proximity preserved network embedding," in the 24th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, London, UK, Aug.19-23 2018.

[9] X. Liu, T. Murata, K.-S. Kim, C. Kotarasu, and C. Zhuang, "A general view for network embedding as matrix factorization," in the Twelfth ACM Int'l Conf. on Web Search and Data Mining, Melbourne VIC, Australia, Feb.11-15 2019, pp. 375–383.

[10] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in the 20th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, New York, NY, USA, Aug.24-27 2014, pp. 701–710.

[11] A. Grover and J. Leskovec, "node2vec: scalable feature learning for networks," in the 22nd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, San Francisco, CA, USA, Aug.13-17 2016, pp. 855–864.

[12] S. Cui, B. Xia, T. Li, M. Wu, D. Li, Q. Li, and H. Zhang, "Simwalk: learning network latent representations with social relation similarity," in 12th Int'l Conf. on Intelligent Systems and Knowledge Engineering, Nanjing, China, Nov.24-26 2017.

[13] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, "Don't walk, skip! online learning of multi-scale network embeddings," in the 2017 IEEE/ACM Int'l Conf. on Advances in Social Networks Analysis and Mining, Sydney, Australia, Jul.31-Aug.3 2017, pp. 258–265.

[14] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in the 23th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, Halifax, NS, Canada, Aug.13-17 2017.

[15] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: large-scale information network embedding," in the 24th IInt'l Conf. on World Wide Web, Florence, Italy, May 18-22 2015, pp. 1067–1077.

[16] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec," in the Eleventh ACM Int'l Conf. on Web Search and Data Mining, Marina Del Rey, CA, USA, Feb.5-9 2018, pp. 459–467.

[17] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in the 22nd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, San Francisco, CA, USA, Aug.13-17 2016, pp. 1225–1234.
[18] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: higher-order graph neural networks," in The Thirty-Third AAAI Conf. on Artificial Intelligence, Honolulu, Hawaii, USA, Jan.27-Feb.1 2019.
[19] Y. Xiao, D. Xiao, B. Hu, and C. Shi, "Ane: network embedding via adversarial autoencoders," in 2018 IEEE Int'l Conf. on Big Data and Smart Computing, Shanghai, China, Jan.15-17 2018, pp. 66–73.
[20] W. Yu, C. Zheng, W. Cheng, C. C. Aggarwal, D. Song, B. Zong, H. Chen, and W. Wang, "Learning deep network representations with adversarially regularized autoencoders," in the 24th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, London, UK, Aug.19-23 2018, pp. 2663–2671.
[21] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: graph representation learning with generative adversarial nets," in the 32nd AAAI Conf. on Artificial Intelligence, New Orleans, Louisiana, USA, Feb.2-7 2018.
[22] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in the 27th Int'l Conf. on Neural Information Processing Systems, Montreal, Quebec, Canada, Dec.8-13 2014, pp. 2672–2680.
[23] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in the Thirty-First AAAI Conf. on Artificial Intelligence, San Francisco, CA, USA, Feb.4-9 2017, pp. 203–209.
[24] Y. Zhang, T. Lyu, and Y. Zhang, "Cosine: community-preserving social network embedding from information diffusion cascades," in the Thirty-Second AAAI Conf. on Artificial Intelligence, New Orleans, LA, USA, Feb.2-7 2018, pp. 2620–2627.
[25] S. Cavallari, V. W. Zheng, and H. Cai, "Learning community embedding with community detection and node embedding on graphs," in the 2017 ACM on Conf. on Information and Knowledge Management, Singapore, Singapore, Nov.6-10 2017, pp. 377–386.
[26] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, and L. Lin, "A unified framework for community detection and network representation learning," IEEE Trans. Knowl. Data Eng., vol. 31, no. 6, pp. 1051–1065, Jun. 2019.
[27] Y. Jia, Q. Zhang, W. Zhang, and X. Wang, "Communitygan: community detection with generative adversarial nets," in the World Wide Web Conf. 2019, San Francisco, USA, May.13-17 2019, pp. 784–794.
[28] M. M. Keikha, M. Rahgozar, and M. Asadpour, "Community aware random walk for network embedding," Knowl.-Based Syst., vol. 148, no. 15, pp. 47–54, Feb. 2018.
[29] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," Phy. Rev. E, vol. 78, p. 046110, Oct. 2008.
[30] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," ACM Trans. Intell. Syst. Technol., vol. 2, no. 3, p. 27, Apr. 2011.
[31] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, "Finding statistically significant communities in networks," PLoS One, vol. 6, no. 4, p. e18961, Apr. 2011.
[32] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: the state-of-the-art and comparative study," ACM Comput. Surv., vol. 45, no. 4, p. 43, Aug. 2013.
[33] M. Chen, K. Kuzmin, and B. K. Szymanski, "Extension of modularity density for overlapping community structure," in 2014 IEEE/ACM Int'l Conf. on Advances in Social Networks Analysis and Mining, Beijing, China, Aug.17-20 2014, pp. 856–863.
[34] H. Sun, W. Jie, L. Wang, S. Ma, G. Han, Z. Wang, and W. Xing, "A parallel self-organizing overlapping community detection algorithm based on swarm intelligence for large scale complex networks," Future Generation Computer Systems, vol. 89, pp. 265–285, Aug. 2018.
[35] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve," Radiology, vol. 143, no. 1, pp. 29–36, Apr 1982.
[36] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," Inf. Retrieval, vol. 3, no. 2, pp. 127–163, Jul. 2000.
[37] R. Zafarani and H. Liu, "Social computing data repository at ASU," 2009. [Online]. Available: http://socialcomputing.asu.edu

HANLIN SUN received his B.S. degree in computer science and technology from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2003 and his Ph.D. degree in computer science and technology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2010.

From 2010 to 2014, he was a Lecture with the the School of Computer Science and Technology, Xi'an University of Posts and Telecommunications. Since 2014, he has been an Associate Professor with the same school. From August, 2018 to July, 2019, he was a Visiting Scholar with the Department of Computer Science, Northwestern University, Illinois, USA. His research interests include computer networks, big data analytics, complex networks, and information network mining.

Dr. Sun is a member of China Computer Federation.

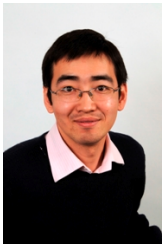
WEI JIE received his B.S. and M.S. degrees in compuer science and engineering from the Beihang University, Beijing, China, in 1993 and 1996, respectively. He received his Ph.D. degree in computer engineering from the Nanyang Technological University, Singapore, in 2000.

From 2001 to 2007, he was a Senior Research Engineer with the National High Performance Computing Research Institute of Singapore. From 2007 to 2009, he was a Senior Researcher with the National Academy of Social Sciences Information, University of Manchester, UK. Since 2009, he has been an Associate Professor in Computing with the School of Computing and Engineering, University of West London. He is the General Chair of the IEEE workshop on Security in e-Science and e-Research, and has served as Program Committee Member for more than 40 international conferences and workshops. His research interests include cloud computing, big data processing and analytics, computing security technologies, and multi-disciplinary research.

Dr. Jie is a Fellow of the Higher Education Academy, UK.

ZHONGMIN WANG received the M.S. degree in mechatronic engineering and the Ph.D. degree in mechanical manufacture and automation from the Beijing Institute of Technology, Beijing, China, in 1993 and 2000, respectively.

He was with Xidian University from 1993 to 1997. From 2004 to 2005, he was a Visiting Scholar with the Robotics Laboratory, Australian National University, Canberra, Australia. Since 2000, he has been a Full Professor with the School of Computer Science and Technology, Xi'an University of Posts and Telecommunications. His current research interests include embedded intelligent perception, intelligent information processing, machine learning, brain-computer interface, and effective computing.



HAI WANG received the B.S. degree in computer and information sciences and the Ph.D. degree in computer sciences from the National University of Singapore in 2000 and 2004, respectively.

From 2001 to 2003, he was a Research Assistant with the School of Computing, National University of Singapore. From 2006 to 2009, he was a Research Associate of Medical Informatics Group, Computer Science Department, University of Manchester. From 2006 to 2009, he was a Research Fellow of the Intelligence-Agents-Multimedia Group, Electronics and Computer Science, University of Southampton. Since 2009, he has been a Lecturer/Senior Lecturer of Computer Science, the School of Engineering and Applied Science, Aston University. His research focuses on formal software engineering and knowledge-based systems and services.

SUGANG MA received his M.S. degree from the Xidian University, Xi'an, China, in 2010. He is currently pursuing the Ph.D. degree in computer science at the Chang'an University, Xi'an, China.

He is a Senior Engineer with the School of Computer Science and Technology, Xi'an University of Posts and Telecommunications. His research interests include intelligent transportation and information system engineering.

Mr. Ma is a Senior Member of the China Communications Society.