Graph Neural Network Based Efficient Subgraph Embedding 2 Method for Link Prediction in Mobile Edge Computing

XiaoLong Deng *, JuFeng Sun, JunWen Lu

Posted Date: 11 May 2023

doi: 10.20944/preprints202305.0797.v1

Keywords: link prediction; graph neural network; graph embedding

Article

Graph Neural Network Based Efficient Subgraph Embedding Method for Link Prediction in Mobile Edge Computing

XiaoLong Deng [1,3], JuFeng Sun [2] and JunWen Lu [3,*]

[1] Cybersecurity School of Beijing University of Posts and Telecommunications, Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education; shannondeng@bupt.edu.cn
[2] Cybersecurity School of Beijing University of Posts and Telecommunications; sunjufeng@bupt.edu.cn
[3] Xiamen University of Technology; jwlu@xmut.edu.cn
* Correspondence: shannondeng@bupt.edu.cn

Abstract: Link prediction is critical to complete the missing links in the network or to predict the generation of new links according to current network structure information, which is vital for analyzing the evolution of the network, such as the logical architecture construction of MEC (Mobile Edge Computing) routing links of 5G/6G Access Network. Link prediction can provide throughput guidance for MEC and select appropriate nodes for message forwarding processing and motivation of this article is to construct a better prediction algorithm to optimize network structure and promote throughput MEC routing links of 5G/6G Access Network. Traditional link prediction algorithms are always based on node similarity, which needs predefined similarity functions, is highly hypothetical and only can be applied to specific network structures without generality. To solve this problem, this paper has proposed a new efficient link prediction algorithm PLAS and its GNN (Graph Neural Network) version PLGAT based on subgraph of target node pair. In order to automatically learn the graph structure characteristics, the algorithm first extracts the h-hop subgraph of target node pair, and then predicts whether the target node pair will be linked according to the subgraph. Experiments on eleven real data sets show that our proposed link prediction algorithm is suitable for various network structures and superior to other link prediction algorithms especially in some 5G MEC Access Networks datasets with higher AUC values.

Keywords: link prediction; graph neural network; graph embedding; 5G MEC network routing links

1. Introduction

In real life, many complex systems can be modeled into complex networks for analysis, such as power network, traffic network, routing network, citation network, 5G/6G space-air-ground communication networks, social network, etc.

In link prediction task, nodes in the network always represent real entities such as routers and switches in network, and associations between entities represent edges. Link prediction is mainly based on current network structure and other information to complete the missing links in current network or predict the possible new connections in the future network [1] such as new routing links which may occur in the 5G/6G space-air-ground communication networks around satellite MEC equipment (Mobile Edge Computing) to transport data-densed computation. As an important research direction in complex networks, link prediction has extensive theoretical research value and practical application value.

In other theoretical research value aspect, link prediction can reveal the evolutionary mechanism of the network and provide a simple and fair comparison method for the evolutionary network [2]. For example, for a certain type of network, many models provide evolutionary methods. The quality of evolutionary methods can be verified by real data sets, but these evolutionary methods are often limited by the scale of evolutionary time or the difficulty of real data set collection. The link prediction

 doi:10.20944/preprints202305.0797.v1

algorithm provides a simple and fair comparison method. According to network structure at a moment $t-n$, the link prediction algorithm can be used to complete the missing link in the current network or to predict the network at time t when new links are generated in the future. Then, the accuracy of different evolution methods can be obtained by comparing with the network at the original moment. And advantages and disadvantages of different evolutionary methods can be obtained by analyzing the accuracy of new generated links.

In other practical applications, for example, in social networks, link prediction algorithm recommends users who have the same interests but are not friends to online social users to improve user stickiness [3]. In Weibo or Facebook, link prediction algorithm can be used to recommend topics or short videos that users are interested in and to predict the popularity of some special topics and short videos [3]. In the field of e-commerce, the relationship graph between users and commodities can be established, and the link prediction algorithm can recommend relevant commodities to users, reduce the time of searching for commodities and improve efficiency [4]. In the protein network, there are many unknown links, and searching for these links requires many crossover experiments, which will waste a lot of manpower and material resources. However, the link prediction algorithm will predict the most likely links, which provides guiding opinions for the experiment, shortens the scope of the experiment, and speeds up the identification process of unknown links. And the experimental cost is reduced [5]. The research on link prediction in complex networks has a wide range of theoretical and practical value.

2. Related Work

In reality, a large number of complex systems can be represented by networks, with nodes representing different entities and edges representing relationships between entities. Link prediction is to predict the missing links in the current network and the generation of new links in the future network through the known network structure and other information [1]. It has a wide range of practical value in friend recommendation [3], product recommendation [4], knowledge graph completion [6] and other fields[29–31].

The heuristic algorithms commonly used in link prediction are based on node similarity [7]. This algorithm assigns a scoring function to each node pair, representing the similarity of node pair, and sorts the unobserved node pairs according to the scoring function. The node pairs with high similarity are more likely to generate new connections. This kind of algorithm can be classified according to the maximum hop number of the maximum neighbor node required to calculate the scoring function [8]. For example, CN [9] and JC [10] link prediction algorithms only need the one-hop neighbor node of the target node pair to calculate the score function, so it belongs to the heuristic algorithm of one-hop node similarity. AA [3] and RA [11] link prediction algorithms need the target node to calculate the two-hop neighbor node when calculating the score function. Therefore, it belongs to the heuristic algorithm of two-hop node similarity. This kind of similarity based heuristic algorithm has become the most common algorithm in link prediction because of its simplicity and effectiveness. However, such algorithms need strong assumptions. When changing from one network structure to another network structure, the assumptions are not consistent with the other network structure. For example, CN algorithm believes that the more common neighbors two nodes have, the more likely they will have links in the future, which is often correct in social networks. However, this is not true in protein interaction networks (the more common neighbors two nodes have, the less likely they are to generate links in the future) [12]. Therefore, it is a significant disadvantage of heuristic algorithms based on node similarity to select appropriate scoring functions for different network structures.

The link prediction algorithm based on machine learning mainly transforms the link prediction task into a binary classification task, in which the node pairs with links are regarded as positive classes, and the node pairs without links are regarded as negative classes. The key to this kind of algorithm mainly lies in the selection of features and classification algorithms. In 2004, Faloutsos et al. [32] introduced a connection subgraph, which can well capture the topology between two nodes in a social network. In 2006, Al et al. [33] extracted the non-topological features of the network based on extracting the topological features of the network, which improved the algorithm's accuracy. In

 doi:10.20944/preprints202305.0797.v1

2007, Liben et al. [34] extracted some network topology features from the citation network, such as CN, AA, Katz, etc., and input them into a supervised learning algorithm for learning and prediction. Their experimental results outperform link prediction algorithms based on individual network topologies. In 2010, Benchettara et al. [35] used the enhanced decision tree algorithm and found that using topological features in the feature set can significantly improve the link prediction algorithm's precision, recall, and F value. In 2014, Fire et al. [36] proposed a set of easily computable graph-structured features and adopted two ensemble learning methods to predict missing links in the network. In 2018, Zhang et al. [37] used the attribute features of nodes as non-topological features to input into supervised learning algorithms, improving link prediction algorithms' accuracy. In 2018, Mandal et al. [38] used a variety of supervised learning algorithms for link prediction in a two-layer network composed of Twitter and Foursquare. In 2022, Kumar et al. [39] introduced the value of node centrality as a sample feature, input it into various supervised learning algorithms for prediction, and achieved the best results on the LGBM (Light Gradient Boosted Machine) classifier. The key to link prediction algorithms based on machine learning lies in selecting feature sets. Such algorithms often extract some topological features of the network as feature sets. However, when solving domain-specific link prediction problems, corresponding domain knowledge is also required to construct its domain-specific features. Compared with the heuristic-based link prediction algorithm, the link prediction algorithm based on machine learning can improve its accuracy, but it also brings time costs for training models and feature selection.

Link prediction algorithms based on graph representation learning mainly map high-dimensional dense matrices (graph data) into low-dimensional dense vectors and then use the mapped vectors for downstream tasks, such as node classification [40,41], graph classification [42,43], link prediction [44], etc. In 2014, Perozzi et al. [45] first proposed the graph representation algorithm DeepWalk for unsupervised learning. The algorithm obtains the sequence of nodes through random walks and inputs the sequence of nodes as sentences into the Skip-Gram model in the Word2Vec algorithm to obtain the node vector representation. In 2015, Tang et al. [23] proposed the LINE algorithm model, which proposed a method of edge sampling so that the vectors of the obtained nodes retain the first-order similarity and second-order similarity. In 2016, Grover et al. [13] proposed the Node2Vec algorithm. Node2Vec is similar to DeepWalk, but Node2Vec uses a biased random walk method, which balances depth-first walk and breadth-first walk, and obtains a higher-quality embedded representation. In 2016, Wang et al. [46] proposed the SDNE (Structural Deep Network Embedding) model. SDNE is a semi-supervised deep learning model that uses a deep network structure to simultaneously optimize the first-order and second-order similarity objective functions and obtains vectors for preserving the graph's local and global structure. In 2016, Cao et al. [47] proposed the DNGR algorithm model. DNGR uses the random walk model (Random Surfing) to generate the probability co-occurrence matrix, calculates the PPMI matrix with the probability co-occurrence matrix, and uses the superimposed denoising automatic encoding machine to extract features to obtain the vector representation of the node. In 2016, Kipf et al. [48] proposed the GCN (Graph Convolutional Network) model. The algorithm considers the influence of neighbor nodes and continuously aggregates the characteristics of neighbor nodes. Embedding neighbor nodes can obtain scalability, and the global information can be described by aggregating the characteristics of neighbor nodes through multiple iterations. In 2016, Kipf et al. [24] proposed the VGAE (Variational Graph Auto-Encoders) model, introducing variational autoencoders into graph data. The distribution of the node vector representation of the known graph is learned through GCN convolution, the representation of the node vector is sampled in the distribution and then decoded (link prediction) to reconstruct the graph. In 2017, Veličković et al. [28] proposed the GAT (Graph Attention Networks) model, which introduced an attention mechanism. When calculating the vector representation of nodes, the model's generalization ability is improved by assigning different weights to the characteristics of nodes. At the same time, a multi-head attention mechanism is introduced, and the features obtained by multiple attention mechanisms are spliced and averaged to obtain the final node representation. In 2017, Hamilton et al. [49] proposed the GraphSage algorithm model for large-scale graph data. By learning an aggregation function, the neighbor nodes are sampled and aggregated to

 doi:10.20944/preprints202305.0797.v1

obtain a new vector representation of the node. In 2018, Chen et al. [50] proposed the HARP algorithm model. HARP selects the starting node by weight and combines it with DeepWalk and Node2Vec to obtain a better embedding representation. In 2018, Schlichtkrull et al. [51] proposed the R-GCN (Relational Graph Convolutional Networks) algorithm model. R-GCN introduced weight sharing and parameter constraints to improve the performance of the link prediction algorithm. In 2018, Sam et al. [52] learned the representation vectors of nodes in historical time slices through the Node2Vec algorithm, concatenated the vectors of nodes in historical time slices to obtain the representation of future time slice links, and finally used supervised learning algorithms to predict future time links state. In 2019, Lei et al. [53] used GCN to learn the network topology features of each time slice, used LSTM (Long Short Term Memory) to capture the evolution pattern of multiple continuous time slice dynamic networks, and then used GAN (Generative Adversarial Network) to generate a representation of links in future time slices. In 2017, Zhang et al. [54] proposed the WLNM link prediction model, which extracts the h-hop closed subgraph of the target node pair and sends the adjacency matrix of the subgraph to the fully connected layer for learning, which improves the link prediction algorithm results. Link prediction methods based on graph representation learning are difficult to capture deeper network structural relationships in complex networks and more complex relationship features between nodes due to limited walk steps and aggregation methods, resulting in lower algorithm accuracy.

Considering that the existing link prediction algorithms are unsuitable for different network structures, capturing the deeper network structure relationships and the more complex relationship characteristics between nodes is difficult, this paper proposes a link prediction algorithm based on subgraph (PLAS Predicting Links by Analysis Subgraph). The algorithm firstly obtains h-hop neighbor nodes of the target node pair to form a subgraph, and then assigns labels to each node of the subgraph. The nodes in the Subgraph are sorted according to labels. Finally, the nodes of the subgraph are input to the full connection layer in a consistent order for classification.

And our contributions are as follows:

1. A subgraph node labeling method is provided, which can automatically learn graph structure features and input nodes of subgraphs into the full connection layer in a consistent order.
2. A link prediction method (PLAS) based on subgraph is proposed, which can be applied to different network structures and is superior to other link prediction algorithms.
3. Based on torch, the link prediction algorithm (PLAS) model based on subgraph is implemented and verified on seven real data sets. Experimental results show that PLAS algorithm is superior to other link prediction algorithms.
4. The existing algorithm PLAS is improved by introducing graph attention network, and a link prediction algorithm (PLGAT) is proposed, which has been verified on seven real data sets and two 5G/6G space-air-ground communication networks. The experimental results show that PLGAT algorithm is superior to other link prediction algorithms. Furthermore, our proposed PLGAT algorithm for link prediction can precisely find out the new links on the Mobile MEC equipment network in 5G/6G to provide better QoS for data transportation.

3. PLAS model framework

This paper proposed a link prediction algorithm PLAS (Predicting Links by Analysis Subgraph) based on Subgraph. PLAS algorithm transforms the link prediction task into a graph classification task, with the target node taking the linked subgraph as a positive sample and the target node taking the unlinked subgraph as a negative sample. Compared with the link prediction algorithm based on machine learning, it uses node labels to learn the graph features of the subgraph automatically and integrates the potential features of the subgraph nodes and node attribute features. The subgraph with multi-features has more comprehensive information, which can improve link prediction accuracy. Figure 1 is the frame diagram of the model, which is mainly divided into four modules: 1. Extraction of subgraphs; 2. Graph labeling algorithm; 3. Subgraph encoding; 4. Fully connected layer learning.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0797.v1

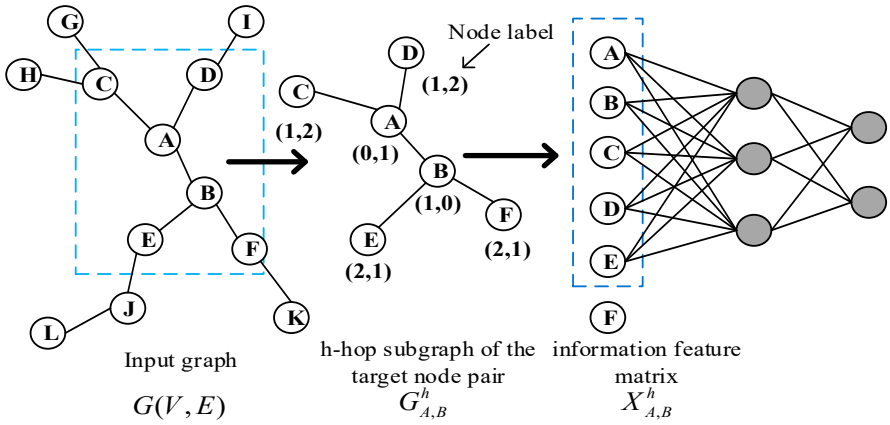

Figure 1. Schematic diagram of PLAS algorithm model

The principle of the PLAS algorithm is shown in Figure 1. Given the undirected and unweighted graph $G(V,E)$ at the input time t, the subgraph of the h-hop of the target node A, B is firstly extracted $G^h_{A,B}$ (1-hop in the graph), and then labels are assigned to each node in the subgraph through the graph labeling algorithm. The nodes in the subgraph are sorted according to labels and the information characteristic matrix $X^h_{A,B}$ is constructed according to their order. Then select the first k node, splicing the corresponding features of the first k node in the information matrix $X^h_{A,B}$ to represent the embedding of the subgraph of target node pair, and finally input it to the fully connection layer for learning. $G^h_{x,y}$ is the closed subgraph of the h-hop of the target node pair (x,y), including all the first-order, second-order and nodes of h-order nodes and edges of corresponding nodes. $X^h_{x,y}$ is information feature matrix that sorts nodes in the subgraph and builds in accordance with their order, which is described as follows:

$$X^h_{A,B} = \begin{pmatrix} F_1^e & F_1^l & F_1^g \\ \vdots & \vdots & \vdots \\ F_n^e & F_n^l & F_n^g \end{pmatrix} \tag{1}$$

Where F_1^e, F_1^l and F_1^g are the explicit attribute feature, implicit attribute feature and graph label of the first node in the information feature matrix. And each row in information feature matrix $X^h_{x,y}$ represents the feature of a node in the subgraph $G^h_{x,y}$. The feature of each node is composed of explicit attributes (attributes of the node itself, such as interest, position, gender, etc.), implicit attributes (Node2Vec [13] embedding nodes in the graph) and the label of each node in the subgraph (network structure features).

3.1. Extraction of subgraphs

In order to learn the topological features of the network, the PLAS algorithm extracts its h-hop closed subgraph for each target node pair. In a natural system, the connection between nodes will be affected by neighbor nodes, and the more intermediate nodes on the node path, the weaker the relationship between two nodes. Although more node structure may lead to more link information, whether the node pair will generate a link will be primarily affected by 1-hop or 2-hop neighbor nodes. In this paper, the h parameter is set to 1 or 2 according to the sparseness of the network structure. When the average number of neighbor nodes of each node in the network is greater than 10, parameter h is selected as 1; when the average number of neighbor nodes of each node in the network is less than 10, parameter h is selected as 2. It ensures that the value of h will not be set too small, resulting in insufficient neighbor information, significantly impacting the prediction results. Moreover, with the decrease of h parameter, the number of neighbor nodes extracted from the target node pairs will decrease exponentially, indicating that the dimension of feature vector of subgraph decreases, which can significantly reduce the training time of the model.

But for large-scale networks, such as social networks, where a superstar has millions of fans, the number of first-order neighbor nodes is very large, which will lead to a memory explosion when the

 doi:10.20944/preprints202305.0797.v1

subgraph is extracted. This paper sets a threshold N to prevent the problem of oversampling of subgraphs. When extracting the subgraph of the target node pair, if the number of subgraph nodes exceeds the threshold value N, the current h-order neighbor nodes will be randomly sampled to equal N, prioritizing sampling nodes with closer hops. For example, we need to extract 2-hop neighbor node subgraphs, while setting the threshold N value with 100, extract first-order neighbor nodes to 50, extract second-order neighbor nodes to 250, and the sum of first-order neighbor nodes and second-order neighbor nodes is greater than the threshold N. Therefore, we need to randomly sample second-order neighbor nodes (randomly select 50 nodes from 250 nodes) so that the number of summary points in the subgraph is equal to the threshold N.

The subgraph extraction algorithm is described as follows:

Algorithm: Subgraph extraction
Input: Target node pair (i,j), graph $G(V,E)$, h-hops neighbor nodes, threshold N
Output: h-hops subgraph S of target node pair (i,j)
$V(s) = \{i,j\}$
$N(temp) = \{\}$
$c = 1$
while $c \leq h$ ***do***
    $N(temp) = T_i^c \cup T_j^c$
    $if\ len(N(temp)) + len(V(s)) > N$:
        $N(temp) = sample(N(temp))$
        $V(s) = N(temp) \cup V(s)$
        break
    $V(s) = N(temp) \cup V(s)$
    $c = c + 1$
end while
$S = SubGraph(V(s))$
return S

The target node h-hop neighbor node subgraph extraction algorithm process is as follows: for the target node pair (i,j) in graph G, first add the target node pair (i,j) to the subgraph node set $V(s)$, and then add the node pair (i,j) first-order neighbors, second-order neighbors to the h-order neighbors to the set $V(s)$. The relationship between all nodes and nodes in the set constitutes a subgraph S.

3.2. Graph labeling algorithm

The graph is an unordered data structure. We need a graph labeling algorithm to sort the nodes in the graph according to their labels to form an ordered sequence so that the fully connected layer can read node feature in a consistent order.

The Weisfeiler Lehman (WL) algorithm [14] is widely used in graph classification, and it is a classic graph labeling algorithm. The main idea of WL algorithm is to use the labels of neighbor nodes to update their own labels iteratively and compress the updated labels into new ones until they converge. The primary process of WL algorithm is as follows:

1. It initializes all nodes in the graph to the same label 1, and each node aggregates its label and the labels of neighbor nodes to construct a label string.
2. The nodes in the graph are sorted in ascending order of label strings, and according to the sorting update to new labels 1, 2, 3, ... nodes with the same label string will get the same new label. For example, suppose that the label of node x is 2, its neighbor label is {3,1,2}, the label of node y is 2, and its neighbor label is {2,1,2}. The label strings of x and y are < 2123 > and < 2122 > respectively. Because < 2122 > is less than < 2123 > in the dictionary order, y will be assigned a smaller label than x in the next iteration.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0797.v1

3. This process is repeated until the node label stops changing. Figure 2 shows updating the nodes' label from 1 to (1, ... 5).

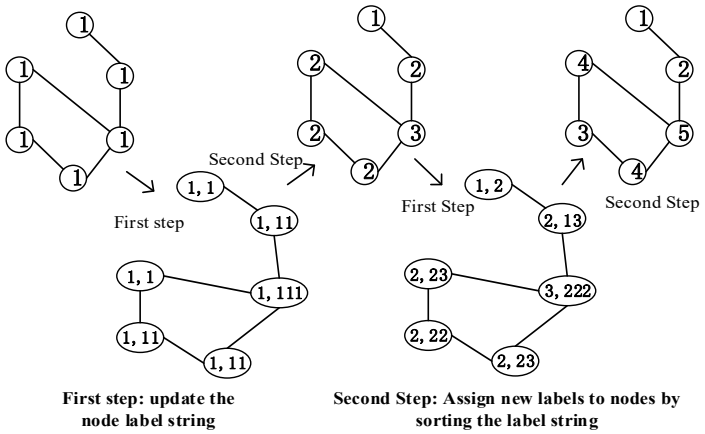

Figure 2. Schematic diagram of WL algorithm

WL algorithm has two key advantages: 1. The final label coding represents the structural role of nodes in the graph. Nodes with similar structure have similar labels in different graphs. 2. It defines the relative order of nodes in the graph, which is also consistent in different subgraphs. However, the WL algorithm treats all nodes in the subgraph equally during initialization and assigns the same label. After multiple rounds of iterations, the final node label encoding makes it impossible to distinguish the target node pair from other nodes in the subgraph, resulting in untargeted model training. Therefore, we designed a new graph labeling algorithm combined with the shortest distance, which can ensure that the nodes in different subgraphs are sorted in a consistent order and distinguish the target node pair from other nodes in the subgraph.

Algorithm: subgraph labeling

Input: Target node pair (i, j), subgraph node list N, subgraph S

Output: Ordered list with labels O

$O \leftarrow (i, (0,1))$
$O \leftarrow (j, (1,0))$
$R = N - \{i, j\}$
for $each\ v \in R\ do$
  $d_i = d(v, i)$
  $d_j = d(v, j)$
  $O \leftarrow (v, (d_i, d_j))$
end for
$sorted\ O\ by\ (d_i, d_j)$
return O

The process of the subgraph labeling algorithm is as follows. The label string (d_i, d_j) for each node consists of the shortest distance from that node to the target link node pair (i, j), where d_i and d_j are the shortest distances from this node to the target nodes i and j respectively. After extracting the subgraph, we put all the extracted nodes into set R. For each node in set R, we calculate the shortest distance (d_i, d_j) from each node to the target node pair (i, j) as the label string, and add it to the sequence list O for uniform sorting. And we allocate the label strings (0,1) and (1,0) for the target link node pair (i, j), respectively. The sequence list O is sorted first by the value of d_i, and then by the value of d_j.

Since the shortest distance (d_i, d_j) from other nodes to the target node pair is always greater than or equal to 1, according to the sorting rules of the sequence list, the label string (0,1), (1,0) of the target link node pair is always smaller than the label string of other nodes. Therefore, the first two

 doi:10.20944/preprints202305.0797.v1

elements in the sequence list O are always target node pairs (i, j), which can well distinguish the target link node from other nodes in the subgraph. The label string of a node defines the structural role in the subgraph centered on the target link node pair. And in the subgraph centered on different links, nodes with the same structural role have similar label strings. Compared with the WL algorithm, the graph labeling algorithm in this paper only needs to calculate the shortest distance from the node in the subgraph to the target node, and there is no iterative process of updating the node label of the sub graph, so it has low time complexity.

3.3. Subgraph encoding

The purpose of subgraph encoding is to represent the subgraph centered on the target node pair as a node information feature matrix $X_{x,y}^{h}$ with specific order. We construct the information feature matrix according to the order of nodes in the sequence list O. The information feature of each node is composed of explicit attribute, implicit attribute and node label.

Explicit attribute: The attribute of the node itself is the attribute feature of the data set. For example, in the social network, the attribute feature of the node is interest, hobby, gender, etc.

Implicit attribute: Forming node embeddings in a graph using a graph embedding algorithm. Node2vec is the common node embedding algorithm, it first obtains a series of node sequences by random walk in the network, and then models such node sequences by processing word vectors to obtain the representation of node vectors in the network.

The label of each node in the subgraph: The node label is represented by the function $f(v) = (d_i, d_j)$, which assigns a label string $f(v)$ to each node in the subgraph. Node labels have three main functions:

1. It can represent that different nodes play different roles in the subgraph. The shorter the shortest path of other nodes in the subgraph relative to the target node pair, the greater its impact on whether the target node will generate links in the future, so it plays a more important role in the subgraph.
2. Graph is an unordered data structure, which has no fixed order. Therefore, it is necessary to sort the nodes in the sub graph through labels, and then input them to the fully connection layer in a consistent order for learning.
3. After extracting the h hop subgraph of the target link node pair (i, j), we calculate the shortest path from other nodes in the subgraph to the target node pair (i, j), and assign a label string to each node in the subgraph. When all node features in the subgraph are spliced and sent to the fully connection layer for learning, the fully connection layer will automatically learn the graph structure features suitable for the current network, including the discovered graph structure features or the undiscovered graph structure features. For example, CN algorithm is to calculate the number of common neighbor nodes of the target node pair. The full connection layer only needs to find the number of nodes with node label (1,1). By assigning a node label string to each node in the graph through the icon algorithm, our algorithm model can automatically learn the graph structure characteristics of the network, so it can be applied to different network structures. The later experimental results show that our algorithm is better in AUC than other link prediction algorithms.

When we obtain the explicit attribute, implicit attribute, and label of each node in the subgraph, we splice the characteristics (explicit attribute, implicit attribute, and node label) of each node in the order of the sequence list, and construct its information feature matrix to represent the whole subgraph.

3.4. Fully connected layer learning

The fully connected layer is used to integrate the information feature matrix $X_{x,y}^{h}$ of the h hop subgraph of different target link node pairs for learning. Since the number of nodes in the h-hop subgraph of each node pair is different, the training of the fully connected layer will be significantly affected by the subgraph with a large number of nodes, while the impact of the node information with a small number of subgraphs will be affected by neglect. Therefore, it is necessary to set a unified

 doi:10.20944/preprints202305.0797.v1

node number threshold K to balance the feature dimension input of different subgraphs. We sort each node in the subgraph using a graph labeling algorithm. When the number of nodes in the subgraph is greater than K, the nodes ranked after K are discarded. When the number of nodes in the subgraph is less than K, we construct virtual nodes so that the number of nodes equals K, and zero vectors represent the feature of the virtual nodes in $X_{x,y}^{h}$. We set the specific K value according to the overall network topology. We sort the number of each subgraph node in training set in ascending order and take the value of the corresponding element at the 0.6 scale index as K. If the K value is set too small, too many nodes need to be deleted, resulting in too much information loss; if the K value is set too large, we need to construct a large number of virtual nodes that do not contain any information, and it will also cause the input feature dimension to be too large, increasing the training time. Therefore, we have compromised selecting the K value according to the network topology.

The neural network structure of the fully connected layer consists of an input layer, a hidden layer, and an output layer. The number of neurons in the input layer is determined by the dimension of each node feature vector multiplied by K, and the number of neurons in the hidden layer is set to 128. And the number of neurons in the output layer is 2, because we converted link prediction into a graph classification problem (two classes), where the subgraph with linked node pairs represents one class, the subgraph without linked node pairs represents the other class. In this paper, the ReLU function was used for activation function and cross entropy loss function for the loss function.

4. Experiments Results

Experiment environment: Stand-alone processor AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz, NVIDIA GeForce RTX 2060, memory 16G, operating system is Windows 10, the programming language is Python, based on the deep learning framework Pytorch.

4.1. PLAS algorithm

We conducted experiments with PLAS model on seven real data sets, and evaluated our model with AUC (Area Under Curve). USAir [15] is the transportation network of American Airlines, NS is the co-author network composed of researchers on the network [16], Pb is the blog network of the U.S. government [17], Yeast is the network composed of protein interactions in yeast [18], Cele is the divine network composed of neurons of a worm [19], power is the power network in the western United States [19], and router is the Internet network composed of routers [20]. Furthermore, we introduced the dataset Bupt5GMEC which is the real network structure of 5G MEC Access Network in BUPT (Beijing University of Posts and Telecommunications) while it demonstrates the inside routing structure of BUPT for 5G MEC access experimental environment through a MEC Hub Node. And this 5G MEC access experimental environment has be used to evaluate some routing links in 5G MEC experiments in Figure 3. Besides that, we also found some other important 5G MEC Access Network with the name CM1-5G and CM2-5G from other 5G MEC experiment joined developed with China Mobile. Finally, the real AS topology dataset ITDK0304S of Global Internet is added to Table 1 to carry out evaluation on the largest subset of real Internet topology [27]. For the construction of the ITDK0304S dataset, because the original dataset has some nodes separated from the overall network structure, there are few links between these nodes, and they are relatively independent of the overall network. As a result, it contains less information and research value and will affect model training. Therefore, we use the FastGN [55] community detection algorithm to divide the original data set. We use several communities that can mainly represent the overall structure to construct the ITDK0304S data set and delete some individual communities with few nodes and nodes not worth researching. Since our model performs link prediction through extracted subgraphs, for large-scale real-world networks, we can adjust the hop count and threshold N of sampling neighbours during subgraph extraction to ensure sufficient information for training without causing oversampling.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0797.v1

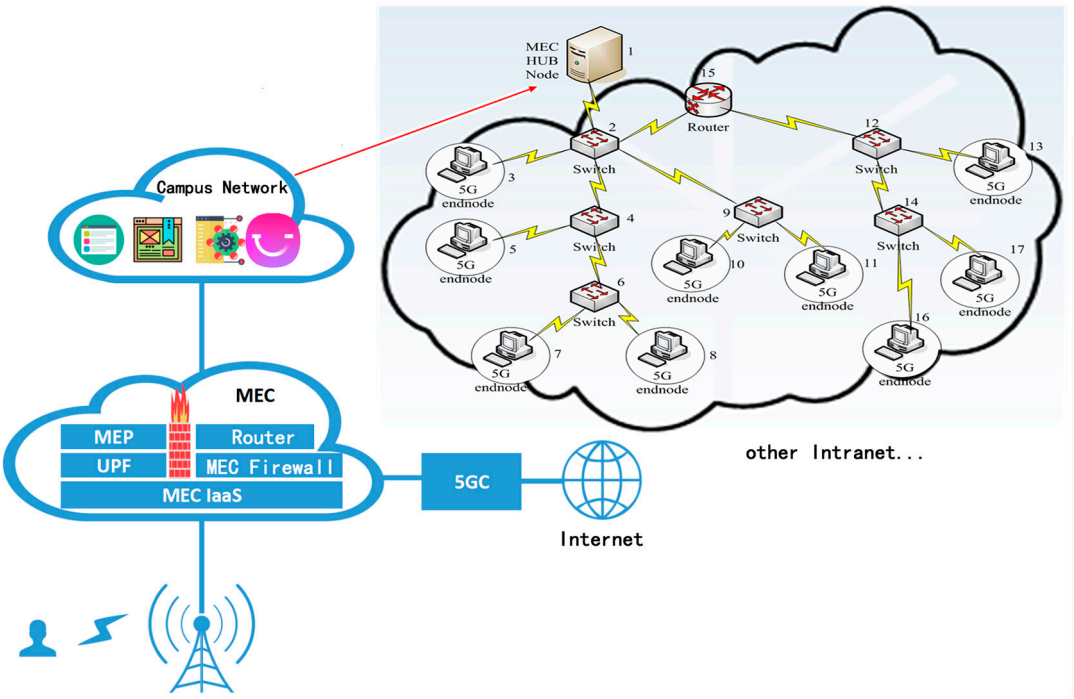

Figure 3. 5G MEC Access Network Structure in BUPT

Table 1 contains the statistical information of each network. We abstract the real-world network structure data into the form of point-edge pairs. We take all the node pairs with edges in each network (as positive samples) and take the same number of node pairs without edges (as negative samples) to form a data set, and then randomly select 90% of the node pairs with and without edges as the training set and the remaining 10% of the data set as the test set. When we use the Node2Vec algorithm to calculate the implicit attributes of nodes, the node sequence obtained by random walk may imply information about whether there is a link between the target node pair, resulting in poor model generalization. Therefore, when the Node2Vec algorithm embeds nodes for each subgraph, links are added to the target nodes without link relationship so that the link information of any target node pair in the training set is the same.

Table 1. Statistical information of network structure of experimental data set

DataSet	$\|V\|$	$\|E\|$
Router	5022	6258
USAir	332	2126
NS	1589	2742
PB	1222	16714
Yeast	2375	11693
Cele	297	2148
Power	4941	6594
Bupt5GMEC	135	338
CM1-5G	1500	5990
CM2-5G	1499	4498
ITDK0304S	3780	10757

We selected five classical heuristic algorithms based on node similarity to compare with our algorithm, which are common neighbors (CN) [16], Jaccard coefficient (JC) [10], Adamic ADAR (AA), resource allocation (RA) and Katz [22].

CN: the number of common neighbors of target node x and y, which is defined as follows:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0797.v1

$$CN(x,y) = |T(x) \cap T(y)| \tag{2}$$

$T(x)$ represents a collection of neighbor nodes x, $|T(x)|$ represents the number of neighbors of node x.

JC: it is similar to CN algorithm. It is the expression of standardization of common neighbors. Its definition is as follows:

$$JC(x,y) = |\frac{T(x) \cap T(y)}{T(x) \cup T(y)}| \tag{3}$$

AA: a neighbor node with a small degree has a larger weight, which is defined as follows:

$$AA(x,y) = \sum_{z \in T(x) \cap T(y)} \frac{1}{log\,|T(z)|} \tag{4}$$

RA: it is similar to AA, but the punishment is different. Its definition is as follows:

$$RA(x,y) = \sum_{z \in T(x) \cap T(y)} \frac{1}{|T(z)|} \tag{5}$$

Katz: it calculates all paths of the target node. The shorter the path, the higher the weight. It is defined as follows:

$$Katz(x,y) = \sum_{l=1}^{\infty} B^l \cdot |path_{x,y}^l| \tag{6}$$

We also selected three methods of link prediction using graph embedding to compare with our model, which are node2vec (n2v) [13], line [23], vgae [24]. They all embed the nodes of the graph, then splice the vectors of the target node pairs and send them to the fully connected layer for classification learning.

We use 50 epochs to train our model. In the process of training, we randomly select 10% of the training set as our verification set for evaluation, and we save the model with the best performance on the verification set. Finally, we use the saved model to predict on the test set.

We first compare with five heuristic based link prediction algorithms, and the results are shown in Table 2 and Figure 4:

Table 2. Comparison results with five heuristic link prediction algorithms (AUC)

DataSet	CN	JC	AA	RA	Katz	**PLAS**
Router	56.57	56.38	56.40	56.38	38.62	**88.81**
USAir	94.02	89.81	95.08	**95.67**	92.90	93.28
NS	93.94	94.40	94.77	94.33	95.03	**98.93**
PB	92.07	87.39	92.31	92.45	92.89	**93.37**
Yeast	89.41	89.30	89.32	89.44	92.30	**95.04**
Cele	84.95	80.18	87.03	87.49	86.45	**87.84**
Power	58.90	58.80	58.83	58,83	65.90	**74.27**

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0797.v1

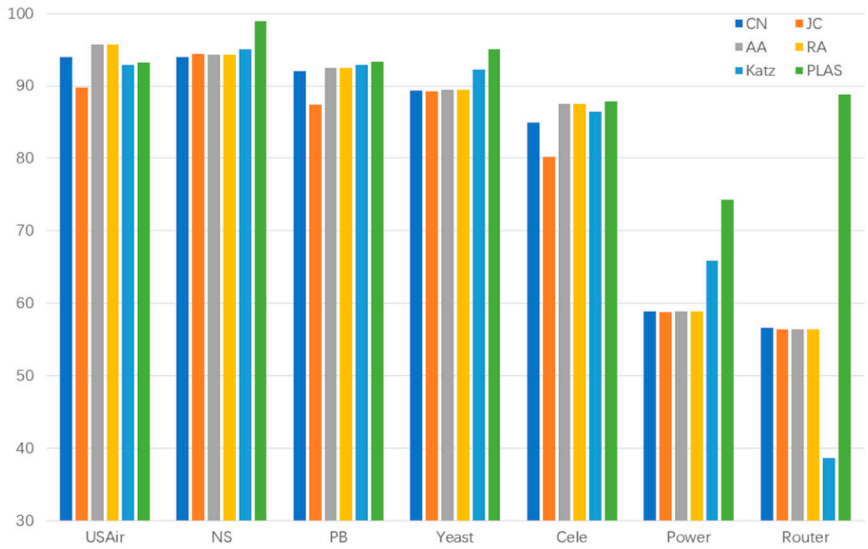

Figure 4. Comparison results with five heuristic link prediction algorithms (AUC)

As shown in Table 2, our algorithm performs better than heuristic link prediction algorithm on data sets on NS, Pb, yeast, Cele, power and router. On the USAir dataset, our algorithm is also better than Katz and JC algorithms. Among them, on sparse data sets such as power and router, the AUC of heuristic link prediction algorithm is about 65% and 55% respectively, while the AUC of our link prediction algorithm can reach about 75% and 85%, which is increased by 10% and 30% respectively. As shown in Table 2, our algorithm performs better than heuristic link prediction algorithm on data sets on NS, Pb, yeast, Cele, power and router. On the USAir dataset, our algorithm is also better than Katz and JC algorithms. Among them, on sparse data sets such as power and router, the AUC of heuristic link prediction algorithm is about 65% and 55% respectively, while the AUC of our link prediction algorithm can reach about 75% and 85%, which is increased by 10% and 30% respectively.

We then compare with three link prediction algorithms based on graph representation. The experimental results are shown in Table 3 and Figure 5:

Table 3. Comparison results with three link prediction algorithms based on graph representation learning (AUC)

Dataset	Node2Vec	LINE	VGAE	**PLAS**
Router	65.46	67.17	61.53	**88.81**
USAir	91.40	81.47	89.30	**93.28**
NS	91.55	80.63	94.04	**98.93**
PB	85.79	76.94	90.70	**93.37**
Yeast	93.68	87.45	93.87	**95.04**
Cele	84.13	69.22	81.87	**87.84**
Power	**76.23**	55.64	71.20	74.27

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0797.v1

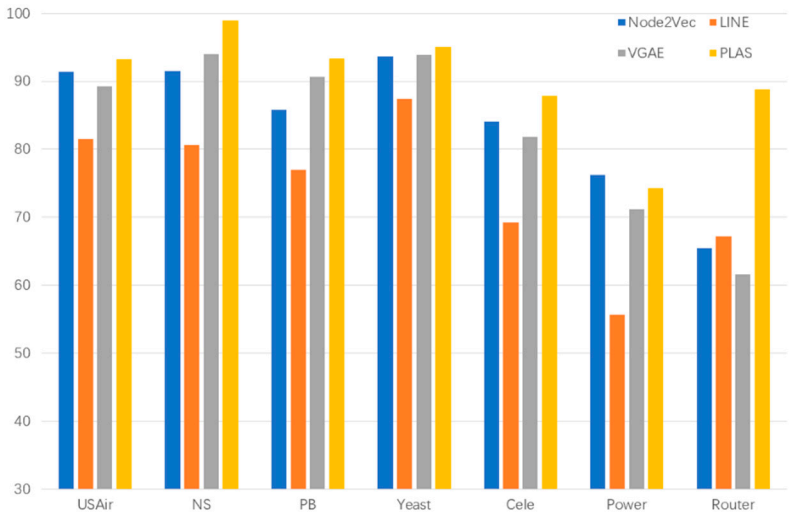

Figure 5. Comparison results with three link prediction algorithms based on graph representation learning (AUC)

As shown in Table 3, the link prediction algorithm based on graph representation has better performance on the sparse data set of power and router than the heuristic link prediction algorithm, but compared with our algorithm, especially on the router data, our algorithm improves the AUC by about 20%. On USAir, NS, Pb, yeast and Cele datasets, our algorithm is better than the link prediction algorithm based on graph representation.

4.2. PLGAT algorithm

The PLAS algorithm treats neighbor nodes equally, lacks consideration of the degree of influence on different neighbor nodes, and may discard some nodes in the training of the fully connected layer, resulting in information loss. Aiming at these problems, we improve PLAS and propose a link prediction algorithm PLGAT (Predicting Links by Graph Attention Networks) based on graph attention network [28]. The algorithm distinguishes the degree of influence of different nodes on the target node through the attention mechanism. It aggregates the information of the discarded nodes through two layers of GAT convolution layers so that the nodes have comprehensive information and improve the accuracy of the link prediction algorithm. In addition, the algorithm defines the relative position of the nodes in the subgraph through the pooling layer module and reflects the directionality of the subgraph (centering on the target node and expanding to the neighbor nodes on both sides), which improves the link prediction results. The PLGAT model framework of our algorithm is shown in Figure 6:

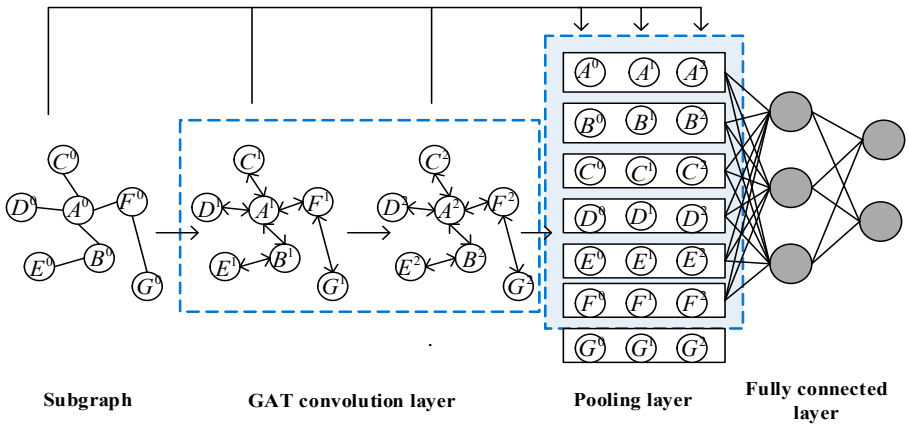

Figure 6. Schematic diagram of PLGAT algorithm

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0797.v1

Compared with the PLAS model, PLGAT takes the subgraph features further processing with GAT. Firstly, the h-hop subgraph of the target node pair was extracted. Then, the subgraph aggregates the information of the neighbor nodes through the GAT convolution layer, and combines the original node features with the new features obtained through the GAT convolution layer to get the new feature representation of the node. Finally, through a pooling layer, nodes in the subgraph are sorted according to specific sorting rules, and the feature of the first K nodes are selected as the subgraph encoding, which is sent to the fully connection layer for classification.

4.2.1. GAT convolution layer

Graph Attention Network (GAT) takes a set of node features $h = \{\vec{h}_1, \vec{h}_2, \dots, \vec{h}_N\}$ as input, where $\vec{h}_i \in \mathbb{R}^F$,N represents the number of nodes, and F indicates the dimension of the node's original features. It generates a new set of features $h' = \{\vec{h}'_1, \vec{h}'_2, \dots, \vec{h}'_N\}$ through the graph attention layer, where $\vec{h}'_i \in \mathbb{R}^{F'}$, F' indicates the dimension of the node's new feature vector.

In order to obtain sufficient expression information, GAT needs to perform at least one learnable linear transformation on the features of input nodes to obtain new node features. Therefore, the algorithm trains a weight matrix w for all nodes, $w \in \mathbb{R}^{F' \times F}$, this weight matrix is the relationship between input features and output features. For each node, a single-layer feed-forward neural network is used to calculate the self-attention coefficient between nodes. The attention coefficient is expressed as:

$$e_{ij} = a(W\vec{h}_i, W\vec{h}_j) \tag{7}$$

a is a single-layer feed forward neural network, and e_{ij} represents the importance of node j to node i. Typically, each node computes the attention coefficient with any other node, but this results in loss of graph structure information. In order to obtain the graph structure features, GAT introduces the masked attention mechanism, and distributes attention to the set of neighbor nodes N_i of node i, and $j \in N_i$. At the same time, GAT uses the SoftMax function to regularize all neighbor nodes j of node i, and adds a LeakyReLU nonlinear function to the output of the feed forward neural network. Therefore, the formula of the complete attention mechanism is as follows:

$$a_{ij} = \frac{exp(\,LeakyReLu(\,\vec{a}^T[W\vec{h}_i || W\vec{h}_j]))}{\sum k \in N_i\, exp(\,LeakyReLu(\,\vec{a}^T[Wh_i || W\vec{h}_k]))} \tag{8}$$

a is a single-layer feedforward neural network, and $\vec{a}$ is the weight matrix between layers in the neural network.

After obtaining the attention coefficient between different nodes through the above formula, the final output characteristics of the node are obtained by aggregating the characteristics of neighboring nodes. The formula is as follows:

$$\vec{h}'_i = \sigma(\sum_{j \in N_i} a_{ij} W\vec{h}_j) \tag{9}$$

σ is a non-linear activation function.

In order to stabilize the learning process of self-attention, GAT uses a multi-head attention mechanism. Specifically, it uses k independent self-attention mechanisms to splice the obtained new features. GAT uses k average instead of splicing operations. Therefore, the final formula of the multi-head attention mechanism is:

$$\vec{h}'_i = \sigma(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in N_i} a_{ij}^k W^k \vec{h}_j) \tag{10}$$

k represents a total number of attention mechanisms, a^k represents the k^{th} attention coefficient, and W^k represents the linear transformation weight matrix under the k^{th} attention mechanism.

 doi:10.20944/preprints202305.0797.v1

In the task of deep learning, it is generally believed that the higher the complexity of the model, the higher the fitting effect. For the graph neural network, whether to add more convolutional layers to improve the model's performance should be decided according to the specific situation of the task. Increasing the depth of the convolutional layer increases the aggregation radius, but when the convolutional layers reach a certain depth, almost every node in the graph contains global graph information. Although a large number of convolutional layers can bring global information beneficial to graph classification tasks, it increases the complexity of the model, resulting in a long training time and model overfitting. Since the category of a node is primarily affected by local neighbor nodes, a small number of convolutional layers can achieve better performance, so we only extract the target node to the neighbor nodes within two hops to form a subgraph, and only use two convolutional layers to make the target node aggregate the global graph information, which improves the accuracy of the algorithm without bringing the complexity of the model.

We use the node features obtained in the subgraph encoding part of PLAS as the original input features F^0. The original feature F^0 is input into the GAT convolution layer for calculation, the feature F^1 is obtained through the first layer of GAT convolution, and the feature F^2 is obtained through the second layer of GAT convolution. We concatenate the three feature to obtain the final feature representation of the node as follows:

$$F = F^0||F^1||F^2 \tag{11}$$

Among them, $||$ represents the connection operation, and the final feature representation of each node obtained through two GAT convolutional layers contains almost the global information of the subgraph.

4.2.2. Pooling layer

The pooling operation of the graph is used to aggregate the global node information to obtain a global graph vector representation. The pooling methods of graph data are mainly divided into two categories, one is hierarchical pooling [25], the other is global pooling [26]. The hierarchical pooling method mainly includes two processes: 1. Calculate the assignment matrix to determines which neighbor nodes are allocated together to form a cluster; 2. The nodes in each cluster are aggregated into a new super node in a certain aggregation way, and then these super nodes are used to form a new graph. F For large-scale graphs, this hierarchical pooling method is very effective. The scale of the graph is reduced through continuous convolution and pooling operations to obtain the final representation of the entire graph. The global pooling method selects appropriate nodes to form a representation of the entire graph after multiple convolutions, which minimizes the loss of information. Therefore, in small-scale graph, global pooling method is superior to hierarchical pooling method. We extract the h-hop neighbor nodes of the target node pair to form the subgraph, which mainly has the following two characteristics:

1. The subgraph is centered on the target node pair and spreads out to the neighbor nodes on both sides, which has strong directionality.
2. The number of neighbor hops in the subgraph is limited, resulting in the subgraph often being a small-scale graph.

For small-scale graphs, using the global pooling method is more appropriate. However, it is difficult for this global pooling method to show the directionality of the subgraph, and it is challenging to select appropriate nodes with uniform rules to form a subgraph feature representation. Based on the global pooling method, we use the graph labeling algorithm in PLAS to sort the nodes in the sub-graphs by a specific sorting rule to solve the above problems. In graph labeling algorithm, the label string (d_i, d_j) consists of two parts, one is the shortest distance d_i from the target node i, and the other is the shortest distance d_j from the target node j. For the target node i, the label character is always (0,1), and for the target node j, the label string is always (1,0). The nodes in the subgraph are first sorted by d_i, and then sorted by d_j if d_i is equal. Sorting in this rule has two main benefits:

 doi:10.20944/preprints202305.0797.v1

1. Because the shortest distance d_i from other nodes in the subgraph to the target node pair (i, j), d_j is greater than or equal to 1, the target node pair (i, j) is always ranked in the first two elements, and the node closer to the target node in the subgraph will be ranked in the front, and the node farther away from the target node will be ranked in the back, which reflects the direction of the subgraph.
2. Nodes in the subgraph will be sorted according to consistent rules. We only need to select the first K nodes to represent the feature expression of the current subgraph, to unify the rules of node selection in the global pooling method. For the case that the number of nodes in the subgraph is greater than K, only nodes after K need to be truncated, and the discarded nodes will not lose all node information, because other nodes in the subgraph have learned the discarded node information through the two layers of GAT convolution layer. If the number of nodes in the subgraph is less than K, simply add virtual nodes, where virtual nodes are represented by the zero vector.

4.3.3. Experimental dataset and settings

We carried out experiments on seven real data sets: USAir[15], NS[17], PB[17], Yeast[18], Cele[19], Power[19] and Router[20], and evaluated the performance of the algorithm with the AUC.

To evaluate our algorithm on real 5G MEC access network environment and real AS topology of Global Internet, we test our algorithm on Bupt5GMEC, CM1-5G, CM2-5G and ITDK0304S network datasets.

For each data set, all the linked node pairs in the current network structure are selected as positive samples, and then an equal number of unlinked node pairs are randomly selected as negative samples, which balances the positive and negative samples. And 80% of the positive samples and 10% of the negative samples are randomly selected as the training set, 10% as the verification set, and the remaining 10% as the test set. The number of neurons in the first hidden layer of the fully connected layer is set to 1024, the number of neurons in the second hidden layer is set to 512, and the number of neurons in the output layer is set to 2.

4.3.4. Experiment results analysis

We use the AUC as evaluation of algorithm, during the training process, we use 50 epochs to train our model, save the model that performs best on the verification set, and use it for prediction on the test set. We repeat the above operation ten times and take the average of the ten experimental results as the final result.

Firstly, we compare with five heuristic based link prediction algorithms, and the experimental results are shown in Table 4:

Table 4. Comparison of PLGAT with five heuristic link prediction algorithms (AUC)

Dataset	CN	JC	AA	RA	Katz	**PLAS**	**PLGAT**
USAir	94.02	89.81	95.08	**95.67**	92.90	88.81	**95.21**
NS	93.94	94.40	**94.77**	94.33	95.03	93.28	**97.96**
PB	92.07	87.39	92.31	92.45	92.89	**98.93**	**94.32**
Yeast	89.41	89.30	89.32	89.44	92.30	**93.37**	**94.41**
Cele	84.95	80.18	87.03	87.49	86.45	**95.04**	**89.52**
Power	58.90	58.80	58.83	58,83	65.90	**87.84**	**79.27**
Router	56.57	56.38	56.40	56.38	38.62	**74.27**	**91.42**
Bupt5GMEC	75.01	75.02	68.75	75.02	23.78	**99.67**	**99.67**
CM1-5G	54.23	53.79	53.32	54.13	50.22	**69.84**	**66.43**
CM2-5G	76.15	72.22	74.79	76.21	50.71	**88.95**	**91.34**
ITDK0304S	85.41	85.41	83.60	85.42	48.77	**98.90**	**97.88**

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0797.v1

As shown in Table 4, Figure 7 and Figure 8, our method PLAS and PLGAT are better than the heuristic link prediction algorithm in the NS, PB, Yeast, Cele, Power, Router, Bupt5GMEC, CM1-5G, CM2-5G and ITDK0304S datasets. Especially in the data set with sparse Power, Router and Bupt5GMEC communication network, our methods are superior to the heuristic link prediction algorithm obviously while the AUC index has increased by about 15% and 30% respectively. On the USAir dataset, our method performs as well as the heuristic link prediction algorithm, which is slightly lower than RA algorithm and better than other algorithms.

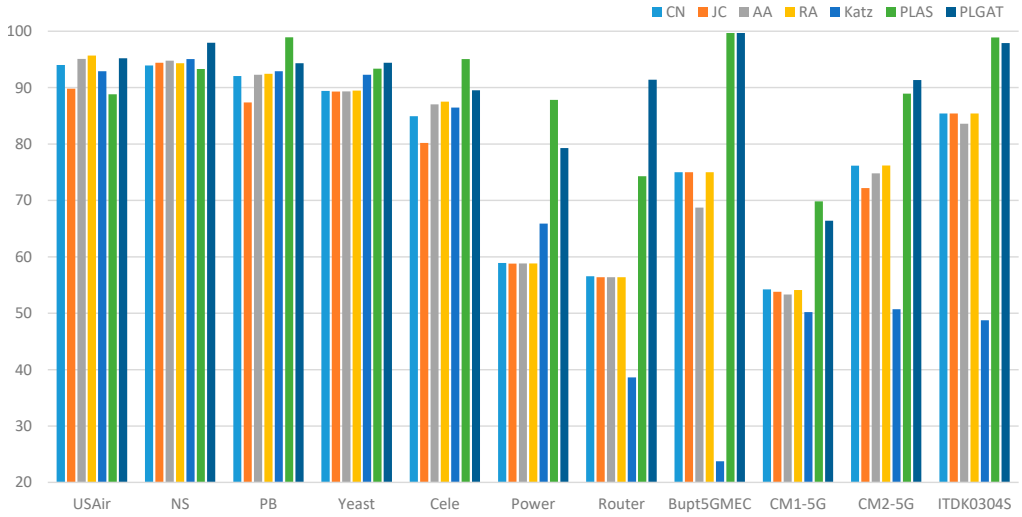

Figure 7. Comparison of PLGAT with five heuristic link prediction algorithms (AUC)-a

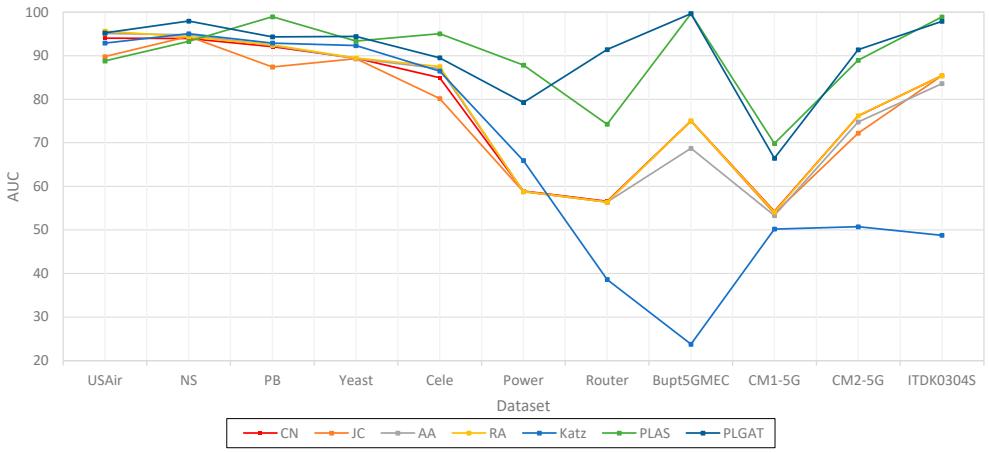

Figure 8. Comparison of PLGAT with five heuristic link prediction algorithms (AUC)-b.

Then PLGAT is compared with three link prediction algorithms based on graph representation learning and the PLAS algorithms, and the results are shown in Table 5:

Table 5. Comparison of PLGAT with three link prediction algorithms based on graph representation (AUC)

Dataset	Node2Vec	LINE	VGAE	**PLAS**	**PLGAT**
USAir	91.40	81.47	89.30	93.28	**95.21**
NS	91.55	80.63	94.04	**98.93**	97.96
PB	85.79	76.94	90.70	93.37	**94.32**
Yeast	93.68	87.45	93.87	**95.04**	94.41
Cele	84.13	69.22	81.87	87.84	**89.52**
Power	76.23	55.64	71.20	74.27	**79.27**

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0797.v1

Router	65.46	67.17	61.53	88.81	**91.42**
Bupt5GMEC	85.61	83.42	87.56	**99.67**	**99.67**
CM1-5G	53.22	56.75	60.34	**69.84**	66.43
CM2-5G	81.24	84.31	83.76	88.95	**91.34**
ITDK0304S	85.87	86.56	87.35	**98.90**	97.88

As shown in Table 5 and Figure 9, compared with PLAS, both algorithms convert link prediction into graph classification tasks. The PLGAT algorithm proposed in this chapter is superior to PLAS algorithm in USAir, PB, Cele, Power, Router, Bupt5GMEC, CM1-5G, CM2-5G and ITDK0304S datasets. In Power data set, our algorithm improves by more than 5%. In terms of Yeast data, the performance of our algorithm was inferior to that of the PLAS algorithm, but the difference was within 1%.

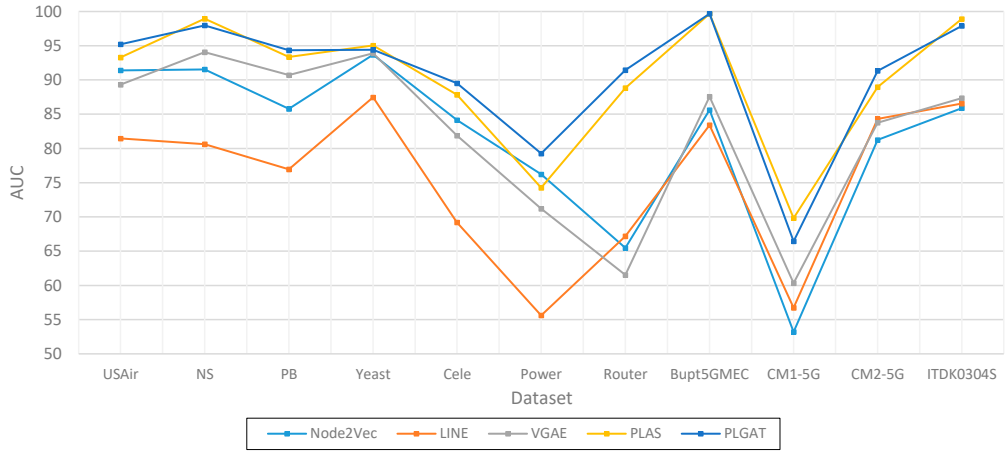

Figure 9. Graphic Comparison of PLGAT with three link prediction algorithms based on graph representation (AUC)

Compared with VGAE algorithm, both algorithms use GNN (Graph Neural Network) to learn neighbor node information. Our algorithm is superior to VGAE algorithm in USAir, NS, PB, Yeast, Cele, Power, Router, Bupt5GMEC, CM1-5G, CM2-5G and ITDK0304S datasets. On the Router data set, our algorithm improved by more than 30%. For the LINE and Node2Vec algorithms, our algorithm outperforms them in all respects. For the heuristic link prediction algorithm, the performance of our algorithm is lower than that of RA algorithm only on USAir data set, and on other data sets, the performance of our algorithm is far better than that of the heuristic link prediction algorithm.

5. Conclusions

Link prediction is a hot research field at present. It is very important for mining and analyzing the evolution of networks. Although the heuristic algorithm based on node similarity is simple and effective, it cannot be applied to all network structures. Finding effective heuristic indexes for different network structures requires a process of repeated experiments. Based on the above problems, we design a link prediction algorithm based on target node pair subgraph, which combines the characteristics of graph structure and node characteristics, and can play a role in different network structures. Finally, we compare five heuristic link prediction algorithms based on node similarity and three link prediction algorithms based on graph embedding on eleven real data sets. The results show that the performance of our algorithm is much better than other link prediction algorithms.

Author Contributions: XiaoLong.Deng and JunWen.Lu contribute methodology and formal analysis, JuFeng.Sun contributes software and validation.

Acknowledgments: Thanks to the 173 Basic Foundation Reinforcement Project of China (No. JSLY-A12017) in network analysis, China National Key Research and Development Project (No. 2021YFB3101900), and Key

Technology Project of Shenzhen city with the name 2021 Research and development of key edge computing gateway technology based on 5G (No. JSGG20201103092802010) and project director Zhou Yong(mario@jcg.com.cn).

References

1. Lü, L. Link Prediction on Complex Networks. *Journal of University of Electronic Science and Technology of China* **2010**, *39*, 651-661.
2. Lü, L.; Zhou, T. *Link Prediction*, 1st ed.; Higher Education Press: Beijing, China, 2013.
3. Adamic; Lada, A et al. Friends and neighbors on the Web. *Social Networks* **2003**, *25*, 211-230.
4. Chen, H.; Li, X.; Huang, Z. Link prediction approach to collaborative filtering. In Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'05), Denver CO, USA, 7-11 June 2005.
5. Cannistraci, C.V.; Alanis-Lobato, G.; Ravasi, T. From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. *Scientific reports* **2013**, *3*, 1-14.
6. Nickel, M.; Murphy, K.; Tresp, V.; Gabrilovich, E. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE* **2015**, 104, 11-33.
7. LIU, W.; CHEN, L. Link prediction in complex networks. *Information and Control* **2020**, *49*, 1-23
8. Zhang, M.; Chen, Y. Link prediction based on graph neural networks. *Advances in Neural Information Processing Systems* **2018**, *31*, 5165-5175.
9. Newman, M.E.J. Clustering and preferential attachment in growing networks. *Physical review E* **2001**, *64*, 025102.
10. Jaccard, P. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bull Soc Vaudoise Sci Nat* **1901**, 37, 547-579.
11. Zhou, T.; Lü, L.; Zhang, Y.C. Predicting missing links via local information. *The European Physical Journal B* **2009**, *71*, 623-630.
12. Kovács, I.A.; Luck, K.; Spirohn, K. et al. Network-based prediction of protein interactions. *bioRxiv* **2019**, *10*, 1240.
13. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco California, USA, 13-17 August 2016.
14. Leman, A.A.; Weisfeiler, B. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya* **1968**, *2*, 12-16.
15. Batagelj, V; Mrvar, A. Pajek. datasets http://vlado. fmf. uni-lj. si/pub/networks/data/mix. USAir97. net, 2006.
16. Newman, M.E.J. Finding community structure in networks using the eigenvectors of matrices. *Physical review E* **2006**, *74*, 036104.
17. Ackland, R. Mapping the US political blogosphere: Are conservative bloggers more prominent? In Proceedings of BlogTalk Downunder 2005 Conference, Sydney, Australia, 20-21 May 2005.
18. Von Mering, C.; Krause, R.; Snel, B. et al. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature* **2002**, *417*, 399-403.
19. Watts, D.J.; Strogatz, S.H. Collective dynamics of 'small-world' networks. *Nature* **1998**, *393*, 440-442.
20. Spring, N.; Mahajan, R.; Wetherall, D. Measuring ISP topologies with Rocketfuel. *ACM SIGCOMM Computer Communication Review* **2002**, *32*, 133-145.
21. Barabási, A.L.; Albert, R. Emergence of scaling in random networks. *science* **1999**, *286*, 509-512.
22. Katz, L. A new status index derived from sociometric analysis. *Psychometrika* **1953**, *18*, 39-43.
23. Tang, J.; Qu, M.; Wang, M. et al. Line: Large-scale information network embedding. In Proceedings of the 24th international conference on world wide web, Florence, Italy, 18-22 May 2015.
24. Kipf, T.N.; Welling, M. Variational graph auto-encoders. *Computing Research Repository* **2016**, *1611*, 07308.
25. Ma, Y.; Wang, S.; Aggarwal, C.C. et al. Graph convolutional networks with eigenpooling. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage AK, USA, 4 – 8 August, 2019.
26. Lee, J.; Lee, I.; Kang, J. Self-attention graph pooling. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, USA, 9-15 June 2019.
27. Guoqiang, Z. An Algorithm for Internet AS Graph Betweenness Centrality Based on Backtrack. *Journal of Computer Research and Development* **2006**, *43*, 114-120.
28. Petar, V.; Guillem, C.; Arantxa, C.; Adriana, R.; Pietro, L.; Yoshua, B. Graph Attention Networks. International Conference on Learning Representations, 2018.
29. Hamad, A. A.; Abdulridha, M. M.; Kadhim, N. M.; Pushparaj, S.; Meenakshi, R.; Ibrahim, A. M. Learning methods of business intelligence and group related diagnostics on patient management by using artificial dynamic system. *Journal of Nanomaterials* **2022**, 1-8.
30. Tao, W.; Hongyu, M.; Chao, W.; Shaojie, Q.; Liang, Z.; Shui, Y. Heterogeneous representation learning and matching for few-shot relation prediction. *ACM Journal of Experimental Algorithmics*, **2022**, *131*, 108830.

31. Xingping, X.; Tao, W.; Shaojie, Q.; Xi-Zhao, W.; Wei, W.; Yanbing, L. NetSRE: Link predictability measuring and regulating, *Knowledge-based systems*, **2020**, *196*, 105800.
32. Christos, F.; Kevin S., M.; Andrew, T. Fast discovery of connection subgraphs. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining* **2004**, 118-127.
33. Hasan, M. A.; Chaoji, V.; Salem, S.; Zaki, M. Link Prediction Using Supervised Learning. Proceedings of SDM'06 Workshop on Link Analysis, Counter terrorism and Security, 2006.
34. David, L., & Jon M., K. The link-prediction problem for social networks. *Journal of the Association for Information Science and Technology* **2007**, *58*, 1019-1031.
35. Nesserine, B.; Rushed, K.; Celine, R. Supervised Machine Learning Applied to Link Prediction in Bipartite Social Networks. Advances in Social Networks Analysis and Mining, 2010, 326-330E-ISBN:978-0-7695-4138-9.
36. Michael, F.; Lena, T.; Rami, P.; Ofrit, L.; Lior, R.; Yuval, E. Computationally efficient link prediction in a variety of social networks. *ACM Transactions on Intelligent Systems and Technology* **2014**, *5*, 1-25.
37. Yu, Z.; Ke-ning, G.; Ge, Y. Method of Link Prediction in Social Networks Using Node Attribute Information. *Computer Science* **2018**, *45*, 41-45.
38. Haris, M.; Miroslav, M.; Sasho, G.; Igor, M. Multilayer Link Prediction In Online Social Networks. Telecommunications Forum. 2018, 823-826.
39. Sanjay, K.; Abhishek, M.; B. S., P. Link prediction in complex networks using node centrality and light gradient boosting machine. *World Wide Web* **2022**, *25*, 2487-2513.
40. Smriti, B.; Graham, C.; S.; M. Node Classification In Social Networks. *Computing Research Repository* **2011**, *1101*, 115-148.
41. Yu, R.; Wenbing, H.; Tingyang, X.; Junzhou, H. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. International Conference on Learning Representations, 2020.
42. Muhan, Z.; Zhicheng, C.; Marion, N.; Yixin, C. An End-to-End Deep Learning Architecture for Graph Classification. *AAAI Conference on Artificial Intelligence* **2018**, 4438-4445.
43. John Boaz, L.; Ryan, R.; Xiangnan, K. Graph Classification using Structural Attention. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining* **2018**, 1666-1674.
44. Víctor, M.; Fernando, B.; Juan Carlos Cubero, T. A Survey of Link Prediction in Complex Networks. *ACM Computing Surveys* **2017**, *49*, 69:1-69:33.
45. Bryan, P.; Rami, A.; Steven, S. DeepWalk: online learning of social representations. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining* **2014**, *1403*, 701-710.
46. DAIXIN, W.; Peng, C.; Wenwu, Z. Structural Deep Network Embedding. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining* **2016**, 1225-1234.
47. Shaosheng, C.; Wei, L.; Qiongkai, X. Deep Neural Networks for Learning Graph Representations. *AAAI Conference on Artificial Intelligence* **2016**, 1145-1152.
48. Thomas N., K.; Max, W. Semi-Supervised Classification with Graph Convolutional Networks. *International Conference on Learning Representations* **2017**, 1609.02907.
49. William L.; H.; Rex, Y.; Jure, L. Inductive Representation Learning on Large Graphs. *Conference on Neural Information Processing Systems* **2017**, *30*, 1024-1034.
50. Haochen, C., Bryan, P., Yifan, H., & Steven, S. HARP: Hierarchical Representation Learning for Networks. *AAAI Conference on Artificial Intelligence* **2018**, *1706*, 2127-2134.
51. Michael, S.; Thomas N.; K.; Peter, B.; Rianne van den, B.; Ivan, T.; Max, W. Modeling Relational Data with Graph Convolutional Networks. *Extended Semantic Web Conference* **2018**, 1703.06103.
52. Sam De, W.; Tim, D.; Sandra, M.; Bart, B.; Jochen De, W. Combining Temporal Aspects of Dynamic Networks with Node2Vec for a more Efficient Dynamic Link Prediction. *Advances in Social Networks Analysis and Mining* **2018**, 1234-1241.
53. Kai, L.; Meng, Q.; Bo, B.; Gong, Z.; Min, Y. GCN-GAN: A Non-linear Temporal Link Prediction Model for Weighted Dynamic Networks. *IEEE International Conference Computer and Communications* **2019**, *1901*, 388-396.
54. Muhan, Z.; Yixin, C. Weisfeiler-Lehman Neural Machine for Link Prediction. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining* **2017**, 575-583.
55. Newman M E, J. Fast algorithm for detecting community structure in networks. *Physical Review E* **2004**, *69*.