# A-GNN: Anchors-Aware Graph Neural Networks for Node Embedding

Chao Liu[1,2], Xinchuan Li[1,2], Dongyang Zhao[1], Shaolong Guo[3], Xiaojun Kang[1,2](✉), Lijun Dong[1,2], and Hong Yao[1,2]

[1] School of Computer Science, China University of Geosciences, Wuhan 430074, China
kangxj@cug.edu.cn
[2] Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China
[3] Sinopec Exploration Company, Chengdu 610041, Sichuan, China

**Abstract.** With the rapid development of information technology, it has become increasingly popular to handle and analyze complex relationships of various information network applications, such as social networks and biological networks. An unsolved primary challenge is to find a way to represent the network structure to efficiently compute, process and analyze network tasks. Graph Neural Network (GNN) based node representation learning is an emerging learning paradigm that embeds network nodes into a low dimensional vector space through preserving the network topology as possible. However, existing GNN architectures have limitation in distinguishing the position of nodes with the similar topology, which is crucial for many network prediction and classification tasks. Anchors are defined as special nodes which are in the important positions, and carries a lot of interactive information with other normal nodes. In this paper, we propose Anchors-aware Graph Neural Networks (A-GNN), which can make the vectors of node embedding contain location information by introducing anchors. A-GNN first selects the set of anchors, computes the distance of any given target node to each anchor, and afterwards learns a non-linear distance-weighted aggregation scheme over the anchors. Therefore A-GNN can obtain global position information of nodes regarding the anchors. A-GNN are applied to multiple prediction tasks including link prediction and node classification. Experimental results show that our model is superior to other GNN architectures on six datasets, in terms of the ROC, AUC accuracy score.

**Keywords:** Graph Neural Network · Node embedding · Link prediction · Node classification · Global structure information

## 1 Introduction

Network applications are ubiquitous in the real world, such as protein-protein interaction networks [1], information networks [2] and biology networks [3]. In network analysis, it is critical to learn effective representations for nodes, which determine the performance of many downstream network tasks, such as node classification [4], link

X. Chu et al. (Eds.): QShine 2019, LNICST 300, pp. 141–153, 2020.
https://doi.org/10.1007/978-3-030-38819-5_9

prediction [5]. Recently, the use of node representation learning (also known as node embedding) to solve network application problems has received wide attention from researchers, which is aimed at preserving the structure of networks in a low-dimensional vector space.

Many node embedding methods have been proposed in recent years, which can be categorized into three types according to the category of node fusion information. The first type is unsupervised node embedding [8, 9]. This type of models integrate the connection information of each node and its neighbor nodes into the vector representation of the node. However, these models cannot capture node attributes, which is important supplementary information for nodes. The second type is attributed node embedding [6, 10], which is proposed to incorporate node attributes to node embedding. The third type is GNN node embedding [7]. In the GNN framework, the embedding of nodes can contain nodes' attribute information and network neighbor information.

However, the existing methods related to GNN node embedding and attributed node embedding cannot capture the global position information of the node within the broader context of the graph structure, which hinders the processing of downstream tasks. Take the node classification task as an example. Figure 1 shows a symmetrical structure, in which nodes $v_1$ and $v_2$ are in the identical symmetrical positions. In GNN and attributed node embedding methods, $v_1$ and $v_2$ are embedded to the same point in the low-dimensional space because they have isomorphic network neighborhoods. Thus, $v_1$ and $v_2$ cannot be distinguished in node representation vector, and they are classified into the same class.

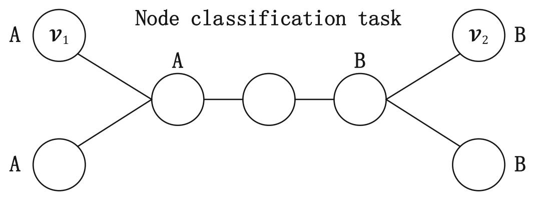

**Fig. 1.** A symmetrical structure example for a simple node classification task, in which nodes $v_1$ and $v_2$ are structurally symmetrical, A and B represent two different node classes respectively.

P-GNN model has partially addressed this problem of distinguishing symmetric nodes [12], which incorporates a node's global positional information with the help of anchors randomly selected in the network. However, P-GNN model has two limitations: (1) Randomly selection of anchors shows the location of the makers is uncertain, causing the P-GNN model unstable; (2) Besides, anchors are usually difficult to be selected at symmetrical locations or the distribution of them is relatively concentrated in a small part. In this case, positional information of most nodes wouldn't work, because the path between anchors and normal nodes in the symmetrical structure are identical. Actually, Anchors are the nodes with important positions and have a lot of information interaction with other nodes, which are not considered in P-GNN model.

Figure 2 gives a further explanation of limitations mentioned above. It shows a large network containing a symmetrical structure (Fig. 2, left). Similar to the notations of Fig. 1, $v_1$ and $v_2$ are in the identical symmetrical positions, A and B stand for two classes of nodes. Assume that the candidate anchor set includes $\alpha$, $\alpha_{1,}$ $\alpha_2$ and $\alpha_3$. Firstly, we explain instability. If $\alpha$ is selected as the anchor, $v_1$ and $v_2$ can be distinguished by P-GNN, because the shortest path ($v_1$, $\alpha$) is different from ($v_2$, $\alpha$). Otherwise if $\alpha_1$ or $\alpha_2$ or $\alpha_3$ is selected as the anchor, $v_1$ and $v_2$ wouldn't be distinguished by P-GNN, because the shortest path from $v_1$ and $v_2$ to the anchor are same. Therefore, selecting $\alpha$ or $\alpha_1$ as anchor will lead two different results for P-GNN model. Secondly, notice that anchor $\alpha$ is in a symmetrical structure, while anchors $\alpha_1$, $\alpha_2$ and $\alpha_3$ are not. But P-GNN model adopts a strategy of randomly selecting anchors, which cannot guarantee that at least one anchor is selected in each symmetric structure.

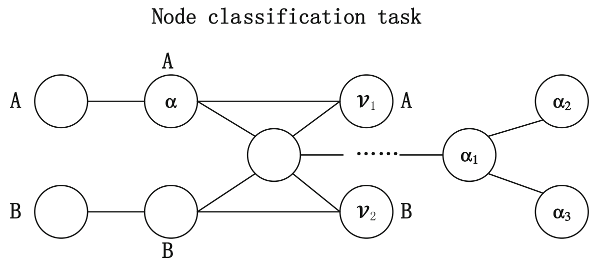

**Fig. 2.** An example of a large network containing a symmetrical structure.

In this paper, we propose Anchors-aware Graph Neural Networks (A-GNN), which use anchors to distinguish normal nodes with different importance. In the selection and computation of anchor nodes, we consider the following two requirements to be met: (1) Anchors should have strong information interaction with other nodes; (2) Anchors could spread as widely as possible in the whole network. We combine the Greedy Algorithm with the Minimum Point Cover Algorithm (**GA-MPCA**) to compute the anchors. Furthermore, the greedy algorithm satisfies the first requirement, and the minimum point cover algorithm satisfies the second one. Through combining with the new anchors computation strategy, A-GNN model supports for incorporating the structure information into the node vector. With the help of anchors, A-GNN learns a non-linear aggregation scheme that combines node feature information from each anchor-set and weighs it by the distance between the node and the anchors.

Overall, the main contributions of this paper are summarized as follows:

(1) We propose Anchors-aware Graph Neural Networks (A-GNN) for improving node embedding, which incorporates the global structure information of a node into its embedding vector with the help of anchors. In this case, nodes can be classified, if they obviously are different in the aspect of position, although they have similar neighborhood nodes.
(2) We propose a new algorithm to compute anchors. The anchors are selected according to the importance of the positions in the whole network. Moreover, we

apply the anchors to the learning of node embedding, which could greatly improves existed GNN architectures.

(3) We evaluate the performance of our A-GNN model for link prediction tasks and pairwise node classification tasks on eight different datasets. The experimental results show that our A-GNN model has significant improvements compared to baselines on both tasks.

The rest of the paper is organized as follows. Section 2 gives a brief review of related works. Section 3 describes the framework of our model. Section 4 reveals our proposed model in detail. In Sect. 5, we introduce the dataset, experiment settings, baseline models, experimental results and discussion. The last section is a conclusion of this paper.

## 2 Related Work

Anchors are the nodes with the important positions, having a lot of information interaction with other nodes. Our model A-GNN integrates the position information of nodes into the vector representation of nodes by introducing anchors. So our model is mainly related to two aspects, one is node importance evaluation, and the other is node embedding.

### 2.1 Node Embedding

According to the category of fusion information, the node embedding related to our work can be divided into three categories: (1) Embedding node's neighbors; (2) Embedding with node's features; (3) Embedding with positional information.

**Embedding with Node's Network Neighbors.** The existing GNN architectures use different aggregation schemes for a node to aggregate its neighbors in the network. For example, Graph Attention Networks aggregate neighborhood information according to trainable attention weights [13]. Message Passing Neural Networks further incorporate edge information when doing the aggregation [14]. However, as Fig. 1 shows, these models cannot distinguish nodes which have similar network neighbors or at symmetric positions in the network.

**Embedding with Node's Features.** Kipf and et al. proposed a heuristics method that alleviates the above issues include assigning an unique identifier to each node [12]. Hamilton and et al. proposed GraphSAGE to concatenate the node's feature in addition to mean/max/LSTM pooled neighborhood information [15]. Zhang and e al. used locally assigned node identifiers plus pre-trained transductive node features [16]. However, these models are not scalable and do not have generalization capabilities for unseen graphs.

**Embedding with Positional Information.** Jiaxuan and et al. proposed Position-aware Graph Neural Network (P-GNN) to capture the position/location of a given node with respect to some maker nodes generated randomly [11]. Although the model can sometimes to distinguish nodes with similar neighbors, it still has unstable limitation, due to its strategy of randomly selecting anchors.

### 2.2 Node Importance Evaluation

There are some traditional methods to evaluate the importance of nodes [21], among which some are based on node deletion, some are based on node affinity, and some are based on shortest paths. Besides, There also are some up-to-date methods are proposed to find important anchors, such as New metrics [22], proposing two type metrics utilized to evaluate the node importance, and GENI [17] is proposed to deal with distinctive challenges involved with predicting node importance in KGs based on GNN. Inspired by these methods and in order to ensure the selected anchors can meet the following two requirements: (1) Have strong information interaction with other nodes; (2) Spread as widely as possible in the whole network. We use **GA-MPCA** to compute the anchors, and combined with the new anchors calculation method, we propose A-GNN model.

## 3 The A-GNN Framework

The purpose of node representation learning is to integrate all useful information related to nodes into the vector representation of nodes. Thus, the high-quality node vectors can achieve good effects in link prediction tasks and node classification tasks. Our proposed A-GNN model can integrate useful information of nodes from three aspects: (1) node's neighborhood structure; (2) node' s attributes; (3) network global position information.

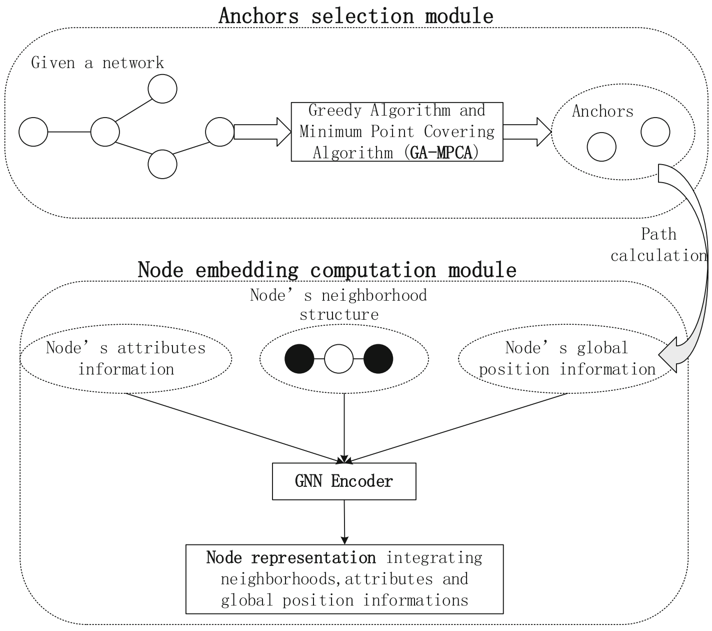
Anchors selection module
Given a network
Greedy Algorithm and Minimum Point Covering Algorithm (GA-MPCA)
Anchors
Path calculation
Node embedding computation module
Node' s neighborhood structure
Node' s attributes information
Node' s global position information
GNN Encoder
Node representation integrating neighborhoods, attributes and global position informations

**Fig. 3.** The framework of A-GNN

Figure 3 gives the framework of A-GNN. As Fig. 3 shows, the A-GNN model can be divided into two mainly modules: (1) The upper layer is anchors selection module, which use GA-MPCA to compute the anchors in the network. The distribution of anchors selected is scattered and the selected anchors have strong information interaction with other nodes; (2) The lower layer is node embedding computation module. The module is used to GNN encoder, for the purpose of to integrate the node's attribute information, neighborhood structure information and global position information into the vector representation of the node.

The workflow of A-GNN model mainly includes the following steps:

- The First step is to input network datasets.
- The second step is to calculate anchors in the network by greedy algorithm and Minimum point covering algorithm.
- The third step is to extract the attribute information, neighborhood structure information contained in the path between anchors and the given node, and its position information relative to the multi-anchors of each given node. Then encode them to the low-dimensional space by GNN.
- The final step is the output the vector representation of each node, which integrates neighborhoods, attributes and global position information.

Compared with the existing node embedding models, our A-GNN model incorporates the global position information of each node, with the help of anchors. Therefore, The A-GNN model can distinguish nodes with similar neighbors or in identical symmetrical positions.

## 4 Proposed Approach

In this section, we mainly illustrate three key aspects of A-GNN model. The first is the strategy of selecting anchors. The second is the method mechanism of node embedding. The last is the description of algorithm implementation.

### 4.1 The Strategy of Anchors Selection

Anchors are the nodes in the important positions with a lot of information interaction with other nodes. Therefore, anchors have an important impact on the whole network, and can be used to provide position information to other nodes. Based on the accuracy position information provided by the anchors, A-GNN model can distinguish nodes, which have similarities in some aspects, but are in different positions. Proper and efficient selection of anchor set can provide high quality position information for nodes, thus improving the accuracy of downstream tasks. Therefore, the selection strategy of the anchor set is very crucial.

As Fig. 2 shows, the strategy of randomly selecting anchors do not satisfy two conditions: (1) one is that the information interaction with other nodes should be rich enough, and (2) the other is that the distribution should be scattered enough. The calculation process of anchors should consider that the importance of each node in a complex network is different. Inspired by previous approaches of evaluating the importance of nodes [31], our A-GNN model proposes a new method, called GA-MPCA to calculate anchors. In terms of information interaction, GA-MPCA believes that the nodes with high degree have strong information communication with other nodes, and they can influence other nodes through the shortest paths. Besides, GA-MPCA adopts first order minimum point covering method to keep the anchors scattered enough.

The strategy of anchor set selection includes three simple steps:

(1) Given a network, we first select the node with the highest degree as the first anchor, and mark it as "anchor" and its one-hop neighbors as "covered". The step belong to the greedy algorithm part.
(2) In the unmarked node geometry, the node with the most connections to the unmarked nodes is found as the next anchor, and mark it as "anchor" and its one-hop neighbors as "covered". This step still belong to the greedy algorithm part.
(3) Repeat (2), until all the nodes in the network are marked "covered" and "anchor", this step belong to minimum point covering algorithm part.

## 4.2 The Node Embedding Mechanism

This subsection mainly consists of two aspect of mechanisms. The one is to utilize GA-MPCA to calculate anchors, the other is to utilize GNN encoder to incorporate the node's attributes, neighborhoods and global position information into the vector representation of the node. It is responsible for the embedding computation process of nodes.

**GA-MPCA.** It is proposed to find anchors in network, which can satisfy the above two requirements: discrete distributions and high interaction with other nodes. To elaborate on the detail process of the algorithm, we use the example shown in Fig. 4. There are totally eight nodes and seven edges, and the red color represents anchors, the light red represents the one-hop neighbors of anchors. Given the network shown in Fig. 4, we infer that nodes $v_7$ and $v_8$ are anchors finally through GA-MPCA.

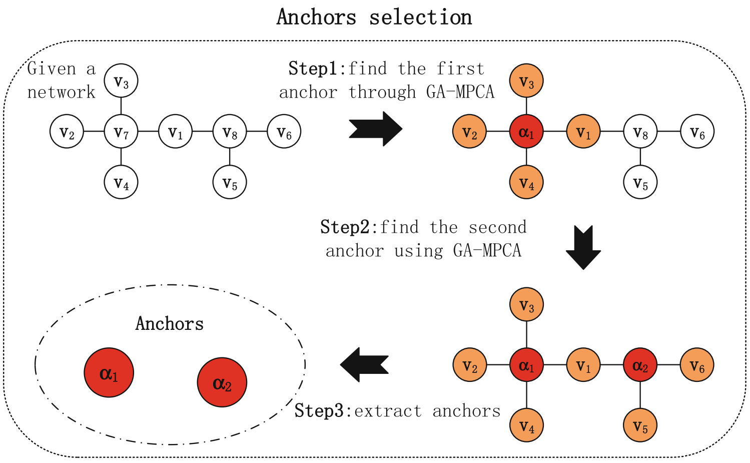

**Fig. 4.** The explanation of GA-MPCA (Color figure online)

For the example of Fig. 4, the steps of GA-MPCA are explained as follows:

(1) Given a network, we firstly use greedy algorithm to find node $v_7$ with the highest degree 4 as the first anchor, then mark it as "anchor" in red, next its one-hop neighbors $v_1$, $v_2$, $v_3$ and $v_4$ are simultaneously marked as "covered" in light red.
(2) Due to there are three nodes still not marked, we continue to calculate the degrees. As a result, the degree of $v_5$ and $v_6$ both are equal to 1, but $v_8$' degree is equal to 2. So we choose $v_8$ as the second anchor, then mark it as "anchor" in red, next its one-hop neighbors $v_5$ and $v_6$ are marked as "covered" in light red.
(3) After the step (2) is completed, all nodes have been marked. So we could extract nodes $v_7$ and $v_8$ as anchors $\alpha_1$ and $\alpha_2$.

**Embedding Computation for Nodes.** In this part, we use GNN encoder to merge the node's attributes, network neighborhood and position information into its vector representation. The calculation process is shown in Fig. 5. In the i-th layer of GNN, the input is a feature vector representation of node $v$. And $h_{\alpha 1}$, …, $h_{\alpha n}$ are the vector of anchors. Function $F_1$ combines the position information into node representation, which is marked as A. Function $AGG_M$ is a trainable function that can transform anchors' features into the given node $v$, represented by M. Then, $M_1$, …, $M_n$ are joined together getting M. We perform two-step operation for M. The one is applying function $AGG_S$ to M, and getting the input vector of next layer. The other is the use of the trainable vector $w$ for projecting M to the output anchors-aware vector $\mathbf{z}_v$ of node $v$.

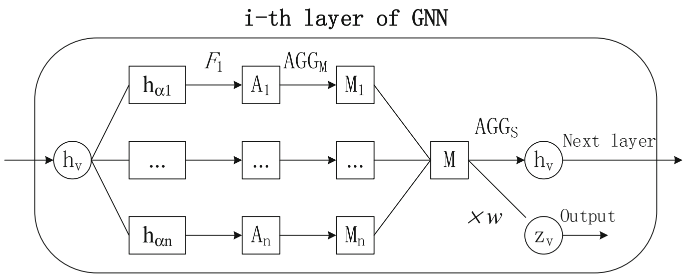

**Fig. 5.** Embedding computation for node v

## 4.3 Algorithm Implementation

In this part, we elaborate the algorithm of anchors selection and GNN encoder in detail.

**Anchors Selection Algorithm (GA-MPCA).** For Algorithm 1, the input can be one or several networks, and the output is the anchors set. We use $G = (\mathcal{V}, \mathcal{E})$ to represent a network, wherein $\mathcal{V}$ represents the node set, $\mathcal{E}$ represents the edge set. Firstly, we go through all the nodes in the network to look for the one with the highest degree. Next, mark it as "anchor" and its neighbors as "covered". Besides, we add them to the set $\mathcal{A}$.

Secondly, we still find the node with the highest degree in $\mathcal{V}$ but not in $\mathcal{V}$, and mark it and its neighbors as well. In addition, we add them to $\mathcal{A}$. Moreover, we repeat step 2, until all nodes in the network are marked. Finally, we go through the network, and add all the nodes marked "anchor" to anchor set S = {$\alpha_1$, …}.

**Algorithm 1** Anchors selection algorithm (**GA-MPCA**)

```
Input: Network $G = (\mathcal{V}, \mathcal{E})$
Output: Anchor set S = {$\alpha_1$, …}
while $\mathcal{V} \neq \mathcal{A}$ do
  for $v \in \mathcal{V}$ and $v \notin \mathcal{A}$ do
    if $v$' degree is the highest do
      mark $v$ "anchor" and its neighbors "covered", and add them to the clique $\mathcal{A}$
    End if
  End for
End while
for $v \in \mathcal{V}$ do
  i = 1
  if $v$ is marked "anchor" do
    $\alpha_i \leftarrow v$
    i = i + 1
  End if
End for
```

**GNN Encoder to Calculate Node's Embedding.** In Algorithm 2, we also represent network with $G = (\mathcal{V}, \mathcal{E})$. Nodes features are represented by $\{y_v, \ldots\}$. There are L layers for GNN, and Anchor set is represented by S = {$\alpha_1$, …, $\alpha_n$}. Firstly, we assign features vectors $\mathbf{h}_{z\{1,\ldots\}}$ to nodes $\mathbf{h}_{v\{1,\ldots\}}$, and regard it as the initial node vector of the first layer neural network. Secondly, we apply the function $F_1$ to combine the given node and anchors' feature information with their network distance to produce A. In Eq. (2), $d_{sp}^{q}(v, \alpha)$ represents the shortest path distance between the given node v and anchor $\alpha$. Note that when the distance is over q, it won't be included in the calculation. Function s($v$, $\alpha$) maps the distance to a (0,1) range.

$$F_1(\mathrm{v}, \alpha, \boldsymbol{h}_v, \boldsymbol{h}_\alpha) = \mathrm{s}(\mathrm{v}, \alpha)\,\mathrm{Concat}\,(\boldsymbol{h}_v, \boldsymbol{h}_\alpha) \tag{1}$$

$$\mathrm{s}(\mathrm{v}, \alpha) = \frac{1}{d_{sp}^{q}(v, \alpha) + 1} \tag{2}$$

Next, trainable aggregation function $\mathrm{AGG_M}$ is applied to A. We transform anchors' features to the given node, which is represented by $\mathrm{M}_{\{1,\ldots\}}$. Then, we combine them into the matrix M. Finally, we perform two operations on M. The one is using AGGs message aggregation function to transform M to the next layer input vector $\mathbf{h}_\mathrm{v}$. The other is projecting M to the anchors-aware vector $\mathbf{z}_\mathrm{v}$ of node $v$ by using the trainable vector $\mathbf{w}$.

## 5 Experiments

The experiment section includes four parts. The first part introduces datasets used for link prediction and node classification tasks. The second part presents the experimental setup, containing inductive learning settings. The third part explains some baseline models, and the last part analyzes the experiment results.

**Algorithm 2** GNN encoder with anchors algorithm

```
Input: Network $G = (\mathcal{V}, \mathcal{E})$; Node input features $\{y_v\}$; Layer $l \in [1, L]$; Anchor set
$S = \{\alpha_1, \ldots, \alpha_n\}$
Output: Anchors-aware embedding $\mathbf{z}_v$ for every node
$h_v \leftarrow y_v$
for $l = 1, \ldots, L$ do
  for $v \in \mathcal{V}$ do
    for $i = 1 \ldots, n$ do
      $A \leftarrow \{F_1(v, \alpha, h_v, h_\alpha), \forall \alpha \in S_i\}$
      $M_i \leftarrow AGG_M(A)$
    End for
    $\mathbf{z}_v \leftarrow \sigma(M \bullet w)$
    $\mathbf{h}_v \leftarrow AGG_S(\{M_v[i], \forall i \in [1,n]\})$
  end for
end for
$\mathbf{z}_v \in R^n, \forall v \in \mathcal{V}$
```

### 5.1 Datasets

We choose some typical synthetic and real datasets (Table 1) to perform our experiments. **Grid**, **Communities** [18], and **PPI** [3] are used for link prediction task, while **Communities**, **Emails** [19], and **Protein** [20] are used to node classification tasks. Note that only nodes in dataset **PPI**, and **Protein** have attributes.

**Table 1.** Statistics on the datasets.

| Dataset | $\|G\|$ | $\|\mathcal{V}\|$ | $\|\mathcal{E}\|$ | #train | #test |
|---|---|---|---|---|---|
| Grid | 1 | 400 | 1216 | 973 | 243 |
| Communities | 1 | 400 | 6080 | 4864 | 1216 |
| PPI | 24 | 56658 | 1269770 | 1015816 | 253954 |
| Email | 7 | 920 | 14402 | 11522 | 2880 |
| Protein | 1113 | 17996 | 68632 | 54906 | 13726 |

### 5.2 Experimental Setup

The proposed A-GNN model is evaluated by two variants. Their differences are reflected in the path calculation of a given node and anchors. (1) P-GNN-F: the variant of P-GNN using truncated 2-hop shortest path distance; (2) P-GNN-E: the variant of P-GNN using exact shortest path distance.

**Table 2.** AUC value for link prediction.

| | Grid | Communities | PPI |
|---|---|---|---|
| GCN | 0.456 ± 0.037 | 0.512 ± 0.008 | 0.769 ± 0.002 |
| GraphSAGE | 0.532 ± 0.050 | 0.516 ± 0.010 | 0.803 ± 0.005 |
| GAT | 0.566 ± 0.052 | 0.618 ± 0.025 | 0.783 ± 0.004 |
| GIN | 0.499 ± 0.054 | 0.692 ± 0.049 | 0.782 ± 0.010 |
| P-GNN-F | 0.694 ± 0.066 | 0.991 ± 0.003 | 0.805 ± 0.003 |
| P-GNN-E | 0.940 ± 0.027 | 0.985 ± 0.008 | 0.808 ± 0.003 |
| P-GNN-F (repetition) | 0.685 ± 0.029 | 0.980 ± 0.007 | 0.786 ± 0.006 |
| P-GNN-E (repetition) | 0.932 ± 0.031 | 0.973 ± 0.000 | 0.793 ± 0.005 |
| A-GNN-F | 0.736 ± 0.066 | 0.988 ± 0.006 | 0.818 ± 0.005 |
| A-GNN-E | **0.969 ± 0.005** | **0.993 ± 0.000** | **0.820 ± 0.003** |

In the experiments of A-GNN model, 80% of the graphs are used to train our A-GNN model and the remaining graphs for testing. For the pairwise node classification task, whether a pair of nodes belongs to the same community is predicted by our model. Note that a pair of nodes that do not belong to the same community are negative.

## 5.3 Baseline Models

In order to prove the validity of our proposed A-GNN model, the classic GNN models and P-GNN model are used as baseline for comparison. Therefore, all above model are trained for the same number of epochs and are set to the same model parameters. The experimental results on both link prediction and pairwise node classification tasks show that our model is about 3% better than the state-of-art P-GNN model in many tasks. Furthermore, in some datasets, the accuracy of our model achieves almost 100%.

**GNN Related Classical Models.** We consider four GNN related classical models, including GCN [12], Graph-SAGE [15], Graph Attention Networks(GAT) [13], and Position-aware Graph Neural Networks (P-GNN) [11].

**A-GNN Model.** Our model consider two variants of A-GNN: (1) A-GNN using truncated 2-hop shortest path distance(A-GNN-F); (2) A-GNN using exactly shortest path distance (A-GNN-E).

## 5.4 Results and Analysis

**Link Prediction.** Link prediction is intended to predict the missing edges in the graph. If two nodes in a low-dimensional vector space are close, they are generally more likely to be linked by an edge. The performance of baseline models and A-GNN model are summarized in Table 2 on link prediction tasks. It can be seen that our A-GNN model achieves better results than any other baseline model of the link prediction tasks. Comparing the datasets, the effect of our model on their improvement is different, ranging from 1% to 4%. The improvements are minimal for both communities and Grid datasets, while obvious for PPI dataset. The reason is that both communities and Grid are small graphs with only 400 network nodes. Therefore, the nodes with similar network neighbors or at symmetric positions can be distinguished by a few makers instead of anchors. However, since each graph in PPI is a large graph, which has about 3000 nodes and 50000 edges on average, and have a lot of symmetrical structures. Thus, anchors can better help A-GNN model to distinguish nodes with similar local structures, compared to randomly selected makers.

**Table 3.** AUC value for pairwise node classification.

| | Communities | Email | Protein |
|---|---|---|---|
| GCN | 0.520 ± 0.025 | 0.515 ± 0.019 | 0.515 ± 0.002 |
| GraphSAGE | 0.514 ± 0.028 | 0.511 ± 0.016 | 0.520 ± 0.003 |
| GAT | 0.620 ± 0.022 | 0.502 ± 0.015 | 0.528 ± 0.011 |
| GIN | 0.620 ± 0.102 | 0.545 ± 0.012 | 0.523 ± 0.002 |
| P-GNN-F | 0.997 ± 0.006 | 0.640 ± 0.037 | 0.729 ± 0.176 |
| P-GNN-E | 1.0 ± 0.001 | 0.640 ± 0.029 | 0.631 ± 0.175 |
| P-GNN-F(repetition) | 0.987 ± 0.002 | 0.676 ± 0.050 | 0.675 ± 0.001 |
| P-GNN-E(repetition) | 0.988 ± 0.005 | 0.668 ± 0.021 | 0.512 ± 0.000 |
| A-GNN-F | 0.988 ± 0.006 | **0.754 ± 0.002** | **0.708 ± 0.003** |
| A-GNN-E | **1.0 ± 0.002** | 0.721 ± 0.009 | 0.633 ± 0.002 |

**Pairwise Node Classification.** In pairwise node classification tasks, we predict whether a pair of nodes belongs to the same community/class. In this case, a pair of nodes that do not belong to the same community are a negative example. The performance of baseline models and A-GNN model are summarized in Table 3 on pairwise node classification tasks. Our A-GNN model achieves the best experimental results in all datasets. Because GNNs focus on learning structure-aware embeddings, these models cannot perform well in distinguishing nodes belonging to different communities but have similar neighbourhood structures. Furthermore, the results of GNNs are about 30% lower than the other two models. A-GNN results are better than P-GNN's, indicating that the anchors adopted by our model are more efficient than randomly generated makers.

## 6 Conclusion

We propose A-GNN model, a new class of GNNs for incorporating the global structure information into the node vector, and utilize node features. We show that A-GNN consistently outperform existing baselines in link prediction and pairwise node classification tasks and both synthetic and real datasets.

**Acknowledgements.** This work was supported in part by the National Key R&D Program of China (Grant No. 2018YFB1004600), the National Science and Technology Major Project of China (Grant No. 2017ZX05036-001) and the National Natural Science Foundation of China (NSFC) (Grant No. 61972365, 61772480, 61672474, 61673354, 61501412).

## References

1. Damian, S., John, H., Helen, C.: quality-controlled protein–protein association networks, made broadly accessible. Nucleic Acid Res., 937 (2016)
2. Ying, R., He, R., Chen, K.: Graph convolutional neural networks for web-scale recommender systems. In: 2018 International Proceedings on ACM SIGKDD Knowledge Discovery and Data Mining, pp. 1576–1585 (2018)
3. Zitnik, M., Leskovec, J.: Predicting multicellular function through multi-layer tissue networks. Bioinformatics **33**(14), i190–i198 (2017)
4. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 2017 International Proceedings on Learning Representation (2017)
5. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: 22th International Proceedings on ACM SIGKDD, pp. 855–864 (2016)
6. Sun, G., Zhang, X.: A novel framework for node/edge attributed graph embedding. In: Yang, Q., Zhou, Z.-H., Gong, Z., Zhang, M.-L., Huang, S.-J. (eds.) PAKDD 2019. LNCS (LNAI), vol. 11441, pp. 169–182. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-16142-2_14
7. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2009)
8. Sun, M., Tang, J., Li, H.: Data poisoning attack against unsupervised node embedding methods. arXiv preprint arXiv:1810.12881 (2018)
9. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: 22nd International Proceedings on ACM SIGKDD Knowledge Discovery and Data Mining, pp. 855–864 (2016)
10. Cheng, Y., Zhi, L., Deli, Z.: Network representation learning with rich text information. In: 2015 International Proceedings on IJCAI (2015)
11. Jiaxuan, Y., Rex, Y., Jure, L.: Position-aware graph neural networks. In: 36th International Proceedings on International Conference on Machine Learning (2019)
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 2017 International Proceedings on International Conference Learning Representations, pp. 1–14 (2017)
13. Velickovic, P., Cucurull, G., Casanova, A.: Graph attention networks (2018)
14. Battaglia, P.W., Hamrick, J.B., Bapst, V.: Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 (2018)
15. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, pp. 1024–1034 (2017)
16. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. Advances in Neural Information Processing Systems (2018)
17. Qin, Q., Wang, D.: Evaluation method for node importance in complex networks based on eccentricity of node. In: Advances in Neural Information Processing Systems, pp. 1324–1334 (2019)
18. Watts, D.J.: Networks, dynamics, and the small-world phenomenon. Am. J. Sociol. **105**(2), 493–527 (1999)
19. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: densification and shrinking diameters. ACM Trans. Knowl. Discov. Data (TKDD) **1**(1), 2 (2007)
20. Borgwardt, K.M., Ong, C.S., Schonauer, S.: Protein function prediction via graph kernels. Bioinformatics **21**(suppl 1), i47–i56 (2005)
21. Xiqing, S., Shoukui, S.: Complex Network Algorithms and Applications, 2nd edn. National Defense Industry Press (2016)
22. Xinbo, A.: New metrics for node importance evaluation in occupational injury network. IEEE Access **7**, 61874–61882 (2019)