scientific reports

Check for updates

OPEN

Jumping knowledge graph attention network for resource allocation in wireless cellular system

Qiushi Sun[1]✉, Zhou Fang[2], Yin Li[3]✉ & Ovanes Petrosian[3,4]

Next-generation wireless networks are characterized by two essential features: ubiquitous connectivity and high-speed data transmission. The realization of these features hinges on the development of rational resource allocation strategies to optimize the utilization of radio resources. This study addresses the beamforming design problem for downlink transmission in multi-cell cellular networks, with a focus on maximizing user data rates while adhering to stringent power constraints. To tackle this challenge, we propose a novel graph learning-based optimization framework that learns the mapping from channel states to beamforming vectors in an unsupervised manner. At the core of this framework is an attention-based graph neural network (GNN), which efficiently captures complex inter-node relationships by dynamically computing the importance of neighboring nodes. Furthermore, a jumping knowledge network is integrated to enhance structural representation learning, enabling the model to adaptively capture diverse neighborhood ranges for each node and mitigate the issue of over-smoothing. Extensive simulations demonstrate that the proposed algorithm significantly outperforms existing benchmark methods, exhibiting robust performance and strong generalization capabilities across a wide range of system parameter configurations.

The problem of optimal resource allocation lies at the heart of wireless communication system design[1]. From the user's perspective, mutual interference among multiple users represents a primary bottleneck, constraining the achievement of high-speed and reliable data transmission over communication links. Effective resource allocation strategies can significantly mitigate such interference, thereby addressing user demands and improving the overall quality of service (QoS)[2]. From the perspective of system operators, the efficient allocation of resources enhances the utilization of expensive hardware infrastructure, leading to substantial reductions in operational expenditures. Consequently, the optimal allocation of power, waveform, and other critical radio resources is indispensable for driving advancements in wireless communications[3].

Beamforming is a crucial technology in wireless communication systems that enhances the efficiency of radio resource utilization[4]. In recent years, beamforming design has emerged as a research hotspot in the field of wireless communications[5]. Traditional methods employ convex optimization theory to design beamforming vectors, aiming to find an upper bound on system performance. Numerical simulation methods require a series of repetitive calculations performed iteratively. Examples of such methods include water filling (WF)[6], weighted minimum mean square error (WMMSE)[7], fractional programming (FP)[8], and heuristic algorithms[9]. Unfortunately, dynamic wireless communication systems with time-varying channel characteristics demand real-time optimization of optimal beamforming. As the dimensionality of the optimization variables increases, these iterative algorithms become computationally intensive, consuming substantial time and rendering them challenging to apply directly[10].

In recent years, the integration of deep learning-based methodologies with traditional optimization algorithms has garnered considerable attention, particularly in addressing challenges inherent to wireless communication systems[11]. This interest is largely driven by the universal approximation theorem, which establishes the capacity of deep neural networks to approximate arbitrary functions, thereby offering a robust framework for optimizing wireless systems[12]. For example, labeled training datasets are often generated using weighted least mean square error techniques, which are the foundation for supervised learning approaches[13,14]. In such frameworks, a

[1]School of Management, Harbin Institute of Technology, Harbin 150001, China. [2]Faculty of Applied Mathematics and Control Processes, St.Petersburg University, St.Petersburg 198504, Russia. [3]School of Mathematics, Harbin Institute of Technology, Harbin 150001, China. [4]Department of Infocommunication Technologies, ITMO University, Saint-Petersburg 197101, Russia. ✉email: sunqiushicn@outlook.com; Dr.liyin@hit.edu.cn

multilayer perceptron (MLP) is trained to map channel state information to optimal beamforming vectors, enabling the effective design of beamforming strategies in interference-prone environments. Alternatively, unsupervised learning techniques have been employed to directly optimize system-level objective functions, yielding optimal beamforming solutions without the need for labeled data[15,16]. Despite the demonstrated efficacy of deep learning in optimization tasks, significant challenges remain, particularly when extending models trained on non-Euclidean data structures to dynamic and evolving scenarios. These limitations underscore the potential constraints of MLPs in achieving end-to-end beamforming learning in complex and adaptive environments.

The connectivity relationships among devices, network topology, and channel states in wireless systems are inherently characterized by non-Euclidean data structures. GNNs offer a powerful framework for incorporating such topological information into neural network architectures, enabling the efficient processing of graph-structured data[12]. As a result, GNNs have emerged as a promising approach to enhance the generalization capabilities and scalability of deep learning-based optimization methods in wireless communication systems[17]. For instance, message-passing GNNs have been successfully applied to power control and beamforming design, effectively addressing the challenge of maximizing the aggregate rate in device-to-device (D2D) networks[18]. Similarly, the interference channel network (ICNet) has been employed in multiple-input-single-output (MISO) systems with statistical channel state information (CSI), directly mapping statistical CSI to beamforming vectors to solve the problem of maximizing energy efficiency under interruption constraints[19]. Furthermore, the branch-and-bound framework has been integrated with GNNs to tackle beamforming and antenna selection problems. In this approach, the GNN module skips intermediate states of the search tree, significantly reducing computational complexity while maintaining optimality[20].

As an emerging paradigm, GNNs extend the capabilities of deep learning to graph-structured data, making them particularly well-suited for applications in wireless communications. Despite their potential, GNNs are not without limitations. As network depth increases, these models are susceptible to overfitting of parameters and excessive smoothing of learned embeddings, which can degrade their performance[21]. Consequently, the development of GNN frameworks that enhance learning capacity while mitigating the effects of transition smoothing remains a significant and unresolved challenge in the field of wireless optimization[22]. Motivated by these ongoing research challenges, this paper explores GNN-based methodologies to address critical issues in resource allocation and interference management within multi-cellular networks. The primary contributions of this work are summarized as follows:

- We propose a generalized framework to address the wireless resource allocation problem by leveraging a graphical representation of multi-cellular networks. Specifically, the data rate maximization problem is reformulated as a graph optimization task, where direct communication links are modeled as nodes and interference links as edges. Building on this representation, we develop a GNN model that integrates message-passing and attention mechanisms. This model is designed to learn the complex mapping from the system state to the optimal resource allocation variables, enabling efficient and scalable optimization in multi-cellular environments.
- We improve the model's learning capability by integrating attention aggregation with jump knowledge networks. The proposed method adaptively adjusts the computation of attention coefficients by layer depth through the ordered gating mechanism of Long short-term memory (LSTM), considering both nearest-neighbor and non-nearest-neighbor nodes from a global perspective. This strategy enhances the representativeness of the embeddings and mitigates the potential over-smoothing problem.
- We validate the proposed method by comparing the performance with several benchmark schemes based on convex optimization and learning through simulations. The experimental results indicate that the proposed method demonstrates superior performance, scalability across different network sizes, and adaptability to various system parameter settings. Additionally, ablation experiments confirm combining attention aggregation and jumping knowledge linking effectiveness.

The rest of this paper is structured as follows. Section 2 reviews related work on graph attention networks and jumping knowledge networks. Section 3 presents the system formulation and graphical representation of the wireless system. Section 4 introduces the GNN-based beamforming optimization framework, detailing the attention and jumping knowledge connection modules. Section 5 provides simulation results, and Section 6 concludes the paper.

Related works

Graph attention network

Classical GNNs aggregate information from a node's neighbors with equal weights through message passing, which may fail to capture the varying influence of different nodes[23]. The attention mechanism, a feature aggregation scheme, addresses this limitation by assigning weights to each neighbor of the current node, allowing updated features to emphasize more important input information[24]. In wireless networks, the attention mechanism can be leveraged to manage interfering relationships between users, thereby optimizing network performance. For example, GAT can be trained using labeled datasets generated by convex optimization algorithms to learn the mapping from channel state information to beamforming vectors[25]. Integrated satellite-terrestrial networks use a GAT-based approach to optimize beamforming vectors between satellites to reduce interference[26]. Additionally, edge-feature enhanced GAT is proposed to learn resource allocation strategies for heterogeneous D2D networks in an unsupervised manner[27].

Over-smoothing of graph neural network

In the field of deep learning, the depth of neural networks has traditionally been a critical factor influencing model performance. For example, in computer vision tasks, convolutional neural networks (CNNs) frequently employ architectures comprising tens or even hundreds of layers[28]. These deeper networks are capable of learning highly complex feature representations, which significantly enhance their predictive capabilities. However, in the context of GNNs, a fundamental challenge arises from the extensive connectivity between nodes, which can lead to the phenomenon of over-smoothing. As node features undergo multiple updates, they tend to become indistinguishable, ultimately degrading the performance of GNN models, particularly in deeper architectures[29].

To mitigate this issue, researchers have proposed a variety of strategies. One widely adopted approach involves the incorporation of jumping connections, which facilitate the propagation of fine-grained information across layers[30,31]. Another promising direction is the use of advanced aggregation functions, such as neighbor aggregation and feature propagation mechanisms, which aim to preserve node-specific information by selectively aggregating data from multi-hop neighbors in each layer[32,33]. Additionally, specialized regularization techniques and node sampling methods have been developed to counteract over-smoothing and improve the overall performance of GNN models.

In this study, we propose a novel GNN-based resource allocation framework for wireless communication systems. Our approach leverages jumping knowledge networks combined with LSTM-based attention aggregation mechanisms to enhance the performance of the GNN model. By integrating these techniques, we aim to address the challenges of over-smoothing and information loss, thereby improving the efficiency and accuracy of resource allocation in wireless networks.

System model and problem formulation

In this section, we introduce the system's basic parameters and mathematical model and formulate the problem of maximizing the system's sum rate as a graph optimization problem.

System model

As shown in Figure 1, we consider a downlink multi-cell communication system where each base station serves M single antenna devices through a shared spectrum band. The resource allocation challenge in cellular networks is associated with the interference multiple access channel (IMAC) setting. In this configuration, the base station at the center of the cellular network provides data services to all user equipment (UEs) within its coverage area. However, adjacent direct links within the same cell cause intra-cell interference to UEs, while direct links to UEs in adjacent cells result in inter-cell interference. Denote the collection of BSs as $\mathcal{N} = \{1, \ldots, N\}$, and denote the collection of UEs as $\mathcal{K} = \{1, \ldots, K\}$. link nk represents the direct link from the n-th BS to the k-th UE. The received signal of k-th UE can be formulated as:

$$y_{nk} = \underbrace{\mathrm{h}_{nk,nk}^{H}\mathrm{x}_{nk}s_{nk}}_{\text{desired signal}} + \underbrace{\sum_{k'\neq k}\mathrm{h}_{nk,nk'}^{H}\mathrm{x}_{nk'}s_{nk'}}_{\text{intra-cell interference}} + \underbrace{\sum_{n'\neq n}\sum_{k'\neq k}\mathrm{h}_{nk,n'k'}^{H}\mathrm{x}_{n'k'}s_{n'k'}}_{\text{inter-cell interference}} + z_{nk}, \tag{1}$$

where $\mathrm{h}_{nk,nk}$ denote the channel state information of the direct link between nk; $\mathrm{h}_{nk,nk'}$ denote the channel state information of the intra-cell interference link between nk' and nk; $\mathrm{h}_{nk,n'k'}$ denote the channel state information of the inter-cell interference link between $n'k'$ and nk; x_{nk} denotes the corresponding beamforming vector. $s_{nk} \sim \mathcal{U}(0,1)$ is the transmit signal. $z_{nk} \sim \mathcal{N}\left(0, \sigma^2\right)$ is the additive white Gaussian noise (AWGN).

Then the signal-to-interference-plus-noise ratio (SINR) of k-th UE can be expressed as:

$$\gamma_{nk} = \frac{\mathrm{h}_{nk,nk}\mathrm{x}_{nk}}{\sum_{k'\neq k}\mathrm{h}_{nk,nk'}\mathrm{x}_{nk'} + \sum_{n'\neq n}\sum_{k}\mathrm{h}_{nk,n'k'}\mathrm{x}_{n'k'} + \sigma^2}, \tag{2}$$

The data rate of direct link nk can be expressed in terms of normalized bandwidth as:

$$C_{nk} = \log_2\left(1 + \gamma_{nk}\right). \tag{3}$$

The main objective of this study is to determine the ideal beamforming design to optimize the data rate of the UE while adhering to the maximum power limit for each transmitter. The problem provided can be expressed as follows:

$$\begin{aligned} &\max_{p_{nk}, w_{nk}} \sum_{n=1}^{N}\sum_{k=1}^{K}\log_2\left(1+\gamma_{nk}\right) \\ &\text{s.t. } 0 \leqslant \ p_{nk} \ \leqslant p_{\max}, \forall n \in \mathcal{N}, k \in \mathcal{K}. \end{aligned} \tag{4}$$

Problem formulation

A graph is a data structure inherently defined in non-Euclidean space, renowned for its ability to model complex data patterns and intricate associative relationships. This structure provides a highly flexible framework for data representation, making it particularly well-suited for capturing the inherent connectivity and relational dependencies present in real-world systems[34]. Owing to its versatility, graphs have found extensive applications

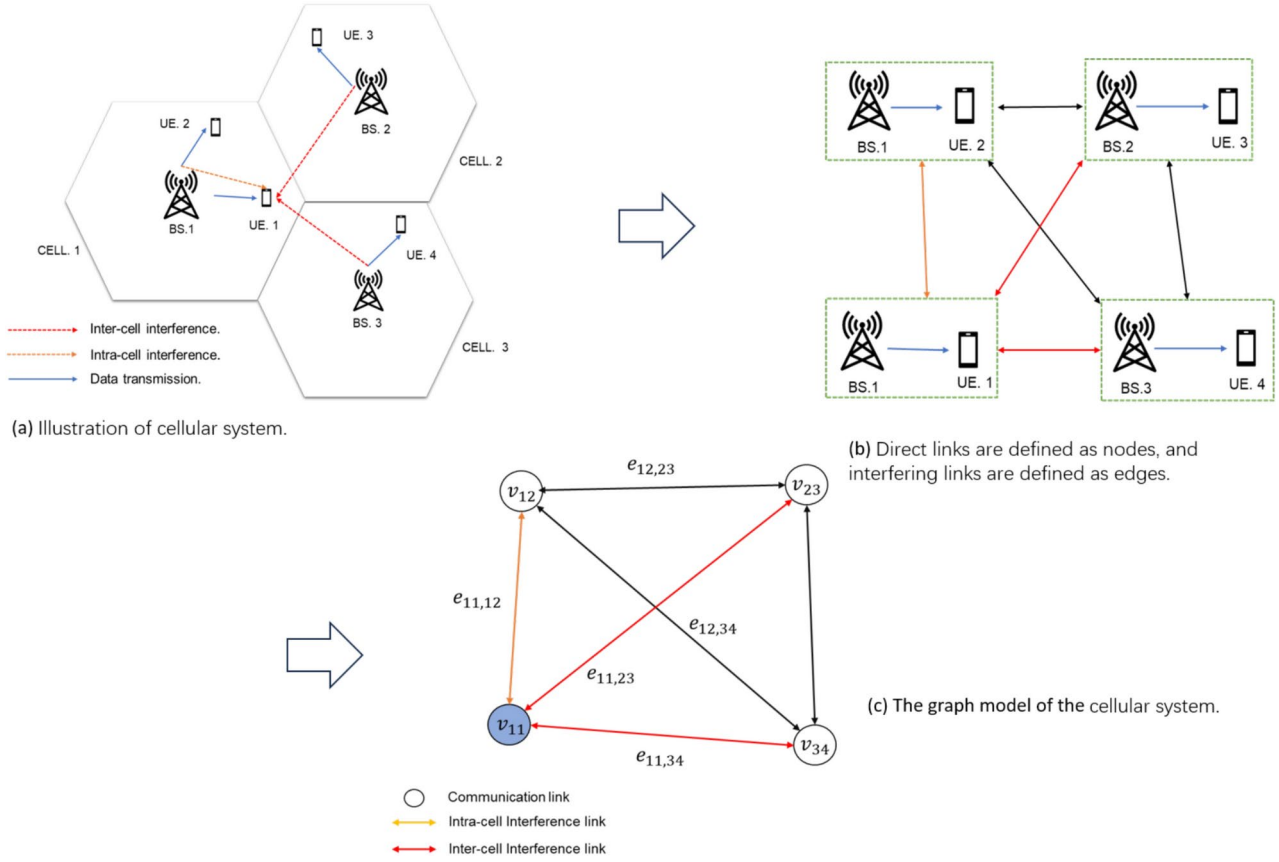

Fig. 1. Cellular system and its graph model. We interpret the graph model from the perspective of the direct link between the 1-st base station and the 1-st user. The circular vertex v_{11} represents this direct communication link, corresponding to the primary signal path within the system, distinct from the interference links depicted separately. The yellow solid arrows indicate intra-cell interference affecting v_{11}, while the red solid arrows represent inter-cell interference impacting v_{11}. The black solid arrows illustrate interference among other direct links.

across a diverse array of domains, including machine learning, deep learning, and beyond. By leveraging the inherent properties of graphs, practitioners can effectively address a wide range of practical challenges, significantly enhancing both the efficiency and accuracy of data processing tasks.

A graph is formally defined by the tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges connecting pairs of nodes. In the context of interference graphs, all direct links are assumed to be of equal significance. Consequently, unweighted bidirectional graphs are employed to model network topologies, ensuring that all links are treated uniformly to interference considerations. Within this framework, each direct link from a base station (BS) to user equipment (UE) is represented as a node $v_i \in \mathcal{V}$, with the corresponding node features denoted by $\vec{h}_i$. These features encapsulate critical information such as the channel state and Gaussian noise. Interference links between distinct UEs are represented as edges in the graph. The topological structure of the graph is succinctly captured by the adjacency matrix $\mathrm{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where A_{ij} corresponds to the element in row i and column j of the matrix. Specifically, $\mathrm{A}_{ij} = 1$ if a connection exists from node i to node j, and $\mathrm{A}_{ij} = 0$ otherwise. The adjacency matrix serves as a powerful tool for encoding the relational structure of the graph, facilitating advanced analysis and computation tasks related to interference management and network optimization.

We define the set of node features in the interference graph as $\mathrm{H} = \left\{\vec{h}_i\right\}$, and the set of beamforming vectors as $\mathrm{X} = \{\mathrm{X}_i\}$. According to the definition of graph G, the objective function can be transformed into a graph optimization problem:

$$\begin{aligned} &\min_{\mathrm{X}} f\left(\mathrm{X}, \mathrm{H}, \mathrm{A}\right) \\ s.t. \quad &q\left(\mathrm{X}, \mathrm{H}, \mathrm{A}\right) \geq 0, \\ &f\left(.\right) = -E\left[\sum_{n=1}^{N}\sum_{k=1}^{K} \log_2\left(1+\gamma_{nk}\right)\right], \end{aligned} \tag{5}$$

$$q(\cdot) = p_{\max} - \|\mathrm{x}_{nk}\|^2 \geqslant 0, \tag{6}$$

where $f(\cdot)$ is the negative value of the objective function and $q(\cdot)$ represents the constraint conditions. Denote the optimal solution of a graph optimization problem by X^*. Our objective is to develop and train a GNN-based model capable of learning a mapping from (H, A) to X^*, thus obtaining a solver that achieves real-time and near-optimal solutions. In our formulation, the original problem, which aims to maximize the sum data rate subject to power constraints, is reformulated into a graph optimization framework where the beamforming vector design is learned through a graph-based model. Specifically, the transformation involves the following key mappings

Methods

In this section, we introduce the proposed algorithm's overall architecture and each part's principles.

Overall architecture of JGAT

We propose a novel architecture, termed Jumping Knowledge Graph Attention Network (JGAT), to address the graph optimization problem defined above. JGAT takes node features and an adjacency matrix as inputs and generates beamforming vectors as outputs. The architecture consists of two key components: an attention network and a jumping knowledge aggregation layer.

In the attention network, node embedding features are iteratively updated by aggregating information from neighboring nodes using attention coefficients. These coefficients quantify the relative importance of each neighbor, enabling the model to dynamically prioritize relevant nodes and capture intricate relationships within the graph structure. This mechanism enhances the ability of the GAT model to effectively encode topological information and node relationships. The initial input features are denoted as $\vec{h}_i^{(\mathrm{ini})}$, while the hidden embeddings output by each attention layer l are represented as $\vec{h}_i^{(l)}$.

The jumping knowledge aggregation layer integrates hidden embeddings from all layers through an LSTM-based attention mechanism, which learns the mapping from the output embedded features to the beamforming vectors. The final aggregated output embeddings are denoted as $\vec{h}_i^{(\mathrm{fin})}$. The overall framework of the proposed JGAT is illustrated in Figure 2.

Graph attention network

The Graph Attention Network (GAT) leverages an attention mechanism to compute the weights of different neighboring nodes within a node's first-degree neighborhood, thereby enabling more expressive and nuanced learning of graph representations. To illustrate, consider layer l: each node i in the graph is associated with a feature vector $\vec{h}_i^{(l-1)}$ from the previous layer, which typically serves as the node's input feature at layer l. Each feature vector is first subjected to a linear transformation via a learnable weight matrix $\mathrm{W} \in \mathbb{R}^{d_v \times d'_v}$, where d_v and d'_v denote the input and output feature dimensions, respectively. This transformation projects the feature vectors into a higher-dimensional space, enhancing the model's capacity to capture complex feature interactions and relationships.

An attention mechanism is introduced to compute the importance of node $j \in \mathcal{N}_i$ to node i. This mechanism utilizes an attention function $\vec{a}$, which performs on node feature pairs. Typically, this function is applied using a single-layer feed-forward neural network, parameterized the weight vector, and outputs a scalar attention coefficient indicating the importance of node j to node i. Consequently, the attention coefficient of node i's adjacent node j can be calculated by the following equation:

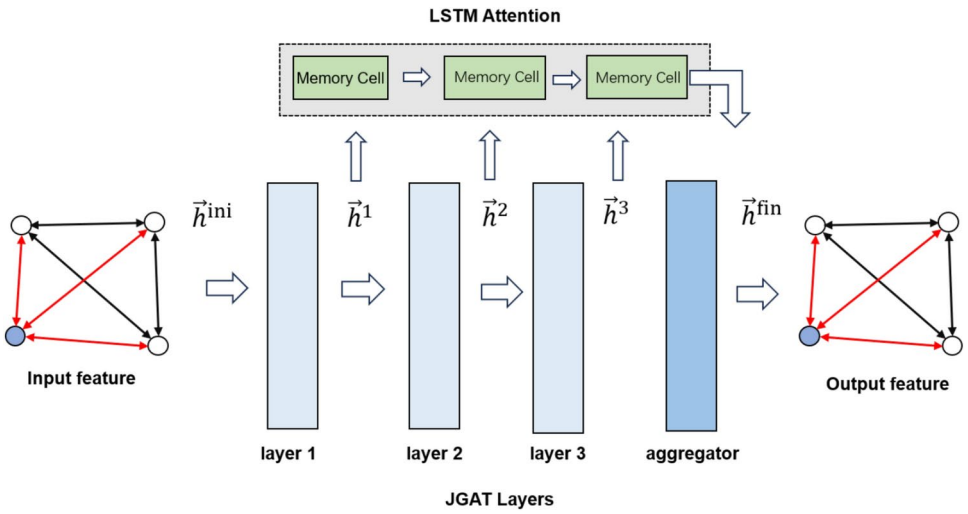

Fig. 2. Illustration of the overall architecture of JGAT.

$$c_{ij} = \sigma\left(\vec{a}^T\left(W\vec{h}_i \,\|\, W\vec{h}_j\right)\right) \tag{7}$$

where σ is a nonlinear activation function, LeakyReLU is used here. Then, the softmax function is employed to normalize the initial attention coefficients to eradicate differences in magnitude and to facilitate a comparison of the importance of different nodes.

$$\alpha_{ij}^{(l)} = \text{soft max}\left(c_{ij}\right) = \frac{\exp\left(\sigma\left(\vec{a}^T\left(W\vec{h}_i^{(l-1)} \,\|\, W\vec{h}_j^{(l-1)}\right)\right)\right)}{\sum_{k\in\mathcal{N}_i, k\neq j}\exp\left(\sigma\left(\vec{a}^T\left(W\vec{h}_i^{(l-1)} \,\|\, W\vec{h}_k^{(l-1)}\right)\right)\right)} \tag{8}$$

Finally, the output of the features by node i at layer l can be obtained by a weighted linear combination of the transformed feature vectors of its first-order adjacent nodes (including itself), where the weights are the normalized attention coefficients. This is expressed as follows:

$$\vec{h}_i^{(l)} = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}^{(l)}W\vec{h}_j^{(l-1)}\right) \tag{9}$$

To smooth the learning process and enhance model performance, GAT typically employs multi-head attention. By running the attention mechanism in parallel, each head independently focuses on the input features, producing distinct output feature vectors. These features are then averaged before being passed to the next layer:

$$\vec{h}_i^{(l)} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^{k,(l)}W^k\vec{h}_j^{(l-1)}\right) \tag{10}$$

Jumping knowledge network

The node embeddings can be organized into a sequence of samples, denoted as $\left\{\vec{h}^{(1)}, \vec{h}^{(2)}, \ldots, \vec{h}^{(l)}\right\}, \cdots$. The sequential information is stored in the memory cells of the LSTM network. These memory cells, corresponding to different hidden layers, form a conveyor belt-like structure through linear interactions, facilitating the continuous and efficient flow of information across layers.

The LSTM memory cell is designed to learn temporal dependencies between sequence elements by dynamically updating its memory based on previously observed elements. To regulate the flow of information, a gating mechanism is introduced, which controls the addition or removal of information from the memory cell. As illustrated in Figure 3, these gates selectively permit information to propagate through the network and are implemented using sigmoid neural network layers combined with element-wise multiplication operations. The LSTM cell comprises three distinct gates: the input gate, the forget gate, and the output gate. Each gate serves a critical function in safeguarding and managing the cell state. The input gate determines the extent to which new information is incorporated into the cell state, while the forget gate decides which information should be discarded. The output gate regulates the information that is passed to the next time step. The unitary state vector, which encapsulates the memory and temporal dependencies between sequence elements, ensures efficient information processing and retention throughout the network.

The forget gate decides which information to discard or keep from the cell state. Take the l th memory cell, for example. It's responsible for processing information H_{l-1} from a previously hidden state and information $\vec{h}^{(t)}$ from the current input passed through a sigmoid function σ:

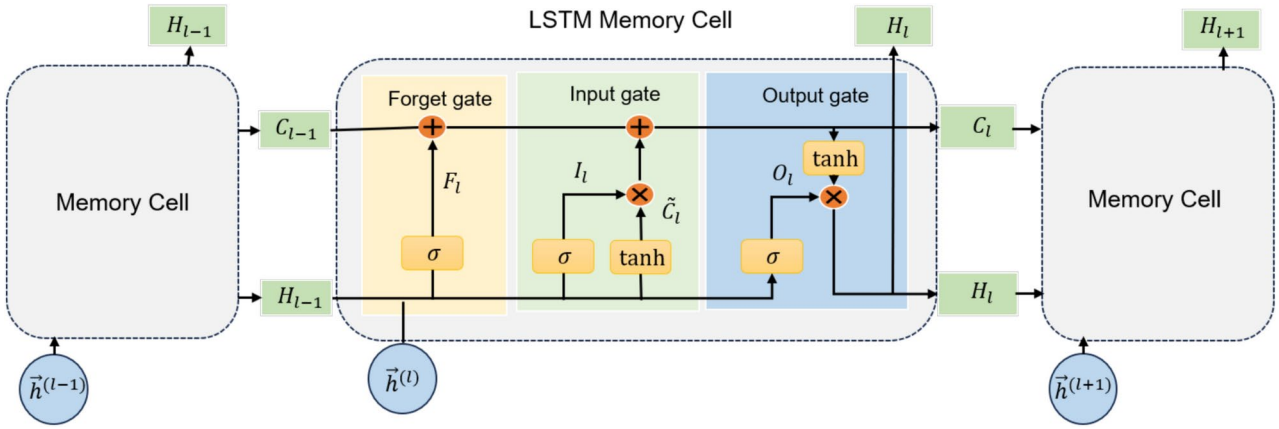

Fig. 3. Illustration of the overall Jumping knowledge network with LSTM aggregating layer.

$$F_l = \sigma\left(W_F\left[H_{l-1}, \vec{h}^{(l)}\right] + b_F\right), \tag{11}$$

where W and b are the learnable parameters of the gate. The output vector F_l ranges between 0 and 1, where values closer to 0 indicate the information to be forgotten and values closer to 1 indicate the information to be retained.

The input gate updates the cell state by evaluating the importance of the incoming information. The previous hidden state and the current are input through the sigmoid function:

$$I_l = \sigma\left(W_I\left[H_{l-1}, \vec{h}^{(l)}\right] + b_I\right). \tag{12}$$

Next, the hidden state and current input are passed to the tanh function to condition the network. The tanh output $\tilde{C}_l$ is then multiplied with the output of the sigmoid as the output and added to the state:

$$\tilde{C}_l = \tanh\left(W_C\left[H_{l-1}, \vec{h}^{(l)}\right] + b_C\right). \tag{13}$$

Subsequently, the output vector of the input gate is combined with the output of the forget gate to create an update to the cell state. The old cell state C_{l-1} is updated to the new cell state C_l by:

$$C_l = F_l \circ C_{l-1} + I_l\tilde{C}_l. \tag{14}$$

Finally, the output gate determines the hidden state of the output. The previous hidden state and the current input are initially passed through a sigmoid function. The updated cell state is subsequently passed through a tanh function, and the output of the tanh function is multiplied by the sigmoid production to determine the hidden state information. The new cell state and the new hidden state are then transferred to the next memory cell.

$$O_t = \sigma\left(W_O\left[H_{l-1}, \vec{h}^{(l)}\right] + b_O\right), \tag{15}$$

$$H_t = O_t \circ \tanh\left(C_t\right). \tag{16}$$

The embeddings of each sequence element learned by the LSTM cell are represented as the hidden features of node output by each neural network layer. The attention mechanism determines each node's most significant neighborhood range *i* by attention coefficient β_i^t. This coefficient evaluates the importance of the hidden features learned in the *l*-th layer for node *i*. We apply a bidirectional LSTM to obtain forward and backward hidden features, $\vec{h}_{i,f}^{(l)}$ and $\vec{h}_{i,b}^{(l)}$, respectively. These hidden features are processed through a single-layer neural network to compute layer-specific attention coefficients for each node. The calculation process of $\beta_i^{(l)}$ is as follows:

$$c_i^{(l)} = \sigma\left(W\left(\vec{h}_{i,f}^{(l)} \middle\| \vec{h}_{i,b}^{(l)}\right)\right), \tag{17}$$

$$\beta_i^{(l)} = \frac{\exp\left(c_i^{(l)}\right)}{\sum_{l=1}^{L}\exp\left(c_i^{(l)}\right)} = \frac{\exp\left(\sigma\left(W\left(\vec{h}_{i,f}^{(l)} \middle\| \vec{h}_{i,b}^{(l)}\right)\right)\right)}{\sum_{l=1}^{L}\exp\left(\sigma\left(W\left(\vec{h}_{i,f}^{(l)} \middle\| \vec{h}_{i,b}^{(l)}\right)\right)\right)}, \tag{18}$$

$$\sum_{l=1}^{L}\beta_i^{(l)} = 1 \tag{19}$$

We can then obtain the optimal embedding for the final output by computing an attention-weighted linear combination of the hidden embeddings:

$$\vec{h}_i^{\text{fin}} = \sum_{l=1}^{L}\beta_i^l\vec{h}_i^{(l)} \tag{20}$$

The LSTM-attention mechanism is node-adaptive, as each node has a distinct attention score.

Initialize: The graph representation $G = (V, E)$.
Output: Model's parameter θ

for $l = 1$ to L **do**
  for Each node $v_i \in V$ **do**
    Collect $\mathcal{N}_i$;
    Calculate the attention coefficient α_{ij}^{l},
    Calculate node embedding
$\vec{h}_i^{(l)} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^{k,(l)}W^k\vec{h}_j^{(l-1)}\right)$;
  end for
end for
Calculate LSTM importance coefficient $\beta^{(l)}$.
Calculate final node features $\vec{h}^{\text{fin}} = \sum_{l=1}^{L}\beta^{(l)}\vec{h}^{(l)}$.
Minimize the loss function $\mathcal{L}(\theta)$;
Update θ.

Algorithm 1. Jumping Knowledge Graph Attention Network

Experiments results

Simulation environment

Consider a cellular network scenario comprising multiple neighboring cells, each with a base station located at its center, jointly managed and scheduled by a centralized collaboration system. Each cell is arranged in a regular hexagonal grid pattern, overlapping coverage areas to form an efficient coverage region[35]. The radius of each cell is 1 kilometer. The maximum power limit for the communication link is 33 dBm. The path loss is modeled as $120.9 + 37.6\log_{10}(d) + 10\log_{10}(z)$(in dB), where d is the distance in meters, and $z$is a lognormal random variable with standard deviation of 8 dB[36]. The small-scale channel fading follows Rayleigh distribution. The noise power is set at −114 dBm.

During the training process, we set the number of epochs to 500. The UE and the small-scale channel are randomly generated for each training sample. We use the ADAM optimizer with a learning rate of $\gamma = 10^{-4}$to update the neural network parameters[37]. The model is trained using unsupervised learning, where the negative value of the objective function is used as the loss function. The training set's system setting is a wireless network with N=9 cells, each BS serving M=4 UEs. After training, we tested the average performance of 100 samples from different scenarios. All simulation results are based on a computing platform configured with an Intel i9-14900 K CPU, Nvidia RTX 4090 GPU, and 64 GB of RAM, with the deep learning portion implemented by PyTorch and Deep Graph Library (DGL).

Benchmarks

Using the system setup described above, we compared the proposed method with several benchmark algorithms. All algorithms were trained and tested on the same dataset in the same environment to ensure a fair comparison. The benchmark algorithms are described below:

- WMMSE[7]: An iterative optimization algorithm that optimizes system performance by minimizing the weighted mean square error. It is typically used for multiuser multi-antenna systems' channel estimation and resource optimization problems. In our experiments, we repeat the computation 50 times for each network implementation and select the best result as the upper limit of performance.
- GAT[25]: A GNN model based on an attention mechanism that learns importance weights between different nodes by introducing attention. GAT can learn feature representations at the node level and can model relationships between nodes flexibly.
- GCN[38]: A GNN model based on graph convolutional operations that updates the representation of each node by aggregating information from neighboring nodes. The update rule of GCN is a weighted summation of the features of each node with those of its neighboring nodes, with the weights determined by the adjacency matrix.

Simulation results

Learning efficiency

We first compare the performance of JGAT and the benchmark algorithms trained on datasets of different sizes. As shown in Figure 4, JGAT slightly outperforms GAT. Specifically, JGAT outperforms GAT by 1.25%, 1.68%, 2.63%, and 3.04% when the number of samples in the training set is 10, 100, 1,000, and 10,000, respectively. When they achieve the same sum rate, the number of training samples required for JGAT is about 10% of that for GAT. When the number of training samples increases, the performance gap becomes more pronounced, highlighting the superiority of the proposed skip connection mechanism in handling sample complexity.

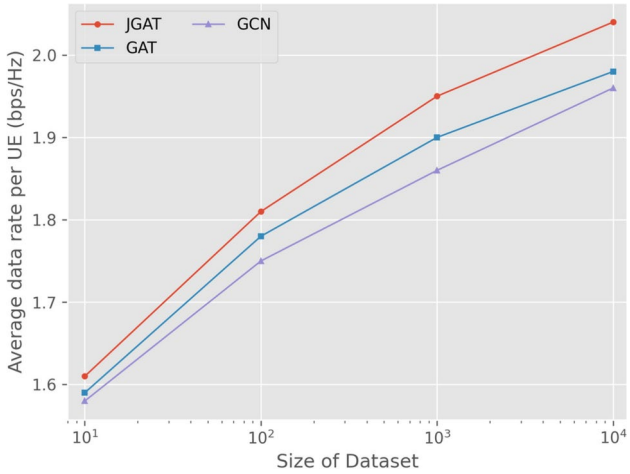

Fig. 4. Average data rate per UE versus the training set size.

Data rate (bps/Hz)				
	Layer			
Model	2	3	4	5
JGAT	1.92	1.99	2.02	2.04
GAT	1.93	1.98	1.27	0.91
GCN	1.96	1.23	1.08	0.89

Table 1. Average data rate per UE versus GNN depth.

MAD				
	Layer			
Model	2	3	4	5
JGAT	0.624	0.475	0.233	0.079
GAT	0.428	0.256	0.107	0.016
GCN	0.384	0.219	0.085	0.009

Table 2. MAD value for each layer.

Ablation experiments

We verify the effectiveness of the LSTM jumping connection mechanism through ablation experiments. Table 1 demonstrates the performance of the GNN model at different depths. It is observed that with increasing depth, the performance of benchmark GAT and GCN sharply declines after reaching the optimum at layers 3 and 2, respectively, due to the over-smoothing problem analyzed above. However, LSTM jumping connections can alleviate the over-smoothing problem, and the expressiveness of JGAT slowly enhances with increasing depth until it reaches the optimum. The experimental results verify the effectiveness and importance of jumping connections.

To explore the over-smoothing problem in GNN depth, we introduce a quantitative index of node feature similarity: the mean average distance (MAD)[39]. MAD is used to compute the mean distance between node features in the graph, with values ranging from 0 to 1, where smaller values indicate more similar node features. Table 2 demonstrates the MAD values of the GNN models for node features at different layers. The MAD values of JGAT decrease more slowly, indicating that jumping connections enhance learning by slowing down the phenomenon of over-smoothing, allowing for a deeper number of network layers. This slower decrease in MAD values suggests that the learning ability of the neural network is roughly positively correlated with depth.

Generalizability to network scale

After training on specific network instances, we test the model's performance in scenarios with different network scales and user densities (N=9, M=4). As shown in Table 3, when the number of cells in the system is kept constant, the density of UEs in the cells increases, leading to more severe intra-cell interference between the communication links, and resulting in degraded performance of the links in all models. Similarly, as shown in Table 4, when the density of UEs in the cell is kept constant, the increasing number of cells results in severe

Data rate (bps/Hz)						
	UE					
Model	1	2	3	4	5	6
JGAT	3.52	2.52	2.23	2.04	1.82	1.73
GAT	3.43	2.45	2.11	1.98	1.77	1.64
GCN	3.39	2.43	2.10	1.96	1.75	1.61
WMMSE	3.14	2.23	1.96	1.82	1.59	1.47

Table 3. Average data rate per UE versus user density.

Data rate (bps/Hz)			
	Cell		
Model	4	9	16
JGAT	2.75	2.04	1.59
GAT	2.48	1.98	1.52
GCN	2.42	1.96	1.49
WMMSE	2.28	1.82	1.48

Table 4. Average data rate per UE versus number network scale.

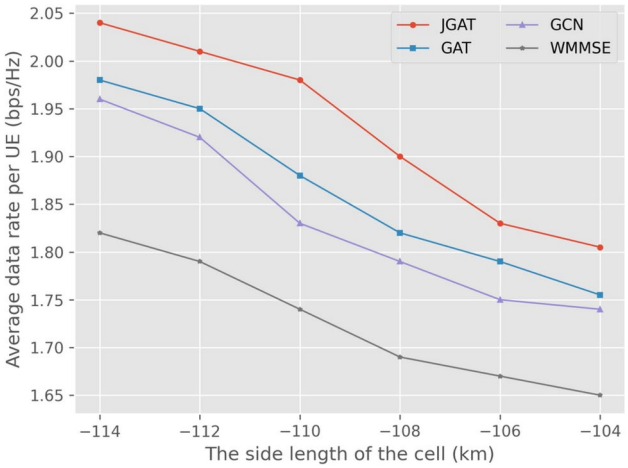

Fig. 5. Average data rate per UE versus cell size.

inter-cell interference generated by the communication links of neighboring cells, causing a decrease in link performance in all models. The experimental results show that JGAT can handle network scenarios where the number of links increases or decreases and maintains superior performance. When the statistical features of the test data deviate from the training data, JGAT shows strong migration ability.

Generalizability to system settings

We test the generalization ability of JGAT under different system settings. The default hexagonal cell length is 1 km, and we zoom in and out at equal intervals of 0.2 km. The distance between the base station and its users ranges from 50 m to 200 m. As the cell size increases, the interference between the cells decreases. Figure 5 demonstrates that the advantage of JGAT over the benchmark remains stable. To further demonstrate JGAT's generalization performance under varying noise power levels, we increment the noise power settings in 2 dBm intervals. From Figure 6, it can be seen that JGAT consistently maintains a higher sum rate compared to the benchmark, proving the advantage of the proposed algorithm in generalizing to scenarios with different system settings.

Robustness to imperfect CSI

We design experiments to evaluate the performance of JGAT in imperfect channel state information (ICSI) scenarios. Imperfect CSI occurs in real-world wireless networks due to factors such as the complexity and real-time nature of channel conditions, making it challenging for the receiver to obtain complete and accurate channel state information. To simulate this, we randomly set the features of some nodes to null values according to a

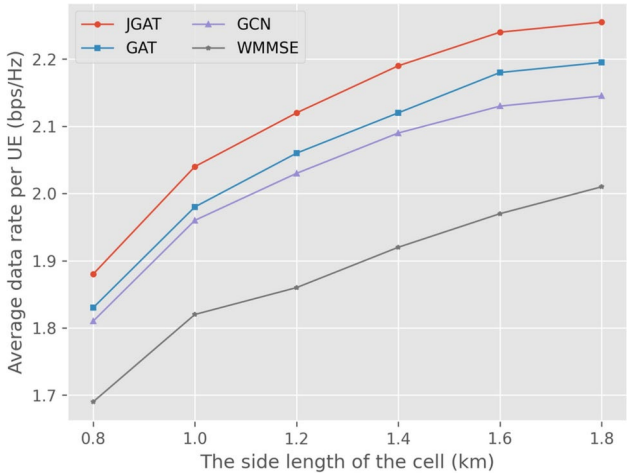

Fig. 6. Average data rate per UE versus noise level.

RP						
	CSI					
Model	1.0	0.8	0.6	0.4	0.2	0.0
JGAT	100%	98.2%	96.4%	92.7%	83.4%	51.9%
GAT	100%	97.6%	96.1%	92.4%	83.1%	52.3%
GCN	100%	97.5%	95.8%	92.3%	82.9%	52.4%

Table 5. Relative performance to CSI proportion.

specified ratio and calculate the performance of the benchmark algorithm under both complete and incomplete CSI conditions. This allows us to assess the robustness of the algorithm. As shown in Table 5, JGAT maintains its performance lead even as the ratio of available CSI information decreases. Remarkably, it achieves a relative performance (RP) of 83.4% when only 20% of the CSI information is avaliable. These experiment results verify the robustness of JGAT in handling ICSI scenarios, demonstrating its ability to perform reliably under less-than-ideal conditions in practical wireless networks.

Conclusions

This paper investigates a GAT-based unsupervised learning framework for optimizing beamforming designs to maximize the average user rate. By modeling the interference channel in a multiuser cellular network as a graph, the mapping of CSI to beamforming vectors is directly learned. To enhance the expressiveness of the model and alleviate the transition smoothing problem, we apply the attention mechanism with the hopping connection mechanism to the message-passing phase. The experimental results demonstrate that the proposed method maintains its advantages in scenarios with different system settings, achieving a higher average rate than the benchmark methods. Additionally, the proposed JGAT exhibits good generalization ability across different cell sizes, user densities, and interference levels and remains robust in partially missing CSI.

Data availability

The datasets used and analyzed during the current study are available from the corresponding author on reasonable request.

Received: 30 July 2024; Accepted: 29 April 2025
Published online: 20 May 2025

References

1. Lin, M. & Zhao, Y. Artificial intelligence-empowered resource management for future wireless communications: A survey. *China Communications* **17**, 58–77 (2020).
2. Bhushan, N. et al. Network densification: the dominant theme for wireless evolution into 5g. *IEEE Communications Magazine* **52**, 82–89 (2014).
3. Strinati, E. C. et al. Reconfigurable, intelligent, and sustainable wireless environments for 6g smart connectivity. *IEEE Communications Magazine* **59**, 99–105 (2021).
4. Xu, Y., Gui, G., Gacanin, H. & Adachi, F. A survey on resource allocation for 5g heterogeneous networks: Current research, future trends, and challenges. *IEEE Communications Surveys & Tutorials* **23**, 668–695 (2021).
5. Zheng, B., You, C., Mei, W. & Zhang, R. A survey on channel estimation and practical passive beamforming design for intelligent reflecting surface aided wireless communications. *IEEE Communications Surveys & Tutorials* **24**, 1035–1071 (2022).

6. Ning, B., Chen, Z., Chen, W. & Fang, J. Beamforming optimization for intelligent reflecting surface assisted mimo: A sum-path-gain maximization approach. *IEEE Wireless Communications Letters* **9**, 1105–1109 (2020).
7. Lin, T., Cong, J., Zhu, Y., Zhang, J. & Letaief, K. B. Hybrid beamforming for millimeter wave systems using the mmse criterion. *IEEE Transactions on Communications* **67**, 3693–3708 (2019).
8. Shen, K. & Yu, W. Fractional programming for communication systems—part i: Power control and beamforming. *IEEE Transactions on Signal Processing* **66**, 2616–2630 (2018).
9. Sun, Q., Wu, H. & Petrosian, O. Optimal power allocation based on metaheuristic algorithms in wireless network. *Mathematics* **10**, 3336 (2022).
10. Ahmad, I. et al. The challenges of artificial intelligence in wireless networks for the internet of things: Exploring opportunities for growth. *IEEE Industrial Electronics Magazine* **15**, 16–29 (2020).
11. Al Kassir, H. *et al.* A review of the state of the art and future challenges of deep learning-based beamforming. *IEEE Access* **10**, 80869–80882 (2022).
12. Hornik, K., Stinchcombe, M. & White, H. Multilayer feedforward networks are universal approximators. *Neural networks* **2**, 359–366 (1989).
13. Chowdhury, A., Verma, G., Rao, C., Swami, A. & Segarra, S. Unfolding wmmse using graph neural networks for efficient power allocation. *IEEE Transactions on Wireless Communications* **20**, 6004–6017 (2021).
14. Zhu, M., Chang, T.-H. & Hong, M. Learning to beamform in heterogeneous massive mimo networks. *IEEE Transactions on Wireless Communications* **22**, 4901–4915 (2022).
15. Ahmed, I., Shahid, M. K., Khammari, H. & Masud, M. Machine learning based beam selection with low complexity hybrid beamforming design for 5g massive mimo systems. *IEEE Transactions on Green Communications and Networking* **5**, 2160–2173 (2021).
16. Dinh-Van, S., Hoang, T. M., Trestian, R. & Nguyen, H. X. Unsupervised deep-learning-based reconfigurable intelligent surface-aided broadcasting communications in industrial iots. *IEEE Internet of Things Journal* **9**, 19515–19528 (2022).
17. He, S. et al. An overview on the application of graph neural networks in wireless networks. *IEEE Open Journal of the Communications Society* **2**, 2547–2565 (2021).
18. Zhang, X. *et al.* Scalable power control/beamforming in heterogeneous wireless networks with graph neural networks. In *2021 IEEE Global Communications Conference (GLOBECOM)*, 01–06 (IEEE, 2021).
19. He, C. *et al.* Icnet: Gnn-enabled beamforming for miso interference channels with statistical csi. *IEEE Transactions on Vehicular Technology* (2024).
20. Shrestha, S., Fu, X. & Hong, M. Optimal solutions for joint beamforming and antenna selection: From branch and bound to graph neural imitation learning. *IEEE Transactions on Signal Processing* **71**, 831–846 (2023).
21. Chen, D. et al. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *In Proceedings of the AAAI conference on artificial intelligence* **34**, 3438–3445 (2020).
22. Waikhom, L. & Patgiri, R. A survey of graph neural networks in various learning paradigms: methods, applications, and challenges. *Artificial Intelligence Review* **56**, 6295–6364 (2023).
23. Zhou, J. et al. Graph neural networks: A review of methods and applications. *AI open* **1**, 57–81 (2020).
24. Velickovic, P. et al. *Graph attention networks. stat* **1050**, 10–48550 (2017).
25. Li, Y., Lu, Y., Zhang, R., Ai, B. & Zhong, Z. Deep learning for energy efficient beamforming in mu-miso networks: A gat-based approach. *IEEE Wireless Communications Letters* **12**, 1264–1268 (2023).
26. Liu, R., Fu, Y., Hu, B., Wang, H. & Chen, S. A gat based robust beamforming method in satellite-terrestrial integrated network. In *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, 1–6 (IEEE, 2024).
27. Sun, Q., He, Y. & Petrosian, O. Resource allocation in heterogeneous network with node and edge enhanced graph attention network. *Applied Intelligence* **54**, 4865–4877 (2024).
28. Tateno, K., Tombari, F., Laina, I. & Navab, N. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6243–6252 (2017).
29. Wu, X., Ajorlou, A., Wu, Z. & Jadbabaie, A. Demystifying oversmoothing in attention-based graph neural networks. *Advances in Neural Information Processing Systems* **36** (2024).
30. Xu, K. *et al.* Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, 5453–5462 (PMLR, 2018).
31. Yang, F., Zhang, H., Tao, S. & Hao, S. Graph representation learning via simple jumping knowledge networks. *Applied Intelligence* **52**, 11324–11342 (2022).
32. Peng, Y., Guo, J. & Yang, C. Learning resource allocation policy: Vertex-gnn or edge-gnn? *IEEE Transactions on Machine Learning in Communications and Networking* (2024).
33. Eschenburg, K. M., Grabowski, T. J. & Haynor, D. R. Learning cortical parcellations using graph neural networks. *Frontiers in neuroscience* **15**, 797500 (2021).
34. Dong, X., Thanou, D., Toni, L., Bronstein, M. & Frossard, P. Graph signal processing for machine learning: A review and new perspectives. *IEEE Signal processing magazine* **37**, 117–127 (2020).
35. Meng, F., Chen, P., Wu, L. & Cheng, J. Power allocation in multi-user cellular networks: Deep reinforcement learning approaches. *IEEE Transactions on Wireless Communications* **19**, 6255–6267 (2020).
36. Remy, J.-G. & Letamendia, C. *LTE standards* (John Wiley & Sons, 2014).
37. Chandriah, K. K. & Naraganahalli, R. V. Rnn/lstm with modified adam optimizer in deep learning approach for automobile spare parts demand forecasting. *Multimedia Tools and Applications* **80**, 26145–26159 (2021).
38. Nguyen, H. B., Van Hai, D., Bui, T. D., Chau, H. N. & Nguyen, Q. C. Multi-channel speech enhancement using a minimum variance distortionless response beamformer based on graph convolutional network. *International Journal of Advanced Computer Science and Applications* **13** (2022).
39. Zhang, X., Xu, Y., He, W., Guo, W. spsampsps Cui, L. A comprehensive review of the oversmoothing in graph neural networks. In *CCF Conference on Computer Supported Cooperative Work and Social Computing*, 451–465 (Springer, 2023).

Acknowledgements

This work was supported by the Ministry of Science and Higher Education of the Russian Federation, Goszadanie (State Assignment) No. FSER-2025-0013

Author contributions

Q.S. was responsible for algorithm design, conducting comparative experiments, analyzing the results, and preparing the first draft. Z.F. was responsible for preparing the revised draft. O.P. was responsible for directing the research. Y.L. provided project support and supervised the completion of the manuscript. All authors reviewed the manuscript.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to Q.S. or Y.L.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.