Spatio-Temporal Adaptive Reinforcement Learning for Task Offloading in Mobile Edge Computing*

Boyao Qu, Mingqiang Zhu, Tianhao Peng, Boyang Shang, and Yuyin Ma†

Beijing Jiaotong University, Beijing, China
{boyaoqu, mqzhu, th.peng, byshang, mayuyin}@bjtu.edu.cn

Abstract

Mobile Edge Computing (MEC) reduces latency by offloading intensive tasks to edge servers, yet dynamic multi-user environments make offloading highly challenging. Deep reinforcement learning (DRL) has been widely adopted, but prior approaches often fail to jointly capture network topology and temporal dynamics. To overcome this limitation, we propose spatio-temporal adaptive reinforcement learning (STAR), which formulates the offloading problem as a markov decision process (MDP) with the MEC network represented as a heterogeneous graph. STAR combines an attention-based spatial perception layer to extract graph features with a temporal memory layer to model state evolution, enabling proactive and context-aware offloading. Comprehensive simulations demonstrate that STAR reduces mean processing delay (MPD) by over 25% and offloading failure rates (OFR) by approximately 26%. Beyond efficiency improvements, the learned policy generalizes well, consistently achieving low latency and high reliability in novel and large-scale network environments.

Keywords: Mobile Edge Computing, Task Offloading, Deep Reinforcement Learning

1 Introduction

With the rapid development of 5G and future 6G wireless networks, a variety of novel mobile applications, such as Augmented Reality (AR), online gaming, and the Internet of Things (IoT), have flourished. While creating new security challenges for the mobile internet, these computation-intensive and latency-sensitive applications also impose significant challenges on mobile devices, which are typically constrained by limited processing power and battery capacity. To address the issue, Mobile Edge Computing (MEC) has emerged as a promising paradigm [1]. By deploying servers with powerful computational capabilities at the edge of the wireless network, in close proximity to mobile users, MEC can provide rich resources to support demanding services.

Within the MEC framework, task offloading is a key technology that allows mobile devices to delegate their heavy computational tasks to nearby edge servers for execution, thus reducing latency and energy consumption [2]. Despite its potential, designing an effective task offloading strategy remains a significant challenge due to the dynamic and complex nature of the MEC environment. The state of the system, which includes wireless channels that vary in time and arrivals of stochastic tasks, evolves rapidly [3], making the offload decision a complex sequential problem. Furthermore, offloading decisions for individual users are highly interdependent and require management through a collaborative framework [4]. The action of one user can affect the performance of others through shared wireless resources and interference, which introduces

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. 81, December 16-18, 2025, Sapporo , Japan. © The copyright of this paper remains with the author(s).

†Corresponding author

complex relationships defined by the network topology. In many real-world IoT scenarios, tasks are not monolithic but are composed of multiple subtasks with intrinsic dependencies [5]. Moreover, modern networks are increasingly heterogeneous, integrating various wired and wireless technologies to support specialized services such as e-health and industrial automation [6] [7]. As network paradigms evolve towards decentralization, new challenges such as ensuring data privacy also emerge [8]. Consequently, traditional reinforcement learning agents using flattened state vectors cannot effectively perceive this structural and relational information, often resulting in suboptimal offloading strategies.

A key and often overlooked challenge is to capture the temporal dynamics of the network. For example, the trend of the workload of a server over time is a more valuable indicator of future performance than its instantaneous state. However, most existing methods are memoryless, making decisions based only on a snapshot of the current network state. This reactive approach prevents the agent from anticipating future network conditions and making forward-looking proactive offloading decisions.

To address the aforementioned challenges, we introduce a novel framework named STAR (Spatio-Temporal Adaptive Reinforcement learning) which is designed to make intelligent task offloading decisions by concurrently processing both spatial and temporal information within the MEC network. The STAR agent utilizes a graph neural network (GNN) to explicitly model the network topology, thereby comprehending the complex interdependencies among users and servers. To overcome the limitations of reactive, memoryless approaches, the spatial representation is further augmented with a recurrent neural network (RNN) based memory module. This unique GNN-RNN architecture allows the agent to first perceive a holistic snapshot of the network's spatial state at each timestep, and then to process the sequence of these snapshots to identify critical temporal trends.

By comprehending both the current network state and temporal evolution, the design significantly enhances the efficiency of task offloading decisions, markedly reducing mean processing delay and offloading failures in complex, dynamic network environments. A key advantage is the system's adaptability and generalizability; experimental results demonstrate that the learned policy can be effectively transferred to larger, unseen network topologies, maintaining high performance where traditional learning-based methods degrade sharply. The flexibility and scalability offered by the proposed approach provide a pivotal solution for managing computational resources in next-generation mobile networks, offering significant research and application value.

The subsequent sections of this paper are organized as follows. Section 2 provides a review of existing literature on MEC task offloading. Section 3 details the system model and formulates the task offloading challenge as a markov decision process (MDP). Section 4 describes our proposed STAR framework, focusing on its spatio-temporal design. Section 5 evaluates the framework's effectiveness using comprehensive simulations. The paper concludes in Section 6, which also outlines potential avenues for future work.

2 Related Work

To overcome the limitations of traditional optimization and heuristic algorithms in dynamic MEC environments, deep reinforcement learning (DRL) has been widely adopted. However, early DRL methods often lack global topology awareness, relying instead on local perspectives—for example, Sohaib et al. [9] considered only the nearest server, while Huang et al. [10] used a central controller for task migration. Other paradigms, such as Zheng et al.'s [11] offloading-caching framework, generally lack DRL's adaptive decision-making. To better capture network architecture, recent studies introduced graph neural networks. Gao et al. [12] applied graph recurrent networks to large-scale systems, Hu et al. [13] achieved decentralized D2D offloading via graphs, and Chen et al. [14] designed a two-stage load-balancing scheme, though at the cost of coordination complexity. These works significantly advanced modeling of spatial complexity in MEC.

However, GNN-based methods mostly capture spatial dependencies at a single time, while MEC also entails crucial temporal dynamics. To address this, researchers employed recurrent neural networks: Tang and Wong [15] used LSTMs to estimate long-term costs under fluctuating loads, and Zhang et al. [16] leveraged GRUs to predict future tasks. Wang et al. [17] applied a Sequence-to-Sequence (S2S) model for dependent tasks, and Wu et al. [18] used a similar model for user mobility, though these approaches neglect network topology. Other efforts explored multi-objective optimization (Xiao et al. [19]) or intelligent collaboration (Song et al. [20]), but without jointly exploiting spatio-temporal structures. This reveals a gap: a unified framework combining deep spatial perception with temporal trend analysis. Our work addresses this by integrating GNNs' spatial modeling with RNNs' temporal memory to enable proactive, forward-looking offloading decisions.

3 System Model and Problem Formulation

To address the task offloading problem, a formal foundation for the MEC environment is first established. The analysis begins with a system architecture defining the properties of key entities such as users and edge servers. Mathematical models for network communication, task characteristics, and computation processes are then introduced. This formalization defines the agent's operating environment, culminating in the formulation of the dynamic offloading problem as a markov decision process, which is essential for the reinforcement learning solution proposed in the subsequent section.

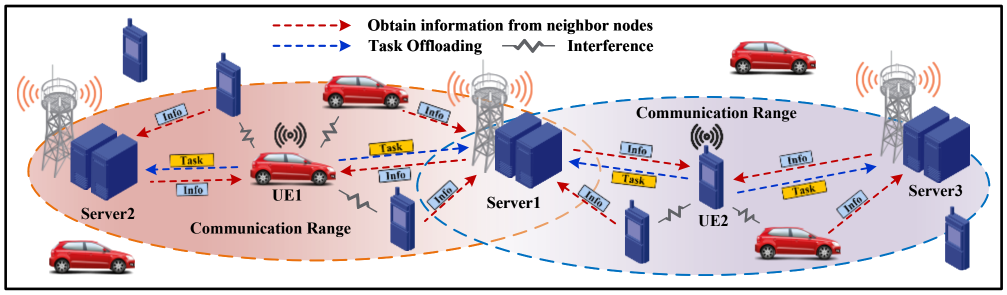


Figure 1: System model of MEC network

3.1 System Overview

The system under study is a multi-user, multi-server MEC environment, with the architecture illustrated in Fig. 1. It operates in discrete time slots indexed by $t \in \{0, 1, 2, \ldots\}$. The architecture has two layers: the terminal device layer and the edge computing layer, connected by a wireless access network. The base station acts not only as a communication relay but also as an information hub that assists decentralized agents with local state information.

The terminal device layer comprises M heterogeneous user equipments (UEs), $\mathcal{M} = \{1, 2, \ldots, M\}$, each with an independent agent. UEs, such as smartphones, vehicles, and AR glasses, are randomly distributed in a two-dimensional area. Each UE $m \in \mathcal{M}$ has distinct properties, including local computational capacity f_m^{local} (cycles/s), maximum transmission power $P_m^{\max}$, and may stochastically generate a new task in any slot.

The edge computing layer consists of N MEC servers, $\mathcal{S} = \{1, 2, \ldots, N\}$, usually co-located with access points (e.g., 5G BSs). Each server $s \in \mathcal{S}$ has heterogeneous but powerful computation capacity F_s^{edge}. While agents act locally, they are aided by the base station, which maintains a broader view and provides information for topologically aware decisions, avoiding the cost of global information collection.

Moreover, each agent must decide for any pending task between two exclusive actions: (1) local execution on the UE CPU, or (2) offloading via wireless link to an MEC server for remote processing. This creates a trade-off between computation and communication latency: local execution is limited by UE capacity, whereas offloading reduces computation delay but may incur transmission variability. The goal of this work is to design intelligent policies that enable agents to balance this trade-off and optimize overall system performance.

3.2 Component Model

Based on the system architecture, the models for network communication, task characteristics, and computation processes are now formalized.

3.2.1 Network and Communication Model

We model the MEC network at each time slot t as a heterogeneous graph, $G_t = (\mathcal{V}, E_t)$, where $\mathcal{V} = \mathcal{M} \cup \mathcal{S}$ is the set of all user and server nodes. Each node possesses a feature vector representing its state. The feature vector for a user node $m \in \mathcal{M}$ is defined as $v_m^t = [p_m, P_m^{\max}, f_m^{\mathrm{local}}, D_m^t, C_m^t, Q_m^{\mathrm{local},t}]$, which includes its geographical coordinates, maximum transmission power, local CPU frequency, current task's data size, required computation cycles, and local task queue latency, respectively. The feature vector for a server node $s \in \mathcal{S}$ is $v_s^t = [p_s, F_s^{\mathrm{edge}}, Q_s^{\mathrm{edge},t}]$, representing its coordinates, CPU frequency, and task queue latency.

The communication link between user m and server s exists if their distance $d_{m,s}$ is within the maximum range $D_{\max}$. The wireless transmission rate is determined by the signal-to-interference-plus-noise ratio (SINR). When a set of users $\mathcal{M}_{\mathrm{tx}}(t) \subseteq \mathcal{M}$ are simultaneously offloading, the SINR for user $m \in \mathcal{M}_{\mathrm{tx}}(t)$ at its target server s is given by:

$$\mathrm{SINR}_{m,s}^t = \frac{P_m^{\mathrm{tx}} \cdot h_{m,s}}{\sum_{j \in \mathcal{M}_{\mathrm{tx}}(t), j \neq m} P_j^{\mathrm{tx}} \cdot h_{j,s} + \sigma^2}, \tag{1}$$

where P_m^{tx} is the transmission power of user m, $h_{m,s}$ is the channel gain between m and s (determined by path loss), and σ^2 is the background noise power. The achievable data rate $R_{m,s}^t$ is then calculated using the Shannon-Hartley theorem:

$$R_{m,s}^t = B \log_2(1 + \mathrm{SINR}_{m,s}^t), \tag{2}$$

where B is the channel bandwidth.

3.2.2 Task Offloading Decision Model

We assume that at each time slot t, each user $m \in \mathcal{M}$ may generate a new computational task with a certain probability. A task is characterized by a tuple $\text{Task}_m^t = (D_m^t, C_m^t)$, where D_m^t is the size of the task's input data in bits, and C_m^t is the total number of CPU cycles required to complete the computation. Both D_m^t and C_m^t are randomly drawn from predefined uniform distributions, reflecting the heterogeneity of mobile applications. If no task is generated for user m at time slot t, we set $D_m^t = 0$ and $C_m^t = 0$. The total processing delay of a task is contingent upon the binary offloading decision $o_m^t \in \{0, 1\}$.

Case 1: Local Execution ($o_m^t = 0$)

If the task is executed locally, its total delay $T_{m,\text{local}}^t$ consists of the waiting time in the local queue and the actual computation time:

$$T_{m,\text{local}}^t = Q_m^{\text{local},t-1} + \frac{C_m^t}{f_m^{\text{local}}}, \tag{3}$$

where $Q_m^{\text{local},t-1}$ is the remaining processing time of tasks already in the user's local queue from the previous time slot.

Case 2: Edge Offloading ($o_m^t = 1$)

If the task is offloaded to a selected server $k_m^t \in \mathcal{S}_m^t$, the total delay $T_{m,k,\text{offload}}^t$ is the sum of three components: transmission delay, queuing delay at the server, and execution delay at the server:

$$T_{m,k,\text{offload}}^t = T_{m,k,\text{trans}}^t + Q_k^{\text{edge},t-1} + T_{m,k,\text{exec}}^t, \tag{4}$$

where the transmission delay is $T_{m,k,\text{trans}}^t = D_m^t / R_{m,k}^t$, $Q_k^{\text{edge},t-1}$ is the server's queue latency from the previous time slot, and the edge execution delay is $T_{m,k,\text{exec}}^t = C_m^t / F_k^{\text{edge}}$. An offloading attempt is considered a failure if the transmission time exceeds a maximum tolerable threshold, i.e., $T_{m,k,\text{trans}}^t > \tau_{\max}$.

3.3 Optimization Objective

The overarching objective of this work is to find an optimal, dynamic offloading policy, denoted by π^*, that minimizes the long-term average cost for all users in the system. This cost is a composite of Mean Processing Delay (MPD) and penalties incurred from Offloading Failures (OFR). Formally, the optimization problem can be stated as:

$$\min_{\pi} \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}_{\pi} \left[\frac{1}{M} \sum_{m=1}^{M} C_m^t \right], \tag{5}$$

where C_m^t represents the total cost for user m at time slot t, which encompasses both processing delay and any applicable failure penalties. T is the total number of time slots, and the expectation $\mathbb{E}_{\pi}[\cdot]$ is taken over the stochastic nature of task arrivals and wireless channels under policy π. To solve this complex stochastic optimization problem, we formulate the dynamic and sequential task offloading problem as a MDP, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$.

State Space ($\mathcal{S}$): The state S_m^t for an agent on user m at time slot t is the localized heterogeneous graph G_m^t provided by the base station.

Action Space ($\mathcal{A}$): The action A_m^t for agent m follows a two-stage decision process: (1) a binary offloading decision $o_m^t \in \{0, 1\}$, and (2) a server selection $k_m^t \in \mathcal{S}_m^t$ if $o_m^t = 1$.

Reward Function ($\mathcal{R}$): To align the agent's learning objective with our system-level goal of cost minimization, we define the immediate reward as the negative of the total cost incurred: $R_{t+1}^{m} = -C_{m}^{t}$.

Objective ($\mathcal{O}$): By defining the reward in this manner, the RL agent's objective of finding a policy π_θ that maximizes the expected cumulative discounted reward, $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t R_{t+1}\right]$, becomes mathematically equivalent to solving the optimization problem stated in Equation (5).

4 STAR Design and Implementation

STAR's core design principle is to endow each decentralized agent with spatio-temporal awareness. This is achieved through a unique neural network architecture that integrates a GNN to comprehend spatial topology with a RNN to model temporal dynamics. A centralized training, decentralized execution (CTDE) paradigm is employed to effectively train the resulting policy.

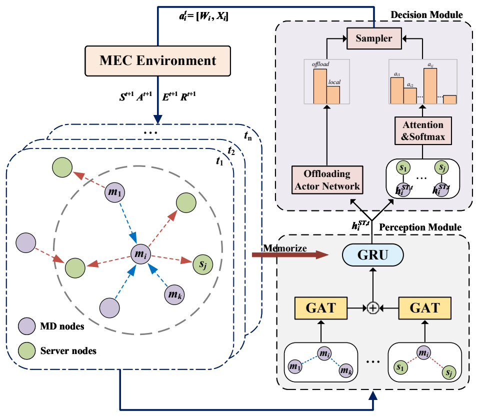


Figure 2: Architecture of our proposed STAR

4.1 Network State Representation

The dynamic MEC network at each time slot t is modeled as a heterogeneous graph, denoted by $G_t = (\mathcal{V}, \mathcal{E}_t)$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}_t$ is the set of edges. This graph serves as the fundamental data structure for the agent's perception module.

The set of all nodes $\mathcal{V}$ is a union of two distinct types: a set of M UE nodes, $\mathcal{M} = \{m_1, m_2, \ldots, m_M\}$, and a set of N MEC server nodes, $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$. Each UE node $m_i \in \mathcal{M}$ is characterized by a time-varying feature vector $v_i^t = [p_i, P_i^{\max}, f_i^{\text{local}}, D_i^t, C_i^t, Q_i^{\text{local},t}]$, representing its geographical coordinates, maximum transmission power, local CPU frequency, current task's data size, required computation cycles, and local task queue latency, respectively. Each server node $s_j \in \mathcal{S}$ is defined by a time-varying feature vector $u_j^t = [p_j, F_j^{\text{edge}}, Q_j^{\text{edge},t}]$, denoting the server's coordinates, its powerful CPU frequency, and its current task queue latency.

The set of edges $\mathcal{E}_t$ represents the relationships between nodes and consists of two primary types. The first type is user-server edges, where a communication edge (m_i, s_j) exists if the distance between UE m_i and server s_j is within a predefined communication range; this edge is attributed with the channel gain between the two nodes. The second type is user-user edges, where a logical edge (m_i, m_k) exists if two UEs are within each other's interference range. This edge is attributed with their inter-distance and is crucial for modeling mutual interference.

The graph is considered dynamic because the node features, particularly task-related attributes (D_i^t, C_i^t) and queue latencies $(Q_i^{\text{local},t}, Q_j^{\text{edge},t})$, change at every time slot.

4.2 Markov Decision Process

To address the complexities of dynamic task offloading, we formulate the problem as a partially observable MDP (POMDP) from the perspective of each decentralized agent. This is because each agent only has access to a localized observation of the total network state. The POMDP is formally defined by the tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{R})$.

The true state of the environment at each time slot t is the complete global graph G_t, which is generally inaccessible to individual agents. Instead, each agent m_i receives a localized and variable-sized observation $\mathcal{O}_i^t$ from the base station, consisting of a subgraph of its local network environment, including its own state, neighboring UEs, and accessible servers. Based on this partial observation, the agent selects a hierarchical action $\mathcal{A}_i^t = (o_i^t, k_i^t)$, which comprises a binary offloading decision ($o_i^t \in \{0, 1\}$) and a subsequent server selection ($k_i^t \in \mathcal{S}_i^t$) if offloading is chosen. The reward function is designed to guide the agent towards minimizing system costs; the immediate reward R_{t+1}^i is defined as the negative of the total cost C_i^t, which includes the task's processing delay and a large penalty for any offloading failures.

4.3 Spatio-temporal Adaptive Learning

The STAR framework, as depicted in Fig. 2, is comprised of a spatio-temporal representation module for perception and a policy network for decision-making. The entire system is trained under a CTDE paradigm.

The agent's perception process begins with the spatio-temporal representation module. To handle the variable-sized graph observation O_t^i, the agent first employs a GNN with an attention mechanism (GAT-like) to extract a fixed-length spatial feature vector $h_i^{S,t}$. The attention mechanism allows the agent to learn the relative importance of its neighbors by computing attention coefficients, α_{ij}, which are used to weight the features of neighboring nodes during aggregation. Subsequently, a GRU-based memory module takes the sequence of these spatial vectors $\{\ldots, h_i^{S,t-1}, h_i^{S,t}\}$ as input. The GRU's hidden state integrates historical information, producing a final spatio-temporal feature vector $h_i^{ST,t}$, which serves as the agent's rich, context-aware belief state.

This spatio-temporal feature $h_i^{ST,t}$ is then fed into the decision-making module, which is a dual-head policy network. The first head, a multi-layer perceptron (MLP), outputs the probability distribution over the offloading actions $\{p_{\text{local}}, p_{\text{offload}}\}$. The second head computes another probability distribution over the available servers. The final action A_t^i is sampled from these two distributions.

The entire network is trained end-to-end using the reinforce algorithm. This policy gradient method updates the policy parameters θ by performing gradient ascent on the objective of maximizing the expected long-term reward $J(\theta)$. After a full episode trajectory is collected, the policy gradient is estimated as:

$$\nabla_\theta J(\theta) \approx \frac{1}{T} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(A_t \mid S_t) G_t, \tag{6}$$

where G_t is the total return of the episode, used here as an unbiased estimate of the future return from time step t. The network parameters are then updated according to the following rule:

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta J(\theta), \tag{7}$$

where α is the learning rate.

5 Experiment and Result Analysis

STAR is validated through extensive simulations, where it is benchmarked against representative methods on key efficiency and reliability metrics, demonstrating superior effectiveness and learning efficiency. Its adaptability and generalization are further confirmed via zero-shot testing on large-scale, unseen network topologies, showcasing its readiness for dynamic, real-world MEC environments.

5.1 Experimental Design

To validate our proposed method, we constructed a simulation platform using Python with the PyTorch and PyTorch Geometric libraries. All experiments were executed on a server equipped with an NVIDIA GPU. The simulation models a square area where UEs are randomly distributed and MEC servers are placed at fixed locations. We test all methods across a range of network scales, from small configurations (e.g., 15 UEs, 3 servers) to large-scale environments (e.g., 75 UEs, 15 servers).

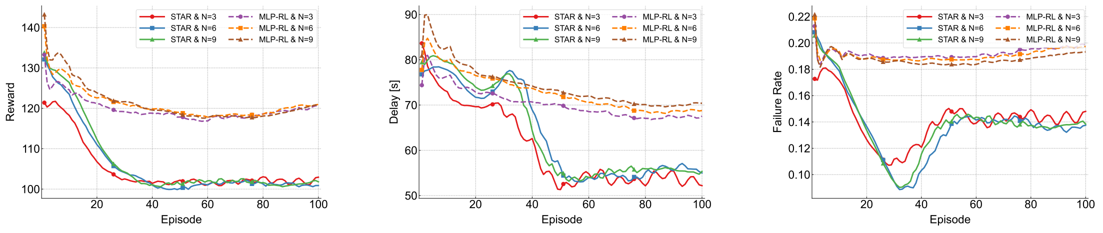
The performance of the proposed STAR agent is benchmarked against four representative baseline methods. The first is MLP-RL, a learning-based agent using a standard MLP that cannot perceive the network's topological structure. The other three are heuristic strategies: Local-Only, where all computational tasks are executed on the local device; Max-SINR, where all users attempt to offload tasks to the edge server with the best channel condition; and shortest-queue-first (SQF), an intelligent heuristic where users offload to the server with the shortest current task queue.

The performance is measured using two primary metrics. The first is MPD(s), defined as the average time from a task's generation to its completion, which evaluates the overall system efficiency. The second is OFR, defined as the proportion of offloaded tasks that fail to complete transmission within a predefined time threshold, which measures the reliability of an offloading policy. For both metrics, lower values indicate better performance.

5.2 Result Analysis

The results of the first experiment are presented in Fig. 3, which compares the learning curves of our proposed STAR method and the MLP-RL baseline across three network scales in terms of training rewards, MPD and OFR.

The STAR agent's convergence to a more optimal policy is evidenced in Fig. 3a by its lower final reward value (approximately 102) compared to the baseline's 118. This enhanced performance is further quantified by a marked reduction in mean processing delay to 50–55 seconds, in contrast to the baseline's 68–72 seconds (Fig. 3b). Concurrently, Fig. 3c reveals a significantly lower offloading failure rate of approximately 0.14 for STAR, whereas the baseline remains higher at 0.19–0.20. This sustained performance advantage across varying network scales underscores the superior scalability of the proposed spatio-temporal approach, affirming



(a) Rewards. (b) Meaning process delay. (c) Offloading failure rate.

Figure 3: Comparison of the performance of STAR and MLP-RL at different network scales

STAR's ability to manage the complex resource and interference dependencies inherent in larger systems—a capability the topology-agnostic MLP-RL agent lacks.

The results of the second experiment, designed to evaluate the adaptability and generalization capabilities of our method, are summarized in Table 1. This experiment subjects the agents to a two-phase procedure: a sequential training and fine-tuning phase on networks with an increasing number of servers ($N = \{3, 5, 7, 9\}$), followed by a zero-shot testing phase on unseen, larger network topologies ($N = \{11, 13, 15\}$).

Table 1: Performance Comparison on Adaptability and Generalization

Method	Training and Fine-tuning								Testing					
	N=3		N=5		N=7		N=9		N=11		N=13		N=15	
	MPD/s	OFR	MPD/s	OFR	MPD/s	OFR	MPD/s	OFR	MPD/s	OFR	MPD/s	OFR	MPD/s	OFR
STAR	**58.750**	**0.134**	**55.798**	**0.127**	**55.580**	**0.139**	**55.008**	**0.138**	**61.439**	**0.142**	**62.840**	**0.147**	**62.258**	**0.143**
MLP-RL	74.091	0.192	81.637	0.196	84.748	0.198	82.596	0.196	222.071	0.235	222.722	0.297	170.502	0.293
Max-SINR	118.866	0.321	126.671	0.324	136.015	0.328	134.533	0.324	134.927	0.322	135.218	0.327	135.979	0.327
SQF	82.189	0.324	80.083	0.320	86.961	0.322	86.312	0.323	87.375	0.325	91.784	0.325	88.160	0.326
Local-Only	223.861	/	223.213	/	225.986	/	225.228	/	225.964	/	229.233	/	226.295	/

The experimental results demonstrate that the STAR model achieves superior performance in both adaptability and generalization compared to all baselines. During the training and fine-tuning phase, STAR consistently maintains the lowest MPD and OFR, outperforming the next-best MLP-RL method by approximately 33% in MPD at $N = 9$. Critically, when tested on larger, unseen topologies up to $N = 15$, STAR exhibits robust generalization with only a graceful performance degradation. In stark contrast, the MLP-RL baseline fails completely, with its MPD more than doubling, which powerfully validates the efficacy of STAR's spatio-temporal architecture in learning transferable principles of network dynamics.

6 Conclusion

This paper proposes STAR, a novel spatio-temporal reinforcement learning framework for dynamic task offloading in complex MEC environments. By integrating a GNN for spatial perception with a RNN for temporal memory, STAR enables agents to comprehend both network topology and its temporal dynamics for proactive decision making. Experimental results demonstrate that STAR significantly improves system performance, reducing mean processing delay by over 25% and offloading failures by approximately 26%. Furthermore, the framework exhibits remarkable adaptability, maintaining robust performance and low latency in large-scale, unseen topologies where traditional, topology-agnostic methods fail. This framework offers a promising solution for intelligent resource management in next-generation MEC systems..

The spatio-temporal framework presented in this paper achieves a new level of intelligent and adaptive offloading in dynamic MEC environments. However, as network environments become increasingly complex and user demands more diverse, it is imperative to continually enhance the framework. Future work will focus on three key areas: first, incorporating user mobility models to handle highly dynamic topologies; second, introducing multi-objective optimization to balance latency with energy consumption; and third, exploring more advanced policy gradient algorithms, such as PPO, to further improve training stability and sample efficiency. We are committed to refining the general framework for spatio-temporal decision-making in wireless networks, aiming to fully unleash the innovation potential of this research direction.

7 Acknowledgements

This work was supported in part by the National Key R&D Program of China under Grant 2023YFB2904200, and in part by the Research Fund of Beijing Jiaotong University under Grant W25B0900010.

References

[1] Mobasshir Mahbub, Md. Shamrat Apu Gazi, Sayed Al Arabi Provat, and Md. Saiful Islam. Multi-access edge computing-aware internet of things: Mec-iot. In *2020 Emerging Technology in Computing, Communication and Electronics (ETCCE)*, pages 1–6, 2020.

[2] Xiaowei Liu, Shuwen Jiang, and Yi Wu. A novel deep reinforcement learning approach for task offloading in mec systems. *Applied Sciences*, 12(21), 2022.

[3] Han Hu, Weiwei Song, Qun Wang, Rose Qingyang Hu, and Hongbo Zhu. Energy efficiency and delay tradeoff in an mec-enabled mobile iot network. *IEEE Internet of Things Journal*, 9(17):15942–15956, 2022.

[4] Fei Song, Yuyin Ma, Ilsun You, and Hongke Zhang. Smart collaborative evolvement for virtual group creation in customized industrial iot. *IEEE Transactions on Network Science and Engineering*, 10(5):2514–2524, Sep. 2023.

[5] Furong Chai, Qi Zhang, Haipeng Yao, Xiangjun Xin, Ran Gao, and Mohsen Guizani. Joint multi-task offloading and resource allocation for mobile edge computing systems in satellite iot. *IEEE Transactions on Vehicular Technology*, 72(6):7783–7795, 2023.

[6] Fei Song, Letian Li, Ilsun You, and Hongke Zhang. Enabling heterogeneous deterministic networks with smart collaborative theory. *IEEE Transactions on Network Science and Engineering*, 8(4):2798–2813, Oct 2021.

[7] Fei Song, Mingqiang Zhu, Yutong Zhou, Ilsun You, and Hongke Zhang. Smart collaborative tracking for ubiquitous power iot in edge-cloud interplay domain. *IEEE Internet of Things Journal*, 7(7):6046–6055, 2020.

[8] Fei Song, Yuyin Ma, Zhenhui Yuan, Ilsun You, Giovanni Pau, and Hongke Zhang. Exploring reliable decentralized networks with smart collaborative theory. *IEEE Communications Magazine*, 61(8):44–50, August 2023.

[9] Muhammad Sohaib, Sang-Woon Jeon, and Wei Yu. Hybrid online–offline learning for task offloading in mobile edge computing systems. *Trans. Wireless. Comm.*, 23(7):6873–6888, July 2024.

[10] Sheng-Zhi Huang, Kun-Yu Lin, and Chin-Lin Hu. Intelligent task migration with deep qlearning in multi-access edge computing. *IET Communications*, 16(11):1290–1302, 2022.

[11] Yi-fan Zheng, Ning Wei, and Yi Liu. Collaborative computation for offloading and caching strategy using intelligent edge computing. *Mobile Information Systems*, 2022(1):4840801, 2022.

[12] Zhen Gao, Lei Yang, and Yu Dai. Fast adaptive task offloading and resource allocation in large-scale mec systems via multiagent graph reinforcement learning. *IEEE Internet of Things Journal*, 11(1):758–776, 2024.

[13] Zheyuan Hu, Jianwei Niu, Tao Ren, Xuefeng Liu, and Mohsen Guizani. Sitoff: Enabling size-insensitive task offloading in d2d-assisted mobile edge computing. *IEEE Transactions on Mobile Computing*, 24(3):1567–1584, 2025.

[14] Xing Chen, Zewei Yao, Zheyi Chen, Geyong Min, Xianghan Zheng, and Chunming Rong. Load balancing for multiedge collaboration in wireless metropolitan area networks: A two-stage decision-making approach. *IEEE Internet of Things Journal*, 10(19):17124–17136, 2023.

[15] Ming Tang and Vincent W.S. Wong. Deep reinforcement learning for task offloading in mobile edge computing systems. *IEEE Transactions on Mobile Computing*, 21(6):1985–1997, 2022.

[16] Xiao Zhang, Yuxiong He, Youhuai Wang, Xiaoming Chen, Shi Jin, and Ye Liang. Task offloading based on gru model in iot. In *Proceedings of the 2022 6th International Conference on Electronic Information Technology and Computer Engineering*, EITCE '22, page 151–156, New York, NY, USA, 2023. Association for Computing Machinery.

[17] Jin Wang, Jia Hu, Geyong Min, Wenhan Zhan, Albert Y. Zomaya, and Nektarios Georgalas. Dependent task offloading for edge computing based on deep reinforcement learning. *IEEE Transactions on Computers*, 71(10):2449–2461, 2022.

[18] Chao-Lun Wu, Te-Chuan Chiu, Chih-Yu Wang, and Ai-Chun Pang. Mobility-aware deep reinforcement learning with seq2seq mobility prediction for offloading and allocation in edge computing. *IEEE Transactions on Mobile Computing*, 23(6):6803–6819, 2024.

[19] Zhu Xiao, Jinmei Shu, Hongbo Jiang, John C. S. Lui, Geyong Min, Jiangchuan Liu, and Schahram Dustdar. Multi-objective parallel task offloading and content caching in d2d-aided mec networks. *IEEE Transactions on Mobile Computing*, 22(11):6599–6615, 2023.

[20] Fei Song, Letian Li, Ilsun You, Shui Yu, and Hongke Zhang. Optimizing high-speed mobile networks with smart collaborative theory. *IEEE Wireless Communications*, 29(3):48–54, June 2022.