International Journal of Innovative Computing, Information and Control
Volume **20**, Number **6**, December **2024**

ICIC International ©2024 ISSN 1349-4198
pp. **1803–1817**

DYNAMIC NETWORK REPRESENTATION LEARNING METHOD COMBINING GRAPH NEURAL NETWORKS AND TEMPORAL INFORMATION

Zhixiao Li

Department of Mathematics and Statistics
Cangzhou Normal University
No. 16, Guofeng South Avenue, Yunhe District, Cangzhou 061000, P. R. China
lzx2014@caztc.edu.cn

Received January 2024; revised May 2024

Abstract. *Network data plays an increasingly important role in various fields such as social life, economy, and scientific research. However, traditional network representation learning methods often overlook temporal information in the network. To address the above issues, a converter network is used to effectively associate node representations at different times. A temporal heterogeneous information network is designed. Finally, the output data of the two models are combined for dynamic network representation. The experimental results showed that the F1 values of this model were significantly higher than those of other models on five different datasets, reaching 94.67%, 93.45%, 89.43%, 89.47%, and 95.53%, respectively. After training, the model error gradually decreased and converged to around 0.35 and 0.30. In summary, the dynamic network representation method proposed in the study can better understand and predict the behavior of dynamic networks. Furthermore, it can be better applied to various network analysis tasks, such as node classification, link prediction, and community discovery.*
Keywords: Graph neural network, Temporal heterogeneous information network, Dynamic network representation, Converter network, Temporal information

1. **Introduction.** With the advent of the digital age, online data is showing explosive growth. How to effectively represent and learn information from network data has become an important research question. Traditional network representation learning methods usually handle network data statically, ignoring the dynamic and temporal information of network data. However, in real-world networks, node states and edge connectivity relationships often change over time. This dynamic and temporal information is crucial for accurately understanding and predicting network behavior [1]. Therefore, how to effectively capture and learn this dynamic and temporal information has become an urgent problem in the current field of network representation learning. In recent years, Graph Neural Network (GNN) has received widespread attention and research as a powerful network representation learning tool. GNN can update the node representation by aggregating the information of neighboring nodes, thereby capturing the topology and node feature information of the network, and learning the low dimensional dense representation of the network through deep learning. However, traditional GNN models are usually static and lack effective processing mechanisms for temporal information and structural changes in dynamic networks, which limits the application of GNN in dynamic network representation learning [2,3]. In response to the above issues, a Transformer network is used to effectively associate node representations at different times. A Metapath-Hierarchical Attention Based Temporal Heterogeneous Information Network Representation Learning

DOI: 10.24507/ijicic.20.06.1803

(MATHNRL) method is constructed. The output results of the GraphSAGE network and the MATHNRL network are combined in an average weighted manner. A GraphSAGE MATHNRL network that combines the advantages of GraphSAGE network and MATHNRL network is designed. The research is divided into four parts. The second part introduces the relevant research on using graph neural networks and dynamic network representations both domestically and internationally. The third part proposes a new dynamic network representation method combining the proposed GraphSAGE network and MATHNRL network. The fourth part validates the effectiveness and applicability of the designed GraphSAGE-MATHNRL network. Finally, the fifth part concludes the paper.

2. **Related Work.** As an effective deep learning model, graph neural networks have received increasing attention from researchers in recent years. Chae et al. proposed an accident diagnosis algorithm based on graph neural networks to reduce human errors and support operator decision-making safety issues in nuclear power plants. This security diagnosis algorithm combined convolutional neural networks. Compared with traditional convolutional neural networks, this method improved diagnostic accuracy by 23.1% [4]. Zhao et al. proposed a new framework HGSL to address the noise or missing heterogeneous graphs in reality. The optimal structure and GNN parameters of heterogeneous graphs were added to achieve classification tasks. The experimental results showed that this method significantly reduced the incidence of heterogeneous image storage noise or missing information [5]. Zhu et al. proposed the Neural Bellman-Ford Graph Network (NBFNet) in conjunction with the Bellman-Ford algorithm to address the high complexity of link prediction on graphs. The experimental results showed that NBFNet outperformed existing methods in various settings. Compared to traditional GNN, the accuracy had increased by 12.3% [6]. Li et al. have designed a traffic flow prediction method based on dynamic graph convolutional recursive networks to more accurately process data in transportation networks. The method utilizes and extracts dynamic features from node attributes through design, and the results show that the method consistently outperforms baseline levels on different datasets [7]. Bhattacharya et al. have designed a deep learning framework called DeepInfNode based on deep graph convolutional neural networks to detect influential nodes in complex social networks. It can identify nodes with structural centrality in the network. Experimental results show that the accuracy of this method is as high as 99.1%, significantly better than existing state-of-the-art methods [8].

Wang et al. proposed a self supervised learning model based on graph neural networks to address the expensive and time-consuming labeled data in molecular machine learning. This method could significantly improve the accuracy of molecular attribute prediction. Time consumption was reduced by 14.56% [9]. Deng and Hooi proposed a method combining structural learning and graph neural networks for anomaly event detection in high-dimensional time series data. Meanwhile, attention weights were used to explain detected anomalies. The experimental results showed that this method was more accurate in detecting anomalies, accurately capturing the correlation between sensors than the benchmark method [10]. Huo et al. designed a distillation based graph neural network model to effectively handle incomplete graph data, which was refined through feature level and structure level learning models. The results showed that the model had good robustness in processing incomplete graph data on 8 benchmark datasets [11]. There was a data loading bottleneck when GPU trained large-scale graph neural networks. Therefore, Bai et al. utilized the structural information of the graph and data access patterns to implement effective caching strategies. The experimental results showed that PaGraph could significantly reduce data loading time and improve overall training performance compared to existing methods in a single server multi GPU environment [12]. Wang et al. designed a

deep learning framework based on graph convolutional graph neural networks to address the poor quality of graphic layouts generated by clothing design systems. Compromise multiple predefined aesthetic standards were used to generate layouts. The experimental results showed that it could effectively draw any shape while flexibly adapting to different aesthetic standards [13].

In summary, graph neural networks have sufficient theoretical and practical foundations in the application of dynamic network representation. Although related research has effectively improved the performance of processing network data, these methods mainly focus on static networks and ignore temporal dynamic features. Processing dynamic networks is difficult to accurately capture network data node features. To this end, a new GraphSAGE-MATHNRL network is designed by combining graph neural networks with temporal heterogeneous information networks to analyze dynamic network characteristics more comprehensively.

3. A Dynamic Network Representation Learning Method Based on Graph Neural Networks and Temporal Information.

This chapter is divided into two sections. The first section is based on the GraphSAGE network for dynamic network representation, and a series of optimizations are made to the algorithm. The second section mainly combines temporal heterogeneity information to design a GraphSAGE-MATHNRL dynamic network representation model.

3.1. Dynamic network representation based on graph neural network.

A dynamic network refers to a network where the connectivity between nodes is constantly changing over time. Specifically, a dynamic network can consist of a series of static network snapshots. Each snapshot represents the network structure at a certain point in time. All the snapshots form a continuous time series [14-16]. In dynamic networks, the addition and deletion of nodes and edges, edge weights, node labels, text, attributes, and other additional information can all change over time. At present, there are two main trends in academic research on dynamic networks. One approach is to view the original dynamic graph as a whole. All edges have a time attribute that represents the establishment time [17]. Another approach is to divide the original dynamic graph into a series of subgraphs with equal time intervals. Each subgraph represents a network snapshot at a certain time, which can be regarded as a static network. The schematic diagrams of the two networks are shown in Figure 1.

GraphSAGE network is an unsupervised graph neural network used to learn node representations from graph data. It learns node representations by transmitting information between neighboring nodes, thereby capturing similarities and differences between nodes. Compared to existing results, the GraphSAGE network can generate embeddings for unseen vertices. In addition, the performance of node classification and link prediction problems is also quite prominent. The GraphSAGE network consists of three main components, namely the convolutional layer, pooling layer, and fully connected layer. The convolutional layer is responsible for processing the input graph data. Based on convolution operation, the features of nodes and their neighboring nodes are aggregated to obtain the representation of each node. The pooling layer down samples the output of the convolutional layer to reduce the number of nodes and obtain the core representation of each node. Finally, the fully connected layer maps the output of the pooling layer to the final node representation. The training process is shown in Figure 2.

In addition, the GraphSAGE network can also handle larger graphs, effectively representing node and relationship features. Existing node information is used to represent new nodes, handle heterogeneous and directed graphs, and effectively handle dynamic

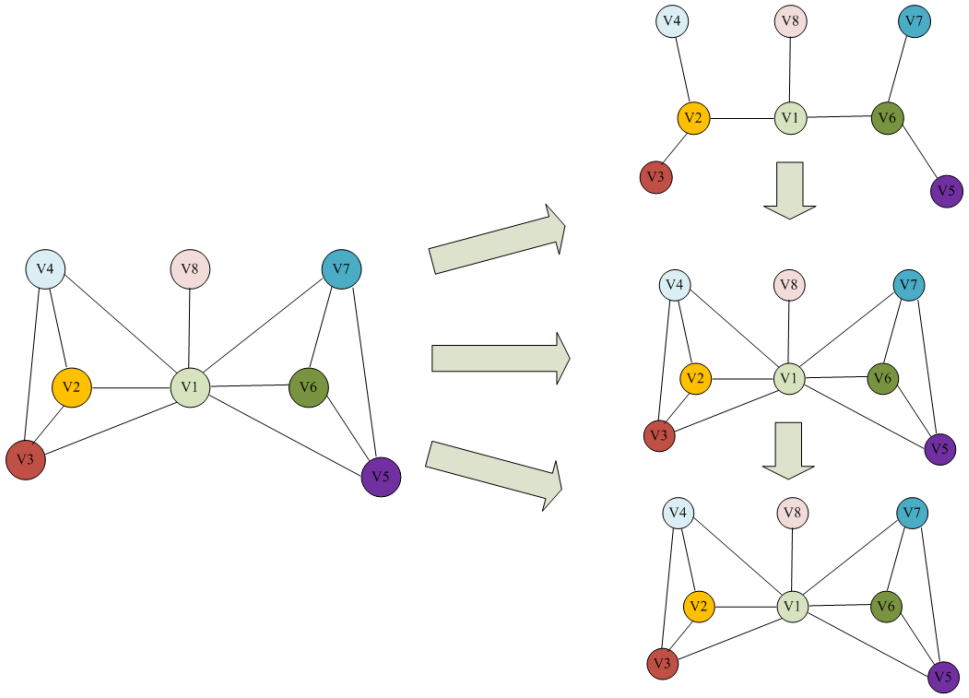


FIGURE 1. Structure diagram of two types of networks

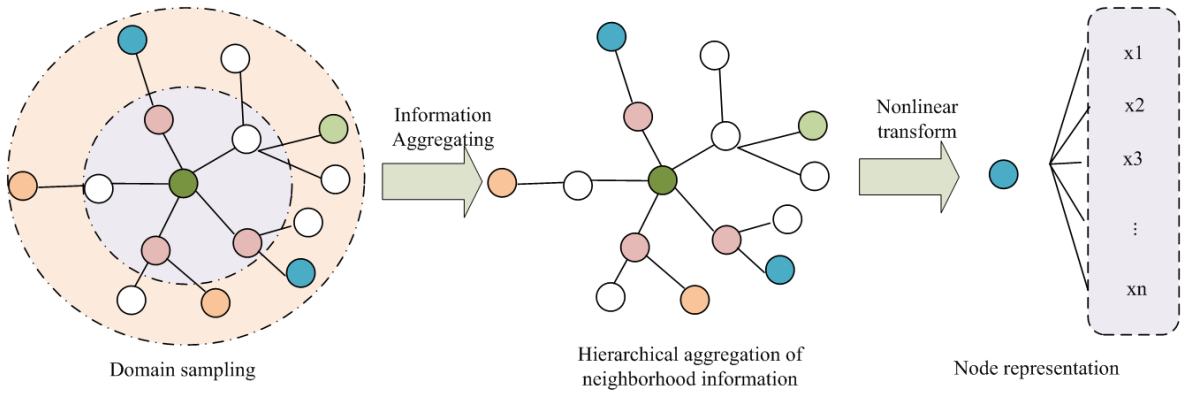


FIGURE 2. Schematic diagram of the training process of GraphSAGE network

graphs. When cutting dynamic graphs, a larger time span is usually chosen, which results in multiple edges between node pairs in the subgraph. To accurately capture this dynamic change, the GraphSAGE network has been extensively optimized to better extract node features and vector representations. A new weighted average aggregation function is used to measure the contribution of neighbors. The aggregation form of any node in the graph is shown in Equation (1).

$$L^k(v) = \sigma\left(W \cdot \left(Weight\_average(\Phi(v))concatL^{k-1}(v)\right)\right) \tag{1}$$

In Equation (1), *concat* represents the concatenation operation of vectors. v represents any node in the graph. $\Phi(v)$ represents the set of neighboring nodes of v. The average weigh value is shown in Equation (2).

$$Weight\_average(\Phi(v)) = \sum_{i \in \Phi(v)} a_{iv} \cdot L^{k-1}(i) \tag{2}$$

The a_{iv} in Equation (2) is shown in Equation (3).

$$a_{iv} = \frac{frequency(iv)}{\sum_{j \in N(v)} frequency(jv)} \tag{3}$$

In Equation (3), $frequency(iv)$ represents the current number of connections between two nodes. When training multiple GraphSAGE networks simultaneously, each network needs to independently calculate and update parameters. This leads to significant computing resources consumption, especially when the network size is large. Because each network requires a large amount of matrix operations and gradient updates, it consumes a large amount of computing resources. To address the computational resource consumption and inconsistent node embedding space, the parameter sharing concept is adopted. The GraphSAGE algorithm shares the same weights and bias parameters on all subgraphs, reducing the computing resources. Ensure that nodes at different times are embedded in the same representation space to better capture the evolution process of the network. The GraphSAGE algorithm obtains low dimensional embedded representations of each node in the graph. However, it is worth noting that the representations of the same node at different times are independently distributed, which means that their vector representations are mathematically unrelated. Therefore, how to associate these independently distributed vectors to obtain a more comprehensive and accurate node representation has become a key issue that urgently needs to be solved. The Transformer model is a machine learning model used for Natural Language Processing (NLP) and other sequence tasks. Unlike traditional Recurrent Neural Network (RNN), the Transformer model can process input sequences in parallel, making it more efficient during training and inference stages [18,19]. When processing sequence data, the Transformer network can capture long-term dependencies within the sequence. Therefore, it can effectively associate node representations at different times. The schematic diagram of the Transformer network structure is shown in Figure 3.

The same node on the timeline is analogized as each word in a sentence. Its representation at all times constitutes long sequence data. The Transformer attention mechanism includes two parts: self attention and multi head attention [20]. The self attention mechanism is used to calculate the attention weights between nodes to determine the correlation between different nodes. The traditional self attention mechanism calculates the similarity score between queries and keys. These scores are applied to the value matrix to generate a weighted sum. To reduce the model's dependence on information, Transformer introduces a multi head attention mechanism. Each input is mapped to a query vector P. The key vector is U. Then perform a normalized point multiplication operation on the query vector and the key vector, as shown in Equation (4).

$$P \cdot U = |P| * |U| * \cos\theta \tag{4}$$

In Equation (4), θ represents the angle between two vectors. Then perform attention calculation, as shown in Equation (5).

$$Attention(P, U, V) = \text{softmax}\left(\frac{Q^T U}{\sqrt{d_u}}\right) V \tag{5}$$

In Equation (5), V represents the value vector. Then the input P, U, V is split into headers. The dimensions of the query vector and key vector in each header are shown in Equation (6).

$$d_u = d_v = \frac{m+n}{h} \tag{6}$$

In Equation (6), h represents that the model is divided into h attention heads. Afterwards, multiple attention results are connected to obtain the multi head attention operation

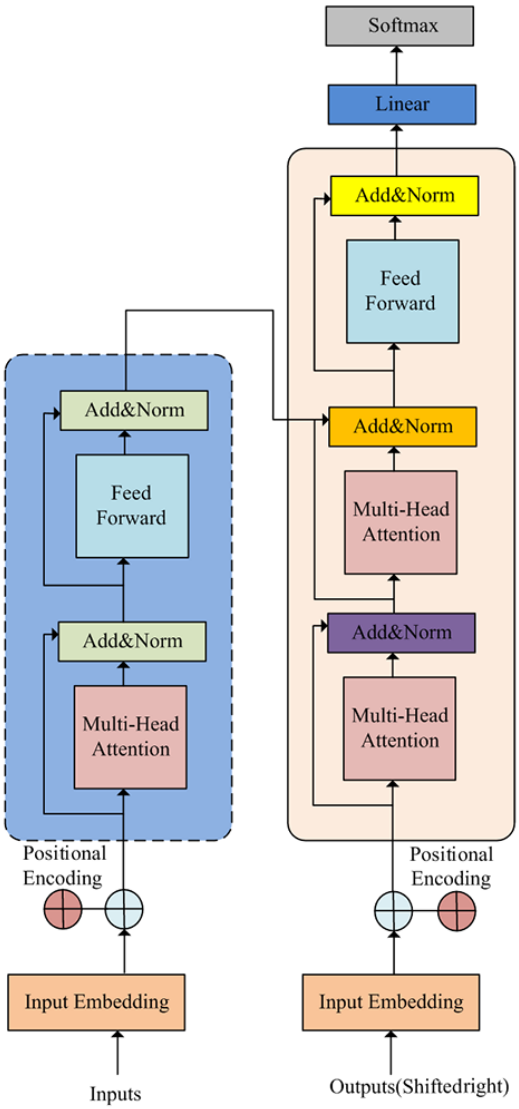


FIGURE 3. Schematic diagram of Transformer model structure

result, as shown in Equation (7).

$$Multi\text{-}Attention(P, U, V) = Concat(head_1, head_2, \ldots, head_h)W^0 \tag{7}$$

In Equation (7), W^0 represents the parameter used for linear transformation after connecting multiple attention results. $head_1$ represents the output result of the ith attention head, as shown in Equation (8).

$$head_i = Attention\left(PW_i^P, UW_i^U, VW_i^V\right) \tag{8}$$

The representation of all time periods for each network node constitutes a long sequence of data. They are sequentially input into the Transformer network to obtain the final embedding representation of the nodes. This embedding representation not only includes the state of the node at the current moment, but also its behavior and changes in history. In addition, cross entropy is used to describe its loss function at each time period, as shown in Equation (9).

$$J_v = \frac{1}{T}\sum_{t=0}^{T}\sum_{i \in W_{walk(v)}^t} -\log(\beta(< h_t(v), h_t(i) >))$$

$$- \sum_{k \in W^t_{neg(v)}} \log(1 - \beta(< h_t(v), h_t(i) >)) \tag{9}$$

In Equation (9), $W^t_{walk(v)}$ represents the set of random walks starting from node v at a certain time t. $W^t_{neg(v)}$ represents the negative sampling set of the node at that time. This ensures that the node vector can effectively capture the temporal and structural features of the network.

3.2. **Design of dynamic network representation model combining temporal heterogeneity information.** For dynamic network learning representations, using graph neural networks alone may not be sufficient to capture the complexity of temporal heterogeneity information. Therefore, combining temporal heterogeneity information for dynamic network learning representation has become crucial. Temporal heterogeneous information network refers to a network data structure with temporal and heterogeneous characteristics. In such a network, the relationships between nodes may be different types of edges. Introducing time series information can track the changes of nodes and edges in the network, helping to understand the temporal and evolutionary patterns of the network. In addition, temporal heterogeneous information networks can improve the predictive accuracy of dynamic network representation learning. Considering the temporal and heterogeneity of the network can better capture the changing patterns of nodes and edges in the network, providing more accurate prediction results, such as node attributes and relationships between nodes. The schematic diagram of the temporal heterogeneous information network is shown in Figure 4.

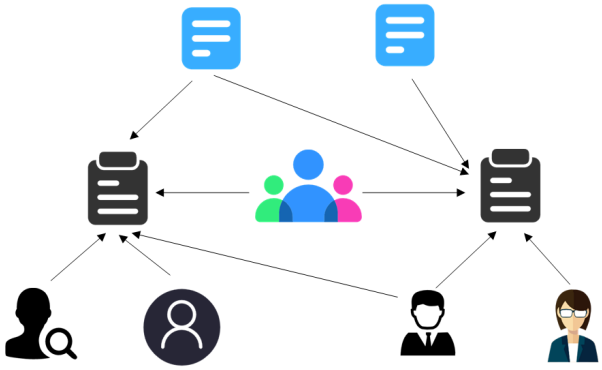


FIGURE 4. Schematic diagram of temporal heterogeneous information network

To effectively capture the dynamic evolution process of nodes and edges in the network, handle heterogeneity relationships, and provide flexible models to improve prediction accuracy, a MATHNRL is designed. MATHNRL integrates three key modules: specific type node mapping, time decaying attention layer, and meta path level attention aggregation, which is used for temporal heterogeneous information network representation learning. It can learn rich structural and semantic information through meta paths, fully considering the influence of temporal attributes in the network, and enhancing the expressive power of the instance's impact on the target node based on the attention layer, using meta path level attention to learn the importance of different meta paths. Firstly, the specific type node mapping module achieves a unified representation of heterogeneous nodes by mapping different types of node features to the same feature space. For any node in the graph, its mapped features are shown in Equation (10).

$$h'_v = P(h_v) = M_\Phi \cdot h_v + b_\Phi \tag{10}$$

In Equation (10), b_Φ represents the characteristics of the initial node v. $P()$ represents the mapping function of the node. M_Φ represents the weight matrix corresponding to the node type Φ. b_Φ represents a bias vector. After the above operation, the feature space inconsistency between heterogeneous nodes can be eliminated. They can better participate in the subsequent learning process. The structure of the MATHNRL model is shown in Figure 5.

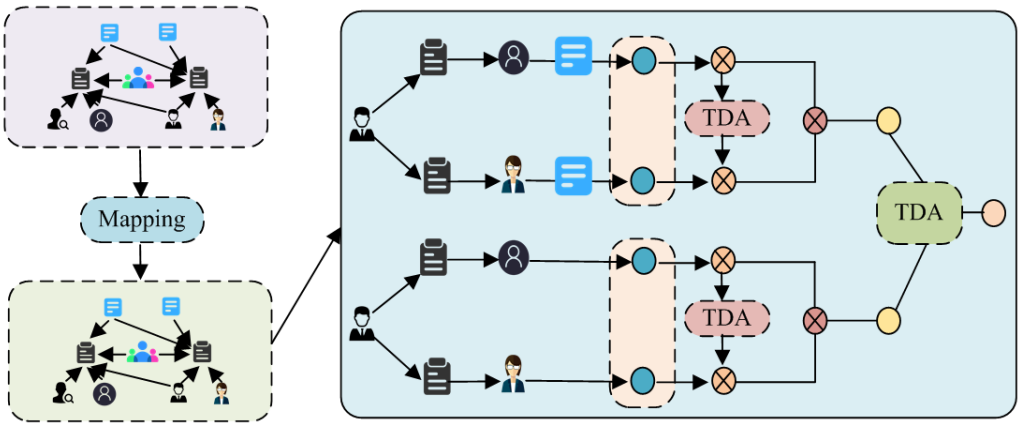


FIGURE 5. Structure diagram of MATHNRL model

In the MATHNRL model, the attenuating attention layer plays a crucial role. It learns the influence of different meta path instances on the target node, and combines the time decay factor to weight the meta path instances at different times. This can better capture the dynamic evolution process of nodes and edges in the network, as well as the temporal relationships between nodes. By using a time decay attention layer, node representations under different meta paths can be obtained. These representations have temporality and diversity. They can more comprehensively describe the evolution process of the network. The loss function is shown in Equation (11).

$$Loss = -\sum_{v \in V_S} \sum_{c=1}^{C} R_v^c \ln(F_v^c) \tag{11}$$

In Equation (11), V_S represents the set of nodes with labels. R_v^c represents the true label of the node. F_v^c represents the predicted label of the node. Any label of the node is represented by C. Finally, the meta path level attention aggregation module plays a role in integrating node representations under different meta paths, as shown in Equation (12).

$$\omega_{\phi j} = \frac{1}{|V|} \sum_{v \in V} q^T \tanh\left(W \cdot h_v^{\phi j} + c\right) \tag{12}$$

In Equation (12), ϕ represents an instance of a certain meta path. $h_v^{\phi j}$ represents the node representation of the jth meta path with specific semantics. c represents the bias vector. V represents a set of nodes. q represents the attention vector of a certain meta path. Then, the weight parameters of each meta path are normalized using the softmax function, as shown in Equation (13).

$$\gamma_{\phi j} = \text{softmax}(\omega_{\phi j}) = \frac{\exp(\omega_{\phi j})}{\sum_{m=1}^{k} \exp(\omega_{\phi j})} \tag{13}$$

After introducing the attention mechanism, this module can dynamically adjust the weights of node representations under different meta paths, thereby better capturing the

contributions of different meta paths to the target node. The final representation can be obtained through meta path level attention aggregation, which integrates information from different meta paths. It has richer semantic expression capabilities. The GraphSAGE network can effectively learn node representations and relational features from graph data. The MATHNRL model can utilize a multi-agent structure to capture complex relationships between nodes. Therefore, the study combines the two to provide more accurate node representation and relationship feature extraction capabilities, thereby better solving graph learning tasks. The schematic diagram of the GraphSAGE-MATHNRL model is shown in Figure 6.

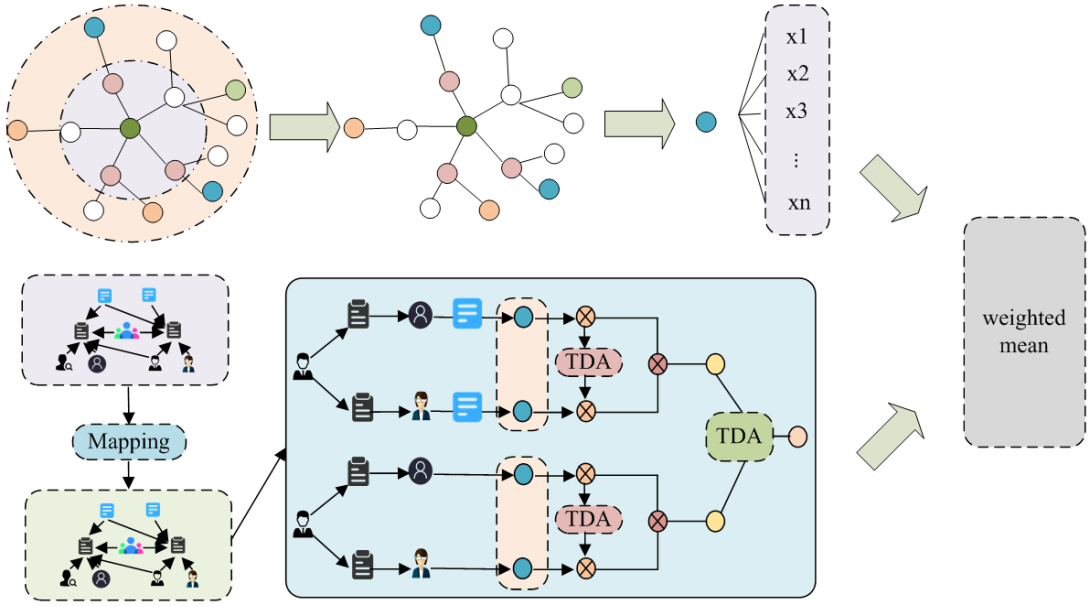


FIGURE 6. Structure diagram of GraphSAGE-MATHNRL model

The GraphSAGE model and the MATHNRL model respectively perform output and then merge node representations, concatenating or fusing the node representations output by the two models. Specifically, the study first utilizes the GraphSAGE network to generate embedding vectors from the local neighborhood structure of nodes in the graph, in order to capture the structural information of nodes. Then use the MATHNRL model to capture complex features between nodes. Then, the two node representations are fused using methods such as weighted average or element by element multiplication to generate a brand new node representation that combines structural information and relational features.

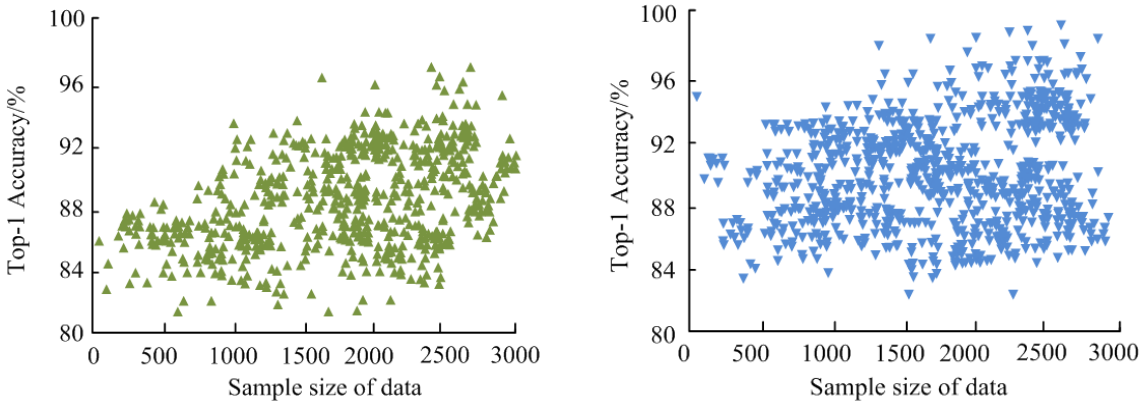
4. Performance Testing and Application Experiments of GraphSAGE-MATHNRL Model.

This chapter is divided into two sections. The first section mainly focuses on performance testing of the GraphSAGE-MATHNRL on different datasets. The second section conducts a series of comparative experiments with other similar models to further verify the superiority of the model.

4.1. Performance testing of GraphSAGE-MATHNRL model.

The experiment first designed GraphSAGE. However, state networks typically contained multiple types of heterogeneous information. To increase the robustness of the model, the GraphSAGE-MATHNRL dynamic network representation model was designed by combining the GraphSAGE model and the MATHNRL model. Firstly, the model was applied to the Email dataset and Flickr dataset for testing. Among them, the Email dataset (https://www.

cs.cmu.edu/∼./enron/) was extracted and generated from email data from a large European research institution. The member and email content data were anonymized. This network represents communication between certain departments within the institution, with each node representing a researcher and each edge representing a user sending an email to another user at a certain time. The Flickr dataset (https://github.com/NVlabs/ffhq-dataset) is constructed by forming links between shared Flickr public images. Edge is formed through images from the same location, submitted to the same gallery, group or collection, shared tags, or photos taken by friends. It is collected from many sources, and images are represented by SIFT to extract features from them. This dataset includes links between 105938 images and 2316948 images. The experiment used Top-1 accuracy (acc1) and Top-5 accuracy (acc5) as evaluation indicators. The experimental results were shown in Figure 7.

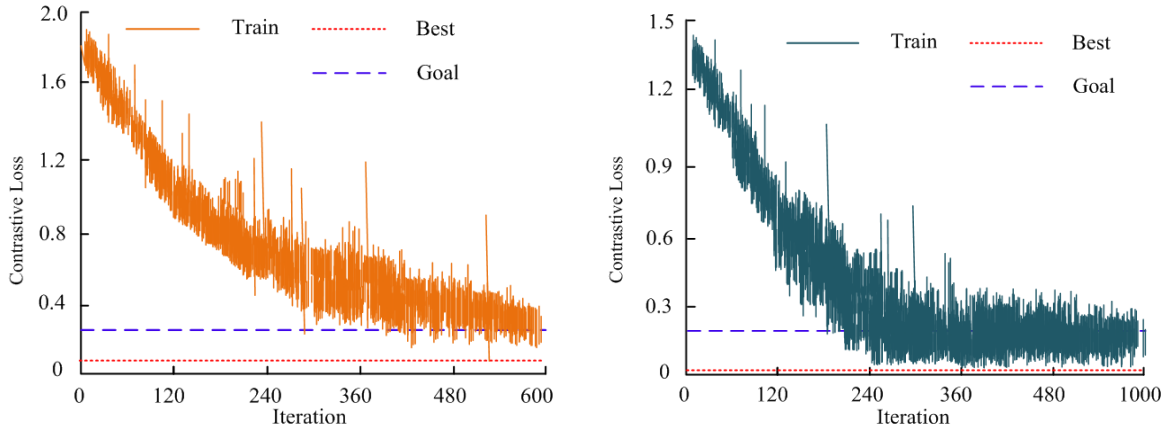


(a) Email (b) Flickr

FIGURE 7. GraphSAGE-MATHNRL accuracy and changes in dataset volume

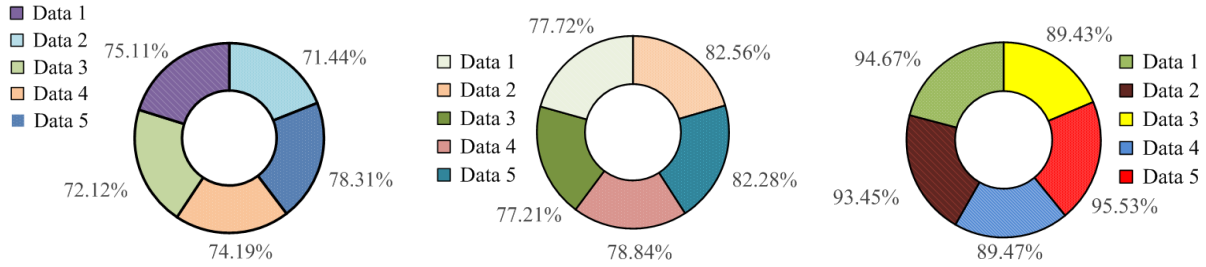
Figures 7(a) and 7(b) respectively represented the accuracy and data size changes of the GraphSAGE-MATHNRL dynamic network representation learning model. From Figure 7, the Top-1 accuracy of the Email dataset was mostly between 80% and 90% before the data volume reached 1000. As the amount of data increased, the accuracy gradually improved. The final Top-1 accuracy remained stable at 84% to 96%. In the Flickr dataset, the Top-5 accuracy ranged from 83% to 92% before reaching a data volume of 1000. As the data volume increased to 2500, the Top-5 accuracy stabilized at 85% to 97%. The design method can better capture more implicit relationship information, extract richer feature representations, and thus better perform node classification prediction. In representation learning, Comparative Loss (CL) was a commonly used loss function. It optimized the representation learning model by calculating the similarity between two samples. The CL of the GraphSAGE-MATHNRL model on two datasets during the training process was shown in Figure 8.

Figures 8(a) and 8(b) showed the training process of the GraphSAGE-MATHNRL model on two different datasets. The loss function was compared with the change in the number of training rounds. The error reduction rate of the model on the two datasets was similar, stable and without significant differences. After training, the model error gradually decreased and converged to around 0.35 and 0.30. These results indicated that the GraphSAGE-MATHNRL model approached the optimal solution during training. The predictive ability gradually improved. To comprehensively understand the performance and generalization ability of GraphSAGE, MATHNRL, and GraphSAGE-MATHNRL algorithms, five datasets were introduced in the experiment. Three algorithms were trained



(a) Email dataset (b) Flickr dataset

FIGURE 8. Comparative loss function values of GraphSAGE-MATHNRL model on two datasets



(a) GraphSAGE (b) MATHNRL (c) GraphSAGE-MATHNRL

FIGURE 9. F1 values for three models

separately. F1 values were used as evaluation indicators. The F1 values of the three models were shown in Figure 9.

Figure 9 showed the training results of GraphSAGE-MATHNRL, GraphSAGE, and MATHNRL on five different datasets. The GraphSAGE-MATHNRL model performed very well in F1 scores on all datasets. The F1 value of the GraphSAGE-MATHNRL model was significantly higher than the other two models, reaching 94.67%, 93.45%, 89.43%, 89.47%, and 95.53%, respectively. This demonstrated that the model had strong learning and prediction capabilities. In contrast, the F1 values of the GraphSAGE model were relatively low, at 75.11%, 71.44%, 72.12%, 74.19%, and 78.31%, respectively. The performance of this model was not ideal. The F1 values of the MATHNRL model were between the first two, with values of 77.72%, 82.56%, 77.21%, 78.84%, and 82.28%, respectively. The above results showed that the GraphSAGE-MATHNRL model successfully optimized the model by combining the GraphSAGE model with the MATHNRL model, increasing its robustness and accuracy. To verify the overall performance of the designed GraphSAGE-MATHNRL model, a node representation learning framework based on graph convolutional networks, GCN_MA [21], an information based causal learning framework GNNs-ICL [22], and a temporal graph neural network T-GNNs [23] were introduced. Four models were tested on the Email and Flickr datasets, and the predicted results are shown in Table 1.

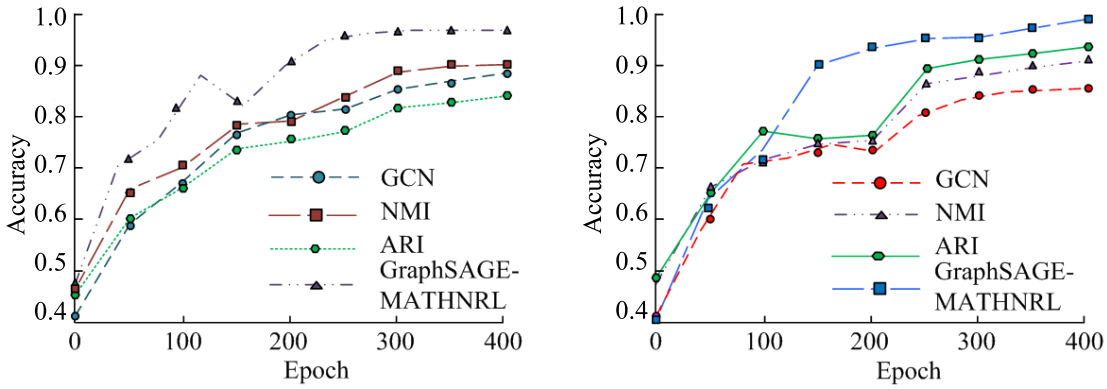
From Table 1, it can be seen that in the Email dataset, the prediction accuracy of the GraphSAGE-MATHNRL model is 0.9442, which is 3.73%, 4.69%, and 3.05% higher than

TABLE 1. Test results of four algorithms on two datasets

Model	Data set	
	Email	Flickr
GCN_MA	0.9137	0.9265
GNNs_ICL	0.8973	0.9543
T-GNNs	0.9069	0.9432
GraphSAGE-MATHNRL	0.9442	0.9678

the other three algorithms, respectively. In the Flickr dataset, the prediction accuracy of the GraphSAGE-MATHNRL model is 0.9678, which is significantly higher than the three models such as GCN_MA. The design algorithm can effectively capture the evolution process of the network, demonstrating its good overall performance.

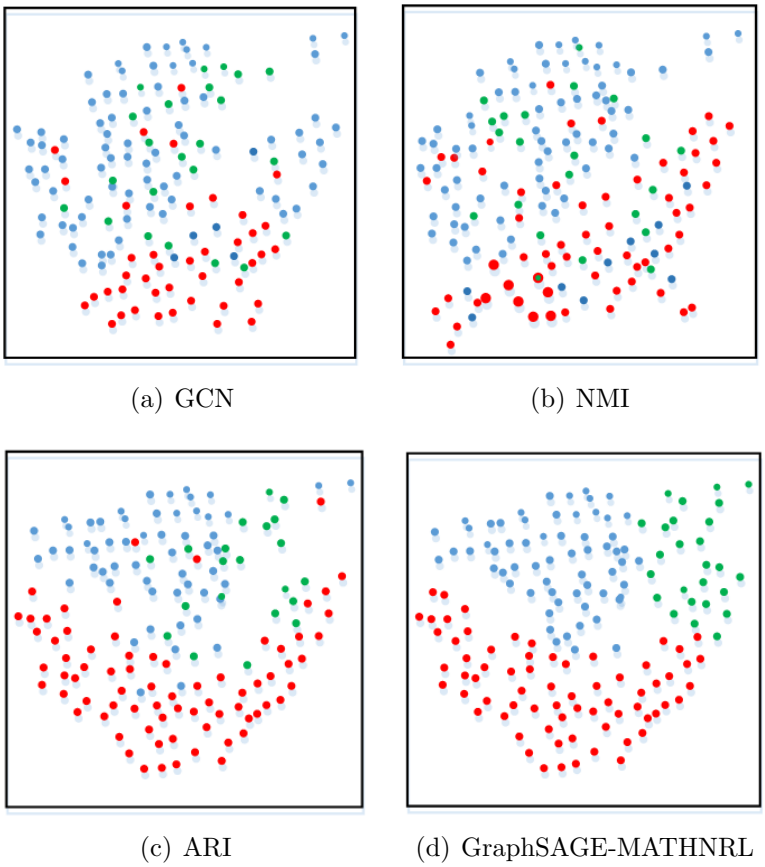
4.2. **Comparative experiment and application analysis of GraphSAGE-MATHNRL model.** To further verify the superiority of the GraphSAGE-MATHNRL algorithm in the dynamic network representation, GCN, Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI) algorithms were introduced in the experiment to compare with the GraphSAGE-MATHNRL algorithm. The model was trained using the Email dataset and Flickr dataset. The experimental results were shown in Figure 10.



(a) Email dataset (b) Flickr dataset

FIGURE 10. Accuracy variation curves during the training process of four algorithms

Figure 10 showed the training accuracy curves of four algorithms on the Email and Flickr datasets. GraphSAGE-MATHNRL had the fastest convergence speed on the Email dataset, reaching 95.5%. The other three algorithms tended to converge after about 300 iterations. ARI had the lowest convergence accuracy, only 83.6%. The convergence accuracy of GCN and NMI was both around 88%. On the Flickr dataset, the accuracy of the GraphSAGE-MATHNRL algorithm did not change significantly. The accuracy of the other three algorithms varied significantly. The ARI algorithm showed the greatest change, with an increase of 8.9%. This may be because GraphSAGE excels at capturing local neighborhood structure information, while MATHNRL can fully utilize temporal information. The combination of the two enables GraphSAGE-MATHNRL to better model the complex evolution process in dynamic networks. To visually demonstrate the effectiveness of the GraphSAGE-MATHNRL algorithm, the experiments projected the representations learned by the four algorithms onto a two-dimensional space. T-distribution random neighborhood embedding was used to visualize nodes on the Flickr dataset. The results were shown in Figure 11.



(a) GCN (b) NMI

(c) ARI (d) GraphSAGE-MATHNRL

FIGURE 11. Visualization of clustering effects of four algorithms

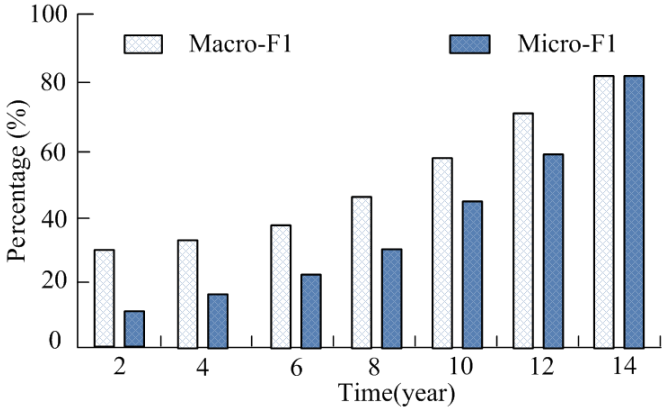


FIGURE 12. The impact of time changes on network representation

In Figure 11, the GCN algorithm performed poorly in processing graph structured data and failed to handle category differentiation well. In contrast, ARI and NMI performed slightly better, but there were limitations. The representations learned by GraphSAGE-MATHNRL exhibited excellent performance. Nodes of the same color were clustered, while nodes of different colors were scattered. To investigate the impact of temporal changes on network representation, the Email dataset was processed over time. Every four years was used as a time node. Macro-F1 and Micro-F1 were used as evaluation indicators to record the current network status. The experimental results were shown in Figure 12.

From Figure 12, as the network continued to develop and evolve, the effectiveness of node classification gradually improved. At the beginning, the Macro-F1 metric of the model was only 30%. However, this indicator reached 90% after 14 years. Similarly, the Micro-F1 indicator also showed a similar performance, increasing from the initial 10% to 90%. Based on these data and trends, over time, the network contained more effective information, which provided richer and more complex sample data for learning.

5. **Conclusion.** Traditional network representation learning methods often only focus on network topology information, while ignoring temporal information in the network. To address the above issues, a new dynamic network representation method is proposed by combining GraphSAGE network and MATHNRL network. In algorithm comparison experiments, the GraphSAGE-MATHNRL algorithm had the fastest convergence speed. After 250 iterations, convergence was achieved, ultimately converging to 95.5%. The convergence speed of the other three algorithms was not significantly different. All tended to converge around 300 iterations. ARI had the lowest convergence accuracy, only 83.6%. The convergence accuracy of both GCN algorithm and NMI algorithm was around 88%. In application experiments, the performance of GCN algorithm in processing graph structured data was not satisfactory. In the figure, nodes of different categories were intertwined with each other. This indicated that the algorithm had not effectively addressed the category differentiation in node representation learning. In contrast, the representations learned by GraphSAGE-MATHNRL exhibited excellent performance. In summary, the GraphSAGE-MATHNRL dynamic network representation method proposed in this study has performed well in algorithm comparison and application experiments, with good performance and accuracy. However, relevant research has not applied this algorithm in a large amount of practical work to verify its effectiveness in practical situations. This is also an aspect that needs to be carried out in subsequent experiments. Meanwhile, for dynamic networks with frequent changes, the model may need to be constantly updated and adjusted to adapt to new network structures. This may lead to poor performance of the model in applications with high real-time requirements. In the future, the calculation methods and optimization algorithms of the model will be improved to reduce training time and resource consumption, enabling it to handle larger scale dynamic networks.

REFERENCES

[1] T. A. Waziri and B. M. Yakasai, Assessment of some proposed replacement models involving moderate fix-up, *Journal of Computational and Cognitive Engineering*, vol.2, no.1, pp.28-37, 2022.

[2] T. A. Waziri and A. Ibrahim, Discrete fix up limit model of a device unit, *Journal of Computational and Cognitive Engineering*, vol.2, no.2, pp.163-167, 2022.

[3] D. Aikhuele, Development of a statistical reliability-based model for the estimation and optimization of a spur gear system, *Journal of Computational and Cognitive Engineering*, vol.2, no.2, pp.168-174, 2022.

[4] Y. H. Chae, C. Y. Lee, S. M. Han and P. H. Seong, Graph neural network based multiple accident diagnosis in nuclear power plants: Data optimization to represent the system configuration, *Nuclear Engineering and Technology*, vol.54, no.8, pp.2859-2870, 2022.

[5] J. Zhao, X. Wang, C. Shi, B. Hu, G. Song and Y. Ye, Heterogeneous graph structure learning for graph neural networks, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol.35, no.5, pp.4697-4705, 2021.

[6] Z. Zhu, Z. Zhang, L. P. Xhonneux and J. Tang, Neural Bellman-Ford networks: A general graph neural network framework for link prediction, *Advances in Neural Information Processing Systems*, vol.34, pp.29476-29490, 2021.

[7] F. Li, J. Feng, H. Yan, G. Jing, F. Yang, F. Sun and Y. Li, Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution, *ACM Transactions on Knowledge Discovery from Data*, vol.17, no.1, pp.1-21, 2023.

[8] R. Bhattacharya, N. K. Nagwani and S. Tripathi, Detecting influential nodes with topological structure via graph neural network approach in social networks, *International Journal of Information Technology*, vol.15, no.4, pp.2233-2246, 2023.
[9] Y. Wang, J. Wang, Z. Cao and A. B. Farimani, Molecular contrastive learning of representations via graph neural networks, *Nature Machine Intelligence*, vol.4, no.3, pp.279-287, 2022.
[10] A. Deng and B. Hooi, Graph neural network-based anomaly detection in multivariate time series, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol.35, no.5, pp.4027-4035, 2021.
[11] C. Huo, D. Jing, Y. Li, D. He, Y. B. Yang and L. Wu, T2-GNN: Graph neural networks for graphs with incomplete features and structure via teacher-student distillation, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol.37, no.4, pp.4339-4346, 2023.
[12] Y. Bai, C. Li, Z. Lin, Y. Wu, Y. Miao, Y. Liu and Y. Xu, Efficient data loader for fast sampling-based GNN training on large graphs, *IEEE Transactions on Parallel and Distributed Systems*, vol.32, no.10, pp.2541-2556, 2021.
[13] X. Wang, K. Yen, Y. Hu and H. W. Shen, DeepGD: A deep learning framework for graph drawing using GNN, *IEEE Computer Graphics and Applications*, vol.41, no.5, pp.32-44, 2021.
[14] T. Chen, X. Zhang, M. You, G. Zheng and S. Lambotharan, A GNN-based supervised learning framework for resource allocation in wireless IoT networks, *IEEE Internet of Things Journal*, vol.9, no.3, pp.1712-1724, 2021.
[15] J. Dai, Y. Chen, L. Xiao, L. Jia and Y. He, Design and analysis of a hybrid GNN-ZNN model with a fuzzy adaptive factor for matrix inversion, *IEEE Transactions on Industrial Informatics*, vol.18, no.4, pp.2434-2442, 2021.
[16] H. Zhou, D. Ren, H. Xia, M. Fan, X. Yang and H. Huang, AST-GNN: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction, *Neurocomputing*, vol.445, pp.298-308, 2021.
[17] X. Wan, K. Xu, X. Liao, Y. Jin, K. Chen and X. Jin, Scalable and efficient full-graph GNN training for large graphs, *Proceedings of the ACM on Management of Data*, vol.1, no.2, pp.1-23, 2023.
[18] L. Zeng, C. Yang, P. Huang, Z. Zhou, S. Yu and X. Chen, GNN at the edge: Cost-efficient graph neural network processing over distributed edge servers, *IEEE Journal on Selected Areas in Communications*, vol.41, no.3, pp.720-739, 2022.
[19] N. Shi, J. Xu, S. W. Wurster, H. Guo, J. Woodring, L. P. Van Roekel and H. W. Shen, A hierarchical and adaptive graph neural network for parameter space exploration of unstructured-mesh ocean simulations, *IEEE Transactions on Visualization and Computer Graphics*, vol.28, no.6, pp.2301-2313, 2022.
[20] J. You, J. M. Gomes-Selman, R. Ying and J. Leskovec, Identity-aware graph neural networks, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol.35, no.12, pp.10737-10745, 2021.
[21] P. Mei and Y. Zhao, Dynamic network link prediction with node representation learning from graph convolutional networks, *Scientific Reports*, vol.14, no.1, 538, 2024.
[22] Z. Zhao, P. Wang, H. Wen, Y. Zhang, Z. Zhou and Y. Wang, A twist for graph classification: Optimizing causal information flow in graph neural networks, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol.38, no.15, pp.17042-17050, 2024.
[23] S. Gao, Y. Li, Y. Shen and L. Chen, ETC: Efficient training of temporal graph neural networks over large-scale dynamic graphs, *Proceedings of the VLDB Endowment*, vol.17, no.5, pp.1060-1072, 2024.

Author Biography

Zhixiao Li obtained bachelor's degree in Mathematics and Applied Mathematics from Ludong University in 2012. In 2014, she obtained master's degree in Applied Statistics for Beijing Institute of Technology. She works as a teacher in Cangzhou Normal University since 2014. Her areas of interest include machine learning and statistical analysis. She has published 6 academic articles, participated in 8 scientific research projects.