Journal of Engineering, Project, and Production Management
2026, 16(3), 2025-208

Dynamic Network Infrastructure Management Using Graph Computation and Incremental Clustering

Lingji Wang
Associate Professor, Saxo Fintech College at Business College, Geely University, Chengdu, 641423, China, E-mail: wangljlingji@outlook.com

Project Management
Received September 9, 2025; revised October 30, 2025; accepted April 23, 2026
Available online May 4, 2026

Abstract: With the rapid development of big data and cloud computing, the scale of network infrastructure continues to expand. Dense equipment and frequent dynamic changes pose new challenges to traditional network management methods. Therefore, a network infrastructure management method that integrates graph computation and incremental clustering is proposed. A graph computation framework is constructed by combining graph sampling and aggregation methods with bidirectional long short-term memory networks, and an incremental clustering algorithm is introduced to achieve dynamic node recognition and region partitioning. The experimental results on the TensorFlow platform showed that the graph computation framework achieved 95.32% node classification accuracy and an F1 score of 0.95. The average clustering time of the incremental clustering algorithm was 1.68 seconds, and the intra-cluster variance was as low as 0.21, which was significantly better than that of traditional methods. In addition, this method achieved task completion rates of 95.28% to 98.25% and 95.71% to 99.33%, verifying its efficiency and robustness in dynamic network environments. The research provides flexible, real-time solutions for the intelligent management of complex network infrastructure, which has important theoretical and practical value.

Keywords: Graph computation; incremental clustering; TensorFlow; big data; network infrastructure.

DOI 10.32738/JEPPM-2025-208

1. Introduction

With the continuous advancement of information technology, new architectures such as big data and cloud computing have been widely applied in national infrastructure and various industry networks, supporting the growing business demands (Gheisari et al., 2023). In the context of digital transformation, big data governance has become a core component of information system operations, and cloud computing has gradually evolved into a key technology foundation supporting national and industry-level infrastructure. With the rise of cloud-edge-end collaborative models, network environments are experiencing unprecedented scale, high heterogeneity in device types, frequent business interactions, and extremely high real-time requirements. This trend not only accelerates the explosive growth of data but also poses unprecedented challenges to traditional network management systems. In practical applications, the network scale rapidly expands, the number of devices continues to increase, and the system structure becomes more complex. The nodes in the network are closely related through links, services, or resource allocation, presenting characteristics such as device density, high-frequency operation, and rapid structural changes (Fang et al., 2023). In this context, network management is facing new challenges. The traditional management method that relies on static configuration and centralized control is no longer able to cope with frequent device access and exit, real-time changes in network topology, and other issues (Aziz et al., 2023). In the dynamic operation of multi-source devices, comprehensively understanding the network status, quickly identifying changes, and making timely management responses has become the key to ensuring network stability and business continuity. However, existing network management methods still have many shortcomings (Chu et al., 2023). Firstly, most methods focus on analyzing the attributes of a single node, neglecting the structural relationships between nodes and making it difficult to identify potential functional groups or abnormal connection patterns (Bouchetara et al., 2024). In addition, many state monitoring methods rely on preset rules or regular checks, making it difficult to detect continuous changes in node behavior, such as traffic fluctuations or performance degradation (Wang et al., 2023). Finally, in environments where large-scale data is continuously generated, traditional clustering or recognition models mostly rely on offline training, lack real-time processing for newly added data, and are difficult to adapt to rapidly changing management needs in complex systems (Guo et al., 2023). Therefore, a network infrastructure management method based on the

TensorFlow platform is proposed, which integrates graph computation and incremental clustering. The network topology is modeled by combining Graph Sample and Aggregation (GraphSAGE), and Bidirectional Long Short-Term Memory (BiLSTM) is used to depict the evolution of node states. An incremental clustering algorithm is introduced to dynamically identify node functions and partition regions. This work aims to use this network infrastructure management method to identify node functions and divide areas, thereby providing flexible, efficient intelligent management for actual network systems.

The novelty of this study lies in the first integrated application of a graph computation framework with an incremental clustering algorithm. This approach addresses the limitations of existing Artificial Intelligence/Machine Learning-based (AI/ML) network infrastructure optimization methods, which often rely on a single technical pathway. Compared with research on topology optimization based on graph convolutional networks, scheduling strategies using deep reinforcement learning, and cluster driven traffic management, this work demonstrates stronger system integration and real-time adaptability. Traditional approaches typically employ graph neural networks for topological embedding or rely solely on clustering algorithms for node grouping and anomaly detection, making it difficult to simultaneously capture both structural relationships and dynamic evolution features. In addition, existing clustering methods are mostly based on offline batch processing, lacking the capability to respond to rapid network changes in real-time.

The contribution of this study lies in constructing an intelligent management framework that integrates GraphSAGE-based structural modeling, Bi-LSTM temporal feature extraction, and incremental clustering. This framework dynamically identifies node functionality and network partitioning, significantly improving adaptability in complex, rapidly evolving network environments. Compared with existing methods, the proposed framework achieves superior performance in node classification accuracy, clustering response speed, and system scalability, representing both a methodological innovation and a practical engineering advancement.

2. Related Works

Currently, as network infrastructure scales and the demand for intelligence increases, traditional management methods are facing severe challenges in energy consumption optimization, system collaboration, intelligent detection, and other areas. Therefore, scholars at home and abroad have proposed a series of innovative solutions. Karthick Raghunath et al. (2024) proposed a hybrid layered computation architecture based on the Internet of Things to address the insufficient monitoring continuity caused by the high energy consumption of wireless network nodes in smart city network infrastructure. By optimizing the cluster node selection algorithm, the energy efficiency of the network infrastructure was improved by 41.04%. Daulat et al. (2024) addressed the fragmentation issue of cross-system collaborative management of urban network infrastructure. They systematically analyzed the networked management system for physical infrastructure, such as roads, and water supply and drainage networks, revealing typical bottlenecks such as data silos and asynchronous decision-making. This provided direction for building a unified management platform for a digital twin driven network infrastructure. Litherland and Andrews (2025) proposed a global modeling framework based on system Petri nets to address the collaborative management of multiple assets in railway network infrastructure. A new methodology for cross-system cost-benefit analysis of complex network infrastructure was established by dynamically integrating the full lifecycle data of heterogeneous assets such as tracks and signal systems. Li et al. (2024) proposed a security management framework that integrated artificial intelligence to address identity authentication at the national level cloud infrastructure. This framework combined multiple model collaboration mechanisms, including threat identification, policy generation, and resource scheduling, to achieve global intelligent control and rapid response to cluster environments, providing technical support for intelligent security management of infrastructure.

With the continuous expansion of network infrastructure scale and the increasing demand for intelligence, traditional static analysis methods face significant challenges in modeling complex spatiotemporal relationships, adapting to dynamic data, and optimizing across multiple objectives. Therefore, the academic community has made a series of breakthroughs in dynamic graph computation and incremental clustering. Yang et al. (2024) built a multi-Graph Convolutional Network (GCN) to address the insufficient spatial features in traditional GCN for predicting traffic flow propagation in urban road networks. A three-dimensional spatial relationship graph was constructed to model the multidimensional spatiotemporal characteristics of traffic propagation flow, providing a new paradigm for intelligent signal control of transportation network infrastructure. Ren (2024) proposed a variational GCN to address the limitations of traditional GCN fixed edge weights in text classification tasks. Its dynamic graph representation learning mechanism provided a transferable technical framework for unstructured data management tasks such as network infrastructure log analysis. Soleymanian et al. (2024) proposed an incremental clustering algorithm based on semantic concepts to address feature drift and concept drift in text data streams. Incremental word embedding and hierarchical modeling were adapted to cluster dynamic text streams, providing an interpretable analysis framework for real-time management of network infrastructure log streams. Dwivedi et al. (2024) developed a multi-objective incremental clustering algorithm to address the lag of traditional clustering methods in dynamic data environments. The algorithm integrated weighted optimization of inter-cluster distance, intra-cluster distance, and density to achieve real-time cluster updates in dynamic data such as plant genomes and network traffic. This provided a new tool for real-time pattern discovery in infrastructure monitoring data.

In summary, although existing research has achieved some progress in energy consumption optimization, intelligent detection, and dynamic analysis of network infrastructure, there are generally deficiencies, such as insufficient dynamic adaptability and limited multi-modal data processing capabilities. Therefore, an innovative network infrastructure management method integrating GraphSAGE, BiLSTM, and an incremental clustering algorithm is proposed, which organically integrates the three to dynamically identify and partition network infrastructure node functions in a “structure-time-increment” manner, filling the gap in real-time intelligent management in large-scale high-frequency change scenarios.

3. Methods and Materials

The research develops a network infrastructure management method that combines graph computation and incremental clustering. First, GraphSAGE is used to structurally model the network topology, and BiLSTM is combined to model the temporal behavior of nodes. Finally, a semi-supervised incremental clustering algorithm based on graph structure is used to achieve dynamic recognition and region partitioning, forming a complete management method.

3.1. GraphSAGE Graph Computation Framework Combined with BiLSTM

With the increasing scale and complexity of network infrastructure, traditional management methods are no longer able to cope with frequent device changes and real-time changes in network topology. A new network infrastructure management method based on the TensorFlow platform is developed, combining network structural information with dynamic changes in node states to address complex network environments. This method uses a graph computation framework as its core, supplemented by temporal modeling techniques to simultaneously capture the structural characteristics of the network and the evolution of node states. The TensorFlow platform supports efficient tensor operations, dynamic graph construction, and distributed training, with good scalability and deployment flexibility. Therefore, it is chosen as the basic platform for underlying modeling and implementation in research. GraphSAGE has the characteristics of neighbor sampling and feature aggregation, which can efficiently process large-scale graph data and generate low dimensional embedded representations of nodes. It is suitable for real-time updates of network infrastructure data in dynamic environments. Therefore, GraphSAGE is taken as the core framework for graph computation in the study. GraphSAGE generates node embeddings through a three-step process, as shown in Fig. 1.

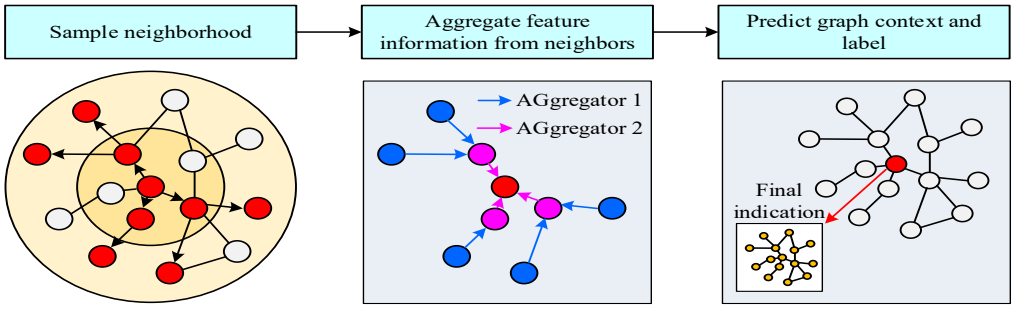


Fig. 1. The overall workflow of GraphSAGE

From Fig. 1, the overall workflow of GraphSAGE includes three main steps: sampling sample neighbors, aggregating neighbor features, and predicting the context and labels of the graph (Pujiastuti and Wahyudi, 2024). In the first step, GraphSAGE improves computational efficiency by sampling a certain number of nodes from each node's neighbors to avoid processing the entire graph's data. Specifically, GraphSAGE selects the neighbors of each node. According to the set number of neighbors, the first and second layers of neighbor nodes are sampled with the node as the center in the graph, forming a neighbor sub-graph of the node, as presented in Eq. (1).

$$N_k(v) = \{\text{Neighbor}(v,k)\} \tag{1}$$

In Eq. (1), $N_k(v)$ represents the k-th layer neighbors sampled from the neighbor nodes of node v. In the second step, GraphSAGE aggregates the feature information of neighbor nodes. To obtain the embedded representation of nodes, GraphSAGE combines the features of neighbor nodes with the features of the target node. In this process, GraphSAGE uses aggregation functions to process the features of neighbor nodes, ultimately forming a node representation. Specifically, the aggregated feature information is shown in Eq. (2).

$$h_{N(v)}^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)} \mid u \in N(v)\right\}\right) \tag{2}$$

In Eq. (2), $h_{N(v)}^{(k)}$ represents the neighbor feature aggregation result of the k-th layer target node v. $h_u^{(k-1)}$ signifies the feature vector of neighbor node u in the previous layer. $N(v)$ signifies the set of neighbor nodes of node v. Finally, in the third step, GraphSAGE combines the node feature with the aggregated features of its neighbors to generate the final node embedding representation through a learning function. Specifically, the representation of the target node is obtained by concatenating the current features of the node with the aggregated features of its neighbors, and then calculating through a weight matrix $W^{(k)}$ and an activation function σ, as presented in Eq. (3).

$$h_v^{(k)} = \sigma\left(W^{(k)} \cdot \left[h_v^{(k-1)} \,\|\, h_{N(v)}^{(k)}\right]\right) \tag{3}$$

In Eq. (3), $h_v^{(k)}$ signifies the final embedding representation of the target node v at the k-th layer. $h_v^{(k-1)}$ signifies the feature of node v in the previous layer. σ signifies a nonlinear activation function, typically using ReLU or sigmoid. However, when applying GraphSAGE to network infrastructure management, due to the constantly changing states of network nodes over time, relying solely on structural information in graph computation cannot fully capture the dynamic

behavior of nodes, resulting in the inability to accurately predict their future states and behaviors. At this point, it is necessary to combine a modeling method that can handle temporal information. Compared with traditional Recurrent Neural Network (RNN), unidirectional Long Short-Term Memory (LSTM), or Gated Recurrent Unit (GRU) models, Bi-LSTM has a stronger ability to extract bidirectional temporal features. It can simultaneously model both the historical evolution and future trends of node states, making it better suited to handling network dynamics with bidirectional dependencies. In addition, Bi-LSTM is structurally mature and stable during training, which performs well at capturing complex dependencies, especially in network infrastructure scenarios characterized by frequent device load fluctuations, traffic anomalies, and topology changes. Based on these considerations, this study adopts Bi-LSTM as the temporal modeling tool, and its structure is shown in Fig. 2.

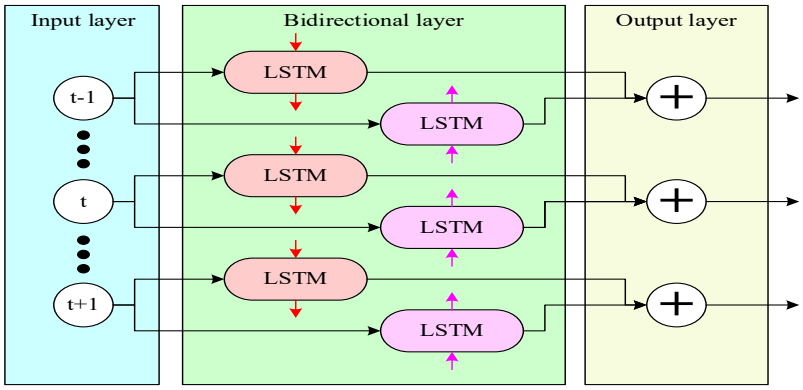


Fig. 2. Structure of BiLSTM

In Fig. 2, BiLSTM processes the forward and backward information of input data through two LSTM layers. Forward LSTM captures past patterns of change from historical node states, while backward LSTM combines future information to further predict changes in nodes (Zhang et al., 2023). BiLSTM processes the forward and backward information of the input sequence separately through two independent LSTM layers. Forward LSTM learns the historical behavior of nodes from past states, while backward LSTM learns to predict the future behavior of nodes from future information. BiLSTM combines the outputs of these two LSTM layers to generate the final node representation, as shown in Eq. (4).

$$\begin{cases} h_t^f = \mathrm{LSTM}\left(x_t, h_{t-1}^f\right) \\ h_t^b = \mathrm{LSTM}\left(x_t, h_{t+1}^f\right) \\ h_t = \left[h_t^f \middle\| h_t^b\right] \end{cases} \tag{4}$$

In Eq. (4), h_t^f and h_t^b represent the output of the forward LSTM at time t and the output of the backward LSTM at time t, respectively. x_t represents the input data at time t, which includes node state characteristics such as bandwidth, load, and traffic in network infrastructure management. h_t represents the final node embedding vector. Therefore, the GraphSAGE graph computation framework combined with BiLSTM is shown in Fig. 3.

In Fig. 3, the GraphSAGE graph computation framework, combined with BiLSTM, adds BiLSTM for temporal data modeling based on the original GraphSAGE module. Firstly, GraphSAGE learns the structural embedding of nodes through neighbor sampling and feature aggregation, obtaining a preliminary representation of each node. Subsequently, the BiLSTM module performs bidirectional modeling on the temporal characteristics of node states, capturing the historical behavior and future trends of nodes. Finally, the outputs of GraphSAGE and BiLSTM are combined through a feature fusion module to generate the final node embedding vector containing structural and temporal information. In the process of feature fusion, the self-attention mechanism is adopted. The node embedding generated by GraphSAGE is h_v^{sage} and that generated by BiLSTM is h_v^{lstm}. The attention mechanism is achieved through dot product attention, as shown in Eq. (5).

$$\begin{cases} \alpha_{sage} = \dfrac{\exp\left(score\left(h_v^{sage}, q\right)\right)}{\sum_i \exp\left(score\left(h_i^{sage}, q\right)\right)} \\ \alpha_{lstm} = \dfrac{\exp\left(score\left(h_v^{lstm}, q\right)\right)}{\sum_i \exp\left(score\left(h_i^{lstm}, q\right)\right)} \end{cases} \tag{5}$$

In Eq. (5), α_{sage} and α_{lstm} represent the attention weights of nodes in the structural and temporal feature spaces, respectively. q represents the query vector, which can be regarded as the global contextual information of the current

task context or classification target, such as the label vector and global node embedding of network state analysis or fault prediction tasks. $score$ represents the attention scoring function, which measures the correlation between embeddings and task objectives. After obtaining two attention weights, the final fusion h_v^{fused} of node v is shown in Eq. (6).

$$h_v^{fused} = \alpha_{sage} \cdot h_v^{sage} + \alpha_{lstm} \cdot h_v^{lstm} \tag{6}$$

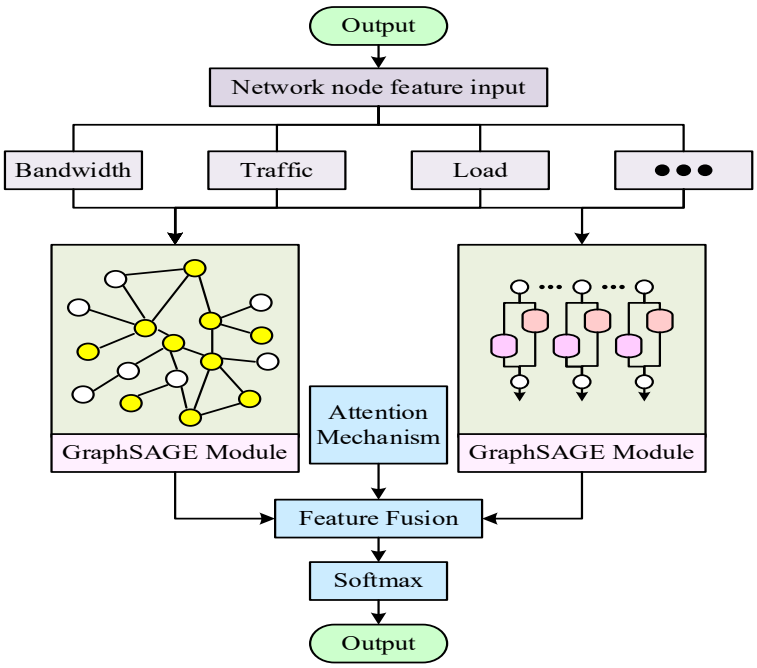


Fig. 3. GraphSAGE graph computation framework combined with BiLSTM

Finally, h_v^{fused} is input into the Softmax layer for node classification or other tasks. The Softmax layer maps the final node representation onto categories, as shown in Eq. (7).

$$P\left(y_k \middle| h_v^{fused}\right) \frac{\exp\left(W_k^T h_v^{fused} + b_k\right)}{\sum_{j=1}^{K} \exp\left(W_j^T h_v^{fused} + b_j\right)} \tag{7}$$

In Eq. (7), W_k and b_k are the weights and bias terms corresponding to category k, respectively. K is the total categories. According to Eq. (1), the sum of all output probabilities can be 1, which enables better network infrastructure management.

3.2. Incremental Clustering Algorithm and Network Infrastructure Management Method

The GraphSAGE graph computation framework, combined with BiLSTM, can effectively extract the temporal variation characteristics of network structure and device state, demonstrating good performance in node modeling and state evaluation. However, in real network management scenarios, with the continuous increase in the number of devices and the continuous addition of new nodes, the network structure frequently changes. Relying solely on static node classification methods is no longer sufficient to meet management needs, and challenges such as recognition delays and decreased maintenance efficiency are prone to occur. Therefore, the study further introduces an incremental clustering algorithm, which gradually updates the classification structure based on existing results, adapts to the dynamic changes of network nodes, and better supports the continuous optimization and efficient management of network facilities. Fig. 4 illustrates the incremental clustering algorithm.

In Fig. 4, the incremental clustering algorithm first takes the node embedding vectors generated by the GraphSAGE graph computation framework combined with BiLSTM as input and constructs the initial cluster structure of existing nodes in the network through the initial clustering stage (Oloruntoba, 2025). Subsequently, the system continuously runs the dynamic monitoring module to sense the added nodes or changes in node status in the network structure. Once a change is detected, the system immediately activates the similarity evaluation and attribution module to determine whether the node should be assigned to an existing cluster, or a new cluster needs to be created to accommodate it. In a specific implementation, the study adopts cosine similarity as a metric for the similarity evaluation process. Assuming that the node to be determined is represented as h_v and the center of the existing j-th cluster is represented as c_j, the cosine similarity between the two is defined as Eq. (8).

$$\mathrm{sim}\left(h_v, c_j\right) = \frac{h_v \cdot c_j}{\left\|h_v\right\| \cdot \left\|c_j\right\|} \tag{8}$$

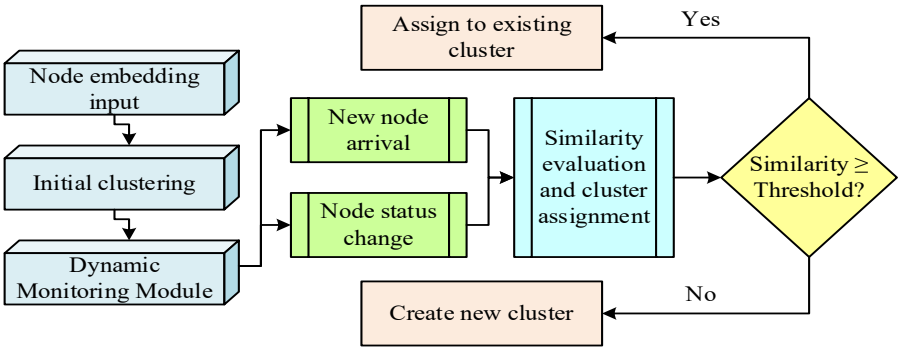


Fig. 4. The process of the incremental clustering algorithm

In Eq. (8), $\cdot$ represents the dot product of vectors. $\|\cdot\|$ represents the L2 norm of the vector. This similarity reflects the relative directional consistency between nodes and cluster centers in the feature space. After calculating the similarity of all clusters, the algorithm selects the cluster $c*$ with the highest similarity and sets a threshold parameter θ to determine whether the current node is close enough to an existing cluster to be included. The parameter θ is initially set by the 25th percentile of the similarity distribution between the training stage nodes and clusters, and then dynamically adjusted during the running process based on the frequency of new cluster generation, balancing clustering accuracy and cluster quantity control (Chen et al., 2025). When the similarity between h_v and $c*$ is the highest and exceeds the set threshold, it is classified into this cluster. If no cluster meets the conditions, it is considered a new category, a new cluster is created, and the structure is updated. To maintain the timeliness and stability of the entire process, after completing each incremental update of nodes, the algorithm recalculates the center of each cluster based on the latest cluster structure, as presented in Eq. (9).

$$\mu_k^{(t+1)} = \frac{1}{\left|C_k^{(t)}\right| + 1}\left(\sum_{u \in C_k^{(t)}} h_u + h_v\right) \tag{9}$$

In Eq. (9), $\mu_k^{(t+1)}$ signifies the center vector of the updated k-th cluster. $C_k^{(t)}$ signifies the set of nodes in the k-th cluster at time t. $\left|C_k^{(t)}\right|$ signifies the original quantity of nodes in the cluster. h_u signifies the embedding vector of each existing node in the cluster (Samani et al., 2023). Therefore, the dynamic evolution process based on the incremental clustering algorithm is illustrated in Fig. 5.

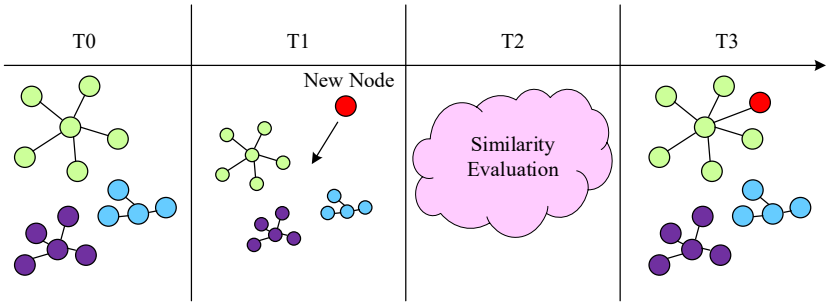


Fig. 5. The dynamic evolution process

In Fig. 5, at the initial time T0, the preliminary clustering of each node in the network has been completed, forming multiple node clusters with similar features. Subsequently, at time T1, a new node accesses the network, triggering a dynamic response. In the T2 stage, the algorithm evaluates the similarity between the new node and the existing cluster and determines whether it matches an existing cluster based on the set criteria. At time T3, if the similarity exceeds the threshold, the new node is classified into the most similar cluster, thereby achieving dynamic evolution and updating of the network structure. The entire process reflects the rapid adaptation of the incremental clustering algorithm to new node access and the self-adjusting ability of the clustering structure, ensuring the continuity and real-time management of network facilities. Finally, the proposed network infrastructure management method based on a graph computation framework and an incremental clustering algorithm is shown in Fig. 6.

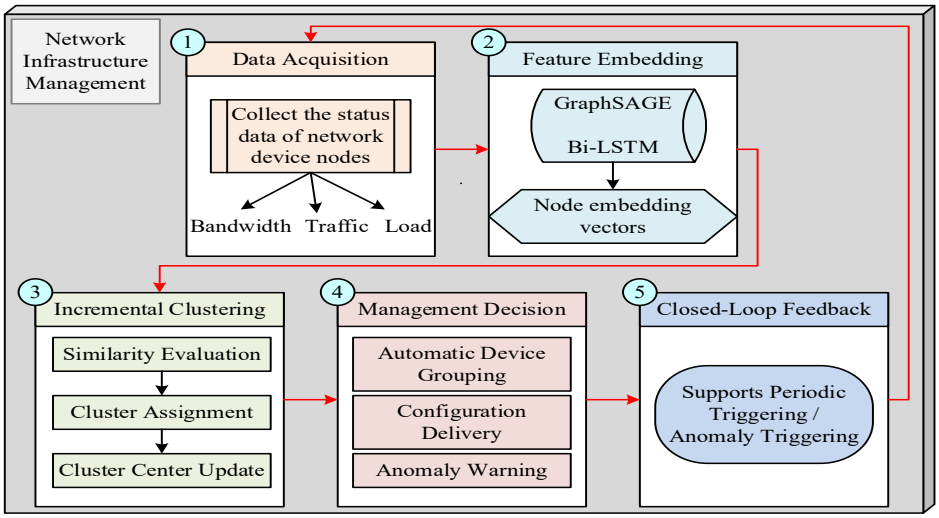


Fig. 6. Network infrastructure management method

In Fig. 6, this method is divided into 5 modules. First, the data collection module is responsible for collecting real-time operational status data from network device nodes, including key indicators such as bandwidth, traffic, and load, as inputs for subsequent modeling. Second, the feature embedding module inputs the collected data into the joint GraphSAGE and BiLSTM model to extract node embedding vectors that integrate structural relationships and temporal evolution features. Third, the incremental clustering module dynamically clusters node vectors based on this, supporting fast clustering updates after new node access or node state changes. Fourth, the management decision-making module automatically performs device grouping, configuration issuance, and abnormal warning operations based on the clustering structure, achieving efficient and intelligent management. Finally, the closed-loop feedback module supports periodic or abnormal triggering mechanisms, which reverse the management results to the data collection and feature modeling stages, enhancing the system's adaptability and responsiveness. The entire network infrastructure management method is implemented based on the TensorFlow platform. The graph computation module takes custom layers to construct GraphSAGE's neighbor sampling and feature aggregation. The temporal modeling part implements the BiLSTM structure through Keras. The feature fusion uses a dot product attention mechanism. The incremental clustering module uses dynamic graphs and efficient tensor operations to dynamically update node clustering, and finally completes node classification and management decision prediction under a unified framework.

4. Results

To verify the effectiveness and superiority of the network infrastructure management method that combines graph computation and incremental clustering, extensive experiments are conducted. The basic environment and settings are displayed in Table 1.

Table 1. Experimental settings

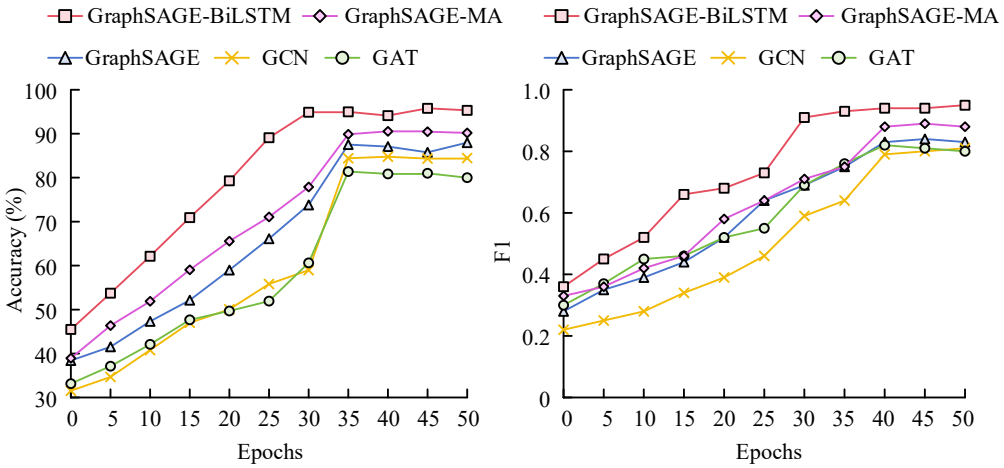
Hardware environment		Software environment	
Processor	Intel Core i9-11900K, 8 cores, 16 threads	Operating System	Ubuntu 20.04 LTS
Memory	64 GB DDR4	Deep Learning Framework	TensorFlow 2.8.0
GPU	NVIDIA RTX 3090, 24 GB GPU	Programming Language	Python 3.8
Storage	1 TB SSD (Solid State Drive)	Libraries and Dependencies	TensorFlow, NumPy, Matplotlib, Scikit-learn, NetworkX
Hyperparameter Settings			
GraphSAGE Model Parameters		Neighbor sampling: 10, Aggregation function: Mean Aggregator, Learning rate: 0.001, Batch size: 64	
BiLSTM Model Parameters		Hidden units: 128, Time steps: 50, Learning rate: 0.001, Optimizer: Adam	
Incremental Clustering Parameters		Initial clusters: 5, Maximum clusters: 100, Cluster update frequency: every 5 minutes, Threshold (similarity): 0.85	
Training Epochs		50 epochs (adjustable based on experimental data	

In Table 1, TensorFlow is an open-source deep learning framework developed by Google and widely used for building

and training machine learning models. This study chooses TensorFlow as the primary implementation platform not only for its maturity and well-established ecosystem, but also because of its significant advantages in graph computation optimization, distributed training, and incremental learning support. Its static computation graph mechanism aligns well with the graph neural network embedding process used in this study, and can efficiently process large-scale network topologies in parallel. Compared with other commonly used deep learning frameworks, TensorFlow also offers stronger advantages in production deployment and scalability. For example, PyTorch excels in flexibility and experimental iteration but relies more on external components for distributed deployment. Apache MXNet is advantageous in lightweight applications but has a less mature ecosystem and weaker community support. Considering computational performance, ecosystem support, and engineering applicability, TensorFlow is more suitable for the large-scale network infrastructure scenarios targeted in this study.

On the basis of Table 1, the study takes the public dataset UNSW-NB15 and the CICIDS 2017 dataset as experimental data sources. The former contains rich network traffic and attack type data, while the latter focuses on the normal and attack behavior of network traffic, both of which are in line with the network infrastructure management method. The dataset is divided into a training set and a testing set in an 8:2 ratio, and both undergo preprocessing steps such as cleaning, feature normalization, and normalization.

The study first verifies the node classification ability of the GraphSAGE graph computation framework combined with BiLSTM, and evaluates the accuracy of the method under device state changes. The basic GraphSAGE, GCN, Graph Attention Network (GAT), and GraphSAGE with Mean Aggregator (GraphSAGE-MA) are selected as comparison algorithms, and the results are shown in Fig. 7.



(a) Comparison of accuracy rates (b) Comparison of F1

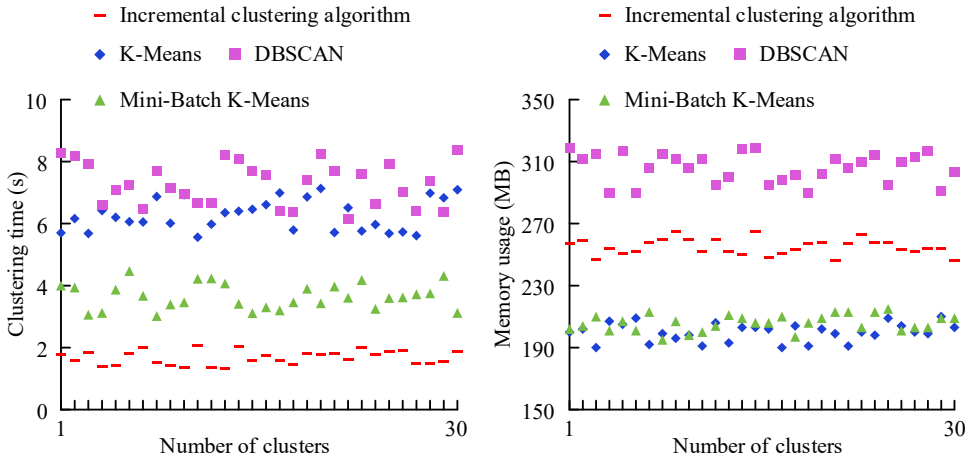
Fig. 7. Comparison of the experimental results

In Fig. 7(a), GraphSAGE-BiLSTM performed the best among all methods, with accuracy increasing from 45.48% to the highest 95.32%. The accuracy of basic GraphSAGE increased from 38.4% to a maximum of 87.97%, while GCN and GAT ultimately improved to 84.47% and 80.06%, respectively. The accuracy of GraphSAGE-MA increased from 38.92% to 90.19%. As shown in Fig. 7(b), the F1 value of GraphSAGE-BiLSTM consistently led the four comparison methods, reaching a maximum of 0.95 in the later stages of training. The node classification ability of the GraphSAGE graph computation framework combined with BiLSTM is stronger. Subsequently, the real-time updating capability of the incremental clustering algorithm in large-scale network environments is validated. The K-Means Clustering (K-Means), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and Mini-Batch K-Means clustering algorithm are selected as comparison methods, and the results are shown in Fig. 8.

According to Fig. 8(a), the average clustering time of the designed algorithm was 1.68 ± 0.23s, which was the shortest and had the smallest standard deviation. The algorithm has strong real-time updating ability and high stability in large-scale network environments. The clustering time of K-Means was relatively long, reaching 6.27 ± 0.51s, the clustering time of DBSCAN was 7.28 ± 0.69s, and the clustering time of Mini-Batch K-Means algorithm was 3.65 ± 0.42s. According to Fig. 8(b), the incremental clustering algorithm had an average memory usage of 255.00 ± 5.19MB, which was low and stable. K-Means had the lowest memory usage, averaging 199.87 ± 5.95MB. DBSCAN had the highest memory usage, reaching 306.10 ± 9.61MB. The Mini-Batch K-Means algorithm had a moderate memory usage, at 205.93 ± 5.25MB. The incremental clustering algorithm can effectively control memory usage while ensuring shorter clustering time. Furthermore, taking intra-cluster variance and inter-cluster distance as indicators to verify the clustering quality in the dynamic changes, the results are presented in Table 2.

According to Table 2, in four different dynamic network environments, the incremental clustering algorithm always maintained the lowest intra-cluster variance, ranging from 0.21 to 0.24, while maintaining the highest inter-cluster distance, ranging from 4.61 to 4.82. This indicates that it can effectively maintain the compactness and discriminability of clustering results under frequent device access, offline, and state change conditions, and has good clustering quality and system stability. In contrast, traditional methods such as K-Means, DBSCAN, and Mini-Batch K-Means performed relatively more

unstable in dynamic environments, with generally higher intra-cluster variances and lower inter-cluster distances. Furthermore, the application effectiveness of the proposed network infrastructure management method is verified. The Rule-Based Network Management system (RBNM) and Deep Forest-based Management method (DFM) are selected as comparison methods. The results are shown in Fig. 9.



(a) Comparison of time (b) Comparison of memory usage

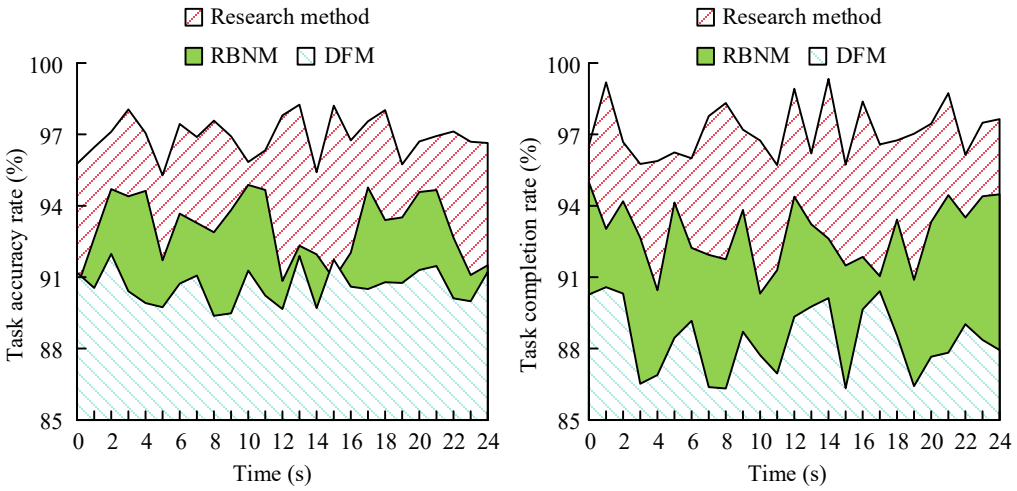
Fig. 8. Comparison results of incremental clustering algorithms

Table 2. Clustering quality in dynamic changes of network environment

Network environment	Clustering method	Intra-cluster variance	Inter-cluster distance
High Device Join Rate	Incremental Clustering Algorithm	0.22	4.76
	K-Means	0.35	3.52
	DBSCAN	0.41	3.28
	Mini-Batch K-Means	0.34	3.92
High Device Dropout Rate	Incremental Clustering Algorithm	0.21	4.82
	K-Means	0.33	3.41
	DBSCAN	0.39	3.12
	Mini-Batch K-Means	0.31	3.79
Dynamic Device State Changes	Incremental Clustering Algorithm	0.23	4.61
	K-Means	0.38	3.59
	DBSCAN	0.42	3.33
	Mini-Batch K-Means	0.31	3.85
Mixed Dynamic Environment	Incremental Clustering Algorithm	0.24	4.71
	K-Means	0.36	3.53
	DBSCAN	0.40	3.18
	Mini-Batch K-Means	0.30	3.87

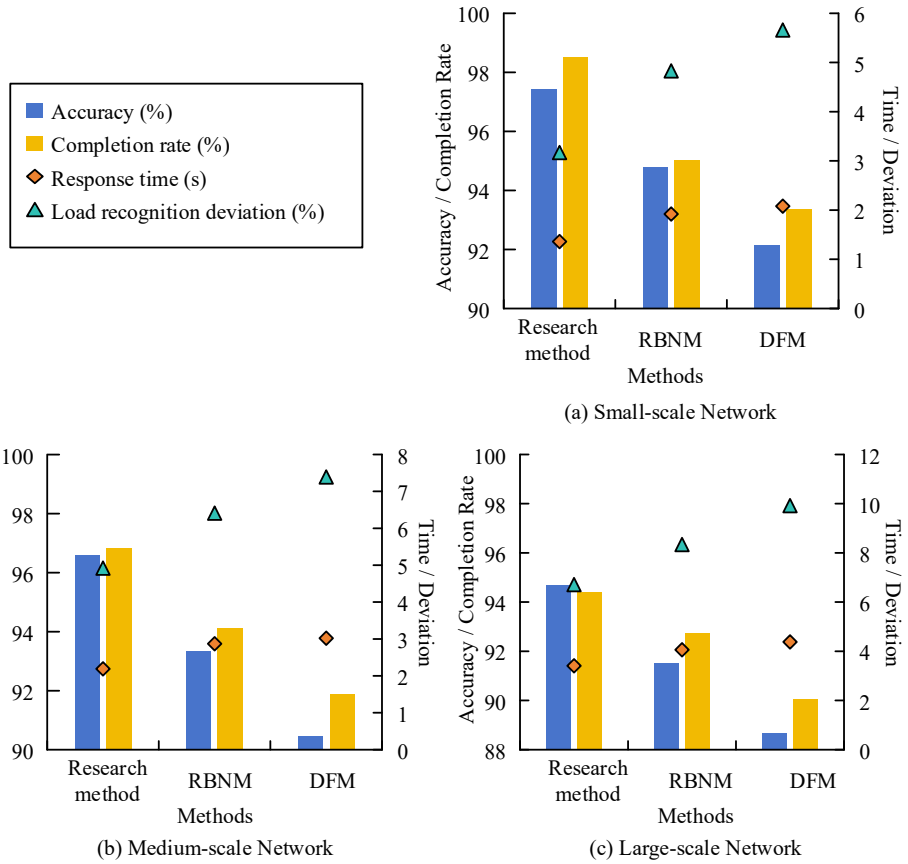
As shown in Fig. 9(a), the accuracy of the research method remained stable at 95.28%-98.25%, with small fluctuations, demonstrating good accuracy and stability. In contrast, the accuracy of RBNM fluctuated between 90.55% and 94.88%, within an acceptable range. The accuracy of DFM was relatively low, with a fluctuation range of 89.38%-91.98%, and its adaptability to dynamic tasks was slightly weaker. As shown in Fig. 9(b), the research method still performed the best in task completion rate, maintaining between 95.71% and 99.33%, with high completion efficiency and strong stability. The task completion rate of RBNM ranged from 90.31% to 94.99%, while the task completion rate of DFM fluctuated between 86.32% and 90.58%, with the lowest task completion rate. The method has better performance in network infrastructure management tasks. Finally, the scalability and generality of the research method in different scale network environments are validated, with small-scale networks (100 nodes), medium-scale networks (500 nodes), and large-scale networks (2,000

nodes) set up, respectively. The results are shown in Fig. 10.



(a) Comparison of task accuracy rates (a) Comparison of task completion rate

Fig. 9. Comparison of application performance



(a) Small-scale Network

(b) Medium-scale Network (c) Large-scale Network

Fig. 10. Results in network environments of different scales

From Fig. 10(a), in small-scale network environments, the research method performed excellently in task accuracy, reaching 97.83%, demonstrating stronger classification and state recognition capabilities. Meanwhile, the response time was only 1.28 seconds, which was the lowest among the three methods, indicating that this method had good real-time processing capability. The task completion rate was also as high as 98.76%, far exceeding that of RBNM and DFM, demonstrating high execution efficiency under light load conditions. In addition, the average load recognition deviation was only 1.34%, with the smallest error, which verified the accuracy and stability of this method in network management. In Figs. 10(b) and 10(c), although the overall performance has slightly decreased, the research method still leads in accuracy, response efficiency, and error control, verifying its good scalability and universality.

5. Conclusion

A management method based on graph computation and incremental clustering was proposed to address the key challenges in dynamic network infrastructure management. This method used GraphSAGE to implement neighbor sampling and feature aggregation of network topology, generating low dimensional node embeddings. Based on the temporal evolution characteristics of BiLSTM bidirectional modeling node states, an incremental clustering algorithm was designed to classify new nodes in real-time and dynamically update the cluster structure. Experimental data showed that the GraphSAGE-BiLSTM framework performed well in node classification tasks, with a final accuracy of 95.32%, which was 7.35%, 10.85%, and 15.26% higher than that of the basic GraphSAGE (87.97%), GCN (84.47%), and GAT (80.06%), respectively. The F1 value remained stable at 0.95. In terms of temporal performance, the incremental clustering algorithm exhibited significant advantages, with an average processing time of only 1.68 ± 0.23, while maintaining a stable memory usage of around 255MB. The clustering quality evaluation results showed that in high device joining rate scenarios, the algorithm could maintain an intra-cluster variance of 0.22 and an inter-cluster distance of 4.76. The method could still maintain a task accuracy of 95.28% and a load recognition deviation of 1.34% in large-scale network testing, with a response time controlled within 3.12s.

The proposed network infrastructure management method based on graph computation and incremental clustering embodies both theoretical innovation and practical value. From an engineering perspective, this approach can be embedded into the existing operation and maintenance systems of network operators or cloud service providers. By modeling the network topology and node states in real-time, it enables automated monitoring, anomaly detection, and dynamic resource scheduling, significantly reducing manual intervention costs while improving operational efficiency and response speed. In terms of deployment scenarios, the model can be applied to high-density and highly dynamic network environments such as smart city transportation systems, energy IoT, and large-scale cloud data centers. For example, in urban sensing systems, it can quickly respond to sensor node access and traffic fluctuations and automatically adjust network resources. In cloud data centers, it can support dynamic partitioning between edge nodes and backbone networks, fault prediction, and energy optimization. Furthermore, from a policy and strategic perspective, this study is of great significance for enhancing the resilience, security, and emergency response capacity of national network infrastructure, providing technical support and decision-making references for future 5G/6G infrastructure development, critical information infrastructure protection, and smart city governance. However, the research has limitations in clustering threshold dependence on empirical distributions and untested ultra large scale networks. In the future, adaptive threshold algorithms and distributed framework optimization can be used to improve application performance in scenarios such as 5G/IoT.

Funding

This research received no specific financial support from any funding agency.

Institutional Review Board Statement

Not applicable.

Declaration of Artificial Intelligence (AI) Tools

The author used Grammarly Academic solely for language editing and readability improvement. The author reviewed and verified all content and takes full responsibility for the accuracy and integrity of the manuscript.

References

Aziz, A., Ioannou, I., Lestas, M., Qureshi, K., Iqbal, A., and Vassiliou, V. (2023). Content-aware network traffic prediction framework for quality of service-aware dynamic network resource management. *IEEE Access*, 11, 99716-99733. doi: 10.1109/ACCESS.2023.3309002.

Bouchetara, M., Zerouti, M., and Zouambi, R. (2024). Leveraging artificial intelligence (AI) in public sector financial risk management: Innovations, challenges, and future directions. *EDPACS,* 69(9), 124-144. doi: 10.1080/07366981.2024.2377351.

Chen, Y., Ding, Y., Hu, Z., and Ren, Z. (2025). Geometrized task scheduling and adaptive resource allocation for large-scale edge computing in smart cities. *IEEE Internet of Things Journal*, 12(10), 14398-14419. doi: 10.1109/JIOT.2024.3525020.

Chu, N. H., Hoang, T., Nguyen, N., Phan, T., Dutkiewicz, E., Niyato, D., and Shu, T. (2023). Metaslicing: A novel resource allocation framework for metaverse. *IEEE Transactions on Mobile Computing*, 23(5), 4145-4162.doi: 10.1109/TMC.2023.3288085.

Daulat, S., Rokstad, M. M., Klein-Paste, A., Langeveld, J. and Tscheikner-Gratl, F. (2024). Challenges of integrated multi-infrastructure asset management: a review of pavement, sewer, and water distribution networks. *Structure and Infrastructure Engineering*, 20(4), 546-565.doi: 10.1080/15732479.2022.2119480.

Dwivedi, R., Tiwari, A., Bharill, N., Ratnaparkhe, M., Soni, R., Mahbubani, R. and Kumar, S. (2024). An incremental clustering method based on multiple objectives for dynamic data analysis. *Multimedia Tools and Applications*, 83(13), 38145-38165.doi: 10.1007/s11042-023-17134-7.

Fang, C., Hu, Z., Meng, X., Tu, S., Wang, Z., Zeng, D., Ni, W., Guo, S. and Han, Z. (2023). DRL-driven joint task offloading and resource allocation for energy-efficient content delivery in cloud-edge cooperation networks. *IEEE Transactions on Vehicular Technology*, 72(12), pp.16195-16207.doi: 10.1109/TVT.2023.3297362.

Gheisari, M., Hamidpour, H., Liu, Y., Saedi, P., Raza, A., Jalili, A., Rokhsati, H. and Amin, R. (2023). Data mining techniques for web mining: a survey. In *Artificial intelligence and applications*, 1(1), 3-10.

Guo, Q., Tang, F., and Kato, N. (2023). Resource allocation for aerial assisted digital twin edge mobile network. *IEEE Journal on Selected Areas in Communications,* 41(10), 3070-3079. doi: 10.1109/JSAC.2023.3310065.

Karthick Raghunath, K. M., Koti, M. S., Sivakami, R., Vinoth Kumar, V., NagaJyothi, G. and Muthukumaran, V. (2024). Utilization of IoT-assisted computational strategies in wireless sensor networks for smart infrastructure management. *International Journal of System Assurance Engineering and Management*, 15(1), 28-34.doi: 10.1007/s13198-021-01585-y.

Li, L., Xiong, K., Wang, G., and Shi, J. (2024). AI-Enhanced Security for Large-Scale Kubernetes Clusters: Advanced Defense and Authentication for National Cloud Infrastructure. *Journal of Theory and Practice of Engineering Science*, 4(12), 33-47. doi: 10.53469/jtpes.2024.04(12).07.

Litherland, J., and Andrews, J. (2025). System Petri net modelling for railway infrastructure asset management. *Infrastructure Asset Management,* 12(2), 110-128. doi: 10.1680/jinam.23.00046

Oloruntoba, O. (2025). AI-Driven autonomous database management: Self-tuning, predictive query optimization, and intelligent indexing in enterprise it environments. *World Journal of Advanced Research and Reviews*, 25(2), 1558-1580. doi: 10.30574/wjarr.2025.25.2.0534.

Pujiastuti, L., and Wahyudi, M. (2024). Advanced graph neural networks for dynamic yield optimization and resource allocation in industrial systems. *Jurnal Teknik Informatika CIT Medicom,* 16(2), 90-102. doi: 10.35335/cit.Vol16.2024.785.pp90-102.

Ren, Z. (2024). VGCN: An enhanced graph convolutional network model for text classification. Journal of Industrial Engineering and Applied Science, 2(4), 110-115. doi: 10.5281/zenodo.13118430.

Samani, Z. N., Mehran, N., Kimovski, D., Benedict, S., Saurabh, N., and Prodan, R. (2023). Incremental multilayer resource partitioning for application placement in dynamic fog. *IEEE Transactions on Parallel and Distributed Systems*, 34(6), 1877-1896. doi: 10.1109/TPDS.2023.3262695.

Soleymanian, M., Mashayekhi, H., and Rahimi, M. (2024). An incremental clustering algorithm based on semantic concepts. *Knowledge and Information Systems*, 66(6), 3303-3335. doi: 10.1007/s10115-024-02063-0.

Wang, J., Bai, L., Chen, J., and Wang, J. (2023). Starling flocks-inspired resource allocation for ISAC-aided green ad hoc networks. *IEEE Transactions on Green Communications and Networking,* 7(1), 444-454. doi: 10.1109/TGCN.2023.3234165.

Yang, H., Li, Z., and Qi, Y. (2024). Predicting traffic propagation flow in urban road network with multi-graph convolutional network. *Complex & Intelligent Systems*, 2024, 10(1), 23-35. doi: 10.1007/s40747-023-01099-z.

Zhang, X., Wu, W., Wang, J., and Liu, S. (2023). BiLSTM-based federated learning computation offloading and resource allocation algorithm in MEC. *ACM Transactions on Sensor Networks,* 19(3), 1-20. doi: 10.1145/3579824.

Lingji Wang obtained her Ph.D. in Philosophy in Management (2025) from SEGi University, Malaysia. Presently, she is an Associate Professor and the Associate Dean of Saxo Fintech College at Geely University and serves as the Head of the Intelligent Marketing Research Center and a Dual-Qualified Teacher. She was invited to preside over the provincial applied brand course “Automotive Brand Management and Practice” and four provincial-level research projects. She has contributed to the compilation of textbooks such as “Foundations of Entrepreneurship” and “Marketing Theory and Practice”. She guided students to win the Silver Award in the “Internet+” Innovation Competition and was recognized as the advising instructor for the first prize winner in the 13th National “CP Group Cup” competition. She has published articles in more than 10 international journals and conference proceedings. Her areas of interest include digital intelligent marketing, big data analytics and management, and financial technology.