IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 20, 2019, accepted May 14, 2019, date of publication June 4, 2019, date of current version June 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2920671

ACE: Ant Colony Based Multi-Level Network Embedding for Hierarchical Graph Representation Learning

JIANMING LV[1], (Member, IEEE), JIAJIE ZHONG[1], JINTAO LIANG[1], AND ZHENGUO YANG[2]
[1]Department of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
[2]School of Computer Science and Technology, Guangdong University of Technology, Guangzhou 510006, China

Corresponding author: Jianming Lv (jmlv@scut.edu.cn)

This work was supported in part by the NSFC under Grant 61876065, in part by the Natural Science Foundation of Guangdong Province, China, under Grant 2018A0303130022, in part by the Science and Technology Program of Guangdong Province, China, under Grant 2016A010101012, in part by the CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, under Grant CASNDST201703, and in part by the Science and Technology Program of Guangzhou, China, under Grant 201904010200.

ABSTRACT As a popularly used technique for feature learning in graphs, network embedding aims to represent each node as a low-dimensional vector to support efficient graph analytic tasks, such as node classification, link prediction, and visualization. The key to this representation method is that the embedding vector of a node should preserve its properties in the graph as much as possible. Most traditional network embedding algorithms only consider the local neighborhood as the context to build the node representation and fail to capture the important hierarchical clustering property ubiquitous in real-world graphs. To solve this problem, we propose ACE, a novel network embedding method, to preserve the features of hierarchical clustering structures. ACE works by using an ant colony-based graph coarsening algorithm to group the nodes according to their relationship to achieve a multi-level clustering pyramid of the input graph. Then, we generate the embedding vectors from multiple layers of the graph pyramid and blend these multi-level vectors into the final representation of nodes based on the PCA dimension reduction algorithm. We demonstrate the effectiveness of ACE over state-of-the-art network embedding algorithms on the node classification tasks in several real-world graph datasets. The experiments show that ACE is easy to be integrated with other network embedding algorithms, such as DeepWalk, Line, node2vec, and SDNE, to significantly improve their performance by up to 22% on Macro F1. The source code and the datasets used in this paper are available on Github (https://github.com/so-link/ACE_Embedding).

INDEX TERMS Ant colony, network embedding, multi-level.

I. INTRODUCTION

Graph is frequently used as a powerful tool to represent real-world complex information, such as social networks, biological networks, World Wide Web, etc. Analyzing these graphs can yield deep insight into the structures, patterns, latent relationship of complex systems. As a typical graph analytic task, node classification aims to predict the most possible labels of nodes. For example, in a protein-protein interaction network, the task is to predict the functional labels of proteins, and in a social network the task can be predicting the tags of users. A common architecture of the node classification application is shown in Fig 1, where each node is represented as a feature vector, and input into the classifier for final decision. The strategies to extract the feature vectors of nodes play important role in the effectiveness and efficiency of the final graph analytic task.

The most naive way to represent nodes in a graph is using the row vectors of adjacency matrix as their feature vectors. However, because many real-world graphs can contain billions of nodes and edges, this method can yield very large and sparse feature vectors, which degrade the performance of classifiers. Recently proposed network embedding algorithms [1]–[6] aim to represent the nodes as low-dimensional vectors, where the nodes with close relationship are mapped to similar embedding vectors to indicate

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

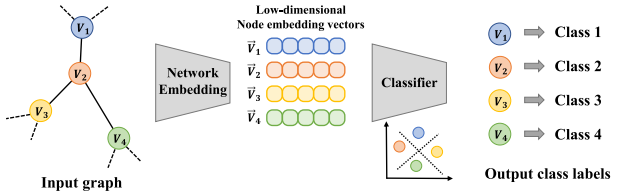


FIGURE 1. An example of node classification using network embedding approach.

related semantic. In this way, a graph can be transformed into an embedding vector space, and processed by a lot of classic vector-based machine learning techniques for rich analytic tasks, such as node classification, link prediction, visualization, etc [7].

The earliest network embedding algorithms [1], [2], which are based on matrix factorization mechanisms, can process small graphs well to achieve low-dimensional node vectors. However, these methods are not scalable to cope with large-scale graphs, because their time complexity are at least quadratic in the number of nodes. The recently proposed random walk based network embedding algorithms, such as DeepWalk [3] and node2vec [4], are more competent to solve this scalability problem. More recently, some deep learning based network embedding algorithms, such as SDNE [5] and GCN [6], are proposed to capture some deeper properties of nodes aided by the complex models with larger capacity.

However, all above embedding algorithms only consider local neighborhoods of nodes, so the generated embedding vectors can only depict the local structures immediately around a node while failing to capture the global hierarchical clustering property in a graph. Recently, a hierarchical embedding method, namely HARP [8], was proposed to capture the features of large-scale structures. As shown in Fig 2, it iteratively coarsens a graph into a series of simplified graphs by randomly merging connected nodes, and then builds the embedding vectors based on these simplified graphs recursively. However, these simplified graphs achieved by randomly merging cannot precisely depict the inner clustering structures of a graph. The nodes with weak relation may have chance to be merged, which may cause a wrong representation of clustering structures in a graph.

In this paper, we propose ACE, a novel ant colony based multi-level network embedding algorithm, which preserves the hierarchical clustering properties of graphs accurately. Specifically, an ant colony based random walk algorithm is utilized in ACE to detect the strength of the relationship between nodes in the context of loop structures in graphs. Then the nodes with strong relation are iteratively merged to form a series of simplified graphs as shown in Fig 2(b), which indicate the clustering structures in multiple granularity and constitute a multi-layer graph pyramid. Subsequently, by applying existing embedding algorithms to these coarsened graphs, the multi-level embedding vectors can be obtained and then blended into the final node representation by the PCA dimension reduction algorithm [9].

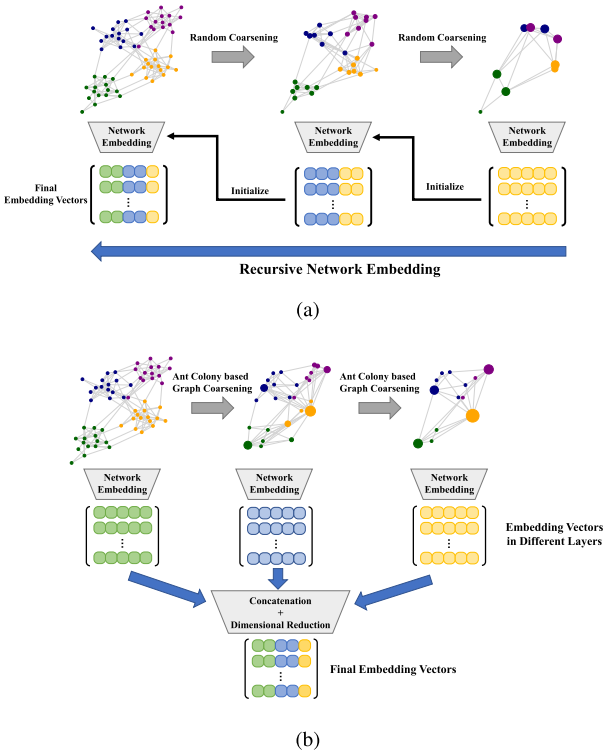


FIGURE 2. Hierarchical embedding. (a) Hierarchical embedding in HARP [8]. (b) Hierarchical embedding in ACE.

As validated in the comprehensive experiments on multiple real-world graph datasets, ACE can effectively capture hierarchical clustering structures in graphs and significantly improve the performance of embedding vectors on the node classification tasks.

Our contributions are the following:

- We propose a novel representation learning paradigm to capture multi-level clustering properties of graphs. Different from the existing network embedding algorithms which only consider local neighborhoods of nodes, ACE applies a specially designed ant colony based network coarsening algorithm to decompose a graph into a multi-level clustering pyramid to capture global clustering structures in different scales.
- We propose a user-friendly multi-level encoding scheme to significantly optimize state-of-the-art network embedding algorithms including DeepWalk [3], LINE [10], node2vec [4], and SDNE [5] without need of changing one line of their source codes. Technically, we apply the PCA dimension reduction algorithms to blend the embedding vectors generated from different layers of the clustering pyramid to encode multi-level clustering properties of graphs.
- Comprehensive experiments are conducted to test the embedding vectors in the node classification tasks on four real-world graph datasets, and the results show that our algorithm can outperform the traditional network

embedding algorithms with improvements as large as 22% on Macro F1.

II. RELATED WORK

The recently proposed network embedding methods can be generally categorized into three main categories: 1) Factorization based dimension reduction methods, like Locally Linear Embedding [1], Laplacian Eigenmaps [2], GraRep [11] and HOPE [12]; 2) Random Walk based graph embedding methods, like Deepwalk [3], LINE [10] and node2vec [4]; 3) Deep Learning based methods, like SDNE [5], GCN [6], [13], and some GAN (generative adversarial network) based embedding methods [14], [15].

Factorization based methods represent the connections between nodes with a matrix and factorize the matrix to achieve the final embedding vectors. Specifically, Locally Linear Embedding (LLE) [1] assumes every node is a linear combination of its adjacent nodes in the embedding space. The solution of LLE to achieve embedding vectors is to factorize the matrix $(I - W)^T(I - W)$, where W is the adjacency matrix of the graph, and keep the d eigenvectors corresponding to the largest d eigenvalues. Laplacian Eigenmaps [2] achieves the embedding vectors through factorizing the normalized Laplacian matrix of the graph. GraRep [11] preserves k-order proximity of the graph by factorizing $1, 2, \ldots, k$-th transition matrix and concatenate the results to get the final embeddings. HOPE [12] tries to preserve higher order proximity of the graph by factorizing a k-order similarity matrix, which can be integrated with different similarity measurements like Rooted Page Rank, Common Neighbors, etc.

With the aid of the Skip-gram model [16], which learns word embeddings from word corpus in an unsupervised manner, several random walk based network embedding methods are proposed in recent years. DeepWalk [3] first generates node sequences by applying random walk to a graph and learns the node vectors using the Skip-gram model [16] with the hierarchical softmax [16]. LINE [10] preserves both first order and second order proximities of nodes, calculates node vectors with the two objectives separately, and concatenates the representation vectors directly. Node2vec [4] obtains node sequences via the biased random walk, which can perform BFS-like or DFS-like random walking to explore structures in different types of graphs. Besides, NetMF [17] unifies DeepWalk, LINE and node2vec. It regards the three methods as factorizing different random-walk matrices.

More recently, some embedding methods integrated with deep learning schemes are proposed. SDNE [5] uses the auto-encoder to encode the embedding vector of a node based on its neighborhood, and utilizes the Laplacian Eigenmaps to force the embedding vectors of directly connected nodes to be closer than the others. In general, SDNE can preserve the first and second order proximity of a graph. GCN (Graph convolutional networks) [6], [13] defines the convolutional operator on graphs using Graph Fourier Transform in spectral domain, constructs the convolutional network by stacking graph convolutional layers like CNN, and learns the node embedding vectors in a semi-supervised manner. The research works [14] and [15] learn network embeddings with GAN (generative adversarial networks).

However, most of the above graph embedding methods only consider local adjacency relation while neglecting long distance dependencies. To solve this problem, Walklets [18] tries to learn multi-scale node representations by skipping some nodes in the random walk sequences. The recently proposed HARP algorithm [8] performs edge collapsing and star collapsing to coarsen a graph, and then prolongs the embedding vectors in a recursive manner to build multi-level representation of nodes. However, although HARP [8] solves the long dependency problem to a certain extent, the random coarsening scheme proposed in HARP cannot capture hierarchical clustering structures in graphs accurately.

III. PRELIMINARIES

A. PROBLEM DEFINITION OF NETWORK EMBEDDING

Given a graph $G = \langle V, E \rangle$, where V is the node set and E is the edge set, the goal of network embedding is to represent each node $V_i \in V$ as a $d(d \ll |V|)$ dimensional vector v_i, namely embedding vector, to indicate its semantic property. The similarity of any two embedding vectors v_i and v_j indicates the relation of these two nodes V_i and V_j. In this way, the nodes of a graph can be mapped into a feature matrix: $M_{d\times|V|}$, the j-th column of which is equal to v_j. $M_{d\times|V|}$ can be fed into a lot machine learning algorithms for the tasks of classification, clustering, prediction, etc.

Most of the existing embedding algorithms such as DeepWalk [3], LINE [10], node2vec [4], and SDNE [5] only consider the short dependency in local neighborship while ignoring global structures of graphs. To capture the structures of graphs in a larger scale, the recently proposed HARP [8] presented a hierarchical embedding solution by constructing embedding vectors based on a series of coarsened graphs, which are constructed by randomly merging the connected nodes. Although this hierarchical presentation can extract some long dependency in a coarsened graph, it cannot precisely depict the hierarchical clustering structures by random merging as shown in Fig 2. This is because not all edges indicate equally strong relation in a graph. The relation of two nodes not only depends on the direct connection between them, but also related to their local context, such as common neighbors or belonging to the same communities. Randomly merging only based on direct connections may cause some nodes with relatively weak relation to be merged preferentially. Thus, as illustrated in Fig 2, the series of coarsened graphs in HARP are far from the original hierarchical clustering structures of the graph.

B. GRAPH CLUSTERING PYRAMID

To capture the hierarchical clustering property clearly, we model a graph as a Graph Clustering Pyramid, the idea of which is similar with the concept Image Pyramid in the

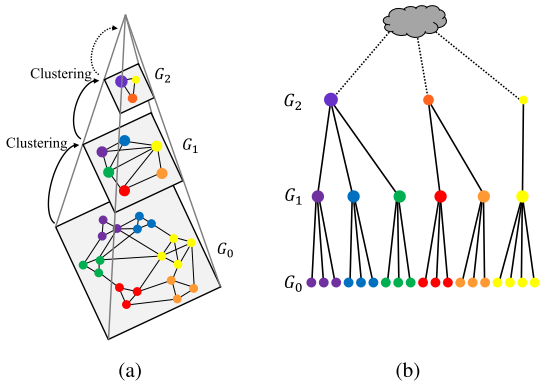


FIGURE 3. Tools to analyze hierarchical structures in a Graph: (a)Graph Clustering Pyramid and (b)Clustering tree of nodes.

image processing field. As shown in Fig 3(a), by clustering the nodes in a graph, we can achieve a series of simplified graphs in higher levels. For each pair of adjacent layers, one node in the upper layer indicates one cluster in the lower one. The Graph Clustering Pyramid can demonstrate multi-scale complex structures in a manner of multi-layer decomposition. The graphs in higher levels indicate the relation between the clusters and show global structures more clearly, while the lower layers show more details about locality. Formally, a Graph Clustering Pyramid is denoted as a series of coarsened graphs:

$$\Psi_G = \langle G_0, G_1, \ldots, G_L \rangle \tag{1}$$

where G_0 is the original graph, and $G_{l+1}(0 \leq l \leq L)$ is the clustering result of G_l.

IV. MULTI-LEVEL NETWORK EMBEDDING

Fig 4 shows the brief architecture of our proposed multi-level network embedding, which includes the following 3 key steps: (1) a novel ant colony based graph coarsening algorithm is applied to achieve a graph clustering pyramid, (2) the network embedding is built based on each level of the graph pyramid, and (3) the multi-level embedding vectors are finally encoded into low-dimensional representations based on Principal Component Analysis (PCA [9]). The main procedure of the network embedding framework is shown in Algorithm 1. Aided by the precise representation of the hierarchical clustering property of graphs, the multi-level network embedding vectors of each node can clearly model the complex relationship in multi-scale graph structures. Detail of the algorithms will be introduced in the following sections.

A. ANT COLONY BASED GRAPH COARSENING

As shown in Fig 3, the core procedure to construct a Graph Clustering Pyramid is to cluster the nodes with tight relation, and merge them in upper-layer coarsened graphs. The key problem is how to define the relation between the nodes in a graph.

The most intuitive way to measure the relation is to use the weight of the direct connection between two nodes. However,

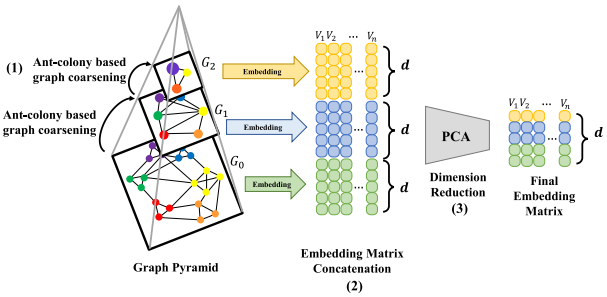


FIGURE 4. Framework of Multi-level Network Embedding.

Algorithm 1 Multi-Level Network Embedding Algorithm

Input:
Graph $G(V, E)$
Basic graph embedding algorithm *EMB*
Representation size d

Output:
Node representation vector matrix $\Phi \in \mathbb{R}^{|V| \times d}$

1: $G_0, G_1, \ldots, G_L \leftarrow ACGraphCoarsening(\mathrm{G})$
2: **for** $G_i \in \{G_0, G_1, \ldots, G_L\}$ **do**
3: $\quad \Phi_i \leftarrow EMB(G_i)$
4: **end for**
5: $\Phi' \leftarrow \{\Phi_0, \Phi_1, \ldots, \Phi_L\}$
6: $\Phi \leftarrow \mathrm{PCA}(\Phi', d)$
7: **return** Φ

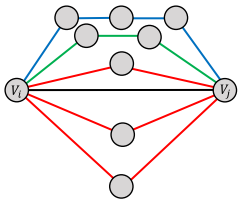


FIGURE 5. Local context of V_i and V_j.

this simple metric cannot reflect the local context of the nodes, which may play an important role in judging their closeness degree. For example, in social networks, two peers with more common neighbors may be closer to each other.

In general, the local context of two nodes (V_i and V_j) in an undirected graph can be defined as a finite set of closed loops such as $\langle V_i, V_{t_1}, V_{t_2}, \ldots, V_j, V_i \rangle$ as illustrated in Fig 5. The more loops containing V_i and V_j indicate stronger relation. Meanwhile, the smaller loops also mean closer relation.

In order to detect the tight relation constrained by loops, we propose an ant colony optimization(ACO [19]) based random walking algorithm (Algorithm 2) to detect the loop structures, as shown in Fig 6(a). Specifically, we initialize several ants to start from every node V_{init} in the graph to perform random walking. The number of ants is proportional to the degree of V_{init}. The targets of these ants are to detect loops in the graph. Once a loop is detected (Line 10 in Algorithm 2), the ant will lay down some pheromone on each edge in

Algorithm 2 Ant Colony Walking $ACWalk(G)$

Input:
Graph $G(V, E)$
Output:
Graph edge pheromone matrix ρ

1: Initialize transition probabilities
2: **repeat**
3: **for** $V_{init} \in V$ **do**
4: $d_{init} \leftarrow$ degree of V_{init}
5: **repeat**
6: $path \leftarrow \{V_{init}\}$
7: **while** length($path$) $<$ max_steps **do**
8: $V_{next} \leftarrow$ select next node according to the transition probabilities
9: append V_{next} to $path$
10: **if** $V_{next} = V_{init}$ **then**
11: **for** consecutive V_i, V_j in $path$ **do**
12: $\rho_{ij} \leftarrow \rho_{ij} + 1/\text{length}(path)$
13: **end for**
14: **break**
15: **end if**
16: **end while**
17: **until** $n * d_{init}$ ant walks
18: **end for**
19: Update the transition probabilities according to Eq (3)
20: **until** k iterations
21: **return** ρ

the loop to indicate a successful detection. The quantity of increased pheromone on each edge is inversely proportional to the length of the loop (Line 12 in Algorithm 2). In this way, the edges contained in more small loops will be assigned with more pheromone, so the quantity of pheromone can reflect the strength of relation between two given nodes.

Algorithm 3 Adaptive Threshold Selection $ATS(\rho)$

Input:
Graph edge pheromone matrix ρ
Output:
$threshold$

1: $\rho' \leftarrow \{\rho_{ij} | \rho_{ij} > 0\}$
2: $l \leftarrow \text{length}(\rho')$
3: Sort ρ' in descending order
4: Draw a line connects $(1, \rho'_1)$ and (l, ρ'_l)
5: $i \leftarrow$ find i such that (i, ρ'_i) has the maximum distance to the line
6: **return** ρ'_i

During the ant walking, the transition probability for an ant in any pair of adjacent nodes u_i to u_j is defined as $P(u_i \rightarrow u_j)$. In the beginning, $P(u_i \rightarrow u_j)$ is proportional to

Algorithm 4 Graph Clustering Pyramid Building

Input:
Graph $G(V, E)$
Output:
Graph Clustering Pyramid $\Psi_G = \langle G_0, G_1, \ldots, G_L \rangle$

1: $L \leftarrow 0$
2: $G_0 \leftarrow G$
3: **while** $|V_L|/|V| > ratio$ **AND** $|E_L|/|E| > ratio$ **do**
4: $\rho \leftarrow ACWalk(G)$ // Algorithm 2
5: $threshold \leftarrow ATS(\rho)$ //Algorithm 3
6: $G_{L+1} \leftarrow$ coarsen G_L based on $threshold$
7: $L \leftarrow L + 1$
8: **end while**
9: **return** $\Psi_G = \langle G_0, G_1, \ldots, G_L \rangle$

edge weight W_{ij}, which is

$$P(u_i \rightarrow u_j) = \frac{W_{ij}}{\sum_k W_{ik}} \tag{2}$$

Similar with the traditional ACO algorithm [19], after each iteration of the ant colony walking, we update the transition probability $P(u_i \rightarrow u_j)$ by the mixture of the edge weight W_{ij} and the amount of edge pheromone ρ_{ij} as:

$$P(u_i \rightarrow u_j) = \frac{W_{ij} \cdot \rho_{ij}^{\alpha}}{\sum_k W_{ik} \cdot \rho_{ik}^{\alpha}} \tag{3}$$

Here α is a hyper-parameter, which is used to adjust the importance of edge pheromone. Moreover, the edge pheromone does not evaporate, in each iteration we preserve the edge pheromone produced in previous iterations. By tuning the transition probability according to the amount of pheromone, ants will concentrate on the edges that are contained in more loops and indicate close relation.

After k iterations of ant-colony walking, there will be more pheromone laid down on the edges inside clusters with strong intra-relationship while less on the edges connecting clusters as shown in Fig 6(b). According to our observations shown in Fig 7(a–c), the pheromone in the edges of real-world graphs usually has a very distinct segmental distribution. There is a portion of edges which have much more pheromone than the others. This interesting result shows that in most of the real-world graph, the edges can be classified clearly into two categories: the edges which indicate strong relationship and work as intra-links in clusters, and the ones which indicate weak relationship and work as extra-links of clusters. On the other hand, Fig 7(d) shows that random graphs tend to have a much smoother pheromone distribution, because there are no distinct remarkable clusters in this kind of graphs.

While constructing the Graph Clustering Pyramid, we can cluster the nodes with strong relation into one super node in upper layer graphs. The simplest solution is to select the edges, on which the quantity of pheromone is higher than a given threshold, and merge the nodes connected with these edges. However, as shown in Fig 7, the pheromone distributions have big differences between different graphs, so we

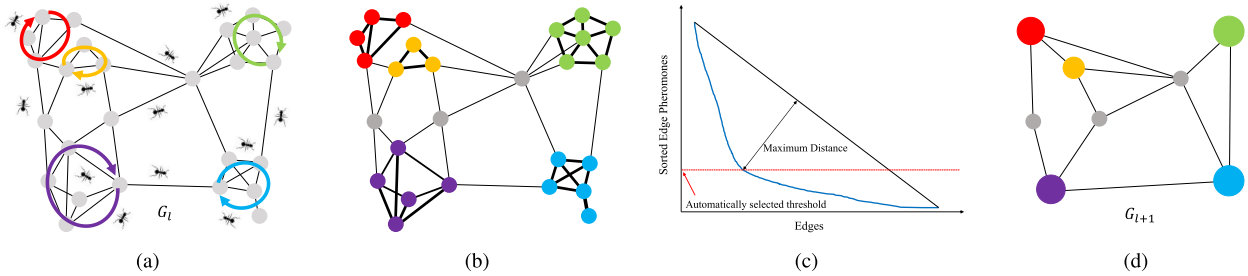


FIGURE 6. A toy example for graph coarsening visualization. (a) Ant walking in the original graph G_l. (b) Edge pheromones on the graph, where edge thickness represents the amount of pheromone on that edge. Different colors of nodes represent different components connected by the edges with large amount of pheromones. (c)Select the threshold automatically according to the sorted edge pheromones distribution.(d) The coarsened graph G_{l+1}.

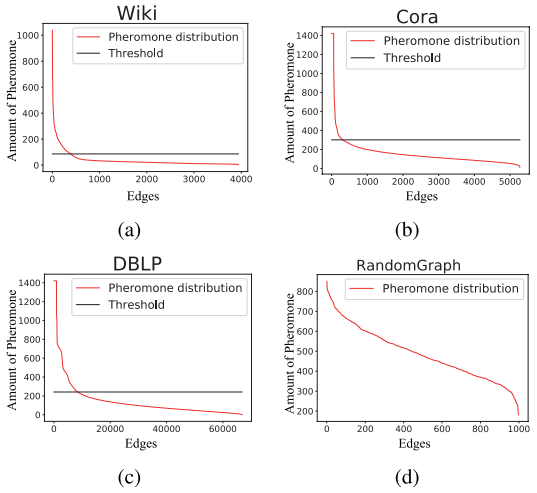


FIGURE 7. The pheromone distribution in different graph. The edges are ranked in descending order according to the pheromones on them. (a) Pheromone distribution of Wiki dataset. (b) Pheromone distribution of Cora dataset. (c) Pheromone distribution of DBLP dataset. (d) Pheromone distribution of randomly generated graph.

cannot use a unified threshold to decide which edges should be collapsed. Hence, we present an adaptive threshold selection algorithm (Algorithm 3) to solve this problem flexibly. Specifically, we sort the edges according to the pheromones on them in descending order, and calculate the distribution of the sorted pheromones as shown in Fig 6(c). The distribution curve usually has an obvious *elbow point*, which can be used as the threshold point. To achieve this point precisely, we can draw a line between the left most point and the right most point of the curve, and use the point with the maximum distance to this line as the threshold point of this curve (Line 5 in Algorithm 3). Edges that have pheromones larger than this threshold will be collapsed.

After selecting the edges to be collapsed on a given graph G_l, we coarsen every component connected by these edges into one super node to achieve a simplified graph G_{l+1}, as shown in Fig 6(d). The edge weights of G_{l+1} are set as the sum of edge weights between the components of G_l. G_{l+1} shows a higher-level coarsened view of the original graph G_l, and exhibits global clustering structures in a larger scale. According to Algorithm 4, after iteratively coarsening the graph, we can achieve a Graph Clustering Pyramid $\Psi_G = \langle G_0, G_1, \ldots, G_L \rangle$, where G_0 is the original graph and G_L is the highest-level abstraction of the graph.

B. ANALYSIS OF GRAPH PYRAMID CONSTRUCTION METHODS

In fact, the proposed ACO based graph coarsening algorithm is not the only way to achieve the multi-level structures of graphs. In this section, we will compare our method with other options theoretically, and show further experimental results in the following sections.

The random graph coarsening method, which is adopted in the recently proposed hierarchical network embedding algorithm HARP [8], is one of the simplest way to obtain the hierarchical structures of graphs. Specifically, HARP iteratively coarsens a graph into a series of simplified graphs by randomly merging connected nodes. The limitation of this method is that the randomly coarsened graphs cannot depict the inner clustering structures of a graph. The nodes belonging to a same super-node in the upper layer graph is not necessary to have strong relation to each other.

Another popularly used way to reveal the hierarchical structures of graphs is community detection [20]–[22]. The goal of the community detection algorithms is to discover the densely connected groups, which have more internal links than external links in a graph. By merging the nodes belonging to a same community, hierarchical clustering structures of the graph can be achieved. However, although the nodes in a detected community are more connected than the ones outside, it is not guaranteed that the relationship between each pair of the nodes in a community has equivalent strength. This is because of the group based measurement of community detection, which does not have strict constrain on the strength of every pair of nodes in a group.

In our multi-level network embedding scheme presented in the following section IV-C, the nodes merged into the same supernode will be assigned with the same embedding vector in the upper layer. Thus, it may be a better solution to only merge the nodes with strong relation. Based on above analysis, we propose the ACO based random walk algorithm, which detects the strength of the relation between each pair of adjacent nodes in the loop based context and merges the nodes with strong relation. This edge based procedure may be more precise to merge the close nodes than the group based com-

munity detection methods [20]–[22]. Comprehensive experiments are conducted to further show the comparison results of above graph coarsening methods in the real-world datasets.

C. MULTI-LEVEL EMBEDDING ENCODING

After we get the Graph Clustering Pyramid $\Psi_G = \langle G_0, G_1, \ldots, G_L \rangle$, we can also achieve a corresponding clustering tree of nodes as shown in Fig 3(b). A node in upper layers indicates a cluster of its descendent nodes in lower layers. For each node $V_j^{(0)}$ in G_0, its ancestor in the upper layer graph $G_x (0 < x \leq L)$ is denoted as $V_{P_j}^{(x)}$.

As shown in Fig 4, based on the Graph Pyramid $\Psi_G = \langle G_0, G_1, \ldots, G_L \rangle$, we can apply any basic graph embedding algorithm (such as DeepWalk [3], LINE [10], node2vec [4], SDNE [5], etc.) to each graph $G_l = \langle V^{(l)}, E^{(l)} \rangle (0 \leq l \leq L)$ to achieve an embedding vector $v_j^{(l)}$ for each node $V_j^{(l)} \in V^{(l)}$. For any given node $V_i^{(0)}$ in original graph G_0, its ancestor in G_l is $V_{P_i}^{(l)}$, and its corresponding embedding vector in this layer is $v_{P_i}^{(l)}$, which is called ***l-layer embedding vector*** of $V_i^{(0)}$. By collecting the l-layer embedding vectors of all nodes in G_0, we can achieve a ***l-layer embedding matrix*** $M_{d \times |V_0|}^{(l)}$, where the i-th column of the matrix indicates the l-layer embedding vector of $V_i^{(0)}$.

After achieving the multi-layer embedding matrixes $\{M_{d \times |V_0|}^{(l)} | 0 \leq l \leq L\}$, we can concatenate these matrices as shown in the step (2) of Fig 4 to get a mixed embedding matrix $M_{dL \times |V_0|}$. Each column of this matrix indicates a dL dimensional embedding vector of a node. To keep the length of the final embedding vector as d, we can apply some dimension reduction methods such as PCA [9] to further reduce the dimension of the embedding vectors to d as shown in step (3) of Fig 4.

The combination of ACE with other graph embedding algorithms is in a quite loose-coupled manner here. As shown in Algorithm 1, the basic graph embedding algorithms are invoked based on their common API (line 3 of Algorithm 1), which takes a graph as input and output embedding vectors. Thus, ACE can be integrated with other graph embedding algorithms without need of modifying their original implementation. This feature makes ACE much easier to extend basic embedding algorithms than HARP [8], which needs to modify the API of original embedding algorithms into a recursive mode.

V. EXPERIMENTS

In this section, we describe the configurations of experiments and evaluate the effectiveness of ACE with the clustering property test and node classification tasks in four real-world graph datasets. A detailed analysis of the parameter sensitivity is also provided to validate the stability of the model.

A. METRICS

To test the performance of the network embedding algorithms, we utilize the inverse version of NCut Score [23] to measure the clustering property of resulting embedding vectors. Given a graph $G = \langle V, E \rangle$, the nodes of which can be grouped into a set of clusters $\langle V_1, \ldots, V_k \rangle$ according to the class labels of nodes, where the nodes belonging to the same cluster $V_i (1 \leq i \leq k)$ have the same label. The ***Inverse NCut Score*** of a network embedding algorithm executed on this graph can be calculated as:

$$\text{iNCut}(V_1, \ldots, V_k) = \sum_{i}^{k} \frac{L(V_i, V_i)}{L(V_i, V - V_i)}, \tag{4}$$

where $L(V_a, V_b)$ defines the closeness of two node sets V_a and V_b as:

$$L(V_a, V_b) = \sum_{v_i \in V_a, v_j \in V_b} \sigma\left(\frac{\vec{v}_i \cdot \vec{v}_j}{|\vec{v}_i||\vec{v}_j|}\right). \tag{5}$$

Here $\vec{v}_i$ and $\vec{v}_j$ are the embedding vectors of node v_i and v_j respectively, and $\sigma(\cdot)$ is the sigmoid function. $L(V_i, V_i)$ measures the cohesion of nodes in one cluster, and $L(V_i, V - V_i)$ measures the coupling of nodes in different clusters. Larger Inverse NCut Score indicates better network embedding, where the embedding vectors of the nodes with same class labels are much closer than the ones with different labels.

Furthermore, we also test the performance of the embedding vectors on real node classification tasks, where the embedding vector of each node is fed into a classifier to predict its labels. A portion of nodes along with their labels are randomly sampled from the graph as training data, and a one-vs-rest logistic regression classifier is trained to predict the labels of the other nodes. The ***Macro-F1*** and ***Micro-F1*** score are utilized to measure the classification results as:

$$Macro - F1 = \frac{1}{|C|} \sum_{c \in C} \frac{2P_c R_c}{P_c + R_c} \tag{6}$$

$$Micro - F1 = \frac{1}{|V|} \sum_{v \in V} \frac{2P_v R_v}{P_v + R_v} \tag{7}$$

Here C is the set of class labels. P_c and R_c represent the precision and recall of the node classification result of class c. V is the set of nodes. P_v and R_v represent the precision and recall of the classification result of node v. Note that P_v and R_v are both equal to the number of correctly predicted labels divided by the number of total labels of node v, so the Micro-F1 score here is equal to the average precision to predict class labels of each node.

TABLE 1. Dataset information.

Dataset	BlogCatalog	Wiki	Cora	DBLP
#Nodes	10312	2363	2708	27199
#Edges	333983	11596	5278	66832
#Classes	39	17	7	4

B. DATASETS

The experiments are conducted on four real-world graph datasets, the basic information of which is shown in Table 1. Specifically, Cora and DBLP are paper citation networks. The Cora dataset is collected by [24] from the Cora website, and the DBLP dataset is built by [18] from the DBLP

dump. The nodes represent papers and the edges indicate citations between papers. The labels of a node represent the research areas of the paper related to the node. Wiki is a web page dataset build by [25], where the nodes are web pages of Wikipedia entries and edges are hyperlinks between the entries. The labels of a node represent the categories of an entry. The BlogCatalog dataset is a social network dataset built by [26] from the BlogCatalog website. The nodes of the BlogCatalog network are users of the BlogCatalog website, the edges are the social relation of users, and the labels of a node represent the categories of the blogs published by a user. These four network datasets are widely used in the node classification tasks and available for public access.

C. COMPARED METHODS

At first, we compare the performance of ACE with the following network embedding algorithms:

- **DeepWalk** [3]: DeepWalk applies random walk on a graph to achieve node sequences, which are used to learn the node vectors aided by the Skip-gram model.
- **LINE** [10]: LINE builds the embedding vectors of each node by considering both first and second order proximities of the node. Skip-gram with negative sampling is adopted to improve the efficiency of the algorithm.
- **node2vec** [4]: node2vec obtains node sequences via the BFS-like or DFS-like random walking, which is able to effectively explore the structures in different types of graphs, and then applies the Skip-gram model to train embedding vectors.
- **SDNE** [5]: SDNE uses the auto-encoder to encode the embedding vector of a node based on its neighborhood, and utilizes the Laplacian Eigenmaps to force the embedding vectors of directly connected nodes to be closer than the others. In general, SDNE can preserve the first and second order proximity of a graph.
- **HARP** [8]: HARP iteratively coarsens a graph into a series of simplified graphs by randomly merging connected nodes, and then builds the embedding vectors based on these simplified graphs in a recursive manner.
- **ACE**: The method proposed in this paper.

The common ground of HARP and ACE is that they both require a basic embedding algorithm to build embedding vectors on each coarsened graph. To facilitate the comparison, we use a compound name to indicate a combination of algorithms, e.g. ACE(DeepWalk) means the ACE algorithm using DeepWalk as the basic embedding algorithm, while HARP(LINE) means the HARP algorithm using LINE as the basic embedding algorithm.

Furthermore, There are Two Key Steps in ACE: **graph coarsening** (section IV-A) and **multi-level embedding building** (section IV-C). In order to validate the effectiveness of each key step, we also test some variation models by replacing these steps with other methods. Specifically, for the graph coarsening step, the candidate methods tested in the experiment are:

- **ACO:** Ant-colony based graph coarsening method proposed in this paper (section IV-A).
- **Random:** The graph coarsening method proposed in HARP [8], which utilizes edge collapsing and star collapsing in random order.
- **Community:** The community detection based graph coarsening method, which recursively applies the Label Propagation Algorithm (LPA) [20], [27] on a graph and merges the nodes in one community.

Meanwhile, for the multi-level embedding building step, we test the following methods:

- **PCA**: The method based on PCA proposed in this paper (section IV-C). We run a graph embedding algorithm on each level of the graphs separately to get multi-level embedding vectors, and then blend them using the PCA dimension reduction algorithm.
- **Recursive** [8]: The recursive embedding method adopted in HARP [8]. It first runs graph embedding algorithm on the upper level graph, then uses the resulting vectors as the initial values of the embedding vectors of the lower level graph to run the graph embedding algorithm. In this way, it prolongs the embedding vectors from top to bottom of the hierarchy to obtain the final embedding vectors.

Each combination of the above procedures is denoted as a compound name. For example, '*DeepWalk* + *Random* + *Recursive*' means using DeepWalk as the basic embedding algorithm, constructing the graph pyramid using the random graph coarsening, and finally building the embedding vectors using the recursive embedding method. 'node2vec + ACO + PCA' means that using node2vec as the basic embedding algorithm, constructing the graph pyramid using the ant-colony based coarsening algorithm, and finally building the embedding vectors by using PCA.

D. PARAMETER SETTINGS

The parameter settings of the models are given here:

1) DEEPWALK

For DeepWalk, the window size s, the walk length l and the number of walks per node η are set to 10, 80 and 10 respectively.

2) LINE

For LINE, the number of negative samples is set to 5.

3) NODE2VEC

For node2vec, we follow the settings in the original paper [4]. On Wiki, Cora, and DBLP, p and q are set to 4 and 0.5, while on BlogCatalog p and q are both set to 0.25. Moreover, the window size $s = 10$, the walk length $l = 80$ and the number of walks per node $\eta = 10$.

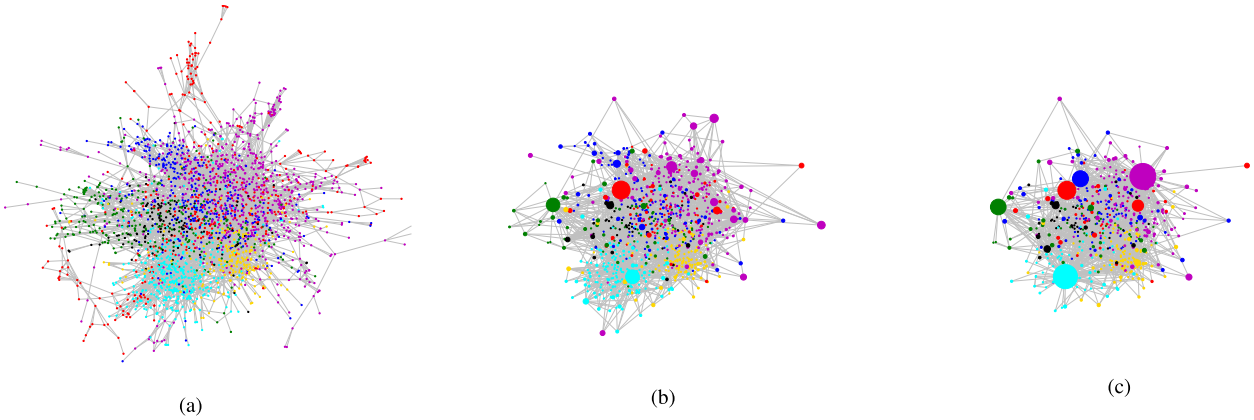


FIGURE 8. Visualization of graphs generated by ant-colony based coarsening algorithm on the Cora dataset. The size of a super-node is proportional to the number of nodes it contains. The color of a node indicates the label of the node. (a) Original graph. (b) Graph of layer #8. (c) Graph of layer #16.

TABLE 2. The Inverse NCut Scores of the embedding vectors.

	BlogCatalog	Wiki	Cora	DBLP
DeepWalk	0.838	0.765	0.963	1.008
HARP(DeepWalk)	0.846	0.791	0.997	1.021
ACE(DeepWalk)	**0.864**	**0.836**	**1.126**	**1.042**
LINE	0.841	0.776	0.961	0.960
HARP(LINE)	0.824	0.758	0.929	0.953
ACE(LINE)	**0.862**	**0.785**	**1.077**	**0.986**
node2vec	0.835	0.764	0.958	1.008
HARP(node2vec)	0.841	0.788	0.994	1.021
ACE(node2vec)	**0.864**	**0.832**	**1.126**	**1.043**
SDNE	0.820	0.668	0.706	0.915
HARP(SDNE)	-	-	-	-
ACE(SDNE)	**0.836**	**0.713**	**0.887**	**0.929**

4) SDNE

For SDNE, we use one 512-dimensional hidden layer for the encoder and the decoder separately. The dimension of the embedding layer of the auto-encoder is identical to the dimension of output embedding vectors.

5) ACE

For the ant-colony based graph coarsening, the number of ant walking iterations k is set to 2 and the power of pheromone α is set to 0.5 on the Wiki, Cora, and DBLP datasets. Meanwhile, α is set to 1.5 to achieve better performance in the BlogCatalog dataset, which has more complex hierarchical structures than the others. The sensitivity analysis of k and α is given in the following sections.

Moreover, for all of the models, the dimension of output embedding vectors d is set to 128.

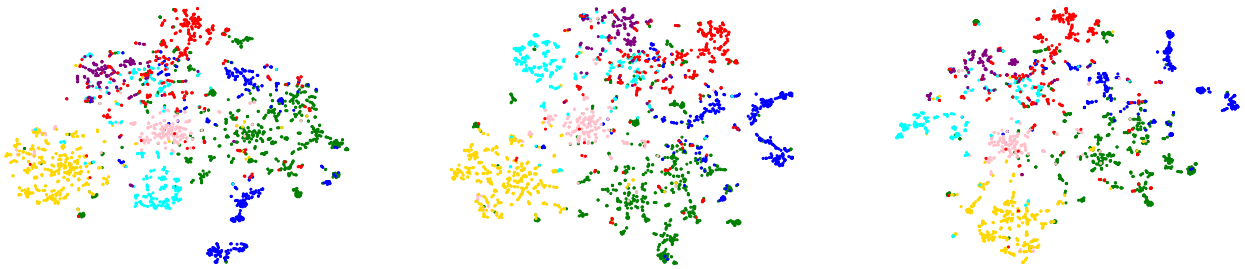
E. ANALYSIS OF CLUSTERING PROPERTY

As reported in Table 2, we analyze the clustering property of resulting embedding vectors by testing the Inverse NCut Score of all compared methods. It is surprising that the scores of the extension model HARP [8] are even lower than the original embedding algorithms in some cases (highlighted with underlines). This may be caused by the incorrect hierarchical representation based on the random coarsening, which may merge the nodes with weak relation. Such imprecise hierarchical representation can have a negative impact on the clustering property of the embedding vectors. Meanwhile, ACE achieves the highest scores in all test cases. The multi-level optimization of ACE can provide obvious improvement comparing to the original basic embedding algorithm such as DeepWalk, LINE and node2vec. This again confirms the superiority of the ant-colony based graph coarsening comparing to the random merging. Fig 8 visualizes an example of the ant-colony based graph coarsening procedure by showing the generated coarsened graphs in different layers of the graph pyramid. It shows that the green and red nodes with strong relation are merged first, and the purple and blue nodes are merged gradually. We can see reasonable clusters in the final graph, in which the purple, green and cyan nodes are clustered correctly. This enables better clustering properties of the generated embedding vectors.

To further illustrate the clustering properties intuitively, we utilize the t-SNE [28] algorithm to visualize the generated embedding vectors in Fig 9, which takes the Cora dataset as an example. The points in the same color represent the nodes with the same class label. Fig 9 shows that the Inverse NCut Score of DeepWalk is the lowest. Accordingly, the boundaries between different clusters are not clear in this case, and blue points are divided into two clusters far away from each other. On the other hand, the boundaries of HARP(DeepWalk) are clearer, which is consistent with a higher Inverse NCut score. For ACE(DeepWalk) which achieves the highest Inverse NCut score, points with the same color are gathered tightly, and boundaries are much clearer, especially in the intersection areas of purple, cyan and red points.

F. NODE CLASSIFICATION

In order to evaluate the performance of the embedding vectors on real graph analytic tasks, we conduct node classification experiments on four real-world datasets with the embedding vectors obtained from different algorithms. The classification process is repeated for 10 times and the average Macro-F1



(a) (b) (c)

FIGURE 9. t-SNE visualization of embedding vectors generated by different algorithms on Cora dataset, where different colors represent different node classes. (a) DeepWalk. The iNCut score is 0.963. (b) HARP(DeepWalk). The iNCut score is 0.997. (c) ACE(DeepWalk). The iNCut score is 1.126.

TABLE 3. The precision of node classification. Here the fraction of labeled nodes is set to 0.5.

	BlogCatalog		Wiki		Cora		DBLP	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
DeepWalk	24.83	38.63	52.47	65.64	80.09	80.97	65.58	68.71
HARP(DeepWalk)	25.73 (↑3.62%)	39.45 (↑2.12%)	53.98 (↑2.88%)	66.81 (↑1.78%)	80.88 (↑0.99%)	81.64 (↑0.83%)	66.43 (↑1.30%)	69.49 (↑1.14%)
ACE(DeepWalk)	**27.51** (↑10.79%)	**40.34** (↑4.43%)	**56.26** (↑7.22%)	**68.23** (↑3.95%)	**82.10** (↑2.51%)	**83.01** (↑2.52%)	**68.48** (↑4.42%)	**71.46** (↑4.00%)
LINE	22.83	37.12	44.13	59.97	75.95	77.62	64.44	68.01
HARP(LINE)	23.04 (↑0.92%)	38.54 (↑3.83%)	47.89 (↑8.52%)	61.66 (↑2.82%)	78.62 (↑3.52%)	79.69 (↑2.67%)	64.91 (↑0.73%)	68.29 (↑0.41%)
ACE(LINE)	**24.53** (↑7.45%)	**38.61** (↑4.01%)	**49.89** (↑13.05%)	**63.70** (↑6.22%)	**80.17** (↑5.56%)	**81.33** (↑4.78%)	**67.57** (↑4.86%)	**70.85** (↑4.18%)
node2vec	25.24	38.80	52.11	65.46	80.27	81.13	65.80	69.02
HARP(node2vec)	25.81 (↑2.26%)	39.26 (↑1.19%)	52.43 (↑0.61%)	65.69 (↑0.35%)	81.05 (↑0.97%)	81.90 (↑0.95%)	66.60 (↑1.22%)	69.67 (↑0.94%)
ACE(node2vec)	**27.54** (↑9.11%)	**40.19** (↑3.58%)	**55.95** (↑7.37%)	**67.91** (↑3.74%)	**82.29** (↑2.52%)	**83.18** (↑2.53%)	**68.40** (↑3.95%)	**71.45** (↑3.52%)
SDNE	17.21	32.48	48.73	63.50	71.15	72.49	56.91	62.09
HARP(SDNE)	-	-	-	-	-	-	-	-
ACE(SDNE)	**21.02**(↑22.14%)	**35.56**(↑9.48%)	**54.06**(↑10.93%)	**67.22**(↑5.8%)	**79.77**(↑12.12%)	**80.38**(↑10.82%)	**64.48**(↑13.30%)	**68.20**(↑9.84%)

TABLE 4. Experimental results of different variation models.

	BlogCatalog		Wiki		Cora		DBLP	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
DeepWalk	24.83	38.63	52.47	65.64	80.09	80.97	65.58	68.71
DeepWalk+Random+Recursive	25.73	39.45	53.98	66.81	80.88	81.64	66.43	69.49
DeepWalk+Random+PCA	22.80	37.76	54.83	66.60	81.53	82.60	67.18	70.10
DeepWalk+Community+Recursive	-	-	53.55	67.17	79.96	80.03	66.75	69.76
DeepWalk+Community+PCA	-	-	53.49	66.74	81.62	82.64	66.67	69.69
DeepWalk+ACO+Recursive	26.62	39.89	52.78	66.02	81.30	82.20	66.24	69.57
DeepWalk+ACO+PCA (**ACE**)	**27.51**	**40.34**	**56.26**	**68.23**	**82.10**	**83.01**	**68.48**	**71.46**
LINE	22.83	37.12	44.13	59.97	75.95	77.62	64.44	68.01
LINE+Random+Recursive	23.04	38.54	47.89	61.66	78.62	79.69	64.91	68.29
LINE+Random+PCA	21.71	35.57	46.71	61.14	77.34	78.54	66.54	69.94
LINE+Community+Recursive	-	-	48.35	62.19	77.70	78.68	65.57	69.03
LINE+Community+PCA	-	-	48.45	62.11	78.39	79.42	66.74	70.05
LINE+ACO+Recursive	23.75	37.55	49.57	62.28	78.57	79.91	64.58	67.89
LINE+ACO+PCA (**ACE**)	**24.53**	**38.61**	**49.89**	**63.70**	**80.17**	**81.33**	**67.57**	**70.85**
node2vec	25.24	38.80	52.11	65.46	80.27	81.13	65.80	69.02
node2vec+Random+Recursive	25.81	39.26	52.43	65.69	81.05	81.90	66.60	69.67
node2vec+Random+PCA	22.02	36.94	54.01	66.71	81.58	82.52	67.24	70.31
node2vec+Community+Recursive	-	-	52.51	65.72	80.64	81.44	66.52	69.56
node2vec+Community+PCA	-	-	53.88	66.74	81.69	82.56	66.61	69.63
node2vec+ACO+Recursive	26.67	39.81	52.49	65.75	80.56	81.43	66.53	69.59
node2vec+ACO+PCA (**ACE**)	**27.54**	**40.19**	**55.95**	**67.91**	**82.29**	**83.18**	**68.40**	**71.45**

and Micro-F1 scores are reported in Table 3. We can see that ACE improves all basic embedding algorithms in all datasets, and outperforms the improvement introduced by HARP. Specifically, in the BlogCatalog dataset with many complex structures, which is very challenging for the graph coarsening algorithms, ACE improves the performance of the basic algorithms by 7%∼22% on Macro-F1, and around 4%∼9% on Micro-F1. In this case, HARP only improves the scores by 1%∼4%. Similarly, on the Wiki dataset, ACE improves the basic algorithms by 7%∼13% and 4%∼6% on Macro-F1 and Micro-F1 respectively. For the two citation networks Cora and DBLP with less complex structures, ACE can still outperform the basic algorithms by 3%∼13% on Macro-F1. Moreover, because the implementation of SDNE[1] does not provide API to set initial values of the embedding vectors, we cannot apply HARP on SDNE and the scores of HARP(SDNE) are marked as '-' in the table.

To validate the effectiveness of the two key steps of ACE (graph coarsening and multi-level embedding), we test the variation models of ACE by replacing these steps with other

[1]https://github.com/palash1992/GEM

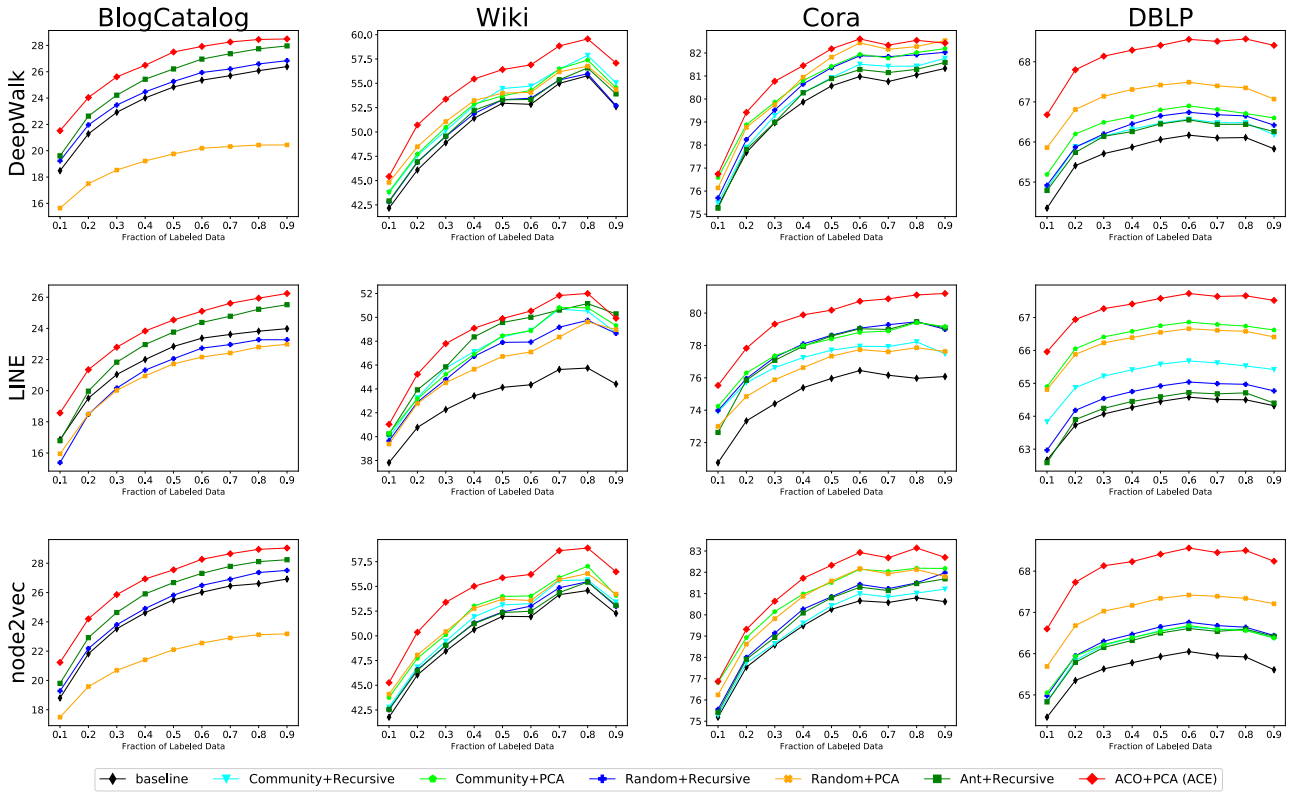


FIGURE 10. The macro-F1 scores of the models with different proportions of labeled data.

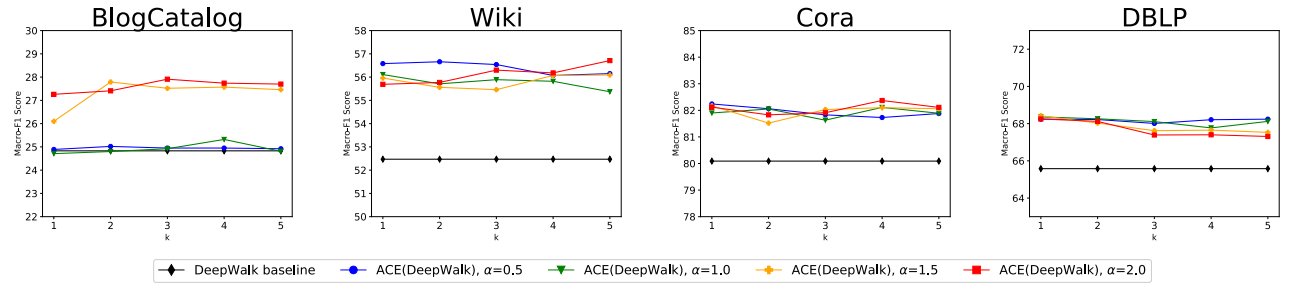


FIGURE 11. Macro-F1 scores run with different k and α combinations.

methods, and show the results in Table 4. For the graph coarsening methods, there is no obvious improvement of *Community* comparing with *Random*, while *ACO* outperforms *Random* significantly in most cases. This proves that the ACO based coarsening are more suitable for improving network embedding than the typical group based community detection algorithms like LPA [20], [27], as discussed in Section IV-B. Meanwhile, it is interesting to find that *ACO* with the aid of *PCA* in the ACE models achieves the best performance, which indicates that the PCA dimension reduction can utilize the hierarchical structures effectively to generate good embedding vectors. Moreover, because the implementation of LPA we used [2] fails to detect the community structures of the BlogCatalog dataset, the scores of *Community* in this case are marked as '−' in the table.

[2]https://networkx.github.io/

To test the stability of the models on the classification tasks with different training configurations of the datasets, we vary the proportion of labeled nodes for classification, and present the Macro-F1 scores in Fig 10. Each column of the figure indicates the experiments on one dataset, and each row is corresponding to the experiments based on one certain basic embedding algorithm. Fig 10 clearly shows that ACE outperforms all other variation models in all configurations.

G. PARAMETER SENSITIVITY

In this section, we study how the parameters affect the performance of ACE. There are two tunable parameters for ACE: k in Algorithm 2 which is the number of iterations in the ant-colony walking while coarsening a graph, and α in Eq (3) which adjusts the importance of edge pheromone. Fig 11 shows the Macro-F1 scores on the test of different combinations of k and α. In most of the test cases (Wiki,

Cora, DBLP), the configuration of $k = 2$ and $\alpha = 0.5$ can achieve the best performance. The results on the BlogCatalog dataset are quite different from the others, where the larger α ($\alpha = 1.5$) is the best. This may be because that BlogCatalog is a scale-free social network with much more complex structures than the others. Larger α can speed up the convergence of the algorithm to locate important intra-links in clusters. On the contrary, in the small Cora network, hierarchical structures are obvious and easy to be detected, so the combination of k and α does not have significant effect on its performance.

VI. CONCLUSION

In this paper, we propose a novel ant colony based network embedding algorithm, namely ACE, to decompose a graph into a clustering pyramid, and blend the embedding vectors which are generated from each layer of the graph pyramid into a multi-level representation of nodes. Comparing to the existing network embedding algorithms which only consider local neighborhood structures, ACE can precisely capture the hierarchical clustering structures of a graph and generate multi-level embedding vectors of the nodes correctly. Furthermore, the user-friendly encoding scheme provided by ACE makes it very easy to integrate with other embedding algorithms like DeepWalk, LINE, node2vec and SDNE to significantly improve their performance, without need of changing their source codes in their original implementations. The experiments based on four real-world graph datasets show the superiority of ACE compared with state-of-the-art network embedding algorithms on the node classification tasks, in which ACE outperforms the traditional network embedding algorithms with improvements as large as 22% on Macro-F1 score.

In the future, we will utilize the labels of nodes to further optimize the ant colony based walking procedure of ACE, and validate the performance of ACE on more graph learning tasks beyond node classification.

REFERENCES

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

[2] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2001, pp. 585–591.

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

[4] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. Int. Conf. Knowl. Discovery Data Mining (SIGKDD)*, 2016, pp. 855–864.

[5] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. Int. Conf. Knowl. Discovery Data Mining (SIGKDD)*, 2016, pp. 1225–1234.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–14.

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

[8] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, "HARP: Hierarchical representation learning for networks," in *Proc. Conf. Artif. Intell. (AAAI)*, 2018, pp. 2127–2134.

[9] I. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer-Verlag, 1989.

[10] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. Int. Conf. World Wide Web (WWW)*, 2015, pp. 1067–1077

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

[12] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. Int. Conf. Knowl. Discovery Data Mining (SIGKDD)*, 2016, pp. 1105–1114.

[13] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 3837–3845.

[14] 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 *Proc. Conf. Artif. Intell. (AAAI)*, 2018, pp. 2508–2515.

[15] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," in *Proc. Conf. Artif. Intell. (AAAI)*, 2018, pp. 2167–2174.

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

[17] 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 *Proc. Int. Conf. Web Search Data Mining (WSDM)*, 2018, pp. 459–467.

[18] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, "Don't walk, skip!: Online learning of multi-scale network embeddings," in *Proc. Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, 2017, pp. 258–265.

[19] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, nos. 2–3, pp. 243–278, 2005.

[20] G. Cordasco and L. Gargano, "Community detection via semi-synchronous label propagation algorithms," 2011, *arXiv:1103.4550*. [Online]. Available: http://arxiv.org/abs/1103.4550

[21] H. Shen, X. Cheng, K. Cai, and M.-B. Hu, "Detect overlapping and hierarchical community structure in networks," *Phys. A, Stat. Mech. Appl.*, vol. 388, no. 8, pp. 1706–1712, Apr. 2009.

[22] X. Wen, W.-N. Chen, Y. Lin, T. Gu, H. Zhang, Y. Li, Y. Yin, and J. Zhang, "A maximal clique based multiobjective evolutionary algorithm for overlapping community detection," *IEEE Trans. Evol. Comput.*, vol. 21, no. 3, pp. 363–377, Jun. 2017.

[23] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

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

[25] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93–106, 2008.

[26] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proc. Int. Conf. Knowl. Discovery Data Mining (SIGKDD)*, 2009, pp. 817–826.

[27] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 76, no. 3, p. 036106, 2007.

[28] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

JIANMING LV received the B.S. degree in computer science from Sun Yat-sen University, China, in 2002, and the Ph.D. degree from the Institute of Computing Technology, University of Chinese Academy of Sciences, in 2008. He is currently an Associate Professor with the South China University of Technology. He has published more than 30 papers at prestigious venues, including the IEEE TPAMI, IEEE TSC, ACM TOMM-CAP, *Computer Networks*, CVPR, and CIKM. His research interests include data mining, computer vision, and distributed computing and privacy. He is also a member of ACM and CCF.

JIAJIE ZHONG received the B.S. degree from the South China University of Technology, China, in 2017, where he is currently pursuing the M.S. degree with the School of Computer Science and Engineering. His research interests include data mining and artificial intelligence.

JINTAO LIANG received the B.S. degree from the South China University of Technology, China, in 2017, where he is currently pursuing the M.S. degree with the School of Computer Science and Engineering. His research interests include data mining and artificial intelligence.

ZHENGUO YANG received the B.E. degree from Shandong Normal University, China, in 2010, the M.E. degree from Zhejiang Normal University, China, in 2013, and the Ph.D. degree from the City University of Hong Kong, in 2017, all in computer science. He is currently a Postdoctoral Fellow with the Guangdong University of Technology. He has published papers on prestigious venues, such as IEEE TPAMI, ACM TOIT, and ACM Multimedia. His research interests include online event detection and transfer learning.