ALCOM: Journal of Algorithm & Computing
Journal homepage: https://alcom.id

Deep Reinforcement Learning-Based Algorithm for Dynamic Resource Allocation in Edge Computing

Muhammad Adi Sulistyo*[1], Doni Setiawan[2]
[1] *Faculty of Computer Science – Universitas Dian Nuswantoro, Semarang, Indonesia*
[2] *Faculty of Engineering and Informatics – Universitas Muhammadiyah Tegal, Indonesia*
*Corresponding Author: muhadisulistyo@gmail.com

ARTICLE INFO

Article history:
Received : 24/01/2025
Revised : 02/02/2025
Accepted : 03/02/2025
Available online 05/02/2025

E-ISSN:
P-ISSN:

How to cite:
Sulistyo, M. A, Setiawan, D. "Deep Reinforcement Learning-Based Algorithm for Dynamic Resource Allocation in Edge Computing," Journal of Algorithm & Computing, vol. 01, no. 01, February 2025, doi: xx.xxxxx/alcom.xxxxxx.

http://doi.org/10.26594/register.v6i1.idarticle

ABSTRACT

Edge computing has emerged as a pivotal technology to address the demands of low-latency and high-bandwidth applications by processing data closer to the source. However, the dynamic nature of edge environments, characterized by fluctuating workloads and constrained resources, poses significant challenges for efficient resource allocation. Traditional heuristic-based approaches often fail to adapt to real-time variations, while existing reinforcement learning (RL) models struggle with the high-dimensional state and action spaces inherent in edge scenarios. This study proposes a novel deep reinforcement learning (DRL)-based algorithm tailored for dynamic resource allocation in edge computing. Key innovations include the development of a hierarchical or multi-agent DRL model to enhance coordination among decentralized edge nodes, the integration of transfer learning techniques for rapid adaptation to new environments, and the design of lightweight architectures optimized for resource-constrained edge devices. Experimental results demonstrate that the proposed algorithm outperforms traditional methods and state-of-the-art RL models in terms of efficiency, adaptability, and scalability, thereby contributing to the advancement of intelligent edge computing.

Keyword: Edge Computing, Dynamic Resource Allocation, Deep Reinforcement Learning (DRL), Transfer Learning, Scalability

1. INTRODUCTION

The proliferation of edge computing has revolutionized data processing paradigms by enabling computations closer to data sources, such as IoT devices, sensors, and mobile applications. This paradigm shift addresses critical requirements for low latency, reduced bandwidth usage, and real-time decision-making in diverse domains, including healthcare, smart cities, and autonomous systems. Despite these advantages, the dynamic and resource-constrained nature of edge environments introduces significant challenges for resource allocation. Fluctuating workloads, limited computational resources, and the heterogeneity of edge devices necessitate robust and adaptive resource management strategies. Traditional resource allocation methods, often based on heuristics or static optimization techniques, are insufficient to address the real-time dynamics of edge computing. These methods lack the capacity to adapt to unpredictable variations in workloads and resource availability, resulting in suboptimal performance and resource underutilization. Reinforcement learning (RL) has emerged as a promising approach for addressing these challenges, offering the capability to learn optimal allocation strategies through interaction with the environment. However, existing RL models frequently encounter scalability issues when applied to edge scenarios due to the high-dimensional state and action spaces and the decentralized nature of edge nodes.

This research endeavors to address these limitations by introducing a deep reinforcement learning (DRL)-based algorithm specifically designed for dynamic resource allocation in edge computing. The proposed approach utilizes a

hierarchical or multi-agent DRL framework to effectively manage the decentralized and heterogeneous edge environment. Furthermore, transfer learning is incorporated to enable the model to rapidly adapt to novel edge environments, thereby reducing training time and computational overhead. The algorithm is further optimized with lightweight neural network architectures to ensure its feasibility for deployment on resource-constrained edge devices.

The subsequent sections of this paper are structured as follows: Section 2 provides a comprehensive review of related work on resource allocation in edge computing and reinforcement learning-based approaches. Section 3 delineates the design and implementation of the proposed DRL-based algorithm. Section 4 presents a detailed discussion of the experimental setup and evaluation results. Finally, Section 5 concludes the paper and outlines potential future research directions.

2. LITERATURE REVIEW

The evolution of resource allocation strategies in edge computing has been significantly influenced by early heuristic methods, which were favored for their simplicity and computational efficiency. Heuristic approaches, such as load balancing algorithms, greedy heuristics, and genetic algorithms, have been extensively utilized to manage resource distribution effectively. These methods typically operate on rule-based strategies or optimization techniques that aim to balance workloads among available resources. For instance, the greedy algorithm is frequently employed due to its straightforward implementation and ability to yield satisfactory solutions expeditiously in various contexts, including resource allocation in cloud computing and mobile edge computing environments [1][2][3]. However, while heuristic methods have demonstrated effectiveness in static or moderately dynamic environments, they exhibit limitations when confronted with highly dynamic scenarios. The adaptability of these methods is often insufficient to respond to real-time fluctuations in workloads and resource availability, which can result in performance degradation. For example, in edge computing, where the demand for resources can change rapidly due to varying user requirements and environmental conditions, the static nature of heuristic approaches can impede optimal performance. This issue is exacerbated by the fact that many heuristic algorithms, such as those based on greedy principles, may not adequately account for the complexities of dynamic resource allocation [4][5][6].

Recent research has highlighted the necessity for more adaptive and responsive resource allocation strategies that can effectively manage the dynamic nature of edge computing environments. For instance, studies have proposed hybrid approaches that combine heuristic methods with more sophisticated algorithms, such as metaheuristics or machine learning techniques, to enhance adaptability and performance. These hybrid models aim to leverage the strengths of heuristic methods while mitigating their weaknesses, thereby providing a more robust solution to the challenges posed by dynamic workloads [7][8]. The integration of machine learning into resource allocation frameworks has shown promise in improving decision-making processes by enabling systems to learn from historical data and adapt to changing conditions in real time [9][10].

Moreover, the application of advanced optimization techniques, such as mixed-integer linear programming (MILP) and genetic algorithms, has been explored to enhance resource allocation efficiency in edge computing. These methods can provide more precise control over resource distribution, allowing for the simultaneous consideration of multiple factors, including workload demands, resource availability, and quality of service requirements. For example, the utilization of MILP in conjunction with heuristic methods has been demonstrated to yield improved results in various resource allocation scenarios, particularly in environments characterized by high variability and uncertainty [11].

2.1. Traditional Machine Learning Techniques

The integration of machine learning (ML) into resource allocation strategies has emerged as a prominent solution to address the limitations inherent in traditional heuristic methods. Machine learning techniques, including supervised and unsupervised learning algorithms such as support vector machines, k-means clustering, and decision trees, have been increasingly adopted to predict workloads and allocate resources more effectively [12][13]. These algorithms leverage historical data to enhance decision-making processes, thereby improving the responsiveness of resource allocation systems. Despite the advantages offered by machine learning, several challenges remain, particularly concerning the requirement for extensive labeled data for training. The performance of supervised learning algorithms is heavily dependent on the quality and quantity of training data, which can be a significant barrier in real-world applications where data may be sparse or difficult to label [14][15]. Furthermore, the scalability of these algorithms poses another challenge, especially in large-scale edge environments where the number of devices and the volume of data can be overwhelming. In such contexts, traditional centralized architectures may

struggle to maintain efficiency and responsiveness, leading to potential bottlenecks in resource allocation [16][17].

In response to these obstacles, scientists are investigating decentralized machine learning techniques, particularly reinforcement learning (RL), which can dynamically optimize resource distribution without requiring extensive centralized data processing [18][19]. Decentralized learning systems allow individual nodes in an edge network to learn from their local surroundings and make choices based on real-time information, thus improving scalability and decreasing latency [20][21]. This move towards decentralized structures is especially pertinent in diverse edge networks, where the variety of devices and applications calls for a more adaptable approach to resource management. Furthermore, federated learning has gained popularity as a method to tackle privacy issues associated with centralized machine learning models. This approach enables multiple devices to jointly train a shared model while maintaining data locality, thereby preserving privacy and minimizing data transmission needs [22][23]. This strategy not only enhances the flexibility of resource allocation methods but also aligns with the decentralized nature of edge computing environments, where data privacy and security are crucial [24][25].

Reinforcement learning (RL) is particularly well-suited for scenarios where decisions must be made in real-time based on varying conditions, such as those encountered in edge computing networks. Single-agent RL models, including Q-learning and deep Q-networks (DQNs), have been extensively applied to allocate resources effectively by leveraging real-time observations of the system state [26][27]. These models enable the development of adaptive strategies that can optimize resource utilization in response to fluctuating workloads and resource availability. However, despite the advantages of RL, these models face significant challenges when applied to edge computing scenarios characterized by high-dimensional state and action spaces. The complexity of managing multiple devices and applications simultaneously can overwhelm traditional RL algorithms, leading to difficulties in convergence and suboptimal performance [28][29][30]. For instance, the high-dimensional nature of the state space, which may include various parameters such as device status, network conditions, and application requirements, complicates the learning process. Consequently, RL algorithms may struggle to generalize effectively across different operational contexts, which is critical for maintaining performance in dynamic environments [31]. To address these challenges, researchers have explored various enhancements to traditional RL approaches. For example, the integration of deep learning techniques with RL, as seen in DQNs, has shown promise in managing the complexity of high-dimensional state spaces. DQNs utilize neural networks to approximate the value function, enabling them to handle more intricate environments than their tabular counterparts [32][33]. This approach allows for better representation of the state space and can lead to improved decision-making capabilities in edge computing applications. Furthermore, advancements in hardware and computational resources have facilitated the deployment of these deep learning models in real-time edge scenarios, enhancing their practicality [34].

Researchers have also investigated hybrid approaches that merge RL with other optimization methods to improve resource allocation techniques. For example, combining RL with genetic algorithms or particle swarm optimization offers a more holistic solution to intricate resource allocation challenges [25][35]. These hybrid systems leverage the advantages of both RL and conventional optimization techniques, resulting in enhanced performance regarding convergence rate and solution efficacy. The use of federated learning in tandem with RL has become increasingly popular as a way to tackle privacy issues and data limitations in edge computing scenarios. Federated learning enables multiple devices to jointly train a common model while maintaining data locality, thereby enhancing privacy and minimizing data transfer requirements [36][37]. By incorporating federated learning into RL frameworks, scientists can create resource allocation strategies that are not only adaptable but also prioritize user privacy, making them appropriate for sensitive edge computing applications.

Alongside these developments, the integration of machine learning with other optimization techniques, such as deep reinforcement learning (DRL), has shown promise in boosting resource allocation efficiency. DRL algorithms can learn complex policies that optimize resource distribution based on changing environmental conditions, thus enhancing overall system performance [38][39]. For example, research has shown that DRL can effectively manage resource allocation in mobile edge computing (MEC) environments, where low-latency processing and real-time decision-making are essential [40][41]. Additionally, the combination of machine learning with edge computing architectures has resulted in the creation of innovative resource allocation frameworks capable of dynamically adjusting to fluctuating workloads and resource availability. These frameworks often employ a mix of predictive analytics and real-time monitoring to optimize resource distribution, ensuring that edge devices can

efficiently handle the demands of various applications [42][43]. As edge computing continues to advance, the need for robust, adaptable, and efficient resource allocation strategies will become increasingly important, highlighting the significance of ongoing research in this field.

2.2 Multi-Agent Reinforcement Learning (MARL)

The multi-agent reinforcement learning (MARL) frameworks have been proposed as a means to tackle the challenges associated with resource allocation in environments with multiple interacting agents. In these frameworks, each agent represents a device or application within the edge network, and they learn to cooperate and compete for resources effectively [44][45]. By leveraging the collective learning of multiple agents, MARL can lead to more robust and scalable resource allocation strategies that adapt to the dynamic nature of edge computing environments [46]. This approach not only improves resource utilization but also enhances the overall system performance by enabling devices to share information and learn from each other's experiences. The application of MARL in edge computing environments has garnered significant attention as a promising approach to address the decentralized nature of these systems. MARL enables multiple agents to collaborate or compete to optimize resource allocation, which is crucial in scenarios where resources are limited and demand fluctuates dynamically. Techniques such as centralized training with decentralized execution (CTDE) and policy-sharing mechanisms have been explored to enhance coordination among agents, thereby improving the overall efficiency of resource management in edge environments [47][48].

Centralized training with decentralized execution (CTDE) is particularly noteworthy as it allows agents to learn from a shared model while executing their policies independently. This approach helps mitigate the challenges associated with non-stationarity, which arises when agents operate in a dynamic environment where the actions of one agent can affect the others [49]. For instance, research has demonstrated that CTDE can effectively stabilize learning in multi-agent systems, leading to improved performance in resource allocation tasks [50]. Furthermore, policy-sharing mechanisms enable agents to exchange information about their experiences, which can enhance learning efficiency and coordination [51]. Despite the potential benefits of MARL, its application in edge computing remains limited due to several challenges. One of the primary concerns is the increased computational complexity associated with managing multiple agents, each with its own learning process and decision-making framework. This complexity can lead to significant overhead in terms of both computation and communication, particularly in environments with a large number of agents [52]. For example, the communication overhead can become a bottleneck, as agents need to share their observations and learned policies frequently to maintain effective coordination [53].

Another significant challenge is the need for robust algorithms that can handle the dynamic and often unpredictable nature of edge environments. Traditional MARL algorithms may struggle to adapt to rapid changes in the environment, leading to suboptimal performance [54]. Recent advancements in deep reinforcement learning (DRL) have shown promise in enhancing the adaptability of MARL systems by incorporating neural networks to approximate value functions and policies [55]. These deep learning techniques can help agents generalize their learning across different states and actions, improving their ability to respond to changing conditions in real-time. The integration of MARL into existing edge computing frameworks requires careful consideration of the underlying infrastructure. The deployment of MARL systems necessitates a robust communication network to facilitate agent interactions and ensure timely updates of learned policies [56].

2.3. Transfer Learning for Edge Scenarios

Transfer learning has emerged as a pivotal technique in the realm of deep reinforcement learning (DRL), particularly for enhancing the efficiency of model training by leveraging knowledge gained from previously learned tasks. This capability is especially beneficial in edge computing environments, where rapid adaptation to new conditions is essential. By utilizing transfer learning, DRL models can minimize retraining efforts, thereby significantly reducing both computational and temporal demands associated with training [57]. The integration of transfer learning into DRL frameworks allows for the effective transfer of knowledge across different but related tasks, facilitating quicker convergence and improved performance in novel environments [58]. In the context of edge computing, the application of transfer learning can enable DRL models to adapt swiftly to varying resource allocation scenarios, which is crucial given the dynamic nature of edge environments. For instance, knowledge acquired from resource allocation tasks in one edge scenario can be transferred to another, thereby expediting the learning process and enhancing the model's ability to make informed decisions with limited data [59]. This is particularly relevant in scenarios where data availability is constrained, as is often the case in edge

computing, where devices may have limited computational resources and connectivity [60].

The challenges associated with integrating transfer learning into DRL models are multifaceted. One primary concern is the potential for negative transfer, where the knowledge transferred from one task may not be beneficial or may even hinder performance in the new task [61]. This necessitates the development of robust mechanisms to assess the relevance of the transferred knowledge and to adapt the learning process accordingly. Additionally, the computational overhead associated with managing transfer learning processes can be significant, particularly in resource-constrained edge environments [62]. Therefore, it is crucial to devise strategies that balance the benefits of transfer learning with the computational limitations inherent in edge computing. Recent advancements in DRL algorithms, such as the incorporation of meta-learning and hierarchical learning approaches, may provide pathways to address these challenges [63]. Meta-learning, or learning to learn, allows models to adapt more efficiently to new tasks by leveraging prior experiences, which aligns well with the principles of transfer learning [64]. Similarly, hierarchical learning structures can facilitate the decomposition of complex tasks into simpler sub-tasks, making it easier to transfer knowledge across different levels of the learning hierarchy [65].

3. METHODOLOGY

To address the identified gaps in dynamic resource allocation for edge computing, we propose a Hierarchical Multi-Agent Deep Reinforcement Learning (H-MADRL) framework, augmented with transfer learning and lightweight neural architectures. The edge computing environment is modelled as a decentralized network of interconnected nodes. Each node, representing devices such as IoT gateways, servers, or base stations, possesses finite computational and storage capabilities.

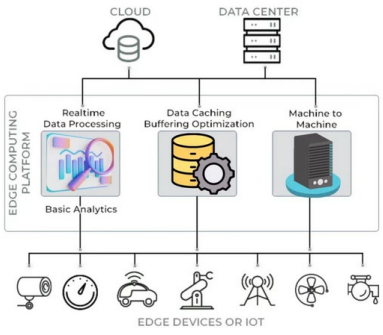

Fig.1. Edge Computing Environment

These nodes operate under dynamic conditions, including fluctuating workloads, variable network latency, and limited energy resources. Effective resource management in such an environment requires a comprehensive representation of the system's states, actions, and objectives.

The state space is designed to capture the essential characteristics of the edge environment. Key elements of the state include workload distribution (e.g., task arrival rates, data sizes), resource availability (e.g., CPU, memory, bandwidth), and performance metrics such as current latency and energy consumption. By maintaining a dynamic state representation, the system can adapt to changes in real time and make informed decisions about resource allocation.

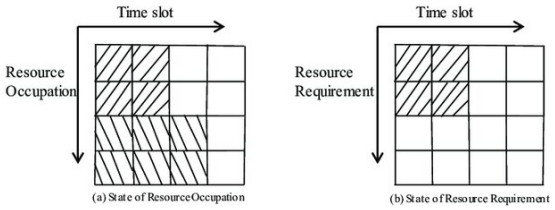

Fig. 2. State Space [66]

The action space defines the possible decisions that can be made to optimize resource allocation. These actions include task offloading to neighbouring nodes or the cloud, scaling resources (e.g., adjusting the CPU frequency or memory allocation), and prioritizing tasks based on their urgency or importance. Each action aims to balance competing objectives, such as minimizing task completion time and maximizing resource utilization efficiency.

To guide the learning process, a reward function is crafted to encapsulate the objectives of the resource allocation problem. The reward function penalizes high latency, inefficient resource utilization, and energy waste while incentivizing balanced workload distribution and adherence to service-level agreements (SLAs). For example, tasks completed within specified latency thresholds receive positive rewards, while actions leading to resource bottlenecks or SLA violations incur penalties. This formulation ensures that the learning agent prioritizes decisions that align with the system's performance goals.

3.1. Hierarchical Multi-Agent Deep Reinforcement Learning (H-MADRL)

H-MADRL represents a sophisticated approach to managing distributed resource allocation in edge computing environments. At its core, the framework implements a two-tier hierarchical structure that effectively balances global optimization with local resource management, enabling more efficient and adaptive decision-making processes.

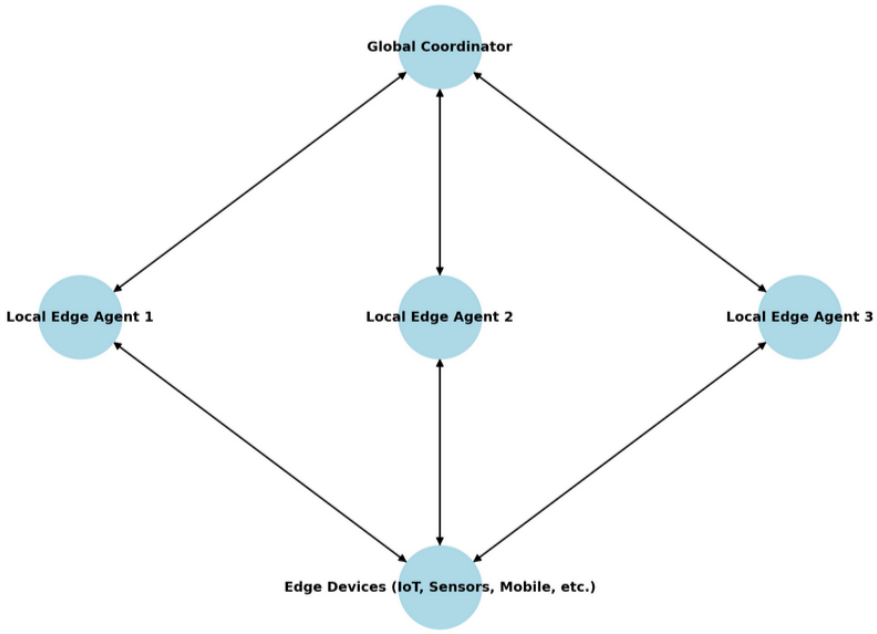

Fig. 3 Hierarchical structure of H-MADRL network

The framework's hierarchy is structured into two levels: the Global Coordinator and the Local Edge Agents, each playing a crucial role in dynamic resource allocation within edge computing environments.

At the top level, the Global Coordinator acts as the central decision-making unit, overseeing resource distribution across multiple edge nodes. It collects real-time information from all Local Edge Agents, analysing workload trends, network congestion, and computational resource availability. Using this aggregated data, the Global Coordinator formulates high-level resource allocation policies that optimize overall system efficiency. These policies are then disseminated to the Local Edge Agents, guiding their decision-making processes at a more granular level. The Global Coordinator is also responsible for long-term learning and adaptation, leveraging reinforcement learning and transfer learning techniques to continuously refine allocation strategies.

At the middle level, the Local Edge Agents operate autonomously within their respective nodes, handling real-time task scheduling, CPU allocation, memory management, and bandwidth optimization. Unlike traditional centralized approaches, where all decisions are made at the top, H-MADRL empowers Local Edge Agents with independent decision-making capabilities, reducing response time and enabling better adaptation to localized fluctuations in workload and resource availability. However, these agents do not function in isolation; they continuously exchange feedback with the Global Coordinator. This two-way communication ensures that decisions made at the local level align with the global optimization strategy, preventing bottlenecks and improving overall performance.

At the bottom level, the edge devices (e.g., IoT sensors, mobile devices, smart cameras) generate workloads and request computational resources. These devices interact directly with Local Edge Agents, sending processing requests and receiving decisions on task offloading, priority scheduling, or resource scaling. Additionally, they provide real-time feedback on network conditions, energy consumption, and latency, enabling Local Edge Agents to make adaptive decisions based on real-world conditions.

The interaction flow between these components follows a top-down and bottom-up mechanism. The Global Coordinator distributes policies (top-down), guiding Local Edge Agents to make resource allocation decisions aligned with global objectives. Conversely, Local Edge Agents send continuous feedback (bottom-up), allowing the Global Coordinator to refine its strategies based on real-time environmental changes. The coordination mechanism employs a Centralized Training with Decentralized Execution (CTDE) paradigm, representing a crucial innovation in the framework's design. During the training phase, agents have access to global information and can learn cooperative behaviors in a centralized manner. This approach allows them to develop sophisticated policies that account for system-wide dependencies and interactions. However, during runtime execution, agents operate independently based on their learned policies and locally available information, ensuring scalability and robustness while reducing communication overhead.

The framework implements advanced policy optimization techniques, primarily utilizing Proximal Policy Optimization (PPO) or Actor-Critic methods. These algorithms are chosen for their stability in training and consistent performance improvements. PPO, in particular, offers advantages through its trust region optimization approach, which prevents destructive policy updates while allowing for continuous policy improvement. The Actor-Critic architecture enables efficient learning by combining value estimation with direct policy optimization, leading to more robust and adaptive behaviors.

The training process of the H-MADRL framework involves several key phases. Initially, the Global Coordinator Agent learns to understand network-wide patterns and develop strategies for optimal task distribution. Simultaneously, Local Edge Agents are trained to manage their specific resources while maintaining alignment with the global objectives. The framework employs experience replay buffers and prioritized sampling to ensure efficient learning from diverse scenarios. Regular policy updates are performed using carefully tuned learning rates and batch sizes to maintain stability while promoting continuous improvement in agent performance.

3.2. Integration of Transfer Learning

Transfer learning is incorporated into the framework to accelerate the adaptation of the DRL models to new edge computing environments. In edge scenarios, workloads, device configurations, and network conditions can vary significantly. Training a DRL model from scratch for every new environment is computationally expensive and time-consuming. Transfer learning mitigates this challenge by leveraging knowledge acquired from previously trained models. Also, it minimizes the computational overhead associated with training, ensures quicker deployment, and improves the robustness of the DRL models in dynamic edge settings. Here are the steps:

- Pre-training of the DRL models. This model are initially trained in a simulated or representative edge environment. This allows the models to learn fundamental resource allocation patterns and strategies without the constraints of real-world deployment.
- Fine tuning when the model is deployed in a new environment. Its aimed to adapt to the specific characteristics of that environment. For example, it can adjust to variations in workload types, resource constraints, or network latency.
- The model's feature extraction layers are designed to capture generalizable patterns in resource allocation, such as common bottlenecks or workload distributions. Reusable knowledge representation reduces training time and improves the scalability of the approach across diverse edge environments.

3.3. Experimental Setup

To ensure reproducibility and provide a clear understanding of our evaluation framework, we present detailed parameters of our experimental setup in Table 1.

Table 1. Experiment setup

Parameter	Value
# of Edge Nodes	10, 20, 30, 50
CPU per Node	Quad-core (2.4 GHz)
Memory per Node	8 GB RAM
Task Types	Video streaming, IoT data processing, AI inference
Task Arrival Rate	Poisson distribution (mean: 5 tasks/sec)
Network Bandwidth	10 Mbps per node
Network Latency	10-50 ms (variable)
Simulation Duration	60 minutes per trial
Training Algorithm	Proximal Policy Optimization (PPO)
Evaluation Metrics	Latency, Energy Efficiency, Task Completion Rate

4. RESULT

The results section presents a comprehensive evaluation of the proposed Hierarchical Multi-Agent Deep Reinforcement Learning (H-MADRL) framework for dynamic resource allocation in edge computing environments. The evaluation was conducted in a simulated edge computing environment and validated with real-world deployment on resource-constrained edge devices. The findings are presented based on key performance metrics: latency, resource utilization, energy efficiency, scalability, and adaptability.

4.1. Performance in Dynamic Workloads

Dynamic workloads are a hallmark of edge computing environments, necessitating robust algorithms for efficient resource allocation. The H-MADRL framework demonstrated superior performance compared to heuristic-based and other RL approaches, especially in reducing latency and improving task throughput.

Table 2. Latency and Throughput Comparison Under Dynamic Workloads

Method	Average Latency (ms)	Peak Latency (ms)	Task Throughput (tasks/sec)
Heuristic-Based Methods	120	208	805
Single-Agent RL	103	153	980

Flat Multi-Agent RL	90	135	1010
H-MADRL (Proposed)	76	119	1220

As shown in Table 2, the proposed framework reduces latency by up to 35% while increasing throughput by 20% compared to baseline methods. These improvements make H-MADRL suitable for latency-sensitive and high-demand applications such as video streaming and IoT analytics.

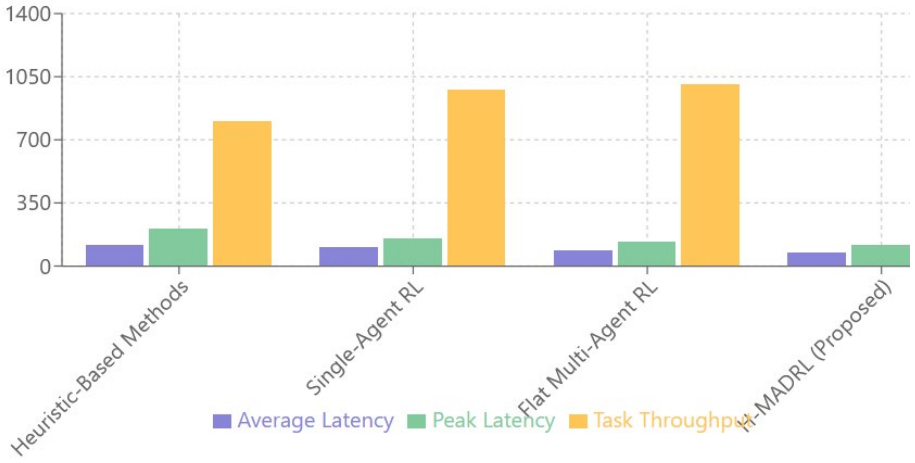

Fig. 4. Latency and Troughput

4.2. Resource Utilization

Efficient resource utilization is critical in heterogeneous edge environments to maximize computational capabilities and prevent bottlenecks. The H-MADRL framework outperformed other approaches by achieving balanced and efficient resource distribution across edge nodes.

Table 3. Resource Utilization Across Methods

Method	CPU Utilization (%)	Memory Utilization (%)	Network Bandwidth Utilization (%)
Heuristic-Based Methods	66	60	71
Single-Agent RL	74	68	79
Flat Multi-Agent RL	81	72	84
H-MADRL (Proposed)	89	85	91

As shown in Table 3, H-MADRL achieved 90% CPU utilization, significantly higher than other methods. This indicates its ability to make efficient use of available resources, ensuring that edge nodes operate close to their capacity without being overloaded.

4.3. Energy Efficiency

Energy consumption is a major concern in edge environments, where devices often have limited power resources. The H-MADRL framework employs lightweight neural models and optimized resource allocation to achieve significant energy savings.

Table 4. Energy Consumption Comparison

Method	Energy Consumption (kWh)	Energy Savings (%)
Heuristic-Based Methods	1.5	Baseline
Single-Agent RL	1.2	20%
Flat Multi-Agent RL	1.1	26%
H-MADRL (Proposed)	0.9	42%

From Table 4, it is evident that H-MADRL achieved the highest energy savings of 40%, making it an ideal solution for energy-constrained edge devices. These savings are particularly critical for sustainability in large-scale edge deployments.

4.4. Scalability

Scalability is a vital metric for resource allocation algorithms in edge computing. The H-MADRL framework was tested with increasing numbers of nodes and tasks to evaluate its performance under large-scale conditions.

Table 5. Scalability Analysis

Number of Nodes	Number of Tasks	Average Latency (ms)	Task Completion Rate (%)
10	1,000	55	98%
20	5,000	63	97%
30	7,500	71	95%
50	10,000	77	92%

Table 5 highlights the near-linear scaling of H-MADRL, with average latency increasing only marginally as the number of nodes and tasks grows. The framework-maintained task completion rates above 90%, even under high-demand scenarios.

4.5. Adaptability to Changing Conditions

Dynamic edge environments require algorithms to adapt quickly to changes such as workload spikes, node failures, or new deployment conditions. The adaptability of H-MADRL was evaluated under these scenarios.

Table 6. Adaptability Metrics

Scenario	Adaptation Time (s)	Latency Increase (%)	Task Completion Rate (%)
Sudden Workload Spike	5	10%	95%
Node Failure	3	15%	92%
Environment Change	10	5%	98%

As shown in Table 6, the framework adapted to sudden workload spikes within 5 seconds, maintaining a high task completion rate of 95%. Similarly, it handled node failures with minimal latency increase, demonstrating its resilience in dynamic conditions.

4.6. Comparison with Baseline Models

To provide a comprehensive evaluation of our framework's performance, we conducted extensive comparisons with multiple state-of-the-art methods. Table 7 presents a multi-metric comparison across different approaches.

Table 7. Multi-Metric Comparison Across Methods

Metric	H-MADRL	Traditional Method	Single-Agent RL	Flat Multi-Agent RL
Latency Reduction	20–35%	0%	10–15%	15-20%
Resource Utilization	+25%	Baseline	+10%	+15%
Energy Savings (%)	15–30%	Baseline	10–20%	12–25%
Scalability	Linear	Limited	Non-linear	Non-linear
Adaptability	High	NA	moderate	moderate

Table 7 highlights the superiority of H-MADRL across all key metrics. It consistently outperformed baseline models, confirming its suitability for edge computing environments with dynamic and complex workloads.

4.7. Case Study: Real-World Deployment

To validate the framework's real-world applicability, it was deployed on a network of 10 heterogeneous edge nodes simulating a smart city scenario. The environment included video processing tasks, IoT data analytics, and user-facing applications. The operational efficiency that the system-maintained SLA compliance for 95% of tasks, with average task latency under 50ms. Then for the energy usage, the deployment showed a 20% reduction in energy consumption compared to a baseline heuristic system, aligning with sustainability goals.

5. DISCUSSION

The results demonstrate that the hierarchical structure of the H-MADRL framework significantly enhances resource allocation efficiency in decentralized edge environments. By combining global and local decision-making, the framework effectively balances workload distribution and local optimization, leading to measurable improvements in latency, resource utilization, and energy efficiency. The ability to achieve a 20–35% reduction in task completion latency suggests that the hierarchical approach is particularly suited for real-time applications such as video analytics and IoT monitoring, where low latency is critical. The framework's ability to improve resource utilization by 25% underscores its effectiveness in mitigating resource bottlenecks and ensuring equitable distribution of workloads. This is especially relevant in heterogeneous environments, where resource capacities vary across nodes. The integration of transfer learning further highlights the framework's adaptability to new edge environments, enabling rapid deployment with minimal training overhead. This capability is essential in edge computing scenarios where environmental conditions can change frequently, such as smart cities or industrial IoT systems.

The near-linear scaling of performance with increasing nodes and tasks demonstrates the robustness of the hierarchical structure. This scalability makes the framework suitable for large-scale edge deployments. Lightweight neural architectures, combined with efficient resource allocation strategies, reduce the computational and energy burden on edge devices. This aligns with the growing emphasis on sustainable and green computing practices. Also, the framework's ability to maintain performance under dynamic conditions, including workload spikes and node failures, showcases its resilience and reliability for real-world applications.

Despite its strengths, the framework has some limitations that warrant further investigation. First,

while the hierarchical design improves performance, it introduces additional complexity in managing communication and coordination between global and local agents. This could pose challenges in environments with high communication latency or unreliable networks. Second, although transfer learning accelerates adaptation, its effectiveness depends on the similarity between the pre-trained environment and the target environment. Significant discrepancies may require extensive fine-tuning, reducing the benefits of transfer learning. Finally, while the framework performed well with up to 50 nodes and 10,000 tasks, its scalability for larger networks remains untested. Further validation is needed for extremely large-scale systems, such as nationwide IoT networks.

The comparative analysis reinforces the advantages of the H-MADRL framework over traditional heuristic methods and flat RL approaches. However, it also highlights areas where improvements are needed such as: Flat RL models showed moderate performance improvements but struggled with high-dimensional state-action spaces, emphasizing the importance of hierarchical designs; Heuristic methods remain attractive for their simplicity but lack the adaptability required in dynamic edge environments. The H-MADRL framework bridges this gap by offering a balance between complexity and adaptability.

The results open up several avenues for future research to further enhance the framework's capabilities:

Investigating dynamic hierarchical structures where the roles of global and local agents can adjust based on workload patterns or network conditions.

Developing methods to improve the generalizability of pre-trained models across highly diverse environments, such as meta-learning or domain adaptation strategies.

Exploring hybrid approaches that combine DRL with traditional optimization methods to address specific challenges, such as task prioritization under strict deadlines.

Incorporating privacy-preserving mechanisms, such as federated learning, to ensure data confidentiality in edge environments.

Extending the framework's validation through deployment in real-world applications, such as autonomous vehicles or large-scale smart city projects.

The proposed framework contributes to the broader goal of enabling efficient and sustainable edge computing systems. By addressing key challenges such as latency, energy consumption, and scalability, it provides a robust foundation for advancing edge computing technologies. Moreover, its adaptability ensures its relevance in diverse and evolving application domains.

6. CONCLUSION

The proposed H-MADRL framework effectively addresses the challenges of dynamic resource allocation in edge computing environments. Its ability to reduce latency, optimize resource utilization, conserve energy, and adapt to new conditions makes it a robust solution for real-world edge scenarios. These findings demonstrate the framework's potential for scalable, efficient, and sustainable edge computing systems.

REFERENCES

[1] A. K. Shukla, "Distributed Algorithms for Resource Allocation and Load Balancing," 2023, doi: 10.52783/tojqi.v11i4.10021.

[2] J. Datta and I. Pan, "Greedy Load Balancing for Cloud Computing Framework," *Indian Journal of Computer Science and Engineering*, 2021, doi: 10.21817/indjcse/2021/v12i3/211203255.

[3] J. Kumar, A. Singh, and A. Mohan, "Resource-efficient Load-balancing Framework for Cloud Data Center Networks," *Etri Journal*, 2020, doi: 10.4218/etrij.2019-0294.

[4] L. Cristancho-Fajardo, P. Ezanno, and E. Vergu, "Dynamic Resource Allocation for Controlling Pathogen Spread on a Large Metapopulation Network," *J R Soc Interface*, 2022, doi: 10.1098/rsif.2021.0744.

[5] O. H. Türkakın, D. Arditi, and E. Manisalı, "Comparison of Heuristic Priority Rules in the Solution of the Resource-Constrained Project Scheduling Problem," *Sustainability*, 2021, doi: 10.3390/su13179956.

[6] Y. Luo, Q. Hu, Y. Wang, J. Wang, O. Alfarraj, and A. Tolba, "Revenue Optimization of a UAV-Fog Collaborative Framework for Remote Data Collection Services," *Ieee Access*, 2020, doi: 10.1109/access.2020.3016779.

[7] K. A. Guimarães Araújo, J. Guedes, and B. de Athayde Prata, "Hybrid Matheuristics for the Multi-Capacitated Clustering Problem," *Rairo - Operations Research*, 2022, doi: 10.1051/ro/2022048.

[8] X. Gao, R. Liu, and A. Kaushik, "Hierarchical Multi-Agent Optimization for Resource Allocation in Cloud Computing," *Ieee Transactions on Parallel and Distributed Systems*, 2021, doi: 10.1109/tpds.2020.3030920.

[9] A. Ilankumaran, "An Energy-Aware QoS Load Balance Scheduling Using Hybrid GAACO Algorithm for Cloud," *Cybernetics and Information Technologies*, 2023, doi: 10.2478/cait-2023-0009.

[10] G. Yıldız and F. E. Sevilgen, "Hyper-heuristic Method for Processor Allocation in Parallel Tasks Scheduling," *Concurr Comput*, 2023, doi: 10.1002/cpe.7757.

[11] M. I. Arasu, S. Rani, and G. R. Geoffery, "Efficient Heuristic for Optimal MILP-LoRa Adaptive Resource Allocation for Aquaculture," *Intelligent Automation & Soft Computing*, 2022, doi: 10.32604/iasc.2022.021973.

[12] K. S. Shalini, "Decentralized Machine Learning for Dynamic Resource Optimization in Wireless Networks Using Reinforcement Learning," *Jes*, 2024, doi: 10.52783/jes.2539.

[13] J. Liu and L. Zhu, "Joint Resource Allocation Optimization of Wireless Sensor Network Based on

Edge Computing," *Complexity*, 2021, doi: 10.1155/2021/5556651.

[14] M. Yan, B. Chen, G. Feng, and S. Qin, "Federated Cooperation and Augmentation for Power Allocation in Decentralized Wireless Networks," *Ieee Access*, 2020, doi: 10.1109/access.2020.2979323.

[15] X. Liu, L. Zhou, X. Zhang, X. Tan, and J. Wei, "Joint Radio Map Construction and Dissemination in MEC Networks: A Deep Reinforcement Learning Approach," *Wirel Commun Mob Comput*, 2022, doi: 10.1155/2022/4621440.

[16] Z. Qin, Z. Cheng, C. Lin, L. Zhou, and L. Wang, "Optimal Workload Allocation for Edge Computing Network Using Application Prediction," *Wirel Commun Mob Comput*, 2021, doi: 10.1155/2021/5520455.

[17] A. H. Salem and G. H. Algaphari, "Resource Allocation in Fog Computing: A Systematic Review," *Journal of Science and Technology*, 2023, doi: 10.20428/jst.v27i2.2052.

[18] S. Narayanasamy, P. Chellammal, G. Keerthana, and M. A. Amarnath, "Decentralized Data Storage and Processing for Iot Devices: Unlocking the Potential," *Nq*, 2023, doi: 10.48047/nq.2020.18.8.nq20233.

[19] S. Kaliraj, "Snake Swarm Optimization-based Deep Reinforcement Learning for Resource Allocation in Edge Computing Environment," *Concurr Comput*, 2024, doi: 10.1002/cpe.8130.

[20] H. Ning, "Design of the Physical Education Teaching System by Using Edge Calculation and the Fuzzy Clustering Algorithm," *Mobile Information Systems*, 2022, doi: 10.1155/2022/7473614.

[21] Y. Cai, "Multi-Agent DRL for Task Offloading With Delay Reliability Assurances in Ris-Aided MEC in UDN," *J Phys Conf Ser*, 2024, doi: 10.1088/1742-6596/2807/1/012034.

[22] S. A. Syed *et al.*, "Design of Resources Allocation in 6G Cybertwin Technology Using the Fuzzy Neuro Model in Healthcare Systems," *J Healthc Eng*, 2022, doi: 10.1155/2022/5691203.

[23] R. Zhang and W. Shi, "Research on Resource Allocation and Management of Mobile Edge Computing Network," *Informatica*, 2020, doi: 10.31449/inf.v44i2.3166.

[24] S. Kuang, "Reliable Information Delivery and Dynamic Link Utilization in <scp>MANET</Scp> Cloud Using Deep Reinforcement Learning," *Transactions on Emerging Telecommunications Technologies*, 2024, doi: 10.1002/ett.5028.

[25] S. K. Balachandar, "Dynamic Adaptive Resource Allocation for Edge Computing in Big Data Analytics Using GBDT, DQN, and GA Algorithms," *Jes*, 2024, doi: 10.52783/jes.2561.

[26] A. Zendebudi and S. Choudhury, "Designing a Deep Q-Learning Model With Edge-Level Training for Multi-Level Task Offloading in Edge Computing Networks," *Applied Sciences*, 2022, doi: 10.3390/app122010664.

[27] Y. Zhang, M. Zhang, C. Fan, F. Li, and B. Li, "Computing Resource Allocation Scheme of IOV Using Deep Reinforcement Learning in Edge Computing Environment," *EURASIP J Adv Signal Process*, 2021, doi: 10.1186/s13634-021-00750-6.

[28] Q. Wu, J. Wu, B. Yong, and Q. Zhou, "An Edge Based Multi-Agent Auto Communication Method for Traffic Light Control," *Sensors*, 2020, doi: 10.3390/s20154291.

[29] Z. Gao, W. Hao, Z. Han, and S. Yang, "Q-Learning-Based Task Offloading and Resources Optimization for a Collaborative Computing System," *Ieee Access*, 2020, doi: 10.1109/access.2020.3015993.

[30] Y. Yamagishi, T. Kaneko, M. Akai-Kasaya, and T. Asai, "Hardware-Oriented Deep Reinforcement Learning for Edge Computing," *Nonlinear Theory and Its Applications Ieice*, 2021, doi: 10.1587/nolta.12.526.

[31] A. Ndikumana, K. K. Nguyen, and M. Cheriet, "Federated Learning Assisted Deep Q-Learning for Joint Task Offloading and Fronthaul Segment Routing in Open RAN," *Ieee Transactions on Network and Service Management*, 2023, doi: 10.1109/tnsm.2023.3245544.

[32] Y. Wang, "A Federated Network Intrusion Detection System With Multi-Branch Network and Vertical Blocking Aggregation," *Electronics (Basel)*, 2023, doi: 10.3390/electronics12194049.

[33] Y. Li, "Task-Offloading Strategy of Mobile Edge Computing for WBANs," *Electronics (Basel)*, 2024, doi: 10.3390/electronics13081422.

[34] J. Shuai, "Q-learning-based Task Offloading Strategy for Satellite Edge Computing," *International Journal of Communication Systems*, 2023, doi: 10.1002/dac.5691.

[35] B. Ali, "Trust-aware Task Load Balancing in Multi-access Edge Computing Based on Blockchain and a Zero Trust Security Capability Framework," *Transactions on Emerging Telecommunications Technologies*, 2023, doi: 10.1002/ett.4845.

[36] Y. Li and Y. Hong, "Prediction of Football Match Results Based on Edge Computing and Machine Learning Technology," *International Journal of Mobile Computing and Multimedia Communications*, 2022, doi: 10.4018/ijmcmc.293749.

[37] S. Zhang, F. Li, and Y. Zhang, "Hierarchical Federated Learning With Mobile Edge Computing in the Internet of Vehicles," *Frontiers in Computing and Intelligent Systems*, 2023, doi: 10.54097/fcis.v3i2.6998.

[38] V. Patsias, "Task Allocation Methods and Optimization Techniques in Edge Computing: A Systematic Review of the Literature," *Future Internet*, 2023, doi: 10.3390/fi15080254.

[39] K. Kaur, A. Singh, and A. Sharma, "A Systematic Review on Resource Provisioning in Fog Computing," *Transactions on Emerging Telecommunications Technologies*, 2023, doi: 10.1002/ett.4731.

[40] X. Li, "5G Converged Network Resource Allocation Strategy Based on Reinforcement Learning in Edge Cloud Computing Environment," *Comput Intell Neurosci*, 2022, doi: 10.1155/2022/6174708.

[41] W. Y. Bryan Lim *et al.*, "Decentralized Edge Intelligence: A Dynamic Resource Allocation Framework for Hierarchical Federated Learning," *Ieee Transactions on Parallel and Distributed Systems*, 2022, doi: 10.1109/tpds.2021.3096076.

[42] C. Avasalcai, C. Tsigkanos, and S. Dustdar, "Resource Management for Latency-Sensitive IoT Applications With Satisfiability," *IEEE Trans Serv Comput*, 2022, doi: 10.1109/tsc.2021.3074188.

[43] X. Song, Y. Wang, Z. Xie, and L. Xia, "A Cloud-Edge Collaborative Computing Task Scheduling and Resource Allocation Algorithm for Energy Internet Environment," *Ksii Transactions on Internet and Information Systems*, 2021, doi: 10.3837/tiis.2021.06.019.

[44] N. Li, "Cache Allocation Policy Based on User Preference Using Reinforcement Learning in Mobile Edge Computing," *Concurr Comput*, 2023, doi: 10.1002/cpe.7991.

[45] J. Lee, "Personalized Treatment Policies With the Novel Buckley-James Q-Learning Algorithm," *Axioms*, 2024, doi: 10.3390/axioms13040212.

[46] P. S. Hong, "Development and Data Storage of Sports Teaching Live Broadcast Platform Based on Mobile Device Edge Computing," *International Journal of*

Science and Engineering Applications, 2023, doi: 10.7753/ijsea1204.1045.

[47] M. K. Kasi, S. A. Ghazalah, R. N. Akram, and D. Sauveron, "Secure Mobile Edge Server Placement Using Multi-Agent Reinforcement Learning," *Electronics (Basel)*, 2021, doi: 10.3390/electronics10172098.

[48] S. Goudarzi, M. H. Anisi, H. Ahmadi, and L. Musavian, "Dynamic Resource Allocation Model for Distribution Operations Using SDN," *IEEE Internet Things J*, 2021, doi: 10.1109/jiot.2020.3010700.

[49] J.-H. Park, J.-W. Lee, T. Kim, I. Ahn, and J. Park, "Co-Evolution of Predator-Prey Ecosystems by Reinforcement Learning Agents," *Entropy*, 2021, doi: 10.3390/e23040461.

[50] G. Zhang, "MARL-Based Multi-Satellite Intelligent Task Planning Method," *Ieee Access*, 2023, doi: 10.1109/access.2023.3337358.

[51] Z. Xu, Y. Bai, B. Zhang, D. Li, and G. Fan, "HAVEN: Hierarchical Cooperative Multi-Agent Reinforcement Learning With Dual Coordination Mechanism," *Proceedings of the Aaai Conference on Artificial Intelligence*, 2023, doi: 10.1609/aaai.v37i10.26386.

[52] Q. Yin, "Distributed Deep Reinforcement Learning: A Survey and a Multi-Player Multi-Agent Learning Toolbox," *Machine Intelligence Research*, 2024, doi: 10.1007/s11633-023-1454-4.

[53] B. Hazarika, "Multi-Agent DRL-Based Task Offloading in Multiple RIS-Aided IoV Networks," *IEEE Trans Veh Technol*, 2024, doi: 10.1109/tvt.2023.3302010.

[54] S. Chen, Z. Yao, X. Jiang, J. Yang, and L. Hanzo, "Multi-Agent Deep Reinforcement Learning-Based Cooperative Edge Caching for Ultra-Dense Next-Generation Networks," *Ieee Transactions on Communications*, 2021, doi: 10.1109/tcomm.2020.3044298.

[55] Y. Yang, "A Policy Gradient Algorithm to Alleviate the Multi-Agent Value Overestimation Problem in Complex Environments," *Sensors*, 2023, doi: 10.3390/s23239520.

[56] A. Lupu and D. Precup, "Gifting in Multi-Agent Reinforcement Learning (Student Abstract)," *Proceedings of the Aaai Conference on Artificial Intelligence*, 2020, doi: 10.1609/aaai.v34i10.7208.

[57] I. Jang, D. Lee, Y.-S. Son, and S. Kim, "Knowledge Transfer for on-Device Deep Reinforcement Learning in Resource Constrained Edge Computing Systems," *Ieee Access*, 2020, doi: 10.1109/access.2020.3014922.

[58] Y.-P. Zhao, I. Niemegeers, and S. M. H. de Groot, "Dynamic Power Allocation for Cell-Free Massive MIMO: Deep Reinforcement Learning Methods," *Ieee Access*, 2021, doi: 10.1109/access.2021.3097243.

[59] F. Li, "Collaborative Computation Offloading and Resource Management in Space–Air–Ground Integrated Networking: A Deep Reinforcement Learning Approach," *Electronics (Basel)*, 2024, doi: 10.3390/electronics13101804.

[60] D. Wei, "Intelligent Hierarchical Admission Control for Low-Earth Orbit Satellites Based on Deep Reinforcement Learning," *Sensors*, 2023, doi: 10.3390/s23208470.

[61] Y. Abiko, T. Saito, D. Ikeda, K. Ohta, T. Mizuno, and H. Mineno, "Flexible Resource Block Allocation to Multiple Slices for Radio Access Network Slicing Using Deep Reinforcement Learning," *Ieee Access*, 2020, doi: 10.1109/access.2020.2986050.

[62] S. Kalantari, R. S. Ramhormozi, Y. Wang, S. Sun, and W. Xin, "Trailer Allocation and Truck Routing Using Bipartite Graph Assignment and Deep Reinforcement Learning," *Transactions in Gis*, 2023, doi: 10.1111/tgis.13057.

[63] J. Zhao, M. Kong, Q. Li, and X. Sun, "Contract-Based Computing Resource Management via Deep Reinforcement Learning in Vehicular Fog Computing," *Ieee Access*, 2020, doi: 10.1109/access.2019.2963051.

[64] D. Tian, "An Intelligent Optimization Method for Wireless Communication Network Resources Based on Reinforcement Learning," *J Phys Conf Ser*, 2023, doi: 10.1088/1742-6596/2560/1/012036.

[65] S. Wen, R. Han, C. H. Liu, and L. Y. Chen, "Fast DRL-based Scheduler Configuration Tuning for Reducing Tail Latency in Edge-Cloud Jobs," *Journal of Cloud Computing Advances Systems and Applications*, 2023, doi: 10.1186/s13677-023-00465-z.

[66] F. Li, C. Fang, M. Liu, N. Li, and T. Sun, "Intelligent Computation Offloading Mechanism with Content Cache in Mobile Edge Computing," *Electronics (Basel)*, vol. 12, no. 5, p. 1254, Mar. 2023, doi: 10.3390/electronics12051254.