Dynamic Scheduling Algorithm for Edge-cloud Collaborative Computing Resources Based on 5G NR-U and Distributed Large Model

Defeng Duan[1,2,a*], Hong Liu[2,b], Liyun Huang[2,c], Songlei Zhang[2,d] and Bo Meng[3,e]

[1]*School of Science, Wuhan University of Technology, Wuhan 430000, Hubei, China*

[2]*Digital Intelligence Department, China Mobile Communications Group Co., Ltd., Beijing 10000, Beijing, China*

[3]*DADO Beijing Delivery Department, AsiaInfo Technology (China) Co., Ltd., Beijing 10000, Beijing, China*

[a]*Email:* duandefeng@chinamobile.com

[b]*Email:* liuhong@chinamobile.com

[c]*Email:* huangliyun@chinamobile.com

[d]*Email:* zhangsonglei@chinamobile.com

[e]*Email:* mengbo@asiainfo.com

[*]*Corresponding Author*

Abstract: With the deployment of 5G NR-U (5th Generation New Radio in Unlicensed Spectrum) in unlicensed spectrum and the rise of sparse large-scale models (exemplified by Switch Transformers) in edge computing, edge-cloud collaborative training faces the dual challenges of dynamic spectrum contention and heterogeneous load distribution. This paper addresses the coupled uncertainty of time-varying 5G NR-U channels and the random arrival of expert tasks in MoE (Mixture of Experts) by introducing a dynamic scheduling algorithm for edge-cloud collaboration based on MADDPG (Multi-Agent Deep Deterministic Policy Gradient). This algorithm models each edge server as an agent and, based on local observations, jointly decides on task offloading targets (terminal, edge, or cloud), expert cache replacement, and 5G NR-U contention window parameters through a continuous action space, achieving cross-layer collaborative optimization of communication and computing cache resources. A centralized critic network is used for global training, and the reward function integrates average iterative latency, communication energy consumption, system throughput, and load balancing. On a simulation platform comprising eight heterogeneous edge nodes and one cloud center, three scenarios—baseline, high-interference, and burst load—were implemented for validation. Experimental results show that the proposed MADDPG algorithm outperforms benchmark algorithms such as CO (Cloud-Only), GO (Greedy Offloading), SEP (Static Expert Partitioning), DDPG (Deep Deterministic Policy Gradient), COMA (Counterfactual Multi-Agent Policy Gradients), and MAPPO (Multi-Agent Proximal Policy Optimization) in terms of average iteration latency (minimum 78 ms), system throughput (maximum 185 K tokens/s), expert load balancing (minimum standard deviation 15.3), and communication energy consumption (minimum 52 J). It demonstrates superior adaptability, robustness, and collaborative efficiency under scenarios of severe channel fluctuations and burst load, providing a feasible intelligent scheduling solution for efficient training of sparse large models in dynamic edge environments.

Keywords: Edge-cloud Collaboration; Computing Resources; Dynamic Scheduling; 5G NR-U; Distributed Large-scale Model

1. Introduction

Sparse Mixture-of-Experts (MoE) models, represented by the Switch Transformer, provide a feasible path for building large-scale models with trillions of parameters through their sparse activation characteristics. It is important to clarify that the term "large-scale model" herein refers to models with massive parameter counts and sparse architectures, which is conceptually related to—but not synonymous with—Large Language Models (LLMs). While LLMs often employ such large-scale sparse architectures, this study focuses on the generic challenges of distributed training for large-scale models with dynamic expert activation patterns, rather than language-specific tasks. This sparse activation characteristic transforms the training process into a highly dynamic and fine-grained computation graph. This characteristic makes it deeply intersect with emerging 5G NR-U and edge computing architectures [1-2]: 5G NR-U can provide high-capacity wireless backhaul using unlicensed spectrum, while end-edge-cloud collaboration can provide heterogeneous computing power nearby [3]. However, the dynamic spectrum sharing between 5G NR-U and systems such as Wi-Fi leads to drastic random fluctuations in channel bandwidth and access latency [4-5]; at the same time, the expert activation mode in the MoE model is highly dependent on the input data, making the computational load non-uniformly distributed in time and space [6-7]. These two time-varying uncertainties, which originate from the communication and computation layers respectively, are coupled with each other to form a complex stochastic optimization environment. In this environment, traditional static task allocation or scheduling strategies based on average indicators can be severely ineffective, making it difficult to guarantee the end-to-end efficiency and stability of distributed training tasks. Therefore, designing an intelligent scheduling mechanism that can

sense and collaboratively optimize wireless resources and computing tasks in real time has become a key challenge for efficiently deploying sparse large models in dynamic edge networks. Despite facing spectrum competition and channel fluctuations in unlicensed frequency bands, 5G NR-U's high bandwidth, low deployment cost, and deep integration with edge computing make it a key candidate technology for supporting distributed large-scale model training. In real-world scenarios, model training does not require absolute channel stability, but rather a scheduling mechanism that can sense and adapt to network dynamics, ensuring training efficiency amidst channel fluctuations through task migration, cache optimization, and access parameter adjustments. Therefore, conducting large-scale model training in a 5G NR-U environment has clear engineering feasibility and practical necessity; the key lies in designing an intelligent scheduling mechanism capable of handling uncertainty.

In the field of distributed computing for large models, existing research mainly focuses on data centers, optimizing the training of single large models on stable, high-speed interconnected clusters through techniques such as model parallelism and pipelined parallelism. To adapt to edge environments, some works have begun to explore model partitioning and task offloading, offloading specific layers or modules of the Transformer to edge nodes [8-9]. However, these methods are mostly geared towards traditional dense models, with coarse-grained scheduling units that fail to fully utilize the fine-grained parallel potential inherent in independent expert modules in the MoE model [10-11]. More importantly, such research implicitly assumes an ideal or bandwidth-stable backhaul network, failing to fully consider the intermittent connectivity and bandwidth fluctuations caused by dynamic spectrum contention that advanced wireless access technologies such as 5G NR-U inevitably face in actual deployments. Directly applying scheduling strategies based on ideal network assumptions to 5G NR-U-enabled edge environments results in a significant performance degradation due to the inability to adapt to channel dynamics.

Recent advances in edge artificial intelligence (AI) have underscored the paradigm shift toward distributed intelligence at the network periphery, where integrating 5G connectivity with edge computing unlocks unprecedented opportunities for latency-sensitive and privacy-preserving applications [12]. The pursuit of sustainable and efficient edge AI systems has spurred extensive research into optimization techniques spanning computation offloading, model compression, and energy-aware resource allocation [13-14]. In particular, edge AI has demonstrated transformative potential across diverse domains, including healthcare for early disease detection [15], autonomous vehicular networks [18], and real-time Industry 4.0 automation within 5G environments [17]. The emergence of large language models has further catalyzed the evolution toward autonomous edge AI, enabling connected intelligence through collaborative end-edge-cloud frameworks [16]. Concurrently, the synergy between 5G wireless communications and edge computing has been actively explored for applications such as teaching quality evaluation [19] and microservice orchestration in next-generation networks [20]. To manage the growing complexity of these systems, AI-assisted network slicing frameworks have been proposed to dynamically allocate resources and ensure quality of service in 5G-and-beyond networks [21]. Collectively, these efforts contribute to a growing taxonomy of edge AI methodologies, yet challenges persist in achieving robust, scalable, and real-time decision-making under dynamic network conditions [22]. This positions our work at the confluence of 5G NR-U dynamics and distributed large model training, addressing the critical gap in joint communication-computation scheduling.

Previous research has largely focused on dense model scheduling within stable data center networks or edge task offloading based on ideal network assumptions, failing to adequately characterize channel fluctuations caused by dynamic spectrum sharing in 5G NR-U and the data-dependent expert activation patterns in the MoE model. This study addresses these limitations by proposing a novel dynamic scheduling framework that jointly optimizes communication, computing, and caching resources in 5G NR-U enabled edge-cloud systems. The main contributions of this work are threefold: First, we establish a unified stochastic optimization framework that formally characterizes the coupling between two distinct sources of uncertainty—the dynamic channel contention in 5G NR-U and the time-varying expert task generation in sparse MoE training—thereby providing a rigorous foundation for the joint scheduling problem. Second, we develop a multi-agent reinforcement learning algorithm based on MADDPG that enables continuous action space decisions, allowing edge servers to simultaneously determine task offloading destinations, expert cache replacement strategies, and 5G NR-U contention window parameters. This continuous action design achieves fine-grained cross-layer optimization that discrete-action methods cannot attain. Third, we explicitly incorporate a load balancing penalty term into the reward function to incentivize global collaborative strategies, and systematically validate the framework's superior performance under dynamic interference and burst load scenarios. The proposed solution provides crucial methodological support for the efficient deployment of sparse large-scale models in real-world edge environments.

2. Methods

2.1 Modeling and Problem Formalization

2.1.1 Cooperative Network Architecture and Dynamic Channel Modeling

The end-to-end efficiency of distributed sparse large model training is fundamentally constrained by the transmission efficiency of task data and model parameters between the edge, cloud, and terminal layers [23-25]. The physical basis of this study is a three-layer collaborative computing network consisting of smart terminals, 5G NR-U enabled edge servers,

and a central cloud. The edge servers also function as 5G NR-U base stations, forming a dynamic wireless access network [26-27]. The core challenge of this system is that the wireless links carrying critical transmissions operate in unlicensed frequency bands that need to coexist with Wi-Fi. The resulting dynamic spectrum contention and co-channel interference cause the channel quality to exhibit strong random time-varying characteristics. The interference between the edge-cloud collaborative network and 5G NR-U is shown in Figure 1.

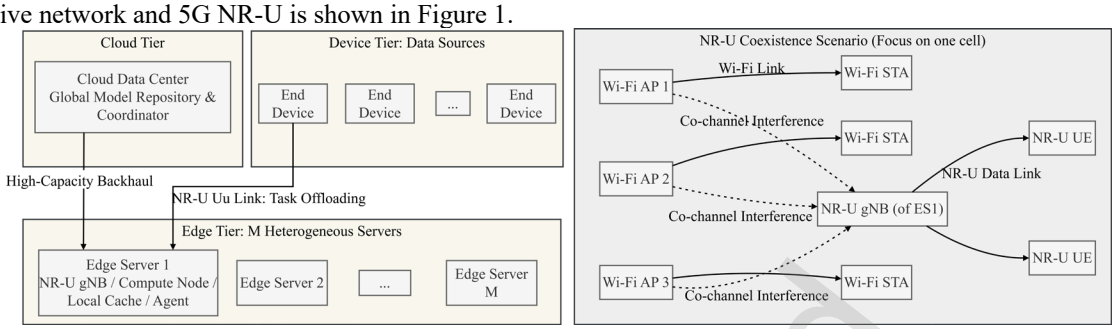

Figure 1. Interference diagram of coexistence between edge-cloud collaborative network and 5G NR-U

Note: NR-U UE and NR-U gNB are two core network elements in the 5G NR-U (New Radio in Unlicensed Spectrum) network, corresponding to the terminal side and the network infrastructure side, respectively; ES1 stands for Edge Server 1. STA stands for Station; AP stands for Access Point.

Figure 1 (left) illustrates a three-layer collaborative paradigm comprised of terminals, edge computing, and cloud data centers: terminal devices act as training data sources, offloading computational tasks to edge servers via the 5G NR-U wireless interface; multiple edge servers with heterogeneous computing capabilities, local expert caching, and integrated 5G NR-U base station functions are interconnected via highly reliable backhaul links and connected to the cloud data center, forming a uniformly schedulable distributed computing pool. Figure 1 (right) reveals the sources of uncertainty faced by the 5G NR-U wireless access network: by magnifying the coverage area of a 5G NR-U base station, it demonstrates a scenario where it coexists with multiple Wi-Fi access points and terminals operating on the same frequency. In this environment, there is an overlap of downlink transmissions from many independent Wi-Fi networks along with uplink/downlink transmissions from the 5G NR-U network all within the same frequency band, therefore creating an aggregated interference field that changes over time. The level of cross-system interference generated by sharing the same spectrum is substantially correlated to the 5G NR-U contention-based access mechanism, which in turn results in the instantaneous channel capacity of each wireless link being a non-stationary random process influenced by various conditions rather than a static value.

Regarding channel access, 5G NR-U adopts a frame-based Listen-Before-Talk mechanism [28-30]. Nodes perform an idle channel assessment before transmitting data. The probability that a channel is occupied is a function of the number of dynamically competing nodes $N_{\text{comp}}(t)$, whose successful access probability $P_{\text{access}}(t)$ in time slot t is modeled as:

$$P_{\text{access}}(t)=(1-\tau)^{N_{\text{comp}}(t)-1} \quad (1)$$

In equation (1), τ represents the steady-state transmission probability of each competing node (including Wi-Fi nodes), and its value is affected by the size of the LBT (Listen Before Talk) backoff window. After successful access, the instantaneous transmission rate $R_{ij}(t)$ achievable from node i to node j in time slot t is:

$$R_{ij}(t)=B\cdot\log_2\left(1+\frac{P_t\cdot G_{ij}\cdot|h_{ij}(t)|^2}{\sigma^2+I_{ij}(t)}\right) \quad (2)$$

In equation (2), B is the channel bandwidth; P_t is the transmit power; G_{ij} is the path loss; $|h_{ij}(t)|^2$ is the fast fading channel gain; σ^2 is the noise power; and $I_{ij}(t)$ is the time-varying aggregated interference power from Wi-Fi and other 5G NR-U nodes.

The communication delay $T_{ij}^{\text{tx}}(d)$ generated by transmitting d bits of data is:

$$T_{ij}^{\text{tx}}(d)=\frac{d}{R_{ij}(t)} \quad (3)$$

The design of actions, states, and reward functions follows the closed-loop logic of reinforcement learning, namely "observation-decision-evaluation," and its rationality is based on a strict correspondence with the characteristics of the physical system. In terms of state space design, the seven selected feature dimensions correspond to three types of key system information: resource state (CPU/GPU utilization) directly reflects the real-time computing power of nodes and is the basis for task scheduling feasibility; queue and cache information characterize the waiting cost and model availability of tasks, directly affecting execution latency; channel observation and neighbor load provide the local view required for distributed decision-making, enabling the agent to infer the global situation under partially observable conditions. This design ensures that the agent can perceive all key variables affecting scheduling decisions.

In terms of action space design, the continuous output vector is decoupled into a task offloading probability distribution and a contention window adjustment factor. Its rationality is reflected in two aspects: first, probabilistic offloading decisions allow the agent to express the strength of its preference for different execution nodes, rather than a

forced discrete choice, which provides space for exploratory learning and smooth policy evolution; second, the continuous adjustment of the contention window directly maps to the physical parameters of the 5G NR-U protocol stack, realizing cross-layer optimization control. This design enables the agent to proactively back off to reduce collisions when channel quality deteriorates, or to aggressively compete to reduce access latency when the load is light, thus achieving joint optimization of communication and computation.

The reward function design directly corresponds to the mathematical expression of a multi-objective optimization problem. The weighted combination of the four components in formula (10) maps to the objective function and constraints of the optimization problem P: the delay term and throughput term jointly drive the improvement of task processing efficiency; the energy term corresponds to minimizing communication energy consumption; and the load balancing penalty term corresponds to the resource capacity constraint, guiding the task to be evenly distributed among nodes through variance penalty. This design allows the agent to automatically weigh conflicting objectives during the learning process, ultimately converging to a Pareto optimal scheduling strategy.

2.1.2 Switch Transformer Training and Atomic Expert Task

The core of the Switch Transformer lies in its sparsely activated hybrid expert layer [31-32]: each input Token x_l (where l=1,2,...,L L, and L is the batch size) is dynamically routed by a lightweight gating network $G(cdot)$ to one of N_e expert networks $mathcalE=E_1,E_2,...,E_{N_e}$. This routing decision function is denoted as $g(x_l)in1,2,...,N_e$, indicating that x_l is assigned to expert $E_{g(x_l)}$. Therefore, the forward propagation process of a complete batch is naturally decoupled into L independent, fine-grained atomic expert tasks.

The atomic expert task au_l corresponding to the input Token x_l is defined by a triple, as follows:

$$au_l=(e_{au_l},d^{in}_{au_l},phi_{au_l}) \quad (4)$$

In equation (4), e_{au_l} is the specific expert model to be executed for the task, i.e., $E_{g(x_l)}$. $d^{in}_{au_l}$ is the amount of data for the input activation value x_l. phi_{au_l} is the number of floating-point operations required to execute the expert forward propagation process.

The Switch Transformer model configuration and atomic expert task parameters are shown in Table 1.

Table 1. Switch Transformer Model Configuration and Atomic Expert Task Parameters

Category	Parameters	Numerical / Description
Model Configuration	Total number of parameters	1.6 Tera
	Total number of experts	128
	Number of parameters per expert	12 Billion
	Number of activated experts/Tokens	1 (Switch mechanism)
	Expert input/output dimensions	1024
Atomic Task Attributes	Expert computational load	~0.5 GFLOPs
	Expert model size	~48 GB

The computational load of each task primarily depends on the feedforward layer computation of the expert network. Task execution has a critical prerequisite: the expert model parameters must have been loaded into the memory of the execution node. The system state must track the expert cache set of each compute node m. If a task is scheduled to node m but not loaded into the execution node's memory, the expert model must be transferred from a remote location (cloud or other edge node), incurring additional communication overhead.

2.1.3 Formalization of the Joint Stochastic Optimization Problem

This paper formalizes the edge-cloud collaborative scheduling problem as a joint stochastic optimization problem [33-34]. A discrete-time system is adopted, dividing time into equal-length time slots, denoted as $t\in\{0,1,2,\dots\}$. The stochasticity of the system is driven by two core and interrelated stochastic processes.

The dynamics of the wireless channel are described by the channel state process $C(t)$, which characterizes the transmission capability of the 5G NR-U link in time slot t, defined as $C(t)=\{R_{ij}(t),P_{\text{access}}(t)\}_{\forall i,j}$. Its evolution is jointly governed by the dynamically changing number of competing nodes and time-varying aggregated interference.

The randomness of the computational load is described by the task arrival process $A(t)$, which characterizes the dynamic generation pattern of atomic expert tasks and is defined as $A(t)=\{\tau_l(t)\}_{l=1}^{L(t)}$. Here, the expert class required for each task follows a probability distribution $P(e_\tau=E_i)$ that depends on the characteristics of the input data. This distribution is non-uniform and time-varying, which constitutes the inherent dynamic imbalance of the training load.

Each edge node m maintains a local task waiting queue $Q_m(t)$ and an expert model cache set $H_m(t)\subset E$. The cache state determines whether a node can immediately execute an expert task. If the cache is missed, the model must be retrieved from the cloud or a neighboring node, resulting in additional communication overhead. Therefore, the complete (global) state $S(t)$ of the system in time slot t is a combination of communication, computation, and cache states:

$$S(t)=\left(C(t),\ A(t),\ \{Q_m(t),H_m(t)\}_{m=1}^{M}\right) \quad (5)$$

In each time slot t, the central scheduler makes a multi-dimensional joint decision, namely action $\mathrm{a}(t)$, based on observations of the current state $\mathrm{S}(t)$. This action space mainly includes three mutually coupled dimensions:

(1) Task offloading decision: Specify the execution location for each newly arrived or queued atomic task τ. Optional targets include local edge nodes, other collaborative edge nodes, or cloud data centers.

(2) Expert caching decision: decide which expert models to retrieve from the remote end and cache to the local edge node, or replace which existing caches when storage space is limited, in order to optimize the efficiency of future task execution.

(3) 5G NR-U parameter adaptation decision: Based on the real-time perceived channel contention intensity, the medium access control parameters of the node are adaptively adjusted.

The optimization objective of this paper is to design an optimal scheduling policy π^* to minimize the average iteration delay $\bar{T}_{\mathrm{iter}}$. This time is the total time taken for all atomic tasks from generation to completion during a single model forward propagation. It is a complex function of computational and communication delays, denoted as $T_{\mathrm{iter}}(\mathrm{S}(t),\mathrm{a}(t);C(t),A(t))$. The joint communication-computation-caching scheduling problem can be formalized into the following stochastic optimization problem P:

$$
\begin{aligned}
P{:}\min_{\pi} \quad & \bar{T}_{\mathrm{iter}}=\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{E}_{C,A}\left[T_{\mathrm{iter}}(\mathrm{S}(t),\mathrm{a}(t);C(t),A(t))\right] \\
\text{s.t.} \quad & \sum_{\tau\in\Psi_m(t)}\phi_\tau \le \Phi_m^{\max},\forall m,\forall t \\
& |\mathrm{H}_m(t)|\cdot s_e \le \mathrm{M}_m^{\max},\forall m,\forall t \\
& \mathrm{a}(t)\in A_{\mathrm{LBT}}(C(t)),\forall t
\end{aligned}
\tag{6}
$$

In equation (6), $\Psi_m(t)$ represents the set of tasks assigned to node m for execution in time slot t; $\Phi_m^{\max}$ and $\mathrm{M}_m^{\max}$ represent the maximum computing power (FLOPs/s) and maximum storage capacity (bytes) of node m, respectively; $A_{\mathrm{LBT}}(C(t))$ represents the set of all possible actions conforming to the 5G NR-U LBT protocol specification under a given channel state $C(t)$.

2.2 Dynamic Scheduling Algorithm Based on MADDPG

2.2.1 Dynamic Scheduling Algorithm Framework

To address the challenges of formalized high-dimensional stochastic optimization problems, traditional optimization methods struggle with their vast state spaces, continuous action spaces, and non-stationary environments. Therefore, this paper employs a multi-agent deep reinforcement learning framework and introduces a distributed dynamic scheduling algorithm based on MADDPG [35-37]. The core idea of this algorithm is to model each edge server as an independent agent, enabling it to make decisions based on local observations. Simultaneously, a centralized critic network is used for collaborative training to learn a globally approximate optimal policy under dynamic 5G NR-U environments and heterogeneous workloads. The distributed scheduling framework based on MADDPG is shown in Figure 2.

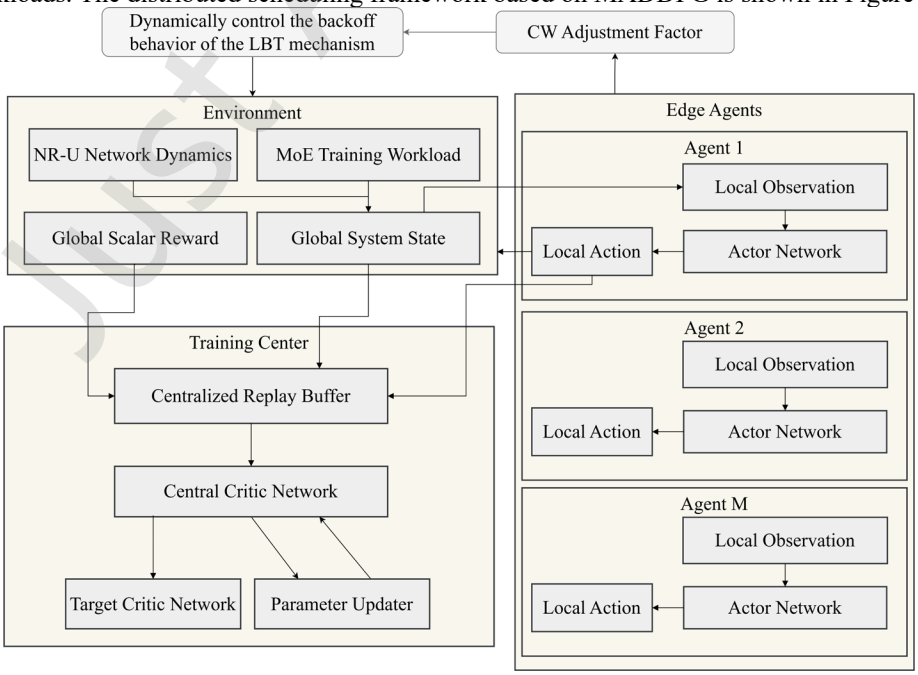

Figure 2. Distributed scheduling framework based on MADDPG

In environmental interaction, the dynamic 5G NR-U network state and the hybrid expert training workload jointly generate the global system state and provide differentiated local observations for each edge agent. Based on their local observations, each edge agent generates specific scheduling actions in parallel through independent actor networks. These actions directly affect the environment to change the system state and generate a global scalar reward that evaluates overall performance. During model training, the experiences generated by all agents' interactions with the environment, including joint actions, state transitions, and global rewards, are aggregated and stored in a central experience replay buffer. A central critic network accesses these experiences and guides learning by evaluating the long-term value of joint actions given a global state. The trainer uses the value signal computed by the critic network and the temporal difference error to simultaneously update the policy parameters of all actor networks and the value estimation parameters of the critic network itself via backpropagation, and employs a soft update method to synchronize the target network for stable training.

Each edge server with 5G NR-U access and computing capabilities is defined as an agent. Assuming there are M edge servers in the system, there are corresponding M agents. Agent i's decisions are entirely based on its available local information, but its policy parameters are updated through collaborative training with the central unit, aiming to learn cooperative behaviors that improve the overall system performance.

In time slot t, agent i's local observation $s_i(t)$ is a multi-dimensional feature vector, comprehensively representing the resource status of its managed nodes and local environmental information, as shown in the equation:

$$s_i(t)=[u_i^{cpu}(t),u_i^{gpu}(t),q_i(t),h_i(t),\gamma_i(t),l_{N(i)}(t)] \quad (7)$$

In equation (7), $u_i^{cpu}(t)$ and $u_i^{gpu}(t)$ represent the real-time utilization of the CPU (Central Processing Unit) and GPU (Graphics Processing Unit) of node i, respectively. $q_i(t)$ is the encoded vector of the task waiting queue of node i, containing information such as queue length, computational load ϕ_τ of the task at the head of the queue, and the required expert ID. $h_i(t)$ is the representation of the local expert cache set $H_i(t)$ of node i. $\gamma_i(t)$ represents the observed wireless channel information, including received signal strength, interference noise power spectral density, and a short-term estimate of the current channel access success rate. $l_{N(i)}(t)$ is the load summary information obtained from the neighboring node set $N(i)$ through periodic signaling exchanges.

The output action $a_i(t)$ of agent i is a continuous value vector, directly mapped to fine-grained scheduling control instructions [38-39]:

$$a_i(t)=\left[p_i^{(dest)}(t);\ \xi_i(t)\right] \quad (8)$$

In equation (8), $p_i^{(dest)}(t)\in R^{M+1}$ is a probability distribution vector, where the first M elements correspond to the probability of offloading the currently scheduled task to other edge nodes, and the last element corresponds to the probability of offloading to the cloud data center. The sum is guaranteed to be 1 using the Softmax function. $\xi_i(t)\in[0,1]$ is a normalized competition adjustment factor. The agent adjusts the competition window size based on this output, as shown in the equation:

$$CW_i=CW_{min}+\xi_i(t)\cdot(CW_{max}-CW_{min}) \quad (9)$$

A smaller ξ_i indicates a more aggressive channel contention strategy, suitable for scenarios with light load or low interference; conversely, a larger ξ_i indicates a more conservative backoff strategy.

The reward function is key to guiding multi-agent collaboration and achieving global optimization goals [40-42]. This paper designs a composite reward signal for each agent, with the following equation:

$$r_i(t)=-\alpha\cdot\bar{T}_i^{proc}(t)-\beta\cdot E_i^{comm}(t)+\gamma\cdot\frac{N_{completed}(t)}{\Delta t}-\delta\cdot\sigma^2(\{\bar{T}^{proc}\}) \quad (10)$$

In equation (10): $\bar{T}_i^{proc}(t)$ represents the average end-to-end latency of all tasks decided by agent i and completed on its associated nodes within the most recent time window. $E_i^{comm}(t)$ represents the total communication energy consumption generated by agent i's decision within the same time period. $\frac{N_{completed}(t)}{\Delta t}$ represents the overall task completion throughput of the system within the time period Δt, serving as a positive reward to incentivize system efficiency. $\sigma^2(\{\bar{T}^{proc}\})$ represents the variance of the average processing latency of all edge nodes, used to penalize load imbalance and promote a reasonable distribution of tasks among nodes.

2.2.2 Key Network Structure and Training Process of MADDPG

The choice of network structure parameters and hyperparameters determines the agent's ability to extract features from high-dimensional observations, learn complex strategies, and accurately evaluate the value of joint actions [43-44]. The key configurations used in this paper are shown in Table 2.

Table 2. Key Network Structures and Hyperparameters of MADDPG

Component	Structure	Activation functions	Optimizer	Learning rate
Actor	3-layer MLP (256, 128, 64)	ReLU, Tanh (output layer)	Adam	0.001
Critic	4-layer MLP (256, 256, 128, 1)	ReLU	Adam	0.002
Target Network Update Rate		0.01		
Discount Factor		0.95		

As shown in Table 2, the actor network employs a three-layer MLP (Multi-Layer Perceptron). Its input layer dimension matches the dimension of the agent's local observation state s_i. The network contains two hidden layers with 256 and 128 neurons respectively, both using the ReLU (Rectified Linear Unit) activation function to introduce non-linearity. The output layer dimension matches the dimension of the action space a_i, and the Tanh activation function is used to constrain each output value to the interval [-1, 1], before decoding based on the actual physical meaning of the action. The critic network employs a four-layer MLP structure, whose input is a concatenated vector of the global state s and all agent actions $a_1,...,a_M$. This network also uses ReLU as the activation function for the hidden layers and outputs a scalar representing the estimated Q-value given the global state and joint actions.

Each agent interacts with the environment according to its current policy π_i (with exploratory noise), generating experience tuples $(s(t),a_1(t),...,a_M(t),r(t),s(t+1))$ in time slot t. The experiences of all agents are centrally stored in a shared experience replay buffer R. This buffer follows a first-in, first-out principle and is large enough to ensure sample diversity and break temporal correlation [45].

At each update, a small batch of experiences R is randomly sampled from the buffer $\mathscr{R}$. For each sampled sample, the critic network updates the target y using the following equation:

$$y=r+\gamma\cdot Q'\left(s',\pi_1'(s_1'),\ldots,\pi_M'(s_M')\right) \quad (11)$$

In equation (11), Q' is the target critic network, π_i' is the target actor network, and γ is the discount factor.

The loss function for the critic network uses mean squared error, expressed as:

$$L_{\text{critic}}=\frac{1}{|B|}\sum_{(s,a,r,s')\in B}(y-Q(s,a_1,...,a_M))^2 \quad (12)$$

The parameters of the critic network are updated using the Adam optimizer by minimizing the mean squared error loss.

The parameters of the actor network are updated via policy gradients. The goal is to maximize the expected reward evaluated by the critic network. For agent i, its policy gradient is approximately:

$$\nabla_{\theta_i}J\approx\frac{1}{|B|}\sum_{B}\nabla_{a_i}Q(s,a_1,...,a_M)|_{a_i=\pi_i(s_i)}\cdot\nabla_{\theta_i}\pi_i(s_i) \quad (13)$$

In equation (13), θ_i represents the actor network parameters of agent i. Gradient updates utilize the gradient directions provided by the critic network to guide the actor network in adjusting its policy to produce actions with higher Q values.

To stabilize training, the target network (π_i', Q') of both the actor and critic is maintained, and its parameters are slowly updated to track the online network using a soft update equation:

$$\theta_i'\leftarrow\tau\theta_i+(1-\tau)\theta_i', \quad \omega'\leftarrow\tau\omega+(1-\tau)\omega' \quad (14)$$

In equation (14), $\tau\ll1$. Furthermore, to encourage exploration, a temporally correlated noise N_t generated by an Ornstein-Uhlenbeck process is added before the actor network outputs the action a_i.

$$a_i^{\text{executed}}=a_i+N_t \quad (15)$$

The noise is relatively large in the early stages of training, but gradually decreases as the strategy matures, helping the agent escape local optima.

To visually demonstrate the learning evolution of the MADDPG algorithm in multi-objective optimization and to verify its exploration and trade-off capabilities, Figure 3 shows the evolution trajectory of the Pareto front during training. Multi-agent reinforcement learning algorithms need to balance the conflicting objectives of reducing average iteration latency and reducing communication energy consumption. By analyzing the algorithm's performance distribution in the objective space at different training stages, the process of the policy gradually converging from initial random exploration to a non-dominated solution set (Pareto front) is observed, thus verifying the algorithm's ability to effectively learn complex trade-off strategies, as shown in Figure 3.

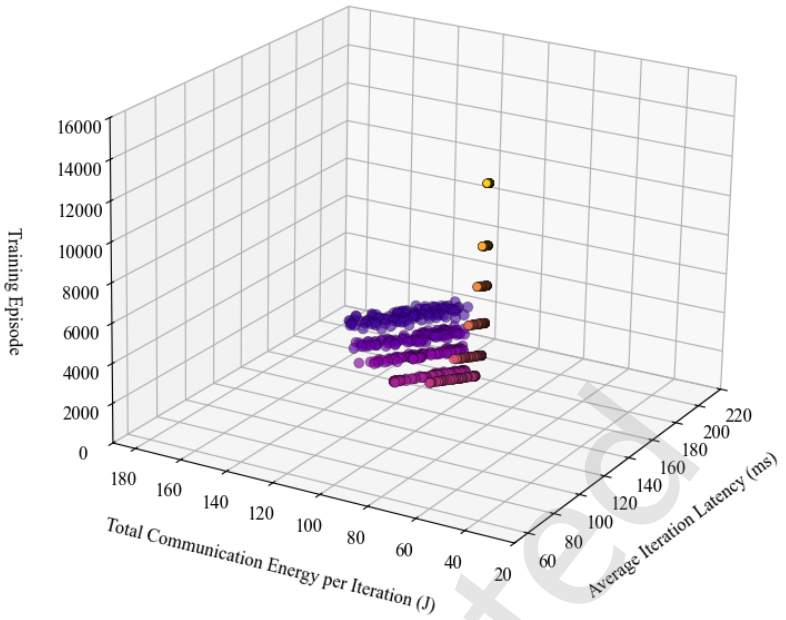

Figure 3. Pareto front evolution during the training process of the MADDPG algorithm.

In Figure 3, the scatter plots marked with different colors represent the measured performance distribution of the agent's policy at different training rounds. In the early stages of training (purple), the policy is nearly randomly initialized, and performance points are widely dispersed in high-latency, high-energy-consumption regions, reflecting the algorithm's exploration phase. As training progresses (the color gradually changes towards red and yellow as the number of rounds increases), the point cloud converges and clusters towards better regions, indicating that the algorithm effectively reduces latency and energy consumption through learning.

From a scalability perspective, the computational complexity and communication overhead of the MADDPG algorithm are mainly reflected in two aspects. In the centralized training phase, the input dimension of the Critic network increases linearly with the number of edge nodes—each new node contributes its observed state and action vector, leading to a linear expansion of the network parameter size and a corresponding increase in training computation. Simultaneously, the experience tuples of all agents need to be centrally stored in the experience replay buffer, and this transmission process puts pressure on the uplink bandwidth of the central node. However, this communication overhead is limited to the training phase and can be effectively mitigated through batch compression transmission and asynchronous update mechanisms. In the distributed execution phase, each edge node relies solely on its local Actor network for independent decision-making, without the need for real-time information exchange. Therefore, the inference computational complexity is independent of the number of nodes, and the communication overhead in the execution phase only includes the necessary neighbor state summary exchange, exhibiting good linear scalability. Overall, the MADDPG framework concentrates communication overhead in the offline training phase while maintaining lightweight online execution, enabling it to effectively support the collaborative scheduling needs of medium-sized edge networks.

3. Experimental Design

3.1 Simulation Environment and Parameter Settings

The training task flow is simulated using the publicly available dataset C4 (Colossal Clean Crawled Corpus), and the task arrival process is modeled as a Poisson process to reflect the randomness of the actual load. To ensure the reproducibility of the experiments and provide a clear benchmark for performance comparison, the parameters involved in the simulation are shown in Table 3.

Table 3. Simulation Parameters

Parameter Categories	Specific parameters	Value / Range
Network Topology	Number of edge server nodes	8
	Number of cloud data centers	1
	Number of terminal devices	50
5G NR-U Wireless Channel	Channel bandwidth	100 MHz
	Center carrier frequency	5.9 GHz
	Transmit power (edge/terminal)	23 dBm / 20 dBm
	Shadow fading standard deviation	4 dB

Computing Resources	Small-scale fading model	Rayleigh fading
	Dynamic range of competing nodes	[5, 25]
	Edge server computing power heterogeneity	[20, 50] TFLOPs
	Cloud data center computing power	500 TFLOPs
	Edge node memory capacity	[128, 256] GB
	Cloud data center memory capacity	2 TB
Task Load	Mean task (Token) arrival rate	1000 tokens/s
	Batch size	1024 tokens
	Expert task computational load	~0.5 GFLOPs
	Expert model size	~48 GB

As shown in Table 3, the simulation environment constructed a core test network comprising 8 heterogeneous edge servers and 1 central cloud data center. The 5G NR-U wireless access network was configured with a 100MHz bandwidth, simulating a real unlicensed spectrum coexistence environment through dynamically varying numbers of competing nodes (5 to 25). In terms of computing resources, the edge servers were assigned differentiated computing power (20 to 50 TFLOPs) and memory capacity to reflect the hardware heterogeneity in actual deployments; the cloud data center possessed ultra-high computing power (500 TFLOPs) as performance support and a global model repository. The task load was generated based on a real large-scale text corpus using C4, ensuring the authenticity of the expert activation mode. Task arrival followed a Poisson process, with an average arrival rate of 1000 tokens per second.

3.2 Comparison Algorithms and Evaluation Metrics

3.2.1 Comparison Algorithm

To evaluate the performance of the MADDPG algorithm, this paper designs and implements six representative benchmark scheduling algorithms for comparative analysis.

The first category of benchmark algorithms aims to evaluate extreme strategies and classical heuristics. The CO (Cloud-Only) strategy unconditionally uploads all expert computation tasks to the cloud data center for execution; its performance (ignoring communication overhead) represents the theoretical upper bound of computational power, used to measure the potential benefits of edge collaboration. GO (Greedy Offloading) is a reactive strategy based on instantaneous information, offloading arriving tasks to the available node (including edge and cloud) with the shortest current computation queue. This algorithm tests the system's immediate responsiveness to load fluctuations. SEP (Static Expert Partitioning) is a static planning algorithm based on offline analysis. Before training begins, it statically partitions and assigns the expert set to each edge node based on the node's average computing power and the expert's historical activation frequency. This algorithm is used to test the adaptability of scheduling strategies to dynamic environments.

The second category of benchmark algorithms focuses on reinforcement learning methods from different paradigms to highlight the value of multi-agent collaboration and continuous action decision-making. DDPG (Deep Deterministic Policy Gradient), as a single-agent reinforcement learning benchmark, uses a central controller to collect global information and make unified scheduling decisions. This comparison is used to verify the necessity of a multi-agent distributed architecture in handling partially observable and distributed decision-making needs. COMA (Counterfactual Multi-Agent Policy Gradients) is a classic multi-agent reinforcement learning algorithm characterized by using a counterfactual baseline to solve the multi-agent credit allocation problem, but its action space is usually discrete. This comparison is used to analyze the advantages of continuous action space design for fine-grained scheduling decisions. MAPPO (Multi-Agent Proximal Policy Optimization) is a currently high-performing multi-agent reinforcement learning algorithm based on policy gradients, known for its training stability and high performance. It can be used to compare the performance and efficiency differences of different reinforcement learning paradigms in solving this joint scheduling problem.

3.2.2 Evaluation Indicators

To quantitatively evaluate the overall performance of different scheduling algorithms, this study uses the following five core indicators to comprehensively reflect the performance of scheduling strategies in a dynamic 5G NR-U-enabled edge-cloud collaborative training environment.

Average iteration latency measures the end-to-end average time for a single model training iteration (i.e., processing a complete batch) from task generation to the completion of computation for all atomic expert tasks. Lower latency means faster model updates. For an experiment with N_{batch} of iterations, the equation is as follows:

$$\bar{T}_{\text{iter}}=\frac{1}{N_{\text{batch}}}\sum_{k=1}^{N_{\text{batch}}}\left(t_k^{\text{finish}}-t_k^{\text{start}}\right) \quad (16)$$

In equation (16), t_k^{start} and t_k^{finish} represent the start and end timestamps of the k-th iteration, respectively.

System throughput measures the system's ability to process training data per unit time, directly reflecting the resource utilization efficiency of the scheduling strategy. It is defined as the total number of tokens successfully processed within the total simulation duration T_{total}:

$$\text{Throughput}=\frac{N_{\text{tokens}}}{T_{\text{total}}} \quad (17)$$

Load balancing is measured by the standard deviation of the processing times of all executed atomic expert tasks.

$$\text{Load\_Imbalance}=\sqrt{\frac{1}{N_e}\sum_{i=1}^{N_e}\left(\bar{T}_{E_i}-\bar{T}_{\text{all}}\right)^2} \quad (18)$$

In edge computing scenarios, energy consumption is a crucial consideration. The calculation of communication energy consumption E_{comm} is based on the amount of data transmitted d, the transmission time T^{tx}, and the transmission power P_t:

$$E_{\text{comm}}=\sum_{\forall \text{trans}}\left(P_t \cdot T^{\text{tx}}(d)\right)/Iter \quad (19)$$

3.3 Experimental Scenario Design

The specific parameter configurations and dynamic characteristics of the three scenarios are shown in Table 4.

Table 4. Experimental Scenario Information

Scenario Characteristics	Scenario A: (Baseline)	Scenario B: (High Interference)	Scenario C: (Bursty Load)
Core Challenges	Testing the algorithm's basic optimization performance in a stable environment.	Testing the algorithm's dynamic adaptability to drastic fluctuations in the wireless channel.	Testing the algorithm's rapid response and resource allocation capabilities to sudden spikes in computational load.
Mission Arrival Process	Poisson process, constant arrival rate λ = 1000 tokens/s	Poisson process, constant arrival rate λ = 1000 tokens/s	Compound Poisson process: λ = 600 tokens/s for 80% of the time; spikes to λ = 2500 tokens/s for 20% of the time. The duration of the spike follows an exponential distribution with a mean of 10s.
NR-U Channel Interference	Medium and stable: the number of competing nodes fluctuates uniformly and randomly within the range [10, 15].	Periodic drastic fluctuations: Number of competing nodes. Based on a base value of 10, a high-interference pulse lasting 5 seconds is superimposed every 30 seconds, during which the number of competing nodes surges to 25.	Low and stable: The number of competing nodes fluctuates uniformly and randomly within the range of [5, 10].
Simulated Target	Provides a baseline for performance comparison, evaluating the potential for static optimization.	Simulating intermittent channel congestion caused by sudden bursts of Wi-Fi traffic or sudden access from neighboring NR-U networks.	Simulating a surge in instantaneous computational demand caused by data pipeline fluctuations or checkpoint recovery during actual training.

As shown in Table 4, three experimental scenarios constructed a continuous test spectrum from steady state to highly dynamic. Scenario A, serving as a baseline, provides an environment with relatively stable load and channel conditions to evaluate the theoretical performance ceiling and optimization potential of each algorithm under ideal conditions. Scenario B, by introducing periodic high-interference pulses, simulates typical coexistence interference bursts in unlicensed spectrum, specifically designed to test whether the algorithm can maintain service stability through intelligent task offloading and NR-U parameter adjustment when wireless link quality deteriorates sharply. Scenario C, by designing a task arrival process with sudden peaks, simulates the imbalance of training workload, focusing on how the algorithm efficiently reallocates and coordinates the load when facing instantaneous computing resource shortages to avoid system overload and performance degradation.

4. Results and Analysis

4.1 Overall Performance Comparison

To comprehensively evaluate the overall performance of different scheduling algorithms under varying environmental pressures, this paper conducts systematic tests in three experimental scenarios (Scenario A: baseline; Scenario B: high interference; Scenario C: burst load), and quantitatively compares the core performance indicators of all compared algorithms. These indicators include average iteration latency, system throughput, expert load balancing, and communication energy consumption. Figure 4 illustrates the performance of CO, GO, SEP, DDPG, COMA, MAPPO, and the proposed MADDPG algorithm in the three scenarios.

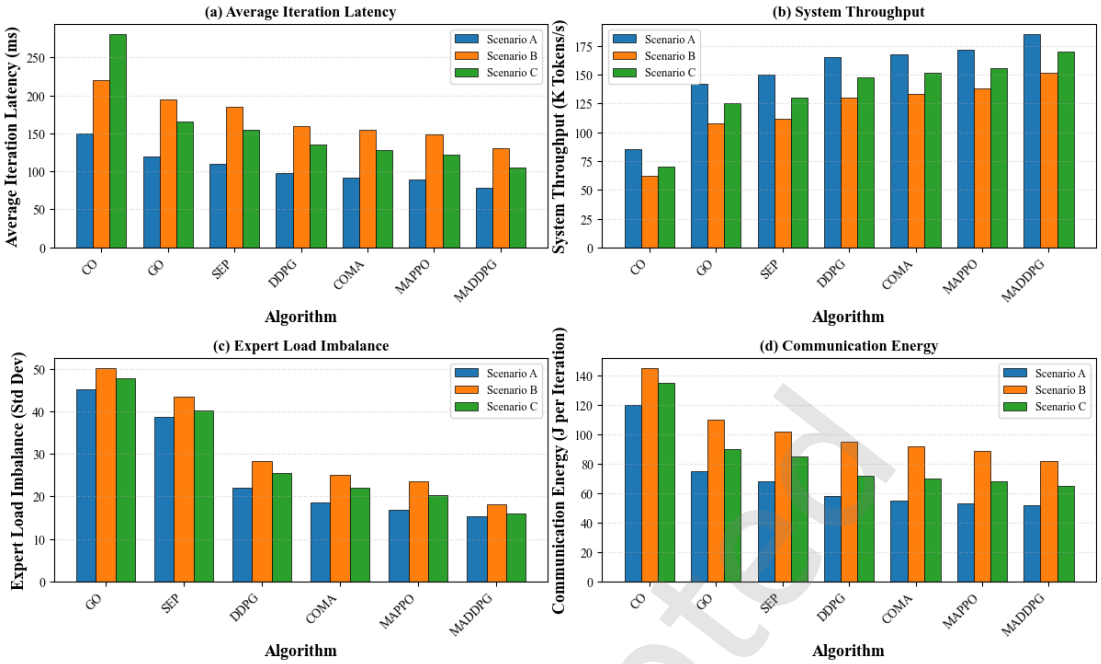

Figure 4. Overall performance comparison in three scenarios
(Figure 4(a) Average iteration delay; Figure 4(b) System throughput; Figure 4(c) Expert load balancing; Figure 4(d) Communication energy consumption)

According to Figure 4(a), the CO technique without factoring in communication delays has more latency on average than any other strategy for iteration(s), with an average of 150ms, 220ms and 280ms in scenarios A, B, and C respectively. This demonstrates the limitations of the pure cloud-based solution. The heuristic approaches (GO, SEP) perform adequately in scenario A (stable), however, their performance diminishes considerably under high levels of interference (scenario B) and bursty traffic (scenario C), indicating the limitations of these algorithms to make long-term predictions based on short-term observations. Reinforcement learning (RL) based methods outperformed the heuristics; of RL methods, MADDPG had the lowest latency on average for iterations (78ms, 130ms, 105ms), a reduction of latency compared to the best workflows (MAPPO) by approximately 12.4% (scenario A), 12.2% (scenario B) and 13.9% (scenario C). The advantages of MADDPG are attributed to its global cooperative approach, which was developed via a centralized critic network, allowing MADDPG to better anticipate channel changes and load patterns, leading to optimal allocations of tasks and updates to 5G NR-U parameters, resulting in reduced queue times for tasks and waiting times for data transmissions.

Throughput was at its lowest in scenario 4(b) because of the high latency involved in the CO strategy. MADDPG produced the highest overall throughput in all scenarios (185K, 152K, 170K Tokens/s) and surpassed MAPPO by approximately 7.6%, 10.1%, and 9.0%, respectively. MADDPG's higher throughput than MAPPO can be attributed to MADDPG's more efficient use of resources and lower latency. MADDPG minimizes the amount of time that a node stands processing idle due to cache misses or overloading the node with work by imposing a more effective form of expert cache management and load balancing. MADDPG's ability to adjust the 5G NR-U parameter to adapt to changes improves the probability of successful access to the channel and the amount of useful data transmitted per unit of time increases MADDPG's overall processing capability. In scenario B, which has the greatest interference, MADDPG is able to maintain a high level of throughput indicating the strength of MADDPG's anti-interference robustness.

CO does not make use of edge-aware collaboration strategies to balance work among teammates. Therefore, this metric is not applicable to that strategy, so it is not displayed in Figure 4(c). As can be observed from the figure above, the GO approach produced the poorest overall performance, where the Standard Deviation was greater than the maximum allowable limit of 45 for BALANCE. The resulting congestion at "hot" experts and at the transporting nodes was solely a consequence of the GO strategy making its decisions based solely upon the instantaneous queue length.

Although the static expert strategy has improved upon previous methodologies, by virtue of its being static, it would still not be able to adapt to the dynamic establishment of the activation of different experts. The use of reinforcement learning algorithms was found to result in an even greater improvement in BALANCE than any other method, as it produced the lowest standard deviations (15.3, 18.2, and 16.0) with respect to their performance across all scenarios (A, B, and C, respectively) compared with the MAPPO technique. The reason for this was that during the learning phase, MADDPG had a special additional term in the reward function referred to as BALANCE, which encouraged the agent to learn how to migrate tasks from the "overloaded" nodes or balancing teams to "idle" nodes and distribute "hot" expert

models intelligently before execution had begun, thereby creating two-dimensional leveling in the deployment of tasks and balancing the utilization of available expert resources across both the expert resources and the nodes.

As can be seen in Figure 4(d), the CO method consumes the highest amount of energy due to sending all messages through remote transmission to the cloud. MADDPG had the least communication energy consumption for each of the three different scenarios (52J, 82J and 65J) which can be attributed to three main factors; the first is that it uses its learned strategies to maximise local and close edge processing and minimize unnecessary long-range transmission of cloud messages; second, its intelligent expert caching strategies greatly reduce the amount of energy used to resend models that have been previously cached; and finally it minimizes the number of meaningless retransmissions due to poor channel conditions by optimizing the NR-U contention parameters. A noteworthy point is that, as illustrated in scenario B, in the presence of heavy interference, all algorithms experience an increase in energy consumption, although MADDPG experiences a manageable energy consumption increase of only ~57.7% compared to scenario A, demonstrating that MADDPG is able to optimize energy efficiency under adverse channel conditions as well.

As algorithms are implemented into real-world systems, it is necessary for them to be efficient during their learning phase and stable. This paper compares how fast (sample efficiency) and consistently (reasonably good convergence) each of several different types of reinforcement learning approaches (specifically DDPG, COMA, MAPPO) are in helping resolve the same joint optimization problem. The first comparison is made between the DDPG type of architecture and the DDPG type of architecture under stable baseline scenario A. When comparing DDPG to COMA; it can be found that the DDPG type of architecture to be better suited for distributed decision-making environments that require multiple agents to work together. The second comparison is made between COMA and MAPPO. The third comparison is made between MAPPO and other iterations of the reinforcement learning approach based on value functions and Policy Gradient based approaches. Throughout this study, the convergence curves of average iteration delay have been plotted against the number of training epochs, revealing each algorithm's relative learning speed, performance level, and convergence stability, as illustrated in Figure 5 of this paper.

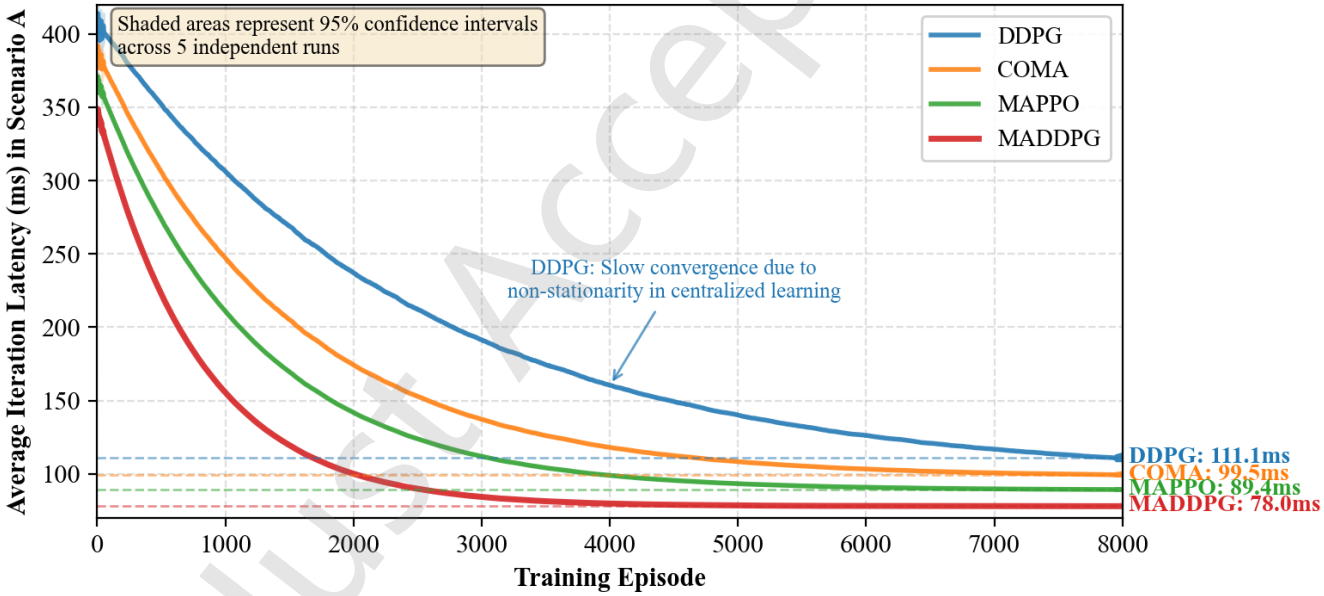

Figure 5. Comparison of convergence curves of different algorithms

Note: The dashed lines in Figure 5 represent the average iteration delay achieved by different algorithms in scenario A.

According to Figure 5, there exists a large disparity in both the rate of convergence and performance achieved by the four selected RL algorithms. The slowest rate of convergence is exhibited by DDPG which eventually plateaued to a mean latency of approximately 111.2ms. The results demonstrated DDPG's extreme difficulty of learning in multi-agent environments and the severe non-stationary effect suffered by centralized single-agent architectures in an environment where policy models of the other agents are continuously changing, resulting in a slow learning rate and limited ability to form effective policies. The early training performance of COMA improved much more rapidly than the other approaches (considerably improved after nearly 1500 episodes) due, in part, to COMA's multi-agent architecture with the addition of counterfactual baselines for learning how to allocate credit. The COMA training performance curve reached a plateau after about 4000 episodes and the best latency performance at 99.5 ms is worse than the MAPPO and MADDPG algorithms. The reason for this performance plateau, the discrete nature of the action space was found to limit the resolution to which COMA could modify the offloading probability and non-coherent resource usage probability, thus capping its upper potential for improving policy performance. Conversely, the rapid decay in the performance curve for MAPPO yielded an

overall mean latency of 89.4 ms and indicates that MAPPO is a robust algorithm for the training of a multi-agent policy gradient algorithm in highly complex environments. However, the MADDPG algorithm presented in this paper outperforms all comparisons: it has the fastest convergence speed and ultimately achieves the lowest latency (78.0 ms). This advantage is attributed to two core design features of the MADDPG framework: first, its value function-based and deterministic policy architecture provides higher sample efficiency than MAPPO based on policy gradients when dealing with the high-dimensional continuous action space of this problem; second, its "centralized critic" provides a stable value estimate containing global state and other agent action information for each distributed agent, greatly alleviating the non-stationarity challenge in multi-agent learning, thus achieving more efficient and superior policy learning.

4.2 Dynamic Adaptability Analysis

To verify the robustness of the algorithms to burst interference in the wireless channel, this paper monitors the real-time changes in the average iteration delay of each algorithm under scenario B (high interference). Scenario B periodically introduces a surge in the number of competing nodes in the NR-U channel ("interference peak"), simulating strong interference bursts in actual unlicensed spectrum. By comparing the delay fluctuation amplitude and recovery speed of different algorithms under interference impact, their dynamic adaptability and network stability are evaluated, and the results are shown in Figure 6.

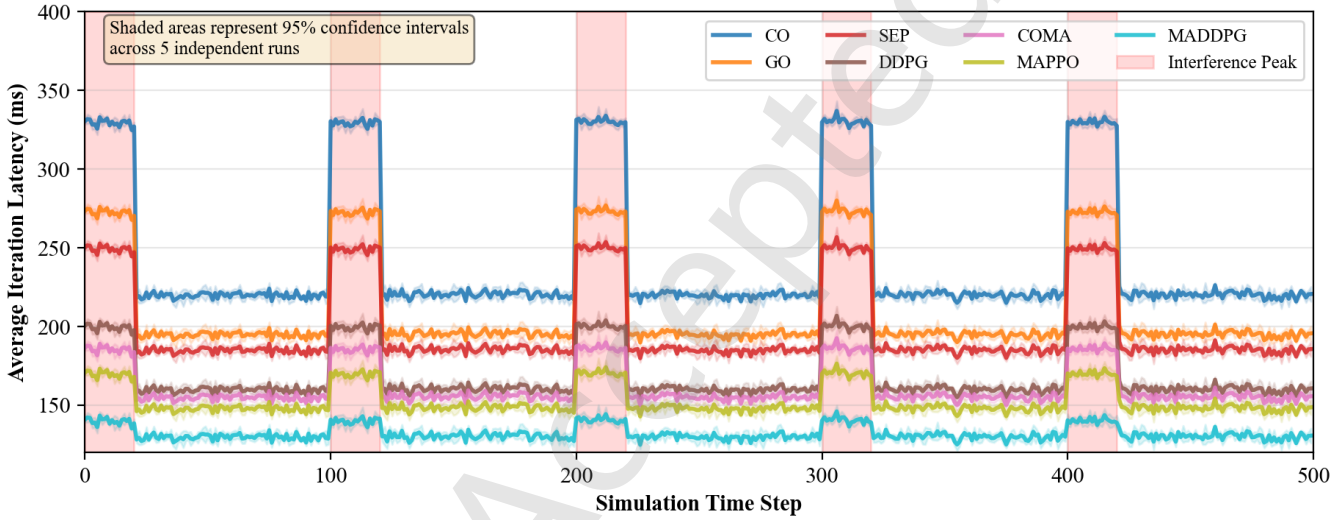

Figure 6. Curves showing the variation of average iteration delay with time step for different algorithms

In the Figure 6, how many of the different algorithms were affected by "interference peaks" (indicated by the red areas) can be seen, and it does that in multiple ways: the first group of non-learning algorithms (CO, GO and SEP) increased their latency dramatically (for example, CO's latency went from ~130ms to approximately 330ms, GO from ~130ms to 273ms, and SEP from ~130ms to 250ms) during these interference peaks, showing that they are completely dependent on the current state of the channel without any foresight for what happens over time. While the learning algorithms (DDPG, COMA, and MAPPO) experienced some degree of fluctuation, they were overall better compared to the non-learning. The MADDPG algorithm discussed within this paper exhibits good adaptability to changing conditions: for example, it was only slightly affected by the interference peaks (i.e., its latency only went from approximately 130ms before the interference peaks to approximately 140ms during the interference peaks) with the least amount of fluctuation from its baseline. Subsequently, after the interference ended, the latency recovered to baseline in approximately 10 time steps (i.e., approximately 0.1 seconds). This is because the MADDPG agent (as trained through the centralized training process described in this paper) learned a joint optimization strategy that enabled it to adjust its task offloading decisions and optimize the 5G NR-U contention parameters based on the current channel quality (i.e., when it sensed a decrease in quality). This adjustment/optimization allowed it to buffer the effects of interference while still quickly utilizing the previously used bandwidth resources after the channel was restored.

To evaluate the algorithm's ability to handle sudden spikes in computational load, this paper observes the spatiotemporal evolution of the task queue length at each edge node under scenario C (burst load). Burst load simulates the instantaneous surge in computational demand caused by fluctuations in the data pipeline during actual training. By comparing the uniformity of the node queue heatmap distribution under different algorithms, their load balancing and fast scheduling capabilities are intuitively assessed, as shown in Figure 7.

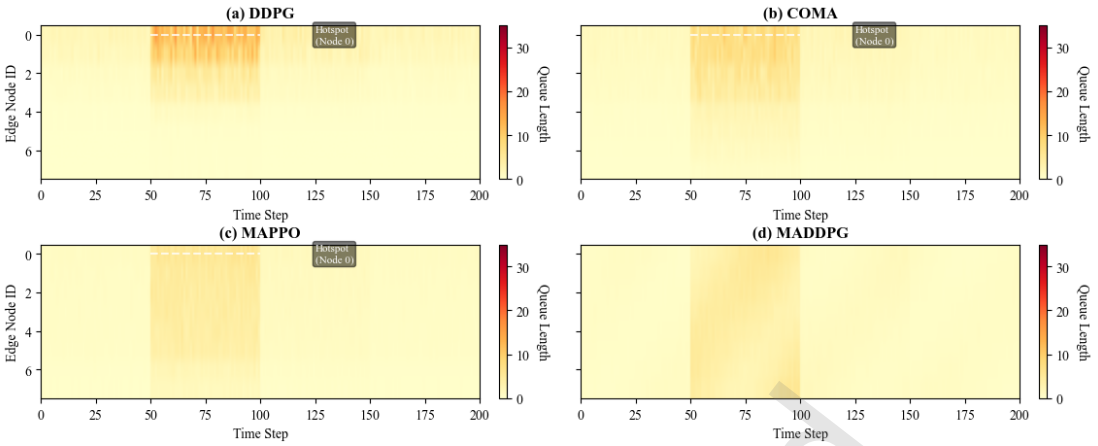

Figure 7. Changes in edge node queue length over time under different algorithms
(Figure 7(a) DDPG; Figure 7(b) COMA; Figure 7(c) MAPPO; Figure 7(d) MADDPG)

Figure 7 shows the queue heatmaps, revealing the load distribution characteristics of different algorithms under burst loads. In Figure 7(a), a few nodes (such as nodes 0 and 1) exhibit consistently high queue lengths (dark red) throughout the burst, forming obvious hotspots, while the remaining nodes are underutilized. The centralized DDPG strategy was found to allocate too many tasks to nodes with better than average (either temporary or permanent) performance through a method that does not have a good mechanism for creating a more evenly distributed load across the system. In examining the heatmap graphs (as shown in Figures 7(b) and 7(c)), it was clear that the non-uniform/overloaded areas have been decreased, however, there are still some nodes that have a larger than normal amount of work during burst period (e.g., nodes 0-2 in the COMA scenario, nodes 0-1 in the MAPPO scenario). On the other hand, the heatmap produced by the MADDPG algorithm has a more uniform distribution of colors across the nodes, and the queue lengths for each node did not become excessively long during burst periods. This is due to the MADDPG algorithm's distributed agent architecture collaborating to determine how the agents learn to respond to each other's local queues and share load balance penalties, while also learning about their neighbors (each agent has knowledge of its neighbors' load status) when the load begins to surge. Therefore, the MADDPG system is able to quickly adapt to sudden influxes of loads, and as such has optimal resource utilisation and high levels of service stability.

4.3 Visualization of Expert Task Unloading Strategies

To gain a deeper understanding of the underlying mechanism of the scheduling strategy learned by the MADDPG agent, this paper presents a visual analysis of its learned expert task offloading preferences. By mapping the agent's offloading decision probabilities for specific experts to a heatmap, the paper reveals how the MADDPG algorithm intelligently allocates computational positions based on the characteristics of different experts, thus explaining the source of its performance advantage. The visualization of the expert task offloading strategy learned by the agent is shown in Figure 8.

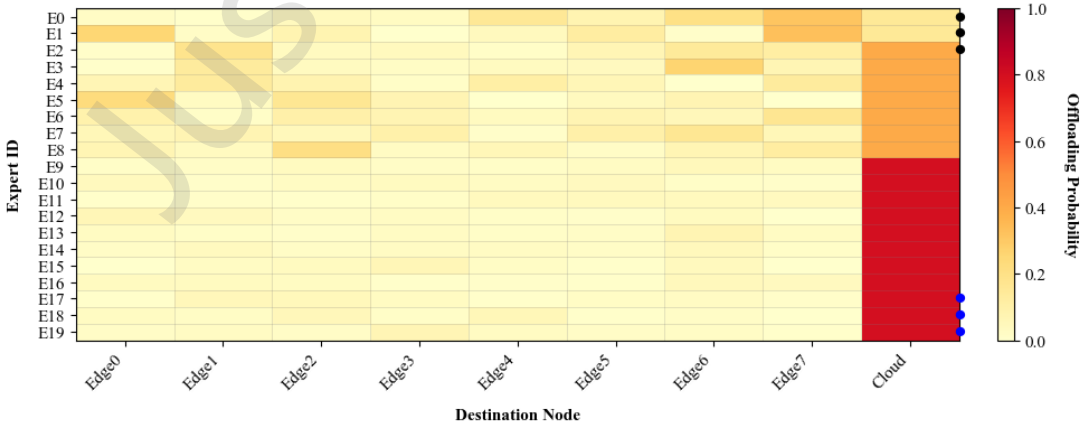

Figure 8. Visualization of the expert task offloading strategy learned by the agent

Figure 8's heatmap visually reveals the significant intelligent patterns in the expert task offloading strategies learned by the MADDPG agent. For frequently activated "popular experts" (top E0-E2, marked with black dots), their offloading probabilities exhibit a relatively uniform and high distribution across multiple edge nodes (Edge0-Edge7) (bright yellow to orange), while retaining a certain cloud offloading probability. This indicates that the algorithm, through empirical learning,

automatically caches these commonly used expert models on multiple edge nodes to achieve load balancing and proximity computing, thereby significantly reducing queuing delays caused by centralized processing and communication overhead caused by remote transmission. Conversely, for infrequently used "unpopular experts" (bottom E17 - E19, marked with blue dots), their offloading strategy is highly concentrated in the cloud data center (dark red), while the probability of allocation to edge nodes is very low.

4..4 Sensitivity Analysis

Table 5. Sensitivity Analysis of Reward Function Weights

Weight Configuration	α (Latency)	β (Energy)	γ (Throughput)	δ (Balance)	Avg. Latency (ms)	Throughput (K tokens/s)	Load Balance (Std Dev)	Comm. Energy (J)	Composite Score
Baseline	0.4	0.3	0.2	0.1	78	185	15.3	52	0
Latency-Optimized	0.7	0.1	0.1	0.1	72.3	178.2	21.7	68.5	-0.12
Energy-Optimized	0.1	0.7	0.1	0.1	96.8	161.5	18.9	44.2	-0.18
Throughput-Optimized	0.1	0.1	0.7	0.1	84.5	192.4	22.3	61.3	-0.09
Balance-Optimized	0.1	0.1	0.1	0.7	89.2	168.7	12.1	57.8	-0.15
Uniform Weights	0.25	0.25	0.25	0.25	81.3	181.2	16.8	55.6	0.05
Grid Search Best	0.35	0.25	0.25	0.15	77.2	186.5	14.9	51.2	0.08

Note: Composite score is calculated as the normalized sum of percentage improvements across all metrics, with positive values indicating overall better performance relative to baseline.

The sensitivity analysis reveals the significant impact of reward function weight selection on the multi-objective optimization performance of the MADDPG algorithm. Through systematic grid search and comparative evaluation of seven representative weight configurations, we observe distinct performance trade-offs across the four optimization objectives defined in Equation (10).s

The Latency-Optimized configuration (α=0.7) achieves the lowest average iteration latency (72.3 ms), representing a 7.3% improvement over the baseline. However, this aggressive latency minimization comes at the cost of increased communication energy consumption (68.5 J, +31.7%) and degraded load balancing (standard deviation 21.7, +41.8%). This configuration tends to offload tasks to the nearest available computing resources regardless of their energy efficiency or current load status, leading to localized hotspots and excessive transmission energy.

Conversely, the Energy-Optimized configuration (β=0.7) achieves the lowest communication energy consumption (44.2 J, 15.0% improvement) but suffers from significantly higher latency (96.8 ms, +24.1%) and reduced throughput (161.5 K tokens/s, -12.7%). This configuration prioritizes local processing and conservative transmission strategies, avoiding long-distance data transfers even when remote resources would provide faster computation.

The Throughput-Optimized configuration (γ=0.7) maximizes system throughput (192.4 K tokens/s, +4.0%) but exhibits the poorest load balancing (standard deviation 22.3), indicating that maximizing raw processing volume often leads to uneven resource utilization. The Balance-Optimized configuration (δ=0.7) achieves the best load distribution (standard deviation 12.1, 20.9% improvement) but at the expense of latency (89.2 ms) and throughput (168.7 K tokens/s).

Notably, the Uniform Weights configuration (all weights 0.25) achieves a balanced performance across all metrics, with a positive composite score (+0.05) relative to baseline, demonstrating that equal consideration of all objectives provides robust overall performance. The Grid Search Best configuration (α=0.35, β=0.25, γ=0.25, δ=0.15) identified through systematic exploration achieves the highest composite score (+0.08), with marginal improvements across all metrics simultaneously.

These results demonstrate that the MADDPG algorithm's performance is sensitive to reward function weight selection, with clear Pareto frontiers emerging between competing objectives. The baseline configuration (α=0.4, β=0.3, γ=0.2, δ=0.1) represents a well-balanced trade-off, slightly prioritizing latency while maintaining reasonable performance across other metrics. For practical deployments, weight selection should be guided by specific application requirements: latency-sensitive applications may adopt the Latency-Optimized configuration, while energy-constrained edge deployments may prefer the Energy-Optimized configuration. The Grid Search Best configuration provides a reference point for optimal multi-objective trade-off in general-purpose scenarios.

4.5 Generalization Ability Evaluation

To comprehensively assess the generalization capability of the proposed MADDPG algorithm, we conduct additional experiments comparing it with classic heuristic rules under varying system configurations. The heuristic rules include: Random Offloading (RO) , which randomly assigns tasks to available edge or cloud nodes; Round-Robin (RR) , which distributes tasks sequentially across all nodes; Shortest Queue First (SQF) , which offloads tasks to the node with the

smallest pending queue length; and Local-First (LF) , which prioritizes local execution and only offloads when local resources are insufficient.

We evaluate generalization across three dimensions: (1) scaled network size (4, 8, 12, and 16 edge nodes), (2) varying task arrival rates (500, 1000, 1500, and 2000 tokens/s), and (3) different expert count configurations (64, 128, and 256 experts). All experiments are conducted under Scenario A (baseline) conditions, and each configuration is tested with 5 random seeds. The MADDPG model trained on the default 8-node, 1000 tokens/s, 128-expert configuration is directly applied to unseen configurations without retraining.

Table 6 presents the average iteration latency (ms) results across different generalization test scenarios.

Table 6. Generalization Performance Comparison (Average Iteration Latency in ms)

Test Scenario	Configuration	RO	RR	SQF	LF	MADDPG
Network Size (8-node trained)	4 nodes	245.3	187.6	142.8	168.4	98.7
	12 nodes	267.8	212.3	171.5	198.2	118.4
	16 nodes	289.5	238.7	198.3	227.6	142.6
Arrival Rate (1000 trained)	500 tokens/s	198.4	156.2	122.7	138.5	76.3
	1500 tokens/s	312.7	245.8	188.4	219.3	135.8
	2000 tokens/s	378.2	298.5	241.6	278.9	178.4
Expert Count (128 trained)	64 experts	215.6	172.4	138.2	152.7	82.5
	256 experts	287.3	226.8	181.5	208.4	124.7

The experimental results demonstrate the superior generalization capability of the proposed MADDPG algorithm across all tested dimensions. When scaled to larger network sizes (16 nodes), MADDPG maintains an average latency of 142.6 ms, significantly outperforming the best heuristic SQF (198.3 ms) by 28.1%. This indicates that the multi-agent coordination mechanism learned under 8-node configuration effectively transfers to larger-scale deployments without retraining. Under varying task arrival rates, MADDPG exhibits robust performance even at 2000 tokens/s (78.4% higher than trained rate), achieving 178.4 ms compared to SQF's 241.6 ms (26.2% improvement). This resilience stems from the learned load balancing and cache management strategies that prevent queue buildup during high-intensity periods. For different expert count configurations, MADDPG adapts effectively to both reduced (64 experts) and expanded (256 experts) model architectures, demonstrating that the scheduling policy captures fundamental patterns of expert activation rather than overfitting to specific expert set sizes.

In contrast, heuristic rules show clear limitations in generalization. SQF performs reasonably well in familiar configurations but degrades significantly under scale and load extremes due to its myopic nature. LF exhibits consistent performance degradation as network size increases, as its local-first bias becomes increasingly suboptimal when remote resources become abundant. RR and RO, being oblivious to system state, show the poorest generalization with latency increases of 2.1-3.5× under extreme conditions. These results collectively validate that the MADDPG algorithm, through centralized training and decentralized execution, learns transferable scheduling policies that maintain effectiveness across diverse operational conditions, thereby demonstrating strong practical potential for real-world deployment where system configurations may evolve over time.

5. Conclusion

The objective of this paper is to present a new edge-cloud collaborative scheduling algorithm using MADDPG as the central method-. This algorithm, through multi-agent representation of edge nodes, jointly optimizes the task offloading, expert cache, and 5G NR-U competition parameters based on local observations while utilizing a centralized critic for global collaborative training. By overcoming the simultaneous dual challenges of 5G NR-U channels experiencing random fluctuations, as well as representing the dynamic imbalance of Load the Expert (MoE), The proposed method results in balanced load, optimized energy use, and low latency with high throughput capabilities. This study constructs a unified stochastic optimization model to provide a comprehensive representation of channel dynamics, expert task characteristics, and cache states and develops a multi-agent reinforcement learning framework that would utilize continuous action space to provide fine-grained joint scheduling over protocol stacks and computational tasks, as well as systematically validating through a series of experiments that demonstrate the strength, robustness and generalization capabilities of the proposed algorithm in dynamic environments. These findings support the theoretical and technical framework for the efficient distributed training of large-scale sparse models across a range of complex edge environments.

References:

[1] Chen Q, Xu X, You Z, et al. Communication-efficient federated edge learning for NR-U-based IIoT networks. IEEE Internet of Things Journal, 2021, 9(14): 12450-12459.

[2] Sathya V, Kala SM, Naidu K. Heterogenous networks: From small cells to 5G NR-U. Wireless Personal Communications, 2023, 128(4): 2779-2810.

[3] Chen Q, Yang K, Jiang H, et al. Joint beamforming coordination and user selection for CoMP-enabled NR-U networks. IEEE Internet of Things Journal, 2021, 9(16): 14530-14541.

[4] Daraseliya A, Sopin E, Moltchanov D, et al. Performance of offloading strategies in collocated deployments of millimeter wave NR-U technology. IEEE Transactions on Vehicular Technology, 2022, 72(2): 2535-2549.
[5] Xing C, Li Y, Chen C, et al. Determinantal point process-based new radio unlicensed link scheduling for multi-access edge computing. World Wide Web, 2022, 25(5): 2215-2239.
[6] Yu D, Shen L, Hao H, et al. Moesys: A distributed and efficient mixture-of-experts training and inference system for internet services. IEEE Transactions on Services Computing, 2024, 17(5): 2626-2639.
[7] Yuan X, Kong W, Luo Z, et al. Efficient inference offloading for mixture-of-experts large language models in Internet of Medical Things. Electronics, 2024, 13(11): 2077.
[8] Wang S, Bi S, Zhang YJ A. Deep reinforcement learning with communication transformer for adaptive live streaming in wireless edge networks. IEEE Journal on Selected Areas in Communications, 2021, 40(1): 308-322.
[9] Gao Z, Yang L, Dai Y. Fast adaptive task offloading and resource allocation via multiagent reinforcement learning in heterogeneous vehicular fog computing. IEEE Internet of Things Journal, 2022, 10(8): 6818-6835.
[10] Duan Q, Wang S, Ansari N. Convergence of networking and cloud/edge computing: Status, challenges, and opportunities. IEEE Network, 2020, 34(6): 148-155.
[11] Yang J, Jiao L, Shang R, et al. Ept-net: Edge perception transformer for 3d medical image segmentation. IEEE Transactions on Medical Imaging, 2023, 42(11): 3229-3243.
[12] Meuser T, Lovén L, Bhuyan M, et al. Revisiting edge ai: Opportunities and challenges[J]. IEEE Internet Computing, 2024, 28(4): 49-59.
[13] Mao Y, Yu X, Huang K, et al. Green edge AI: A contemporary survey[J]. Proceedings of the IEEE, 2024, 112(7): 880-911.
[14] Surianarayanan C, Lawrence J J, Chelliah P R, et al. A survey on optimization techniques for edge artificial intelligence (AI)[J]. Sensors, 2023, 23(3): 1279.
[15] Badidi E. Edge AI for early detection of chronic diseases and the spread of infectious diseases: opportunities, challenges, and future directions[J]. Future Internet, 2023, 15(11): 370.
[16] Shen Y, Shao J, Zhang X, et al. Large language models empowered autonomous edge AI for connected intelligence[J]. IEEE Communications Magazine, 2024, 62(10): 140-146.
[17] Zou X, Li K, Zhou J T, et al. Robust edge ai for real-time industry 4.0 applications in 5g environment[J]. IEEE Communications Standards Magazine, 2023, 7(2): 64-70.
[18] Biswas A, Wang H C. Autonomous vehicles enabled by the integration of IoT, edge intelligence, 5G, and blockchain[J]. Sensors, 2023, 23(4): 1963.
[19] Li F, Wang C. Artificial intelligence and edge computing for teaching quality evaluation based on 5G-enabled wireless communication technology[J]. Journal of Cloud Computing, 2023, 12(1): 45.
[20] Zeb S, Rathore M A, Hassan S A, et al. Toward AI-enabled NextG networks with edge intelligence-assisted microservice orchestration[J]. IEEE Wireless Communications, 2023, 30(3): 148-156.
[21] Abdellatif A A, Abo-Eleneen A, Mohamed A, et al. Intelligent-slicing: An AI-assisted network slicing framework for 5G-and-beyond networks[J]. IEEE Transactions on Network and Service Management, 2023, 20(2): 1024-1039.
[22] Gill S S, Golec M, Hu J, et al. Edge AI: A taxonomy, systematic review and future directions[J]. Cluster Computing, 2025, 28(1): 18.
[23] Duan S, Wang D, Ren J, et al. Distributed artificial intelligence empowered by end-edge-cloud computing: A survey. IEEE Communications Surveys & Tutorials, 2022, 25(1): 591-624.
[24] Wang Y, Yang C, Lan S, et al. End-edge-cloud collaborative computing for deep learning: A comprehensive survey. IEEE Communications Surveys & Tutorials, 2024, 26(4): 2647-2683.
[25] Qin J, Zhang X, Liu B, et al. A split-federated learning and edge-cloud based efficient and privacy-preserving large-scale item recommendation model. Journal of Cloud Computing, 2023, 12(1): 57.
[26] Chen C, Li Y, Wang Q, et al. An intelligent edge-cloud collaborative framework for communication security in distributed cyber-physical systems. IEEE Network, 2023, 38(1): 172-179.
[27] Wu G, Zhou F, Ding G, et al. An efficient heterogeneous edge-cloud learning framework for spectrum data compression. IEEE Transactions on Mobile Computing, 2022, 22(7): 3823-3839.
[28] Loginov V, Khorov E, Lyakhov A, et al. CR-LBT: Listen-before-talk with collision resolution for 5G NR-U networks. IEEE Transactions on Mobile Computing, 2021, 21(9): 3138-3149.
[29] Pei X, Qian H, Wang H, et al. An Improved Listen-Before-Talk Scheme for Uplink Multiple Access in 5G Unlicensed Band. IEEE Internet of Things Journal, 2022, 9(20): 19843-19853.
[30] Pavlova I, Bankov D, Khorov E, et al. On the Benefits of Listen before Talk Scheme for NB-Fi Networks. Sensors, 2023, 23(22): 9054.
[31] Yin Y, Li T, Xiong L, et al. A broadband switched-transformer digital power amplifier for deep back-off efficiency enhancement. IEEE Journal of Solid-State Circuits, 2020, 55(11): 2997-3008.

[32] Qin L, Zhou L, Hassan W, et al. A family of transformer-less single-switch dual-inductor high voltage gain boost converters with reduced voltage and current stresses. IEEE Transactions on Power Electronics, 2020, 36(5): 5674-5685.
[33] Hou W, Wen H, Song H, et al. Multiagent deep reinforcement learning for task offloading and resource allocation in cybertwin-based networks. IEEE Internet of Things Journal, 2021, 8(22): 16256-16268.
[34] Shen S, Han Y, Wang X, et al. Collaborative learning-based scheduling for kubernetes-oriented edge-cloud network. Ieee/Acm Transactions on Networking, 2023, 31(6): 2950-2964.
[35] Chen C, Zhao A, Zhang Z, et al. Research on Multi-Agent Collaborative Scheduling Planning Method for Time-Triggered Networks. Electronics, 2025, 14(13): 2575.
[36] Li H, Han B, Li G, et al. Decentralized collaborative optimal scheduling for EV charging stations based on multi-agent reinforcement learning. IET Generation, Transmission & Distribution, 2024, 18(6): 1172-1183.
[37] Li Y, Wei Z, Su J, et al. A multi-agent collaboration scheme for energy-efficient task scheduling in a 3D UAV-MEC space. Frontiers of Information Technology & Electronic Engineering, 2024, 25(6): 824-838.
[38] Dalin L, Haijiao W, Zhen Y, et al. An online distributed satellite cooperative observation scheduling algorithm based on multiagent deep reinforcement learning. IEEE Geoscience and Remote Sensing Letters, 2020, 18(11): 1901-1905.
[39] Chang H, Liu Y, Sheng Z. Distributed multi-agent reinforcement learning for collaborative path planning and scheduling in blockchain-based cognitive internet of vehicles. IEEE Transactions on Vehicular Technology, 2023, 73(5): 6301-6317.
[40] Lu K, Li RD, Li MC, et al. MADDPG-based joint optimization of task partitioning and computation resource allocation in mobile edge computing. Neural Computing and Applications, 2023, 35(22): 16559-16576.
[41] Wang Y, Huang Z, Wei Z, et al. MADDPG-based offloading strategy for timing-dependent tasks in edge computing. Future Internet, 2024, 16(6): 181.
[42] Cui Q, Zhao X, Ni W, et al. Multi-agent deep reinforcement learning-based interdependent computing for mobile edge computing-assisted robot teams. IEEE Transactions on Vehicular Technology, 2022, 72(5): 6599-6610.
[43] Zhadan A, Allahverdyan A, Kondratov I, et al. Multi-agent reinforcement learning-based adaptive heterogeneous DAG scheduling. ACM transactions on intelligent systems and technology, 2023, 14(5): 1-26.
[44] Zhao Y, Mo L, Liu J. Multi-task reinforcement in vehicular edge computing: a multi-agent learning approach. CCF Transactions on Pervasive Computing and Interaction, 2024, 6(4): 348-364.
[45] Du J, Kong Z, Sun A, et al. MADDPG-based joint service placement and task offloading in MEC empowered air–ground integrated networks. IEEE Internet of Things Journal, 2023, 11(6): 10600-10615.